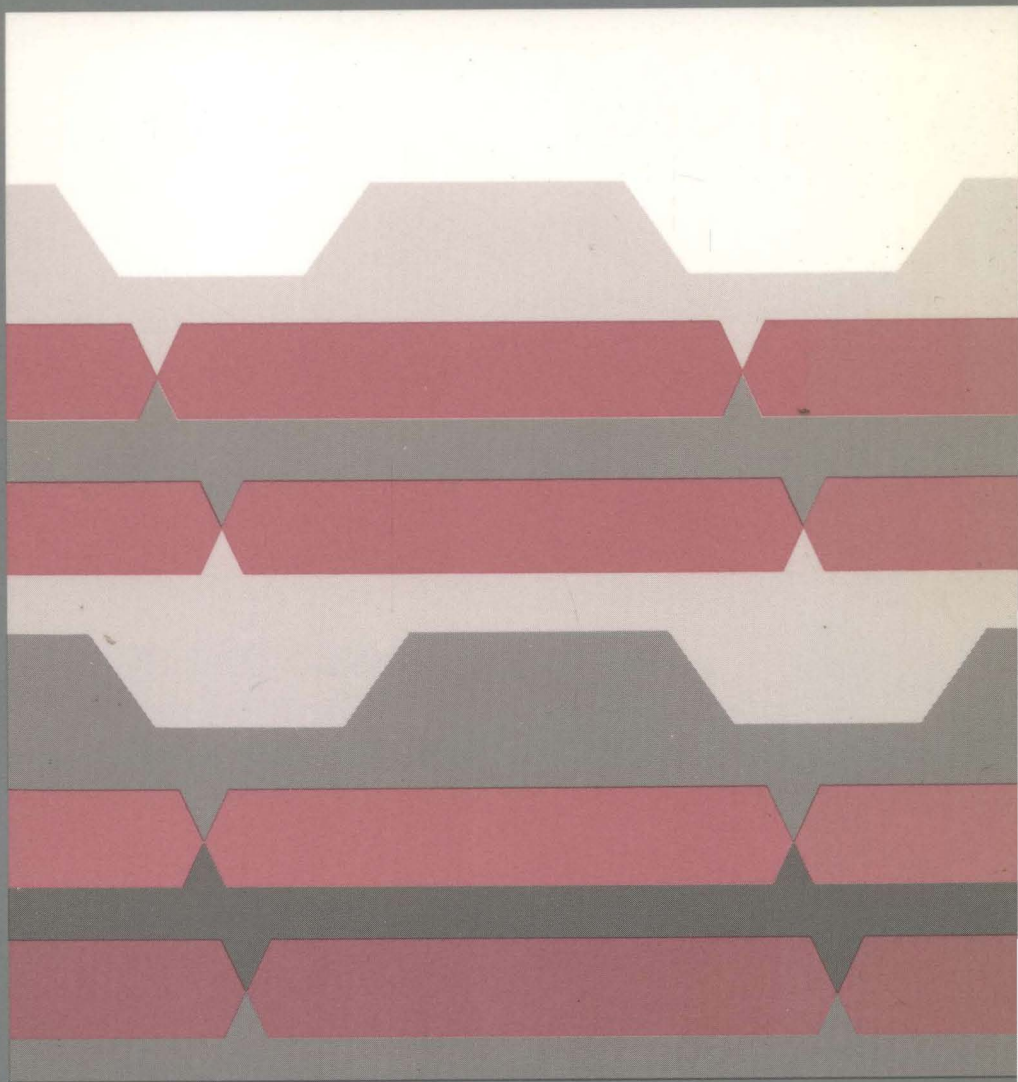
September, 1988

HITACHI®

HD6305/HD63L05 SERIES HANDBOOK

- USER'S MANUALS
- SOFTWARE APPLICATION NOTES
- HARDWARE APPLICATION NOTES



#U08

HD6305/HD63L05 SERIES HANDBOOK

■ USER'S MANUALS
- HD6305 U0
- HD6305 V0
- HD63705 V0
- HD6305X
- HD6305Y
- HD63P05Y
- HD63L05

■ SOFTWARE APPLICATION NOTES:

■ HARDWARE APPLICATION NOTES:

■ APPENDIX: TECHNICAL Q & A

■ SYSTEM APPLICATION NOTES

#U08

HITACHI®

MEDICAL APPLICATIONS

Hitachi's products are not authorized for use in MEDICAL APPLICATIONS, including, but not limited to, use in life support devices without the written consent of the appropriate officer of Hitachi's sales company. Buyers of Hitachi's products are requested to notify Hitachi's sales offices when planning to use the products in MEDICAL APPLICATIONS.

When using this manual, the reader should keep the following in mind:

1. This manual may, wholly or partially, be subject to change without notice.
2. All rights reserved: No one is permitted to reproduce or duplicate, in any form, the whole or part of this manual without Hitachi's permission.
3. Hitachi will not be responsible for any damage to the user that may result from accidents or any other reasons during operation of his unit according to this manual.
4. This manual neither ensures the enforcement of any industrial properties or other rights, nor sanctions the enforcement right thereof.
5. Circuitry and other examples described herein are meant merely to indicate characteristics and performance of Hitachi semiconductor-applied products. Hitachi assumes no responsibility for any patent infringements or other problems resulting from applications based on the examples described herein.
6. No license is granted by implication or otherwise under any patents or other rights of any third party or Hitachi, Ltd.

September 1988

Printed in U.S.A.

HD6305/HD63L05 SERIES HANDBOOK

TABLE OF CONTENTS

Section 4
HD6305X, HD6305Y, HD63P05Y User's Manual

Page

Section 5
HD63L05 User's Manual

Page

Section 6
Software Application Notes

Page

Section 7
Hardware Application Notes

Page

Section 8—Appendix
Technical Q and A (Part I)
8-Bit Single-Chip Microcomputer
HD6305X0, HD6305X1, HD6305X2
HD6305Y0, HD6305Y1, HD6305Y2

Section 8—Appendix
Technical Q and A (Part II)
8-Bit Single-Chip Microcomputer
HD6305U0, HD6305V0

Section 9
HD6305U0, HD6305V0, HD63705V0 User's Manual

Section One

Quick Reference Guide

HITACHI

Quick Reference Guide

HITACHI

■ CMOS 8-BIT SINGLE-CHIP MICROCOMPUTER HD6305 SERIES

Type No.			HD6305U0 HD63A05U0 HD63B05U0	HD6305V0 HD63A05V0 HD63B05V0	HD6305X0 HD63A05X0 HD63B05X0	HD6305X1 HD63A05X1 HD63B05X1
LSI Characteristics	Clock Frequency (MHz)		1.0 (HD6305U0) 1.5 (HD63A05U0) 2.0 (HD63B05U0)	1.0 (HD6305V0) 1.5 (HD63A05V0) 2.0 (HD63B05V0)	1.0 (HD6305X0) 1.5 (HD63A05X0) 2.0 (HD63B05X0)	1.0 (HD6305X1) 1.5 (HD63A05X1) 2.0 (HD63B05X1)
	Supply Voltage (V)		5.0	5.0	5.0	5.0
	Operating Temperature (°C)		0 ~ +70	0 ~ +70 *1	0 ~ +70 *1	0 ~ +70 *1
	Package†		DP-40,FP-54, CP-44	DP-40,FP-54,CP-44	DP-64S, FP-64	DP-64S, FP-64
Functions	Memory	ROM (k byte)	2	4	4	4
		RAM (byte)	128	192	128	128
	I/O Port	I/O Port	31	31	32	24
		Input Port	31	31	55	31
		Output Port	–	–	7	7
					16	–
	Interrupt	External	2	2	2	2
		Soft	1	1	1	1
		Timer	2	2	2	2
		Serial	1	1	1	1
	Timer					
	SCI					
	External Memory Expansion		–	–	–	12k bytes
	Other Features					
EPROM on Chip Type			HD63705V0C HD637A05V0C HD637B05V0C	HD63705V0C HD637A05V0C HD637B05V0C	–	–
EPROM on the Package Type			–	–	HD63P05Y0 HD63PA05Y0 HD63PB05Y0	HD63P05Y1 HD63PA05Y1 HD63PB05Y1
Evaluation Chip			–	–	–	–

*1 Wide Temperature Range (–40 ~ +85°C) version is available.

†DP; Plastic DIP, FP; Plastic Flat Package, CP; Plastic Leaded Chip Carrier (J-bend leads)

HD6305X2 HD63A05X2 HD63B05X2	HD6305Y0 HD63A05Y0 HD63B05Y0	HD6305Y1 HD63A05Y1 HD63B05Y1	HD6305Y2 HD63A05Y2 HD63B05Y2	HD63L05F1
1.0 (HD6305X2) 1.5 (HD63A05X2) 2.0 (HD63B05X2)	1.0 (HD6305Y0) 1.5 (HD63A05Y0) 2.0 (HD63B05Y0)	1.0 (HD6305Y1) 1.5 (HD63A05Y1) 2.0 (HD63B05Y1)	1.0 (HD6305Y2) 1.5 (HD63A05Y2) 2.0 (HD63B05Y2)	0.125
5.0	5.0	5.0	5.0	3.0
0 ~ +70*1	0 ~ +70*1	0 ~ +70*1	0 ~ +70*1	-20 ~ +75
DP-64S, FP-64	DP-64S, FP-64	DP-64S, FP-64	DP-64S, FP-64	DP-64S, FP-80
–	8	8	–	4
128	256	256	256	96
31	24	55	32	31	24	31	24	20	20
	7		7		7		7		–
	–		16		–		–		(19)
2	2	2	2	1
1	1	1	1	1
2	2	2	2	1
1	1	1	1	–
• 8-bit x 1 (with 7-bit prescaler) • 15-bit x 1 (combined with SCI)				• 8-bit x 1 (with 7-bit prescaler)
Synchronous				–
16k bytes	–	8 k bytes	16k bytes	–
• Low power dissipation modes (Wait, stop and standby)				• 8-bit A/D converter • LCD driver (6 x 7 segment) • Low power dissipation modes (Standby and halt)
–	–	–	–	–
–	HD63P05Y0 HD63PA05Y0 HD63PB05Y0	HD63P05Y1 HD63PA05Y1 HD63PB05Y1	–	–
–	–	–	–	HD63L05E

Section Two

• Addressing Modes
• CPU Architecture
• Instruction Set

- Addressing Modes
- CPU Architecture
- Instruction Set

Section 2
Addressing Modes, CPU Architecture, and Instruction Set
Table of Contents

NOTE

This section 2 applies only to HD6305, HD63P05, and HD63705 devices. The addressing modes, CPU architecture and instruction set for HD63L05 can be found in section 5.

1.1 Addressing Modes

Ten different addressing modes are available.

(1) Immediate

Refer to Fig. 1-2. The immediate addressing mode provides access to a constant which does not vary during execution of the program.

This access requires an instruction length of 2 bytes. The effective address (EA) is PC and the operand is fetched from the byte that follows the operation code.

(2) Direct

Refer to Fig. 1-3. In the direct addressing mode, the address of the operand is contained in the 2nd byte of the instruction. The user can gain direct access to memory up to the lower 255th address. All RAM and I/O registers are on page 0 of address space so that the direct addressing mode may be utilized.

(3) Extended

Refer to Fig. 1-4. The extended addressing is used for referencing to all addresses of memory. The EA is the contents of the 2 bytes that follow the operation code. An extended addressing instruction requires a length of 3 bytes.

(4) Relative

Refer to Fig. 1-5. The relative addressing mode is used with branch instructions only. When a branch occurs, the program counter is loaded with the contents of the byte following the operation code. EA = (PC) + 2 + Rel., where Rel. indicates a signed 8-bit data following the operation code. If no branch occurs, Rel. = 0. When a branch occurs, the program jumps to any byte in the range +129 to -127. A branch instruction requires a length of 2 bytes.

(5) Indexed (No Offset)

Refer to Fig. 1-6. The indexed addressing mode allows access up to the lower 255th address of memory. In this mode, an instruction requires a length of one byte. The EA is the contents of the index register.

(6) Indexed (8-bit Offset)

Refer to Fig. 1-7. The EA is the contents of the byte following the operation code, plus the contents of the index register. This mode allows access up to the lower 511th address of memory. Each instruction when used in the index addressing mode (8-bit offset) requires a length of 2 bytes.

(7) Indexed (16-bit Offset)

Refer to Fig. 1-8. The contents of the 2 bytes following the operation code are added to content of the index register to compute the value of EA. In this mode, the complete memory can be accessed. When used in the indexed addressing mode (16-bit offset), an instruction must be 3 bytes long.

(8) Bit Set/Clear

Refer to Fig. 1-9. This addressing mode is applied to the BSET and BCLR instructions that can set or clear any bit on page 0. The lower 3 bits of the operation code specify the bit to be set or cleared. The byte that follows the operation code indicates an address within page 0.

(9) Bit Test and Branch

Refer to Fig. 1-10. This addressing mode is applied to the BRSET and BRCLR instructions that can test any bit within page 0 and can be branched in the relative addressing mode. The byte to be tested is addressed depending on the contents of the byte following the operation code. Individual bits within the byte to be tested are specified by the lower 3 bits of the operation code. The 3rd byte represents a relative value which will be added to the program counter when a branch condition is established. Each of these instructions should be 3 bytes long. The value of the test bit is written in the carry bit of the condition code register.

(10) Implied

Refer to Fig. 1-11. This mode involves no EA. All information needed for execution of an instruction is contained in the operation code. Direct manipulation on the accumulator and index register is included in the implied addressing mode. Other instructions such as SWI and RTI are also used in this mode. All instructions used in the implied addressing mode should have a length of one byte.

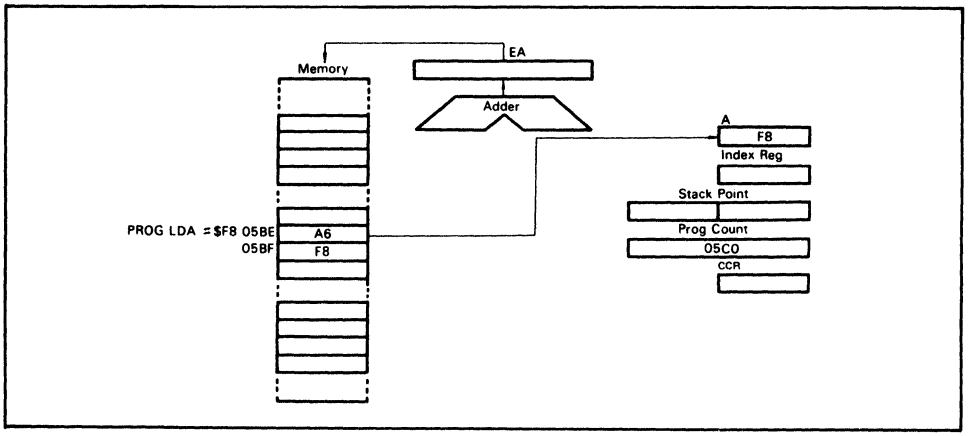


Fig. 1-2 Example of Immediate Addressing

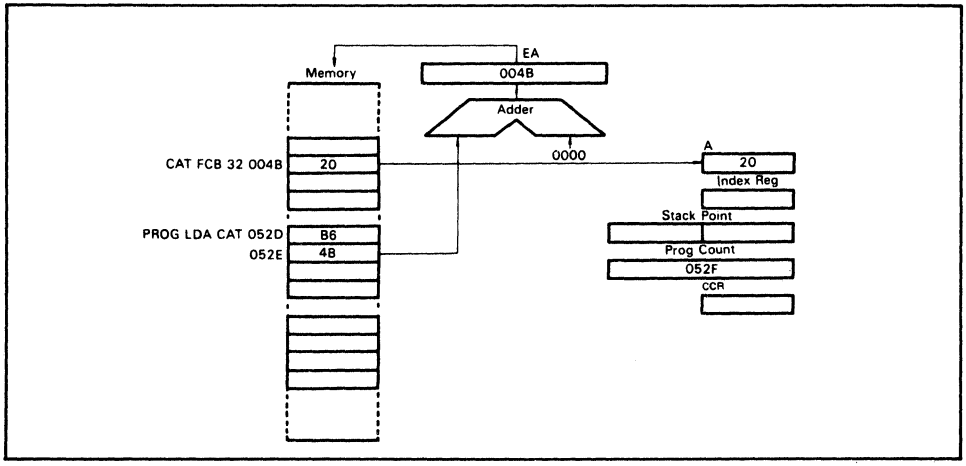


Fig. 1-3 Example of Direct Addressing

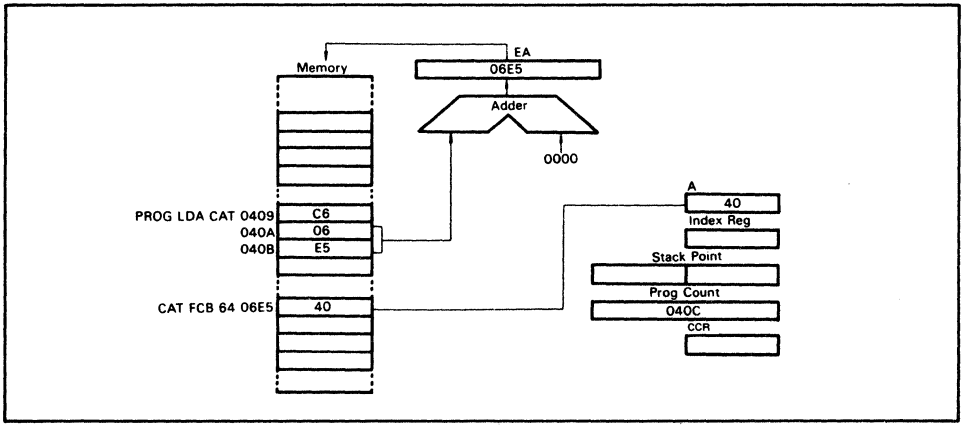


Fig. 1-4 Example of Extended Addressing

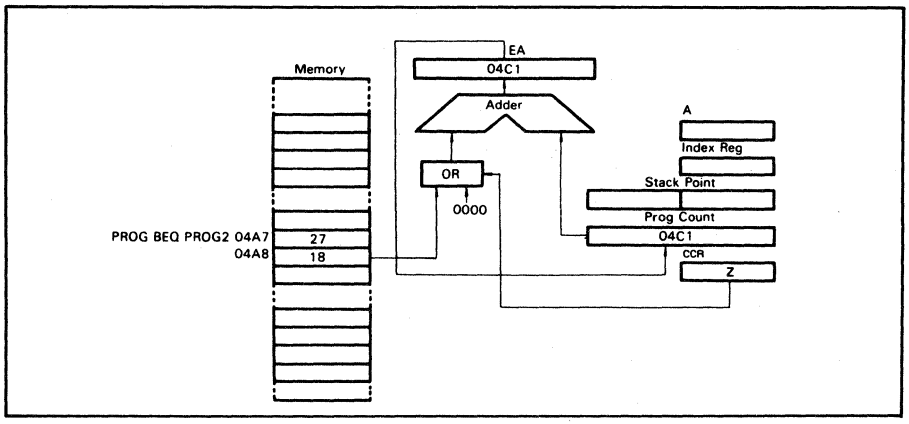


Fig. 1-5 Example of Relative Addressing

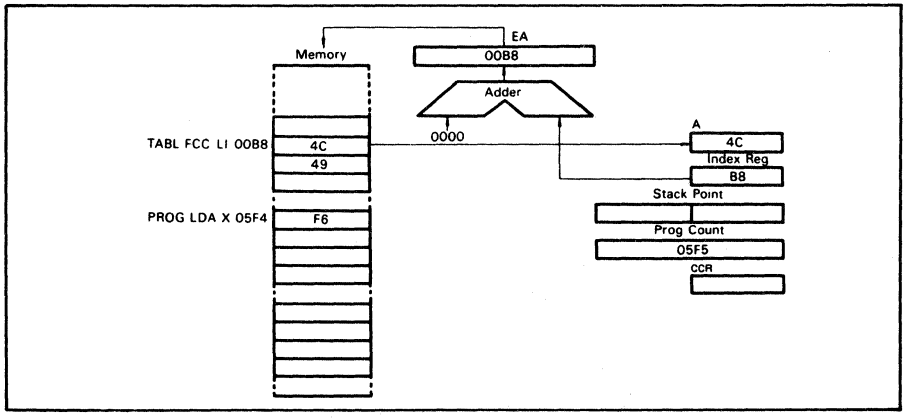


Fig. 1-6 Example of Indexed (No Offset) Addressing

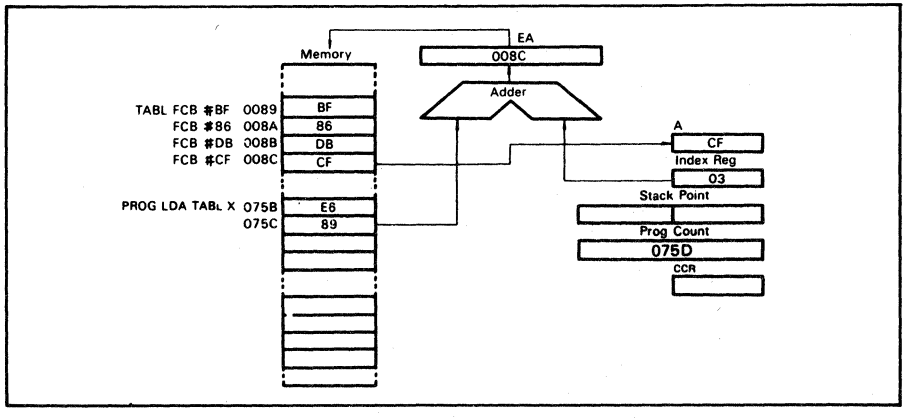


Fig. 1-7 Example of Indexed (8-bit Offset) Addressing

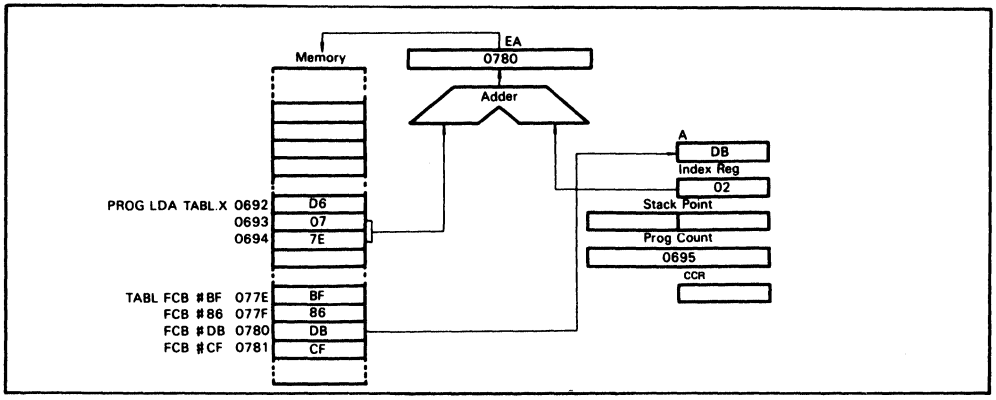


Fig. 1-8 Example of Indexed (16-bit Offset) Addressing

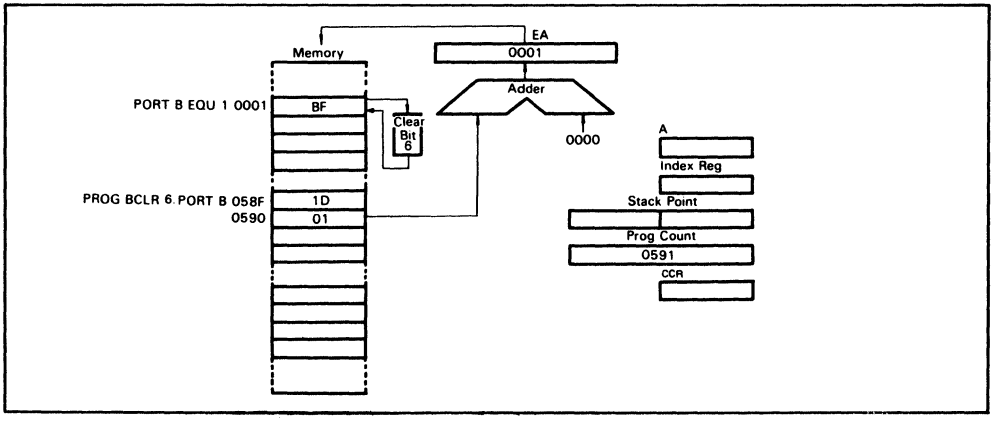


Fig. 1-9 Example of Bit Set/Clear Addressing

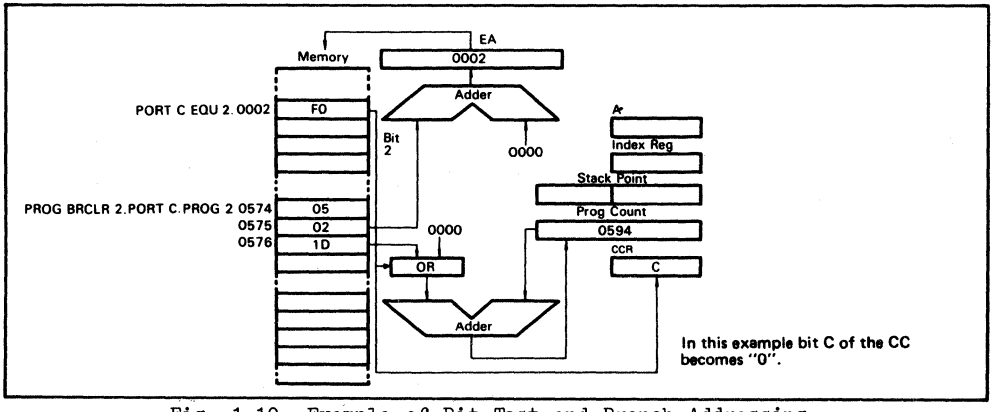


Fig. 1-10 Example of Bit Test and Branch Addressing

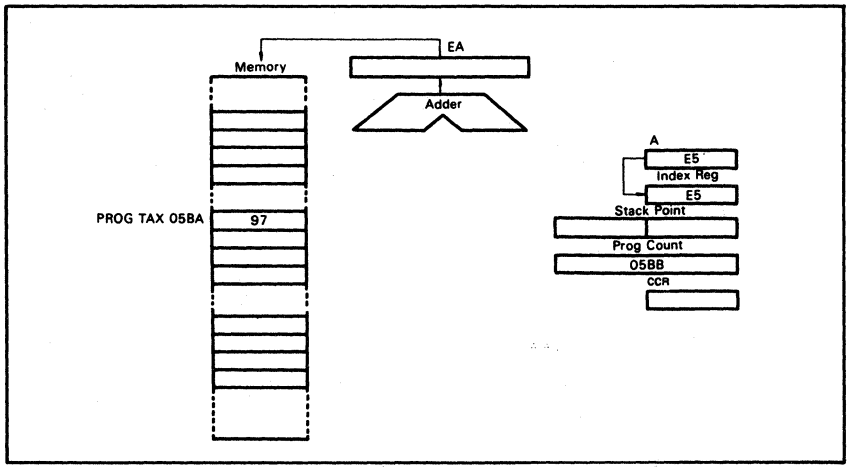


Fig. 1-11 Example of Implied Addressing

1.2 CPU Registers

This CPU contains five registers available to the programmer. They are shown in Fig. 1-12.

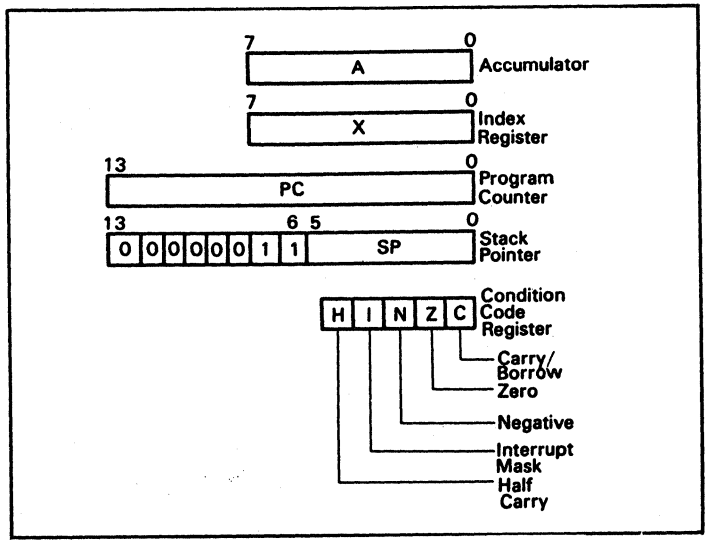


Fig. 1-12 Programming Model

(1) Accumulator (A)

The accumulator is an 8-bit general purpose register which holds operands and results of the arithmetic operations or data manipulations.

(2) Index Register (X)

The index register is an 8-bit register used for the indexed addressing mode. It contains an 8-bit value which is added to an offset to create an effective address.

The index register can also be used for data manipulations with read-modify-write instructions.

When not performing addressing operations, the register can be used as a temporary storage area.

(3) Program Counter (PC)

The program counter is a 14-bit register which contains the address of the next instruction to be executed.

(4) Stack Pointer (SP)

The stack pointer is a 14-bit register containing the address of the next free location of the stack. Initially, the stack pointer is set to location $00FF. It is decremented as a data is pushed on to the stack and incremented as data is then pulled out of the stack. The 8 high-order bits of the stack pointer are fixed to 00000011. During an MCU reset or a reset stack pointer (RSP) instruction, the pointer is set to location $00FF. Subroutines and interrupts may be nested down to $00C1, which allows programmers to use up to 31 levels of subroutine calls and 12 levels of interrupt responses.

(5) Condition Code Register (CCR)

The condition code register is a 5-bit register indicating the results of the instruction just executed. These bits can be individually tested by conditional branch instructions. Each bit is described in the following paragraphs.

(a) Half Carry (H)

When set, this bit indicates that a carry occurred between bit 3 and 4 during an arithmetic operation (ADD, ADC).

(b) Interrupt (I)

Setting this bit masks all interrupts except for software ones. If an interrupt occurs while this bit is set, the interrupt is latched and is processed as soon as the interrupt bit (I) is cleared. (More precisely the interrupt enters the servicing routine after the instruction next to the CLI is executed.)

(c) Negative (N)

When set, this bit indicates that the result of the last arithmetic, logical, or data manipulation is negative. (Bit 7 in the result is a logical "1".)

(d) Zero (Z)

When set, this bit indicates that the result of the last arithmetic, logical, or data manipulation is zero.

(e) Carry/Borrow (C)

When set, this bit indicates that a carry or borrow occurred during the last arithmetic operation. This bit is also affected by bit test and branch instructions, shifts and rotates.

1.3 Instruction Set

The HD6305U0, HD6305V0, and HD63705V0 MCU provide object codes upward compatible with the HD6805 family. They are designed to save execution time of key instructions in order to improve through put.

3 additional instructions; DAA, WAIT, STOP; are available.

The HD6305U0, and HD6305V0, HD63705V0 MCU have 62 basic instructions. They can be classified into five categories: register/memory, read-modify-write, branch, bit manipulation, and control. The details of each instruction are shown in the following tables. All the instructions in a given type are presented in individual tables.

(1) Register/Memory Instruction

Most of these instructions use two operands. One operand is either the accumulator or the index register. The other is obtained from memory by using one of the addressing modes. The unconditional jump (JMP) and the jump to subroutine (JSR) instructions have no register operand. Refer to Table 1-1.

(2) Read-Modify-Write Instructions

These instructions read a memory location or a register, modify or test its contents, and write the modified value back to memory or to the register. The test for zero (TST) instruction is an exception to the read-modify-write instructions since it does not write data. Refer to Table 1-2.

(3) Branch Instructions

The branch instruction cause a branch from the program when a certain condition is met. Refer to Table 1-3.

(4) Bit Manipulation Instructions

These instructions are used on any bit in the lower 255 bytes of the memory. Two groups are available, one either sets or clears and the other performs the bit and test branch operations. Refer to Table 1-4.

(5) Control Instructions

These instructions control the MCU operation during program execution. Refer to Table 1-5.

(6) Alphabetical Listing

Table 1-6 lists the complete instruction set in alphabetical order.

(7) Operation Code Map

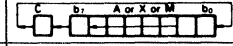
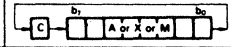
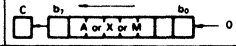
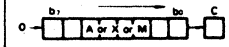
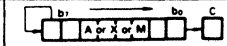
Table 1-7 is an operation code map for the instructions used on the MCU.

Table 1-1 Register/Memory Instructions

Operations	Mnemonic	Addressing Modes: Immediate			Direct			Extended			Indexed (No Offset)			Indexed (8-Bit Offset)			Indexed (16-Bit Offset)			Boolean/ Arithmetic Operation	Condition Code				
		OP	#	~	OP	#	~	OP	#	~	OP	#	~	OP	#	~	OP	#	~		H	I	N	Z	C
Load A from Memory	LDA	A6	2	2	B6	2	3	C6	3	4	F6	1	3	E6	2	4	D6	3	5	M →A	•	•	^	^	•
Load X from Memory	LDX	AE	2	2	BE	2	3	CE	3	4	FE	1	3	EE	2	4	DE	3	5	M →X	•	•	^	^	•
Store A in Memory	STA	—	—	—	B7	2	3	C7	3	4	F7	1	4	E7	2	4	D7	3	5	A →M	•	•	^	^	•
Store X in Memory	STX	—	—	—	BF	2	3	CF	3	4	FF	1	4	EF	2	4	DF	3	5	X →M	•	•	^	^	•
Add Memory to A	ADD	AB	2	2	BB	2	3	CB	3	4	FB	1	3	EB	2	4	DB	3	5	A+M →A	^	•	^	^	^
Add Memory and Carry to A	ADC	A9	2	2	B9	2	3	C9	3	4	F9	1	3	E9	2	4	D9	3	5	A+M+C →A	^	•	^	^	^
Subtract Memory	SUB	A0	2	2	B0	2	3	C0	3	4	F0	1	3	E0	2	4	D0	3	5	A−M →A	•	•	^	^	^
Subtract Memory from A with Borrow	SBC	A2	2	2	B2	2	3	C2	3	4	F2	1	3	E2	2	4	D2	3	5	A−M−C →A	•	•	^	^	^
AND Memory to A	AND	A4	2	2	B4	2	3	C4	3	4	F4	1	3	E4	2	4	D4	3	5	A · M →A	•	•	^	^	•
OR Memory with A	ORA	AA	2	2	BA	2	3	CA	3	4	FA	1	3	EA	2	4	DA	3	5	A+M →A	•	•	^	^	•
Exclusive OR Memory with A	EOR	A8	2	2	B8	2	3	C8	3	4	F8	1	3	E8	2	4	D8	3	5	A+ M →A	•	•	^	^	•
Arithmetic Compare A with Memory	CMP	A1	2	2	B1	2	3	C1	3	4	F1	1	3	E1	2	4	D1	3	5	A−M	•	•	^	^	^
Arithmetic Compare X with Memory	CPX	A3	2	2	B3	2	3	C3	3	4	F3	1	3	E3	2	4	D3	3	5	X−M	•	•	^	^	^
Bit Test Memory with A (Logical Compare)	BIT	A5	2	2	B5	2	3	C5	3	4	F5	1	3	E5	2	4	D5	3	5	A · M	•	•	^	^	•
Jump Unconditional	JMP	—	—	—	BC	2	2	CC	3	3	FC	1	2	EC	2	3	DC	3	4		•	•	•	•	•
Jump to Subroutine	JSR	—	—	—	BD	2	5	CD	3	6	FD	1	5	ED	2	5	DD	3	6		•	•	•	•	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 1-2 Read/Modify/Write Instructions

Operations	Mnemonic	Addressing Modes: Implied(A)			Implied(X)			Direct			Indexed (No Offset)			Indexed (8-Bit Offset)			Boolean/Arithmetic Operation	Condition Code				
		OP	#	~	OP	#	~	OP	#	~	OP	#	~	OP	#	~		H	I	N	Z	C
Increment	INC	4C	1	2	5C	1	2	3C	2	5	7C	1	5	6C	2	6	A+1 →A or X+1 →X or M+1 →M	•	•	^	^	•
Decrement	DEC	4A	1	2	5A	1	2	3A	2	5	7A	1	5	6A	2	6	A−1 →A or X−1 →X or M−1 →M	•	•	^	^	•
Clear	CLR	4F	1	2	5F	1	2	3F	2	5	7F	1	5	6F	2	6	00 →A or 00 →X or 00 →M	•	•	0	1	•
Complement	COM	43	1	2	53	1	2	33	2	5	73	1	5	63	2	6	$\overline{A}$ →A or $\overline{X}$ →X or $\overline{M}$ →M	•	•	^	^	1
Negate (2's Complement)	NEG	40	1	2	50	1	2	30	2	5	70	1	5	60	2	6	00−A →A or 00−X →X or 00−M →M	•	•	^	^	^
Rotate Left Thru Carry	ROL	49	1	2	59	1	2	39	2	5	79	1	5	69	2	6		•	•	^	^	^
Rotate Right Thru Carry	ROR	46	1	2	56	1	2	36	2	5	76	1	5	66	2	6		•	•	^	^	^
Logical Shift Left	LSL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6		•	•	^	^	^
Logical Shift Right	LSR	44	1	2	54	1	2	34	2	5	74	1	5	64	2	6		•	•	0	^	^
Arithmetic Shift Right	ASR	47	1	2	57	1	2	37	2	5	77	1	5	67	2	6		•	•	^	^	^
Arithmetic Shift Left	ASL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6	Equal to LSL	•	•	^	^	^
Test for Negative or Zero	TST	4D	1	2	5D	1	2	3D	2	4	7D	1	4	6D	2	5	A−00 or X−00 or M−00	•	•	^	^	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 1-3 Branch Instruction

Operations	Mnemonic	Addressing Modes			Branch Test	Condition Code				
		Relative								
		OP	#	~		H	I	N	Z	C
Branch Always	BRA	20	2	3	None	●	●	●	●	●
Branch Never	BRN	21	2	3	None	●	●	●	●	●
Branch IF Higher	BHI	22	2	3	C+Z=0	●	●	●	●	●
Branch IF Lower or Same	BLS	23	2	3	C+Z=1	●	●	●	●	●
Branch IF Carry Clear	BCC	24	2	3	C=0	●	●	●	●	●
(Branch IF Higher or Same)	(BHS)	24	2	3	C=0	●	●	●	●	●
Branch IF Carry Set	BCS	25	2	3	C=1	●	●	●	●	●
(Branch IF Lower)	(BLO)	25	2	3	C=1	●	●	●	●	●
Branch IF Not Equal	BNE	26	2	3	Z=0	●	●	●	●	●
Branch IF Equal	BEQ	27	2	3	Z=1	●	●	●	●	●
Branch IF Half Carry Clear	BHCC	28	2	3	H=0	●	●	●	●	●
Branch IF Half Carry Set	BHCS	29	2	3	H=1	●	●	●	●	●
Branch IF Plus	BPL	2A	2	3	N=0	●	●	●	●	●
Branch IF Minus	BMI	2B	2	3	N=1	●	●	●	●	●
Branch IF Interrupt Mask Bit is Clear	BMC	2C	2	3	I=0	●	●	●	●	●
Branch IF Interrupt Mask Bit is Set	BMS	2D	2	3	I=1	●	●	●	●	●
Branch IF Interrupt Line is Low	BIL	2E	2	3	INT=0	●	●	●	●	●
Branch IF Interrupt Line is High	BIH	2F	2	3	INT=1	●	●	●	●	●
Branch to Subroutine	BSR	AD	2	5	——	●	●	●	●	●

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 1-4 Bit Manipulation Instructions

Operations	Mnemonic	Addressing Modes						Boolean/ Arithmetic Operation	Branch Test	Condition Code				
		Bit Set/Clear			Bit Test and Branch									
		OP	#	~	OP	#	~			H	I	N	Z	C
Branch IF Bit n is set	BRSET n(n=0···7)	—	—	—	2·n	3	5	——	Mn=1	●	●	●	●	∧
Branch IF Bit n is clear	BRCLR n(n=0···7)	—	—	—	01+2·n	3	5	——	Mn=0	●	●	●	●	∧
Set Bit n	BSET n(n=0···7)	10+2·n	2	5	—	—	—	1→Mn	——	●	●	●	●	●
Clear Bit n	BCLR n(n=0···7)	11+2·n	2	5	—	—	—	0→Mn	——	●	●	●	●	●

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Table 1-5 Control Instructions

Operations	Mnemonic	Addressing Modes			Boolean Operation	Condition Code				
		Implied								
		OP	#	~		H	I	N	Z	C
Transfer A to X	TAX	97	1	2	A→X	●	●	●	●	●
Transfer X to A	TXA	9F	1	2	X→A	●	●	●	●	●
Set Carry Bit	SEC	99	1	1	1→C	●	●	●	●	1
Clear Carry Bit	CLC	98	1	1	0→C	●	●	●	●	0
Set Interrupt Mask Bit	SEI	9B	1	2	1→I	●	1	●	●	●
Clear Interrupt Mask Bit	CLI	9A	1	2	0→I	●	0	●	●	●
Software Interrupt	SWI	83	1	10		●	1	●	●	●
Return from Subroutine	RTS	81	1	5		●	●	●	●	●
Return from Interrupt	RTI	80	1	8		?	?	?	?	?
Reset Stack Pointer	RSP	9C	1	2	$FF→SP	●	●	●	●	●
No-Operation	NOP	9D	1	1	Advance Prog. Cntr. Only	●	●	●	●	●
Decimal Adjust A	DAA	8D	1	2	Converts binary add of BCD charcters into BCD format	●	●	∧	∧	∧*
Stop	STOP	8E	1	4	0 → I	●	0	●	●	●
Wait	WAIT	8F	1	4	0 → I	●	0	●	●	●

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

* Are BCD characters of upper byte 10 or more? (They are not cleared if set in advance.)

Table 1-6 Instruction Set (in Alphabetical Order)

Mnemonic	Addressing Modes										Condition Code				
	Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8-Bit)	Indexed (16-Bit)	Bit Set/ Clear	Bit Test & Branch	H	I	N	Z	C
ADC		×	×	×		×	×	×			∧	●	∧	∧	∧
ADD		×	×	×		×	×	×			∧	●	∧	∧	∧
AND		×	×	×		×	×	×			●	●	∧	∧	●
ASL	×		×			×	×				●	●	∧	∧	∧
ASR	×		×			×	×				●	●	∧	∧	∧
BCC					×						●	●	●	●	●
BCLR									×		●	●	●	●	●
BCS					×						●	●	●	●	●
BEQ					×						●	●	●	●	●
BHCC					×						●	●	●	●	●
BHCS					×						●	●	●	●	●
BHI					×						●	●	●	●	●
(BHS)					×						●	●	●	●	●
BIH					×						●	●	●	●	●
BIL					×						●	●	●	●	●
BIT		×	×	×		×	×	×			●	●	∧	∧	●
(BLO)					×						●	●	●	●	●
BLS					×						●	●	●	●	●
BMC					×						●	●	●	●	●
BMI					×						●	●	●	●	●
BMS					×						●	●	●	●	●
BNE					×						●	●	●	●	●
BPL					×						●	●	●	●	●
BRA					×						●	●	●	●	●

(to be continued)

Condition Code Symbols:

H	Half Carry (From Bit 3)	C	Carry/Borrow
I	Interrupt Mask	∧	Test and Set if True, Cleared Otherwise
N	Negative (Sign Bit)	●	Not Affected
Z	Zero	?	Load CC Register From Stack

Mnemonic	Addressing Modes										Condition Code				
	Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8-Bit)	Indexed (16-Bit)	Bit Set Clear	Bit Test & Branch	H	I	N	Z	C
BRN					×						●	●	●	●	●
BRCLR										×	●	●	●	●	^
BRSET										×	●	●	●	●	^
BSET									×		●	●	●	●	●
BSR					×						●	●	●	●	●
CLC	×										●	●	●	●	0
CLI	×										●	0	●	●	●
CLR	×		×			×	×				●	●	0	1	●
CMP		×	×	×		×	×	×			●	●	^	^	^
COM	×		×			×	×				●	●	^	^	1
CPX		×	×	×		×	×	×			●	●	^	^	^
DAA	×										●	●	^	^	^
DEC	×		×			×	×				●	●	^	^	●
EOR		×	×	×		×	×	×			●	●	^	^	●
INC	×		×			×	×				●	●	^	^	●
JMP			×	×		×	×	×			●	●	●	●	●
JSR			×	×		×	×	×			●	●	●	●	●
LDA		×	×	×		×	×	×			●	●	^	^	●
LDX		×	×	×		×	×	×			●	●	^	^	●
LSL	×		×			×	×				●	●	^	^	^
LSR	×		×			×	×				●	●	0	^	^
NEG	×		×			×	×				●	●	^	^	^
NOP	×										●	●	●	●	●
ORA		×	×	×		×	×	×			●	●	^	^	●
ROL	×		×			×	×				●	●	^	^	^
ROR	×		×			×	×				●	●	^	^	^
RSP	×										●	●	●	●	●
RTI	×										?	?	?	?	?
RTS	×										●	●	●	●	●
SBC		×	×	×		×	×	×			●	●	^	^	^
SEC	×										●	●	●	●	1
SEI	×										●	1	●	●	●
STA			×	×		×	×	×			●	●	^	^	●
STOP	×										●	0	●	●	●
STX			×	×		×	×	×			●	●	^	^	●
SUB		×	×	×		×	×	×			●	●	^	^	^
SWI	×										●	1	●	●	●
TAX	×										●	●	●	●	●
TST	×		×			×	×				●	●	^	^	●
TXA	×										●	●	●	●	●
WAIT	×										●	0	●	●	●

Condition Code Symbols:

H	Half Carry (From Bit 3)	C	Carry/Borrow
I	Interrupt Mask	^	Test and Set if True, Cleared Otherwise
N	Negative (Sign Bit)	●	Not Affected
Z	Zero	?	Load CC Register From Stack

Table 1-7 Operation Code Map

	Bit Manipulation		Branch	Read/Modify/Write					Control		Register/Memory						
	Test & Branch	Set/ Clear	Rel	DIR	A	X	,X1	,X0	IMP	IMP	IMM	DIR	EXT	,X2	,X1	,X0	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	← HIGH
0	BRSET0	BSET0	BRA	NEG					RTI*	—	SUB						0
1	BRCLR0	BCLR0	BRN	—					RTS*	—	CMP						1
2	BRSET1	BSET1	BHI	—					—	—	SBC						2
3	BRCLR1	BCLR1	BLS	COM					SWI*	—	CPX						3
4	BRSET2	BSET2	BCC	LSR					—	—	AND						4
5	BRCLR2	BCLR2	BCS	—					—	—	BIT						5
6	BRSET3	BSET3	BNE	ROR					—	—	LDA						6
7	BRCLR3	BCLR3	BEQ	ASR					—	TAX*	—	STA				STA(+1)	7
8	BRSET4	BSET4	BHCC	LSL/ASL					—	CLC	EOR						8
9	BRCLR4	BCLR4	BHCS	ROL					—	SEC	ADC						9
A	BRSET5	BSET5	BPL	DEC					—	CLI*	ORA						A
B	BRCLR5	BCLR5	BMI	—					—	SEI*	ADD						B
C	BRSET6	BSET6	BMC	INC					—	RSP*	—	JMP(−1)					C
D	BRCLR6	BCLR6	BMS	TST(−1)	TST			TST(−1)	DAA*	NOP	BSR*	JSR(+2)		JSR(+1)		JSR(+2)	D
E	BRSET7	BSET7	BIL	—					STOP*	—	LDX						E
F	BRCLR7	BCLR7	BIH	CLR					WAIT*	TXA*	—	STX				STX(+1)	F
	3/5	2/5	2/3	2/5	1/2	1/2	2/6	1/5	1/*	1/1	2/2	2/3	3/4	3/5	2/4	1/3	

LOW

(NOTES) 1. "—" is an undefined operation code.
2. The lowermost numbers in each column represent a byte count and the number of cycles required (byte count/number of cycles). The number of cycles for the mnemonics asterisked (*) is as follows:

RTI	8	TAX	2
RTS	5	RSP	2
SWI	10	TXA	2
DAA	2	BSR	5
STOP	4	CLI	2
WAIT	4	SEI	2

3. The parenthesized numbers must be added to the cycle count of the particular instruction.

(11) Additional Instructions

The following new instructions are used on the HD6305, and HD63705.

DAA Converts the contents of the accumulator into BCD code.

WAIT Causes the MCU to enter the wait mode.
Refer to "2.9 Low Power Consumption Mode".

STOP Causes the MCU to enter the stop mode.
Refer to "2.9 Low Power Consumption Mode".

1.4 Bit Manipulation

The HD6305, HD63705 MCU can use a single instruction (BSET or BCLR) to set or clear one bit of the RAM or an I/O port.

Every bit of memory or I/O within page 0 ($00 ∿ $FF) can be tested by the BRSET or BRCLR instruction depending on the result of the test, the program can branch to required destinations. Since bits in the RAM, or I/O can be manipulated, the user may use a bit within the RAM as a flag or handle a single I/O bit as an independent I/O terminal. Fig. 1-13 shows an example of bit manipulation and the validity of test instructions. In the example, the program is configured assuming that bit 0 of port A is connected to a zero cross detector circuit and bit 1 of the same port to the trigger of a triac.

The program shown can activate the triac within 10μs from zero-crossing through the use of only 7 bytes on the ROM. The internal timer provides a required time of delay and pulse width modulation of power is also possible.

```
                    :
SELF 1.     BRCLR 0, PORT A, SELF 1
            BSET 1, PORT A
            BCLR 1, PORT A
                    :
                    :
```

Fig. 1-13 Example of Bit Manipulation

1.5 Symbols and Abbreviations

Shown below are the meanings of symbols and abbreviations.

(1) Operation

() = contents

← = movement direction

+ = addition

− = subtraction

∧ = AND

∨ = OR

⊕ = Exclusive OR

$\bar{x}$ = NOT

(2) Register symbols in CPU

ACCA = accumulator A

CCR = condition codes register

IX = index register, 8 bits

PC = program counter, 14 bits

PCH = the six most significant bits of program counter

PCL = the eight least significant bits of program counter

SP = stack pointer, 6 bits

(3) Memory and addressing codes

M = stored address

MH = the eight most significant bits of stored address

ML = the eight least significant bits of stored address

M+1 = stored address M plus 1

Msp = stored address indicated by stack pointer

Imm = immediate value

Disp = displacement value = M − (IX)

D = displacement value = M − (IX)

DH = displacement value = the eight most significant bits

DL = displacement value = the eight least significant bits

Rel = relative value

IMPLIED = implied addressing

RELATIVE = relative addressing

ACCUMULATOR = accumulator addressing

INDEX REG. = index register addressing
IMMEDIATE = immediate addressing
DIRECT = direct addressing
EXTENDED = extended addressing
INDEXED 0 BYTE OFFSET = indexed addressing 0 byte offset
INDEXED 1 BYTE OFFSET = indexed addressing 1 byte offset
INDEXED 2 BYTE OFFSET = indexed addressing 2 byte offset
EA = effective address

(4) Contents of bits 0 through 4 of condition codes register

C = carry - borrow bit 0
Z = zero bit 1
N = negative bit 2
I = interrupt mask bit 3
H = half carry from bit 3 to bit 4 bit 4

(5) Status of each bit before execution of instruction

An = bit n of ACCA (n = 7, 6, 5, ..., 0)
Mn = bit n of M (n = 7, 6, 5, ..., 0)
Xn = bit n of IX (n = 7, 6, 5, ..., 0)

(6) Status of each bit on result after execution of instruction

Rn = bit n of result (n = 7, 6, 5, ..., 0)

(7) Symbols on instruction's format

P = each addressing mode on Immediate, Direct, Extended and index of 0, 1 and 2 byte offset
Q = each addressing mode on Direct and index of 0 and 1 byte offset
A = accumulator addressing mode
X = index register addressing mode
DR = direct addressing mode
dd = relative operand (8 bits)
n = bit n of memory (n = 7, 6, 5, ..., 0)

(8) Status of HD6305's interrupt pin

$\overline{INT}$ = status of interrupt pin (high, low)

Arithmetic Operation

ADC

ADC (ADd with Carry)

Format

ADC P

Operation

ACCA ← (ACCA) + (M) + (C)

Condition Codes

H: Set if a carry occurs from bit 3; otherwise reset.
I: Not affected.
N: Set if the most significant bit of the result is set; reset otherwise.
Z: Set if the result is 0; otherwise reset.
C: Set if a carry occurs from the most significant bit of the result; otherwise reset.

Description

Adds the contents of the carry bit C to the sum of the contents of ACCA and M and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	ADC	#Imm	A9	Imm		2	2
DIRECT	ADC	M	B9	M		2	3
EXTENDED	ADC	M	C9	MH	ML	3	4
INDEXED 0 BYTE OFFSET	ADC	0,X	F9			1	3
INDEXED 1 BYTE OFFSET	ADC	Disp,X	E9	D		2	4
INDEXED 2 BYTE OFFSET	ADC	Disp,X	D9	DH	DL	3	5

Example

```
0100 B6 02      LDA     VAL2        (EXVAL5,EXVAL6)+(VAL1,VAL2)
0102 CB 0006    ADD     EXVAL6      *          =(EXVAL5, EXVAL6)
0105 C7 0006    STA     EXVAL6      *
0108 B6 01      LDA     VAL1        *
010A C9 0005    ADC     EXVAL5      ***
010D C7 0005    STA     EXVAL5      *
```

ADD (ADD without carry)

Format

ADD P

Operation

ACCA ← (ACCA) + (M)

Condition Codes

H: Set if a carry occurs from bit 3 otherwise reset.
I: Not affected.
N: Set if the significant bit of the result is 1; otherwise reset.
Z: Set if the result is 0; otherwise reset.
C: Set if a carry occurs from the most significant bit of the result; otherwise reset.

Description

Adds the M contents to ACCA contents, and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	ADD	#Imm	AB	Imm		2	2
DIRECT	ADD	M	BB	M		2	3
EXTENDED	ADD	M	CB	MH	ML	3	4
INDEXED 0 BYTE OFFSET	ADD	0,X	FB			1	3
INDEXED 1 BYTE OFFSET	ADD	Disp,X	EB	D		2	4
INDEXED 2 BYTE OFFSET	ADD	Disp,X	DB	DH	DL	3	5

Example

```
0110 B6 10      LDA     VAL1    (VAL1)+(WORK)=(RESULT)
0112 BB FF      ADD     WORK    *
0114 B7 50      STA     RESULT  *
```

Logical Operation
AND

AND (logical AND)

Format

AND P

Operation

ACCA ← (ACCA) ∧ (M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Not affected.

Description

Performs logical AND between the ACCA contents and the M contents, and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	AND	#Imm	A4	Imm		2	2
DIRECT	AND	M	B4	M		2	3
EXTENDED	AND	M	C4	MH	ML	3	4
INDEXED 0 BYTE OFFSET	AND	O,X	F4			1	3
INDEXED 1 BYTE OFFSET	AND	Disp,X	E4	D		2	4
INDEXED 2 BYTE OFFSET	AND	Disp,X	D4	DH	DL	3	5

Example

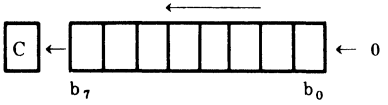
```
0107 F6          LDA     0,X     ERASE UPPER 4 BITS
0108 A4 0F       AND     #$0F    *
010A F7          STA     0,X     (RESTORE)
010B 5C          INC     X       *
010C 20 F2       BRA     LOOP    *
```

ASL (Arithmetic Shift Left)

Format

ASL Q
ASL A
ASL X

Operation



Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Set if the most significant bit is "1" before shifting; otherwise reset.

Description

Shifts the contents of ACCA, IX or M 1 bit to the left. The bit 0 is loaded with "0". The carry bit C is loaded with the bit 7 of ACCA, IX or M.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	ASL	A	48			1	2
INDEX REG.	ASL	X	58			1	2
DIRECT	ASL	M	38	M		2	5
INDEXED 0 BYTE OFFSET	ASL	0,X	78			1	5
INDEXED 1 BYTE OFFSET	ASL	Disp,X	68	D		2	6

Example

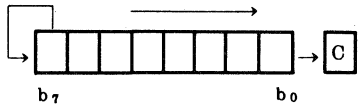
```
010E B6 FF              LDA     WORK
0110 48        CHECK    ASL     A       BRANCH FOLLOWING BIT
0111 25 2E              BCS     BITON   *       7-6-5-4-3-2-1-0
     0113      BITOFF   EQU     *
0113 AE 64              LDX     #100
```

ASR (Arithmetic Shift Right)

Format

ASR Q
ASR A
ASR X

Operation



Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Set if the most significant bit is "1" before shifting; otherwise reset.

Description

Shifts the contents of ACCA, IX or M 1-bit to the right. The bit 7 is not affected. The bit 0 is loaded into the carry bit C.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	ASR	A	47			1	2
INDEX REG.	ASR	X	57			1	2
DIRECT	ASR	M	37	M		2	5
INDEXED 0 BYTE OFFSET	ASR	0,X	77	D		1	5
INDEXED 1 BYTE OFFSET	ASR	Disp,X	67			2	6

Example

```
0143 37 FF      ASR   WORK      BRANCH OPTION (KEEPING BIT 7)
0145 25 17      BCS   OPT0
0147 37 FF      ASR   WORK
0149 25 1B      BCS   OPT1
014B 37 FF      ASR   WORK
014D 25 46      BCS   OPT2
```

BCC (Branch if Carry Clear)

Format

BCC dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (C)=0

Description

Tests the state of the C bit and causes a branch if C is "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BCC	Rel	24	Rel		2	3

Example

```
014F B6 20          LDA     VAL2
0151 CB 0600        ADD     EXVAL6
0154 24 11          BCC     NORMAL    KETA  AGARI  NASHI
                    *
0156 5C             INC     X         KETA  AGARI
```

BCLR (Bit CLeaR bit n)

Format

BCLR n, DR

Condition Codes

Not affected.

Operation

Mn ← 0

Description

Clears the bit n (n = 0 through 7) of M. The other bits are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	BCLR	0,M	11	M		2	5
DIRECT	BCLR	1,M	13	M		2	5
DIRECT	BCLR	2,M	15	M		2	5
DIRECT	BCLR	3,M	17	M		2	5
DIRECT	BCLR	4,M	19	M		2	5
DIRECT	BCLR	5,M	1B	M		2	5
DIRECT	BCLR	6,M	1D	M		2	5
DIRECT	BCLR	7,M	1F	M		2	5

Example

```
0157 B6 03      LDA     CNTRL     ** MAKE CONTROL CODE **
0159 A4 F0      AND     #$F0      *
015B BA FF      ORA     WORK      *
015D B7 03      STA     CONTRL    *
015F 11 03      BCLR    0,CNTRL   CLEAR BIT 0,6,7 ABSOLUTELY
0161 1D 03      BCLR    6,CNTRL
0163 1F 03      BCLR    7,CNTRL
```

BCS (Branch if Carry Set)

Format

BCS dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (C)=1

Description

Tests the C-bit state and causes a branch if C is "1".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BCS	Rel	25	Rel		2	3

Example

```
0165 B6 10          LDA     VAL1
0167 CB 0600        ADD     EXVAL6
016A 25 13          BCS     ABNML     KETA AGARI
                    *
016C C7 0600        STA     EXVAL6    KETA AGARI NASHI
```

BEQ (Branch of EQual)

Format

BEQ dd

Condition Codes

Not affected.

Operation

PC←(PC)+0002+Rel if (Z)=1

Description

Tests the Z-bit state and causes a branch if Z is "1".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BEQ	Rel	27	Rel		2	3

Example

```
016F B6 FF        LDA     WORK
0171 27 18        BEQ     AAAA      WORK = 0
0173 B1 50        CMP     RESULT
0175 27 28        BEQ     BBBB      WORK = RESULT
```

BHCC (Branch if Half Carry Clear)

Format

BHCC dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (H)=0

Description

Tests the H-bit state and causes a branch if H is "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BHCC	Rel	28	Rel		2	3

Example

```
01E7 A1 09                CMP    #$9
01E9 23 02                BLS    DAALOW    $99 --->  INPUT
01EB AE 60      DAAH6     LDX    #$60      HIGH NYBLE NEEDS CORRECTION
                *
01ED 28 15      DAALOW    BHCC   DAAL9
01EF 9F                   TXA
```

Conditional Branch

BHCS

BHCS (Branch if Half Carry Set)

Format

BHCS dd

Condition Codes

Not affected.

Operation

PC ← (PC) + 0002 + Rel if (H) = 1

Description

Tests the H-bit state and causes a branch if H is "1".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BHCS	Rel	29	Rel		2	3

Example

```
01F0 A1 09            CMP     #$9
01F2 23 02            BLS     DAALW1    $99 ---   INPUT
01F4 AE 60      DAAH7 LDX     #$60      HIGH NYBLE NEEDS CORRECTION
                *
01F6 29 16      DAALW1 BHCS   DAAL6
01F8 A4 0F            AND     #$F
```

BHI (Branch if HIgher)

Format

BHI dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (C ∨ Z)=0
i.e. if (ACCA) > (M)
(unsigned binary numbers)

Description

Causes a branch if both C-bit and Z-bit are "0".
When the BHI instruction is executed immediately after either CMP or SUB instruction has been executed, a branch occurs if the minuend represented by the unsigned binary number (i.e. ACCA) is greater than the subtrahend represented by unsigned binary number (i.e. M).

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BHI	Rel	22	Rel		2	3

Example

```
0177 B6 10          LDA     VAL1
0179 B1 20          CMP     VAL2
017B 22 16          BHI     ZIP25     VAL1 > VAL2 (IGNORE SIGN BIT)
                    *
017D B7 FF          STA     WORK      VAL1 --> WORK (LOWER OR SAME)
```

BHS (Branch if Higher or Same)

Format

BHS dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (C)=0

Description

When the BHS instruction is executed after comparing or subtracting unsigned binary, if causes a branch if the register contents are greater than or equal to the M contents.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BHS	Rel	24	Rel		2	3

Example

```
0100 B6 10              LDA     VAL1
0102 B1 20              CMP     VAL2
0104 24 16              BHS     ZIP26       VAL1 >= VAL2  IGNORE SIGN BIT
                *
0106 B7 FF              STA     WORK        VAL1 ---> WORK (LOWER)
```

BIH (Branch if Interrupt line is High)

Format

BIH dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel
if $\overline{INT}$=1 (high)

Description

Tests the external interrupt pin ($\overline{INT}$) state and causes a branch if it is high.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BIH	Rel	2F	Rel		2	3

Example

```
01C6 2F 04            BIH     INTH0     INT LINE CHECK
01C8 A6 28     INTL1  LDA     #$28      OUTPUT DATA = $28
01CA 20 02            BRA     NEXT2
01CC A6 FF     INTH0  LDA     #$FF      OUTPUT DATA = $FF
01CE C7 06E0   NEXT2  STA     PIA       OUTPUT
```

BIL (Branch if Interrupt line is Low)

Format

BIL dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if $\overline{INT}$=0 (low)

Description

Tests the external interrupt pin ($\overline{INT}$) state and causes a branch if it is low.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BIL	Rel	2E	Rel		2	3

Example

```
01D1 2E 04            BIL     INTL2   INT LINE CHECK
01D3 A6 45    INTH3   LDA     #$45    OUTPUT DATA = $45
01D5 20 02            BRA     NEXT4
01D7 A6 00    INTL2   LDA     #$0     OUTPUT DATA = $00
01D9 C7 06E0  NEXT4   STA     PIA     OUTPUT
```

BIT (BIt Test)

Format

BIT P

Operation

(ACCA)/\(M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result of the AND operand is 1; otherwise cleared.
Z: Set if all the bits of the result of the AND operand are 0; otherwise cleared.
C: Not affected.

Description

Performs the logical AND operation between the ACCA contents and M contents and modifies the condition codes respectively. The ACCA contents and M contents are not affected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	BIT	#Imm	A5	Imm		2	2
DIRECT	BIT	M	B5	M		2	3
EXTENDED	BIT	M	C5	MH	ML	3	4
INDEXED 0 BYTE OFFSET	BIT	0,X	F5			1	3
INDEXED 1 BYTE OFFSET	BIT	Disp,X	E5	D		2	4
INDEXED 2 BYTE OFFSET	BIT	Disp,X	D5	DH	DL	3	5

Example

```
0400 B6 10      EVBIT   LDA     VAL1
0402 A5 F8              BIT     #$F8
0404 27 19              BEQ     OK      0 <= BIT ASSIGN (VAL1) <= 7
                *
0406 A6 E3      NG      LDA     #227    SET ERROR NUMBER
0408 CC 0432            JMP     ERROR
```

BLO (Branch if LOwer)

Format

BLO dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (C)=1

Description

Causes a branch when executing BLO after compare if register contents are less than M contents.
BLO is equivalent to BCS.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BLO	Rel	25	Rel		2	3

Example

```
040B B6 10              LDA     VAL1
040D B1 20              CMP     VAL2
040F 25 16              BLO     ZIP27     VAL1 < VAL2 (IGNORE SIGN BIT)
                *
0411 B7 FF              STA     WORK      VAL1 --> WORK  HIGHER OR SAME
```

BLS (Branch if Lower or Same)

Format

BLS dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (C V Z)=1

i.e. if (ACCA) ≤ (M)

Description

Causes a branch if either C-bit or Z-bit is "1". When the BHI instruction is executed immediately after either CMP or SUB instruction has been executed, a branch occurs if the minuend represented by the unsigned binary number (i.e. ACCA) is less than or equal to the subtrahend represented by unsigned binary number (i.e. M).

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BLS	Rel	23	Rel		2	3

Example

```
0413 B6 10              LDA     VAL1
0415 B1 20              CMP     VAL2
0417 23 16              BLS     ZIP28       VAL1 <= VAL2  IGNORE SIGN BIT
                *
0419 B7 FF              STA     WORK        VAL1 ---> WORK (HIGHER)
```

BMC (Branch if interrupt Mask is Clear)

Format

BMC dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (I)=0

Description

Tests the I-bit state and causes a branch if I is "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BMC	Rel	2C	Rel		2	3

Example

```
0217 2C 07          BMC     MSKOFF      INTMSK OFF ?
0219 2E 05          BIL     MSKOFF      INT LINE LOW ?
021B C6 06E0        LDA     PIA         READ DATA
021E B7 FF          STA     WORK
0220 81     MSKOFF  RTS
```

BMI (Branch if MInus)

Format

BMI dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (N)=1

Description

Tests the N-bit state and causes a branch if N is "1".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BMI	Rel	2B	Rel		2	3

Example

```
0425 B6 10              LDA     VAL1
0427 2B 16              BMI     ZIP29       VAL1 < 0
                *
0429 B7 FF              STA     WORK        VAL1 ---> WORK (PLUS)
```

BMS (Branch if interrupt Mask is Set)

Format

BMS dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (I)=1

Description

Tests the I-bit state and causes a branch if I is "1".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BMS	Rel	2D	Rel		2	3

Example

```
0221 2D 01          BMS     MSKON1    INTMSK ON ?
0223 81     MSKOF1  RTS               NO
0224 2D FD  MSKON1  BIL     MSKOF1    INT LINE LOW ?
0226 C6 06E0        LDA     PIA       DATA
0229 B7 FF          STA     WORK
022B 81             RTS
```

BNE (Branch if Not Equal)

Format

BNE dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (Z)=0

Description

Tests the Z-bit state and causes a branch if Z is "0".
Following a compare or subtract instruction, BNE will cause a branch if the arguments were different.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BNE	Rel	26	Rel		2	3

Example

```
020D B6 FF          LDA     WORK
020F 26 18          BNE     CCCC    WORK NOT = 0
0211 B1 50          CMP     RESULT
0213 26 1D          BNE     DDDD    WORK NOT = RESULT
```

BPL (Branch if PLus)

Format

BPL dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel if (N)=0

Description

Tests the N-bit state and causes a branch if N is "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BPL	Rel	2A	Rel		2	3

Example

```
0215 B6 10              LDA     VAL1
0217 2A 16              BPL     ZIP31     VAL >= 0
                  *
0219 B7 FF              STA     WORK      VAL1 ---> WORK (MINUS)
```

BRA (BRanch Always)

Format

BRA dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002+Rel

Description

Causes an unconditional branch to the address gain from the operation shown above.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BRA	Rel	20	Rel		2	3

Example

```
0100 C6 0500            LDA     EXVAL5
0103 B7 50              STA     RESULT
0105 20 1E              BRA     END01     BRANCH TO END01 ALWAYS
                *
     0107       CHECK8  EQU     *
```

Conditional Branch

BRCLR

BRCLR (BRanch if bit n is CLeaR)

Format

BRCLR n, DR, dd

Condition Codes

H: Not affected.
I: Not affected.
N: Not affected.
Z: Not affected.
C: Set if (Mn)=1; otherwise reset.

Operation

PC ← (PC)+0003+Rel if (Mn)=0

Description

Tests the bit n (n = 0 through 7) of M and causes a branch if the contents of Mn are "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BRCLR	0,M,Rel	01	M	Rel	3	5
RELATIVE	BRCLR	1,M,Rel	03	M	Rel	3	5
RELATIVE	BRCLR	2,M,Rel	05	M	Rel	3	5
RELATIVE	BRCLR	3,M,Rel	07	M	Rel	3	5
RELATIVE	BRCLR	4,M,Rel	09	M	Rel	3	5
RELATIVE	BRCLR	5,M,Rel	0B	M	Rel	3	5
RELATIVE	BRCLR	6,M,Rel	0D	M	Rel	3	5
RELATIVE	BRCLR	7,M,Rel	0F	M	Rel	3	5

Example

```
0107 B6 03              LDA     CNTRL       ** SET CONTROL CODE **
0109 A4 0F              AND     #$0F
010B BA FF              ORA     WORK
010D B7 03              STA     CNTRL
                  *             ** ACTION **
010F 09 03 17           BRCLR   4,CNTRL,ENGINE
0112 0F 03 28           BRCLR   7,CNTRL,GASCHK
```

BRN (BRanch Never)

Format

BRN dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002

Description

It does not cause a branch. BRN, which requires 2-byte and 3 cycle long, is the inverse of BRA. This instruction is sometimes available for debugging program.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BRN	Rel	21	Rel		2	3

Example

```
0115 EF 04          STX     4.X
                *
0117 21 FE          BRN     *       ** DELAY **
0119 21 FE          BRN     *
011B 21 FE          BRN     *
011D 21 FE          BRN     *
```

Conditional Branch

BRSET

BRSET (BRanch if bit n is SET)

Format

BRSET n, DR, dd

Operation

PC ← (PC)+0003+Rel if (Mn)=1

Condition Codes

H: Not affected.
I: Not affected.
N: Not affected.
Z: Not affected.
C: Set if (Mn)=1; otherwise reset.

Description

Tests the bit n (n = 0 through 7) of M, and causes a branch if the Mn contents are "1".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BRSET	0,M,Rel	00	M	Rel	3	5
RELATIVE	BRSET	1,M,Rel	02	M	Rel	3	5
RELATIVE	BRSET	2,M,Rel	04	M	Rel	3	5
RELATIVE	BRSET	3,M,Rel	06	M	Rel	3	5
RELATIVE	BRSET	4,M,Rel	08	M	Rel	3	5
RELATIVE	BRSET	5,M,Rel	0A	M	Rel	3	5
RELATIVE	BRSET	6,M,Rel	0C	M	Rel	3	5
RELATIVE	BRSET	7,M,Rel	0E	M	Rel	3	5

Example

```
011F B6 03          LDA     CNTRL     ** SET CONTROL CODE **
0121 A4 8E          AND     #$8E
0123 BA FF          ORA     WORK
0125 B7 03          STA     CNTRL
                 *                    ** ACTION **
0127 00 03 17 PROC1 BRSET   0,CNTRL,OIL
012A 0E 03 28 PROC2 BRSET   7,CNTRL,GAS
```

BSET (Bit SET bit n)

Format

BSET n,DR

Condition Codes

Not affected.

Operation

Mn ← 1

Description

Sets the bit n (n = 0 through 7) of M. All other bits are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	BSET	0,M	10	M		2	5
DIRECT	BSET	1,M	12	M		2	5
DIRECT	BSET	2,M	14	M		2	5
DIRECT	BSET	3,M	16	M		2	5
DIRECT	BSET	4,M	18	M		2	5
DIRECT	BSET	5,M	1A	M		2	5
DIRECT	BSET	6,M	1C	M		2	5
DIRECT	BSET	7,M	1E	M		2	5

Example

```
0100 B6 50          LDA     RESULT
0102 2A 04          BPL     PLUS
                *                       (MINUS)
0104 14 03          BSET    2,CNTRL
0106 16 FF          BSET    3,WORK
     0108   PLUS    EQU     *
0108 B6 20          LDA     VAL2
```

BSR (Branch to SubRoutine)

Format

BSR dd

Condition Codes

Not affected.

Operation

PC ← (PC)+0002
Msp ← (PCL), SP ← (SP)-0001
Msp ← (PCH), SP ← (SP)-0001
PC ← (PC)+Rel

Description

The program counter is increased by "2". The less significant bites (8-bits) of the program counter contents are pushed onto the stack. Then, stackpointer is decreased by "1". The more significant bits (6-bits) of the program counter contents are pushed onto the stack, then stackpointer is decreased by "1". Then a branch occurs to the address specified by the program counter.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BSR	Rel	AD	Rel		2	5

Example

```
010A A6 3B          LDA     #$3B     ACCA = INTERFACE (0011 1011)
010C AD 18          BSR     HAND
                *
010E A6 1E          LDA     #$1E     ACCA = INTERFACE (0001 1110)
0110 AD 28          BSR     FING
```

CLC (CLear Carry)

Format

CLC

Operation

C ← 0

Condition Codes

H: Not affected.
I: Not affected.
N: Not affected.
Z: Not affected.
C: Reset.

Description

Resets the carry bit C in the condition code register.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	CLC		98			1	1

Example

```
0100 26 F8          BNE     CHK83
0102 B7 50          STA     RESULT
0104 98             CLC                 RETURN CODE SET 'OK'
0105 81             RTS
                *
                *
```

CLI (CLear Interrupt mask)

Format

CLI

Condition Codes

H: Not affected.
I: Reset.
N: Not affected.
Z: Not affected.
C: Not affected.

Operation

I ← 0

Description

Resets the interrupt mask bit in the processor condition code register. This enables the microprocessor to service interrupts that occurred through an interrupt request from peripheral equipment.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	CLI		9A			1	2

Example

```
01FA 9B          SEI          INTERRUPT DISABLE
01FB 9C          RSP          RESET STACK POINTER
01FC CD 06F0     JSR   SYSINZ   SYSTEM INITIALIZE
01FF 9A          CLI          INTERRUPT ENABLE
```

CLR (CLeaR)

Format

```
CLR Q
CLR A
CLR X
```

Condition Codes

H: Not affected.
I: Not affected.
N: Reset.
Z: Set.
C: Not affected.

Operation

```
IX   ← 0
  or
ACCA ← 0
  or
M    ← 0
```

Description

The contents of IX, ACCA or M are replaced with "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	CLR	A	4F			1	2
INDEX REG.	CLR	X	5F			1	2
DIRECT	CLR	M	3F	M		2	5
INDEXED 0 BYTE OFFSET	CLR	0,X	7F			1	5
INDEXED 1 BYTE OFFSET	CLR	Disp,X	6F	D		2	6

Example

```
                *                          ** INITIALIZE **
0106 3F 07              CLR        PNTR
0108 3F 08              CLR        PNTR+1
010A 7F                 CLR        0,X
010B 4F                 CLR        A
```

CMP (CoMPare)

Format

CMP P

Operation

(ACCA)-(M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result of the subtraction is "1"; otherwise reset.
Z: Set if the result of the subtraction is 0; otherwise reset.
C: Set if the absolute value of memory is greater than that of the accumulator; otherwise reset.

Description

Compares the ACCA contents with M contents, and affects the condition codes that can be referred to by conditional branch instructions. Both operands are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	CMP	#Imm	A1	Imm		2	2
DIRECT	CMP	M	B1	M		2	3
EXTENDED	CMP	M	C1	MH	ML	3	4
INDEXED 0 BYTE OFFSET	CMP	0,X	F1			1	3
INDEXED 1 BYTE OFFSET	CMP	Disp,X	E1	D		2	4
INDEXED 2 BYTE OFFSET	CMP	Disp,X	D1	DH	DL	3	5

Example

```
0110 E6 07          *       LDA     PNTR,X
0112 A1 41                  CMP     #'A
0114 27 1A                  BEQ     SECTA       ACCA = 'A'
0116 A1 42                  CMP     #'B
0118 27 2A                  BEQ     SECTB       ACCA = 'B'
011A 20 F0                  BRA     INPUT
```

Logical Operation
COM

COM (COMplement)

Format

```
COM Q
COM A
COM X
```

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0"; otherwise reset.
C: Set.

Operation

$IX \leftarrow (\overline{IX}) = \$FF-(IX)$
or
$ACCA \leftarrow (\overline{ACCA}) = \$FF-(ACCA)$
or
$M \leftarrow (\overline{M}) = \$FF-(M)$

Description

Replaces the contents of ACCA, IX or M with its 1's complement.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	COM	A	43			1	2
INDEX REG.	COM	X	53			1	2
DIRECT	COM	M	33	M		2	5
INDEXED 0 BYTE OFFSET	COM	0,X	73			1	5
INDEXED 1 BYTE OFFSET	COM	Disp,X	63	D		2	6

Example

```
                *
     011C       SUBIN   EQU     *
011C 5C                 INC     X
011D E6 07              LDA     PNTR,X
011F 43                 COM     A           MODIFY DATA (REVERSE)
0120 81                 RTS
                *
```

CPX (ComPare indeX register)

Format

CPX P

Operation

(IX)-(M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Set if the absolute value of the contents of the memory is greater than that of the contents of IX; otherwise reset.

Description

Compares the IX contents with M contents. The condition code can be collated by means of the next conditional branch instruction. Both operands are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	CPX	#Imm	A3	Imm		2	2
DIRECT	CPX	M	B3	M		2	3
EXTENDED	CPX	M	C3	MH	ML	3	4
INDEXED 0 BYTE OFFSET	CPX	0,X	F3			1	3
INDEXED 1 BYTE OFFSET	CPX	Disp,X	E3	D		2	4
INDEXED 2 BYTE OFFSET	CPX	Disp,X	D3	DH	DL	3	5

Example

```
0121 A6 CC      LDA     #$CC        ACCA = INTERFACE TO CR OR LF
0123 BE 07      LDX     PNTR
0125 A3 0D      CPX     #$0D
0127 27 18      BEQ     CR          CARRIAGE RETURN
0129 A3 0A      CPX     #$0A
012B 27 28      BEQ     LF          LINE FEED
```

DAA (Decimal Adjust Accumulator)

Format

DAA

Operation

Convert binary Add result into Binary Coded Decimal (BCD).

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of result is "1"; otherwise reset.
Z: Set if the result is "0", otherwise reset.
C: Set or reset with the rule under which DAA and its previous ABA, ADD and ADC is converted into BCD.

Description

Bit Condition of bit C before DAA execution	Lower 4 bits (bit 4 ∿ 7)	Early H bit (Half Carry)	Lower 4 bits (bit 0 ∿ 3)	Value added to ACCA by DAA execution (hexadecimal)	Bit condition of bit C before DAA execution
0	0-9	0	0-9	00	0
0	0-8	0	A-F	06	0
0	0-9	1	0-3	06	0
0	A-F	0	0-9	60	1
0	9-F	0	A-F	66	1
0	A-F	1	0-3	66	1
0	0-2	0	0-9	60	1
0	0-2	0	A-F	66	1
0	0-3	1	0-3	66	1

Add 00, 06, 60 66 (heradecimal) to ACCA according to the table shown left.

If the BCD Add result by ADD and ADC instruction is in ACCA, bit C or bit H, DAA executes above function.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	DAA		8D			1	2

Example

Arithmetic Operation

DEC

DEC (DECrement)

Format

```
DEC Q
DEC A
DEC X
```

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Not affected.

Operation

IX ← (IX)-01
or
ACCA ← (ACCA)-01
or
M ← (M)-01

Description

Subtracts "1" from the contents of ACCA, IX or M.
N and Z bits are set and reset according to the result of this operation.
The C-bit is not affected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	DEC	A	4A			1	2
INDEX REG.	DEC	X	5A			1	2
DIRECT	DEC	M	3A	M		2	5
INDEXED 0 BYTE OFFSET	DEC	0,X	7A			1	5
INDEXED 1 BYTE OFFSET	DEC	Disp,X	6A	D		2	6

Example

```
                    *
012D 4A        LOOP23 DEC          ** MOVE **
012E 2B 07            BMI          NEXT
0130 FE               LDX          0,X       *
0131 DF 0100          STX          $100,X    *
0134 5C               INC          X         *
0135 20 F6            BRA          LOOP23
                    *
     0137      NEXT   EQU          *
```

EOR (Exclusive OR)

Format

EOR P

Operation

ACCA ← (ACCA) ⊕ (M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Not affected.

Description

Performs the logical EXCLUSIVE OR between the ACCA contents and M contents and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	EOR	#Imm	A8	Imm		2	2
DIRECT	EOR	M	B8	M		2	3
EXTENDED	EOR	M	C8	MH	ML	3	4
INDEXED 0 BYTE OFFSET	EOR	0,X	F8			1	3
INDEXED 1 BYTE OFFSET	EOR	Disp,X	E8	D		2	4
INDEXED 2 BYTE OFFSET	EOR	Disp,X	D8	DH	DL	3	5

Example

```
                *                          ** ARRANGE CONTROL CODE **
0137 B6 03              LDA       CNTRL      XXXX XXXX
0139 A8 99              EOR       #$99       1001 1001
013B B7 03              STA       CNTRL
013D 20 14              BRA       ACT01
```

INC (INCrement)

Format

```
INC Q
INC A
INC X
```

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Not affected.

Operation

IX ← (IX)+01
or
ACCA ← (ACCA)+01
or
M ← (M)+01

Description

Adds "1" to the contents of ACCA, IX or M.
N and Z bits are set or reset according to the result of this operation.
The C bit is not affected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	INC	A	4C			1	2
INDEX REG.	INC	X	5C			1	2
DIRECT	INC	M	3C	M		2	5
INDEXED 0 BYTE OFFSET	INC	0,X	7C			1	5
INDEXED 1 BYTE OFFSET	INC	Disp,X	6C	D		2	6

Example

```
0100 4C          LOOP3   INC     A           *
0101 A1 64               CMP     #100
0103 22 1B               BHI     EXIT        CHECK COUNTER (100 TIMES)
0105 FE                  LDX     0,X
0106 DF 0300             STX     $300,X      MOVE
0109 5C                  INC     X           *
010A 20 F4               BRA     LOOP3
```

JMP (JuMP)

Format

JMP P

Condition Codes

Not affected.

Operation

PC ← EA

Description

A jump occurs to the instruction stored at the effective address. The effective address is obtained according to the rules for EXTended, DIRect or INDexed addressing.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	JMP	M	BC	M		2	2
EXTENDED	JMP	M	CC	MH	ML	3	3
INDEXED 0 BYTE OFFSET	JMP	0,X	FC			1	2
INDEXED 1 BYTE OFFSET	JMP	Disp,X	EC	D		2	3
INDEXED 2 BYTE OFFSET	JMP	Disp,X	DC	DH	DL	3	4

Example

```
010C B6 10          LDA     VAL1
010E C7 0500        STA     EXVAL5
0111 B6 20          LDA     VAL2
0113 C7 0600        STA     EXVAL6
0116 CC 0333        JMP     END90       GO TO END-ROUTINE
```

JSR (Jump to SubRoutine)

Format

JSR P

Condition Codes

Not affected.

Operation

Note)
PC ← (PC)+n
Msp ← (PCL), SP ← (SP)-0001
Msp ← (PCH), SP ← (SP)-0001
PC ← EA

Description

The program counter is increased by "n"[Note)] in the addressing mode, then pushed onto the 2-byte stack. And the stack point is updated. A jump occurs to the specified address. The effective address is obtained according to the rules for EXTended, DIRect or INDexed addressing.

Note) n is equal to 1, 2 or 3, according to the number of bytes in the instruction code. Refer to the addressing code and the number of MPU cycles shown below.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	JSR	M	BD	M		2	5
EXTENDED	JSR	M	CD	MH	ML	3	6
INDEXED 0 BYTE OFFSET	JSR	0,X	FD			1	5
INDEXED 1 BYTE OFFSET	JSR	Disp,X	ED	D		2	5
INDEXED 2 BYTE OFFSET	JSR	Disp,X	DD	DH	DL	3	6

Example

```
                *                    ** MAIN ROUTINE **
        0119    START   EQU     *
0119 CD 0407            JSR     INTRTN      INITIALIZE
011C CD 04E5            JSR     KBRTN       INPUT FROM KEY-BOARD
011F CD 03AD            JSR     ANARTN      ANALYSE
0122 CD 053C            JSR     PRCRTN      PROCESS
0125 CC 06CB            JMP     ENDRTN      END
```

LDA (LoaD Accumulator)

Format

LDA P

Operation

ACCA ← (M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Not affected.

Description

Loads the contents of memory into the accumulator.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	LDA	#Imm	A6	Imm		2	2
DIRECT	LDA	M	B6	M		2	3
EXTENDED	LDA	M	C6	MH	ML	3	4
INDEXED 0 BYTE OFFSET	LDA	0,X	F6			1	3
INDEXED 1 BYTE OFFSET	LDA	Disp,X	E6	D		2	4
INDEXED 2 BYTE OFFSET	LDA	Disp,X	D6	DH	DL	3	5

Example

```
0128 B6 10      LDA     VAL1
012A B7 FF      STA     WORK
012C F6         LDA     0,X
012D B7 50      STA     RESULT
012F A6 FF      LDA     #$FF
```

LDX (LoaD indeX register)

Format

LDX P

Operation

IX ← (M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of IX is "1"; otherwise reset.
Z: Set if all the bits of IX of the result are "0"; otherwise reset.
C: Not affected.

Description

Loads the M contents into IX. The condition code is set according to data.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	LDX	#Imm	AE	Imm		2	2
DIRECT	LDX	M	BE	M		2	3
EXTENDED	LDX	M	CE	MH	ML	3	4
INDEXED 0 BYTE OFFSET	LDX	0,X	FE			1	3
INDEXED 1 BYTE OFFSET	LDX	Disp,X	EE	D		2	4
INDEXED 2 BYTE OFFSET	LDX	Disp,X	DE	DH	DL	3	5

Example

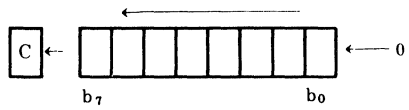
```
0131 BE 10      LDX     VAL1
0133 BF FF      STX     WORK
0135 FE         LDX     0,X
0136 BF 50      STX     RESULT
0138 AE FF      LDX     #$FF
```

LSL (Logical Shift Left)

Format

LSL Q
LSL A
LSL X

Operation



Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Set if the least significant bit of ACCA, IX or memory is "1" before execution of an instruction; otherwise reset.

Description

Shifts the contents of ACCA, IX or M 1-bit to the left. The bit 0 is loaded with "0". The carry bit C is loaded with the most significant bit of ACCA, IX or M.

Addressing Mode and Number of Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	LSL	A	48			1	2
INDEX REG.	LSL	X	58			1	2
DIRECT	LSL	M	38	M		2	5
INDEXED 0 BYTE OFFSET	LSL	0,X	78			1	5
INDEXED 1 BYTE OFFSET	LSL	Disp,X	68	D		2	6

Example

```
013A 38 FF          LSL     WORK    ** MULTIPLY  X 8 **
013C 38 FF          LSL     WORK
013E 38 FF          LSL     WORK
```

LSR (Logical Shift Right)

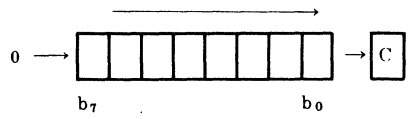
Format

LSR Q
LSR A
LSR X

Condition Codes

H: Not affected.
I: Not affected.
N: Reset.
Z: Set if the result is "0"; otherwise reset.
C: Set if the least significant bit of ACCA, IX, or memory is "1" before execution of an instruction; otherwise reset.

Operation



Description

Shifts the contents of ACCA, IX or M 1-bit to the right. The bit 7 is loaded with "0". The carry bit C is loaded with the least significant bit of ACCA, IX or M.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	LSR	A	44			1	2
INDEX REG.	LSR	X	54			1	2
DIRECT	LSR	M	34	M		2	5
INDEXED 0 BYTE OFFSET	LSR	0,X	74			1	5
INDEXED 1 BYTE OFFSET	LSR	Disp,X	64	D		2	6

Example

```
0140 34 FF          LSR     WORK    ** DIVIDE / 16 **
0142 34 FF          LSR     WORK
0144 34 FF          LSR     WORK
0146 34 FF          LSR     WORK
```

NEG (NEGate)

Format

```
NEG Q
NEG A
NEG X
```

Operation

IX ← (IX)=00-(IX)
or
ACCA ← (ACCA)=00-(ACCA)
or
M ← (M)=00-(M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Set if a borrow occurs; otherwise reset. Set if the contents of ACCA, IX or Memory are other than "0".

Description

Replaces the contents of ACCA, IX or M with its two's complement, and stores ACCA or IX contents into M contents. Note that $80 (-128) is unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	NEG	A	40			1	2
INDEX REG.	NEG	X	50			1	2
DIRECT	NEG	M	30	M		2	5
INDEXED 0 BYTE OFFSET	NEG	0,X	70			1	5
INDEXED 1 BYTE OFFSET	NEG	Disp,X	60	D		2	6

Example

```
                *                CHECK RANGE (RELATIVE ADDRESSING)
0148 A1 81              CMP      #129          CHECK RANGE
014A 24 44              BCC      BERROR        *     BRANCH ERROR
014C 40                 NEG      A             OFFSET
014D 20 14              BRA      SET
                *
```

Unconditional Branch

NOP

NOP (No OPeration)

Format

NOP

Condition Codes

Not affected.

Operation

Description

This is a single-byte instruction which only causes the program counter to be increased. Other registers are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	NOP		9D			1	1

Example

```
014F 9D          NOP          ** DELAY **
0150 9D          NOP
0151 9D          NOP
0152 9D          NOP
0153 9D          NOP
0154 9D          NOP
```

Logical Operation
ORA

ORA (inclusive OR)

Format

ORA

Operation

ACCA ← (ACCA)v(M)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if all the bits of the result are "0"; otherwise reset.
C: Not affected.

Description

Performs logical OR between ACCA contents and M contents, and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	ORA	#Imm	AA	Imm		2	2
DIRECT	ORA	M	BA	M		2	3
EXTENDED	ORA	M	CA	MH	ML	3	4
INDEXED 0 BYTE OFFSET	ORA	0,X	FA			1	3
INDEXED 1 BYTE OFFSET	ORA	Disp,X	EA	D		2	4
INDEXED 2 BYTE OFFSET	ORA	Disp,X	DA	DH	DL	3	5

Example

```
0155 25 06              BCS     SKIP
                *
     0157       ADCN    EQU     *       ** ADDITION CONTROL BIT **
0157 A6 14              LDA     #$14    0001 0100
0159 BA 03              ORA     CNTRL
015B B7 03              STA     CNTRL
     015D       SKIP    EQU     *
```

ROL (ROtate Left)

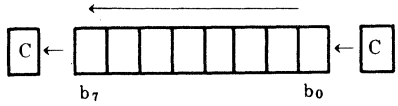
Format

ROL Q
ROL A
ROL X

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set of all the bits of the result are "0"; otherwise reset.
C: Set if the ACCA, IX, or the most significant bit of the memory is "1", before execution of an instruction; otherwise reset.

Operation



Description

Shifts the contents of ACCA, IX or M 1-bit to the left. The bit 0 is loaded with the carry bit C, while the carry bit C is loaded with the most significant bit of ACCA, IX or M.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	ROL	A	49			1	2
INDEX REG.	ROL	X	59			1	2
DIRECT	ROL	M	39	M		2	5
INDEXED 0 BYTE OFFSET	ROL	0,X	79			1	5
INDEXED 1 BYTE OFFSET	ROL	Disp,X	69	D		2	6

Example

```
                    *           ** REPEAT ACTION FOLLOWING CNTRL **
015D 98             REPEAT CLC
     015E           REPEAT EQU  *
015E 39 03                 ROL  CNTRL
0160 25 05                 BCS  ACTION   ACTION & REPEAT OR ESCAPE
0162 9D                    NOP          *
0163 9D                    NOP          ** DELAY **
0164 9D                    NOP          *
0165 20 F7                 BRA  REPEAT
```

ROR (ROtate Right)

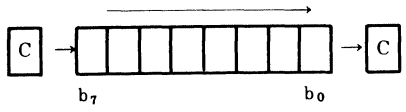
Format

```
ROR Q
ROR A
ROR X
```

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if all the bits of the result are "0"; otherwise reset.
C: Set if the ACCA, IX, or the least significant bit of Memory is "1"; otherwise reset.

Operation



Description

Shifts the contents of ACCA, IX or M one bit to the right. The bit 7 is loaded with the carry bit C, while the bit 0 is loaded with the carry bit C.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	ROR	A	46			1	2
INDEX REG.	ROR	X	56			1	2
DIRECT	ROR	M	36	M		2	5
INDEXED 0 BYTE OFFSET	ROR	0,X	76			1	5
INDEXED 1 BYTE OFFSET	ROR	Disp,X	66	D		2	6

Example

```
                    *                ** REPEAT ACTION FOLLOWING CNTRL **
0167  98                REPT1  CLC
      0168          REPT1  EQU    *
0168  36 03               ROR    CNTRL
016A  25 05               BCS    ACTN1    ACTION & REPEAT OR ESCAPE
016C  9D                  NOP              *
016D  9D                  NOP              ** DELAY  **
016E  9D                  NOP              *
016F  20 F7               BRA    REPT1
```

Stack Pointer Operation
RSP

RSP (Reset Stack Pointer)

Format

RSP

Condition Codes

Not affected.

Operation

SP ← $FF

Description

Resets the stack pointer to the top ($FF) of the stack.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	RSP		9C			1	2

Example

```
0200 9B          SEI              INTERRUPT DISABLE
0201 9C          RSP              RESET STACK POINTER
0202 CD 06F0     JSR     SYSINZ   SYSTEM INITIALIZE
0205 9A          CLI              INTERRUPT ENABLE
```

RTI (ReTurn from Interrupt)

Format

RTI

Condition Codes

Recovers the state saved onto the stack.

Operation

SP ← (SP)+0001, CCR ← (SP)
SP ← (SP)+0001, ACCA ← (SP)
SP ← (SP)+0001, IX ← (SP)
SP ← (SP)+0001, PCH ← (SP)
SP ← (SP)+0001, PCL ← (SP)

Description

Sets the stack contents indicated by SP to CCR, ACCA, IX, PCH, or PCL increasing SP by "1". Note that I = 0 when the interrupt mask bit of CCR saved onto the stack is "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	RTI		80			1	8

Example

```
020C CD 0345     JSR     KEYSCN      KEY INPUT
020F B7 10       STA     INKEY       STORE KEY CODE
0211 CD 0400     JSR     EXSWIN      INPUT EXTERNAL SW
0214 B7 11       STA     INSW        STORE SW CONDITION
0216 80          RTI                 RETURN TO INTERRUPT
```

RTS (ReTurn from Subroutine)

Format

RTS

Condition Codes

Not affected.

Operation

SP ← (SP)+0001, PCH ← (SP)
SP ← (SP)+0001, PCL ← (SP)

Description

Increases SP by "1" and sets the address contents indicated by SP to more significant bits (6-bits) of PC. Increases SP with "1" again, and sets the address contents specified by SP to less significant bits (8-bits) of PC.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	RTS		81			1	5

Example

```
0171 B7 FF          STA     WORK
0173 C6 0500        LDA     EXVALS
0176 B7 50          STA     RESULT
0178 98             CLC                 RETURN CODE SET : OK
0179 81             RTS                 *
                *
```

SBC (SuBtract with Carry)

Format

SBC P

Operation

ACCA ← (ACCA)-(M)-(C)

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the result is "0"; otherwise reset.
C: Set if the absolute value of the contents of memory plus the carry bit C is greater than the absolute value of the contents of ACCA; otherwise reset.

Description

Subtracts the contents of M and the carry bit C from that of ACCA, and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	SBC	#Imm	A2	Imm		2	2
DIRECT	SBC	M	B2	M		2	3
EXTENDED	SBC	M	C2	MH	ML	3	4
INDEXED 0 BYTE OFFSET	SBC	0,X	F2			1	3
INDEXED 1 BYTE OFFSET	SBC	Disp,X	E2	D		2	4
INDEXED 2 BYTE OFFSET	SBC	Disp,X	D2	DH	DL	3	5

Example

```
*                    (VAL1, VAL1+1)-(EXVAL5, EXVAL5+1)
*                                  =(EXVAL5, EXVAL5+1)
*
017A B6 11          LDA     VAL1+1    *
017C C0 0501        SUB     EXVAL5+1  *
017F C7 0501        STA     EXVAL5+1  *
0182 B6 10          LDA     VAL1      *
0184 C2 0500        SBC     EXVAL5    *
0187 C7 0500        STA     EXVAL5    *
```

SEC (SEt Carry)

Format

SEC

Condition Codes

H: Not affected.
I: Not affected.
N: Not affected.
Z: Not affected.
C: Set.

Operation

C bit ← 1

Description

Sets the carry bit C in the condition code register.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	SEC		99			1	1

Example

```
018A 27 F8          BEQ     CHK84
018C B7 50          STA     RESULT
018E 99             SEC                 RETURN CODE SET : NG
018F 81             RTS
                *
                *
```

SEI (SEt Interrupt mask)

Format

SEI

Condition Codes

H: Not affected.
I: Set.
N: Not affected.
Z: Not affected.
C: Not affected.

Operation

I bit ← 1

Description

Sets the interrupt mask bit I in the condition code register. If this bit is set, interrupt from peripheral equipment is disabled until the interrupt mask bit is cleared.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	SEI		9B			1	2

Example

```
0206 9B          SEI                INTERRUPT DISABLE
0207 9C          RSP                RESET STACK POINTER
0208 CD 06F0     JSR      SYSINZ    SYSTEM INITIALIZE
020B 9A          CLI                INTERRUPT ENABLE
```

STA (STore Accumulator)

Format

STA P

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of ACCA is "1"; otherwise reset.
Z: Set if the contents of ACCA are "0"; otherwise reset.
C: Not affected.

Operation

M ← (ACCA)

Description

Stores the ACCA contents into M. The ACCA contents are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	STA	M	B7	M		2	3
EXTENDED	STA	M	C7	MH	ML	3	4
INDEXED 0 BYTE OFFSET	STA	0,X	F7			1	4
INDEXED 1 BYTE OFFSET	STA	Disp,X	E7	D		2	4
INDEXED 2 BYTE OFFSET	STA	Disp,X	D7	DH	DL	3	5

Example

```
0190 B6 10        LDA     VAL1
0192 B7 FF        STA     WORK
0194 B6 50        LDA     RESULT
0196 F7           STA     0,X
0197 A6 FF        LDA     #$FF
0199 D7 0500      STA     EXVAL5,X
```

STOP (STOP)

Format

STOP

Condition Codes

Not affected.

Operation

Description

Enters into the STOP mode by STOP instruction (For details, refer to STOP MODE in "2.9. Low Power Consumption Mode".)

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	STOP		8E			1	4

Example

STX (STore indeX register)

Format

STX P

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of IX is "1"; otherwise reset.
Z: Set if the contents of IX are "0"; otherwise reset.
C: Not affected.

Operation

M ← (IX)

Description

Stores the IX contents into memory. IX contents are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	STX	M	BF	M		2	3
EXTENDED	STX	M	CF	MH	ML	3	4
INDEXED 0 BYTE OFFSET	STX	0,X	FF			1	4
INDEXED 1 BYTE OFFSET	STX	Disp,X	EF	D		2	4
INDEXED 2 BYTE OFFSET	STX	Disp,X	DF	DH	DL	3	5

Example

```
019C BE 10        LDX     VAL1
019E BF FF        STX     WORK
01A0 BE 50        LDX     RESULT
01A2 FF           STX     0,X
01A3 AE FF        LDX     #$FF
01A5 DF 0500      STX     EXVAL5,X
```

SUB (SUBtract)

Format

SUB P

Operation

ACCA ← (ACCA)-(M)

Condition Codes

H: Not affected:
I: Not affected.
N: Set if the most significant bit of the result is "1"; otherwise reset.
Z: Set if the contents of the result are "0"; otherwise reset.
C: Set if the absolute value of the contents of memory is greater than the absolute value of the contents of ACCA; otherwise reset.

Description

Subtracts M contents from ACCA contents, and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	SUB	#Imm	A0	Imm		2	2
DIRECT	SUB	M	B0	M		2	3
EXTENDED	SUB	M	C0	MH	ML	3	4
INDEXED 0 BYTE OFFSET	SUB	0,X	F0			1	3
INDEXED 1 BYTE OFFSET	SUB	Disp,X	E0	D		2	4
INDEXED 2 BYTE OFFSET	SUB	Disp,X	D0	DH	DL	3	5

Example

```
01A8 B6 10      LDA     VAL1    (VAL1)-(WORK)=(RESULT)
01AA B0 FF      SUB     WORK
01AC B7 50      STA     RESULT
```

SWI (SoftWare Interrupt)

Format

SWI

Condition Codes

H: Not affected.
I: Set.
N: Not affected.
Z: Not affected.
C: Not affected.

Operation

PC ← (PC)+0001
Msp ← (PCL), SP ← (SP)-0001
Msp ← (PCH), SP ← (SP)-0001
Msp ← (IX), SP ← (SP)-0001
Msp ← (ACCA), SP ← (SP)-0001
Msp ← (CCR), SP ← (SP)-0001
I bit ← 1
PC ← (SWI interrupt vector address)

Description

All the registers other than the stack pointer (SP) are pushed onto the stack. The interrupt mask bit is then set. Performs vectoring to the address indicated by the contents of the SWI interrupt vector address.

Addressing Mode and Number of Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	SWI		83			1	10

Example

```
01DC A6 FF      LDA     #$FF        *
01DE B7 36      STA     TIMER+1     * TIMER COUNTER SET
01E0 A6 3F      LDA     #$3F        *
01E2 B7 35      STA     TIMER       *
01E4 A6 03      LDA     #3          TIMER CODE SET
01E6 83         SWI                 MONITOR SERVICE CALL
```

TAX (Transfer Accumulator to indeX register)

Format

TAX

Condition Codes

Not affected.

Operation

IX ← (ACCA)

Description

Transfers the ACCA contents to IX. The ACCA contents are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	TAX		97			1	2

Example

```
01AE 97          TAX          SAVE ACCUMULATOR
01AF A6 04       LDA   #4     *
01B1 BB 50       ADD   RESULT ** ADD (RESULT+4)
01B3 B7 50       STA   RESULT *
01B5 9F          TXA          REVIVE ACCUMULATOR
```

TST (TeST)

Format

TST Q
TST A
TST X

Operation

(IX) - 00
or
(ACCA) - 00
or
(M) - 00

Condition Codes

H: Not affected.
I: Not affected.
N: Set if the most significant bit of ACCA, IX or M is "1"; otherwise reset.
Z: Set if the contents of ACCA, IX or M are "0"; otherwise reset.
C: Not affected.

Description

Sets N and Z bits of the condition code register according to the contents of ACCA, IX or M.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	TST	A	4D			1	2
INDEX REG.	TST	X	5D			1	2
DIRECT	TST	M	3D	M		2	4
INDEXED 0 BYTE OFFSET	TST	0,X	7D			1	4
INDEXED 1 BYTE OFFSET	TST	Disp,X	6D	D		2	5

Example

```
01BE 3D 03          TST     CNTRL
01C0 27 18          BEQ     INIT00   CNTRL=$00
                    *
01C2 3D FF          TST     WORK
01C4 2B 28          BMI     MINS00   WORK=(1XXX XXXX)
                    *
```

TXA (Transfer indeX register to Accumulator)

Format

TXA

Condition Codes

Not affected.

Operation

ACCA ← (IX)

Description

Transfers the IX contents to ACCA. IX contents are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	TXA		9F			1	2

Example

```
01B6 97        TAX          SAVE ACCUMULATOR
01B7 A6 04     LDA   #4     *
01B9 BB 50     ADD   RESULT ** ADD (RESULT+4)
01BB B7 50     STA   RESULT *
01BD 9F        TXA          REVIVE ACCUMULATOR
```

Power Control

WAIT

WAIT (WAIT)

Format

WAIT

Condition Codes

Not affected.

Operation

Description

Enters into WAIT mode by WAIT instruction
(Refer to WAIT MODE in "2.9 Low Power Consumption Mode" for details).

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	WAIT		8F			1	4

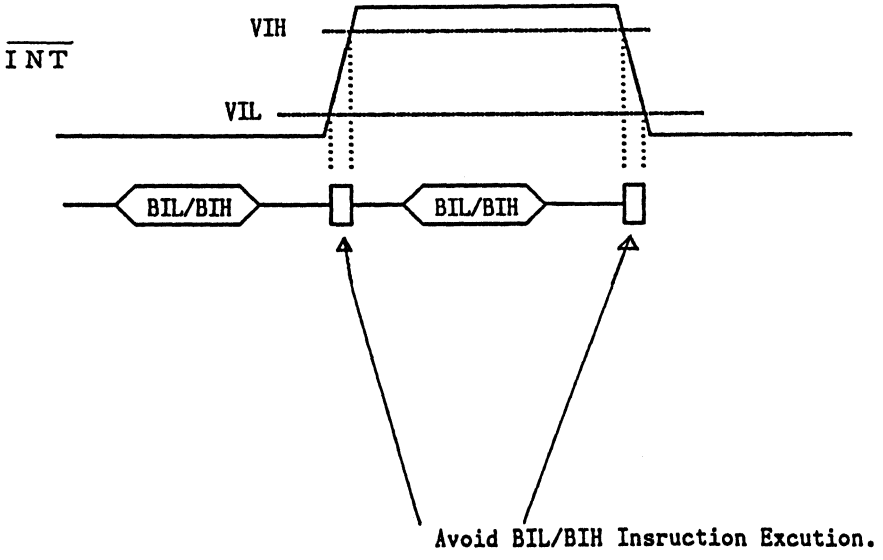
Example

1.7 BIL/BIH Instruction Precaution

(1) Execute Instruction after the $\overline{INT}$ voltage level has stabilized above V_{IH} of below V_{IL}.

(2) $\overline{INT}$ voltage level need to be stabilized while BIL/BIH Instruction Execution.

There may be a malfunction by glitch on control signal if BIL/BIH Instruction Execution has exercised in unstabilized $\overline{INT}$ signal level.



Section Three

• HD6305 U0
• HD6305 V0
• HD63705 V0
User's Manual

HITACHI

Section 3
HD6305U0, HD6305V0, HD63705V0 User's Manual
Table of Contents

1. OVERVIEW

1.1 Features of HD6305U0, HD6305V0, HD63705V0

Hitachi's HD6305U0, HD6305V0 are CMOS 8-bit single-chip microcomputers containing CPU, RAM, I/O on a single chip.

Features of this series are as follows;

(1) Powerful Bit Manipulation Instructions

Dew to upward compatibility with HD6805 family instruction set, powerful bit manipulation instructions are provided so that bit set, bit reset, bit test, and branch would be executed with a single instruction. These bit manipulation instructions are available to I/O and internal RAM as well.

(2) Low Power Consumption

To exploit the CMOS process technology fully, three low power consumption modes; STOP, WAIT and STANDBY; are incorporated.

Table 1-1 Features of HD6305U0, HD6305V0 and HD63705V0

Type No.		HD6305U0	HD6305V0	HD63705V0
Package		DP-40,FP-54,CP-44	DP-40,FP-54,CP-44	DC-40
Memory	ROM (k byte)	2	4	4
	RAM (byte)	128	192	192
I/O Port		31	31	31
Interrupt	external	2	2	2
	soft	1	1	1
	timer	2	2	2
	serial	1	1	1
Timer	8-bit Timer	1	1	1
	15-bit Timer	1	1	1
SCI		1 (clock synchronous)	1 (clock synchronous)	1
Speed Version (5V ±10%)	1MHz	HD6305U0	HD6305V0	HD63705V0
	1.5MHz	HD63A05U0	HD63A05V0	HD637A05V0
	2MHz	HD63B05U0	HD63B05V0	HD637B05V0
Operating Voltage	0.1 ∿ 0.5 MHz	3 ∿ 6.0V	3 ∿ 6.0V	4.5 ∿ 5.5V

Table 1-2 Feature of EPROM ON CHIP TYPE FAMILY

Type	HD63705V0C
Package	DC-40, (with window)
Equivalent Device	HD6305U0, HD6305V0
Process	CMOS
Erasure	Possible

HD63705V0 is fully compatible with HD6305V0.

It is possible to program with EPROM programmers that are commercially available (e.g 27256 type) since HD63705V0 contains 4k bytes of EPROM. And this makes it possible to use them for debugging as well as evaluating the program.

1.2 Block Diagram

A block diagram of HD6305U0, HD6305V0 is given in Fig. 1-1, HD63705V in Fig. 1-2, respectively.

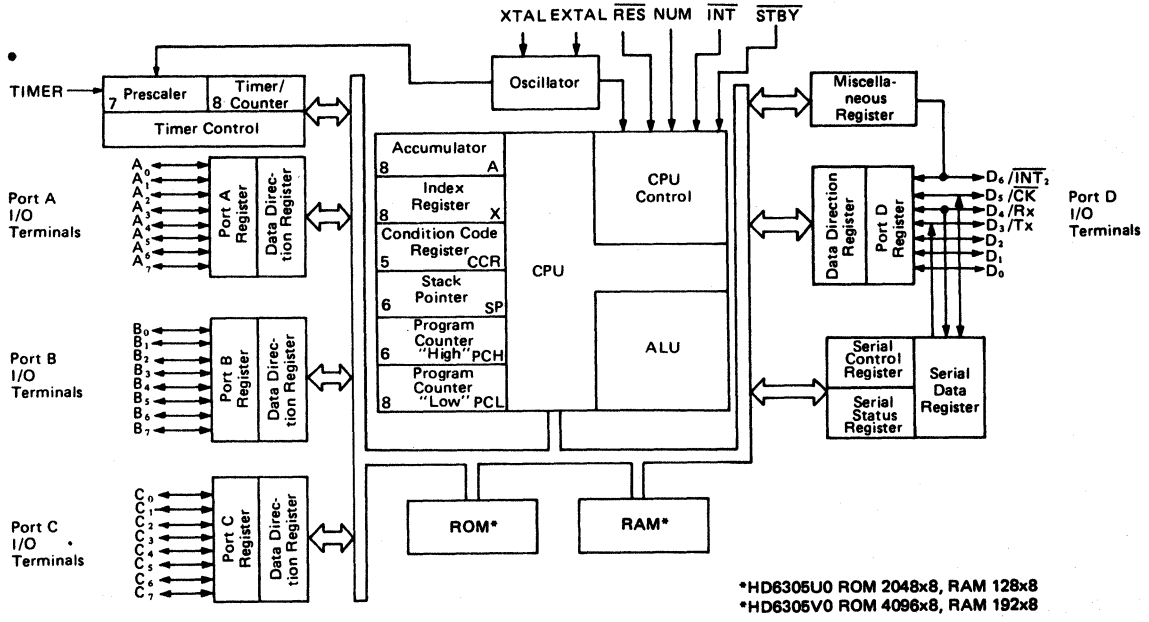


Fig. 1-1 HD6305U0, HD6305V0 Block Diagram

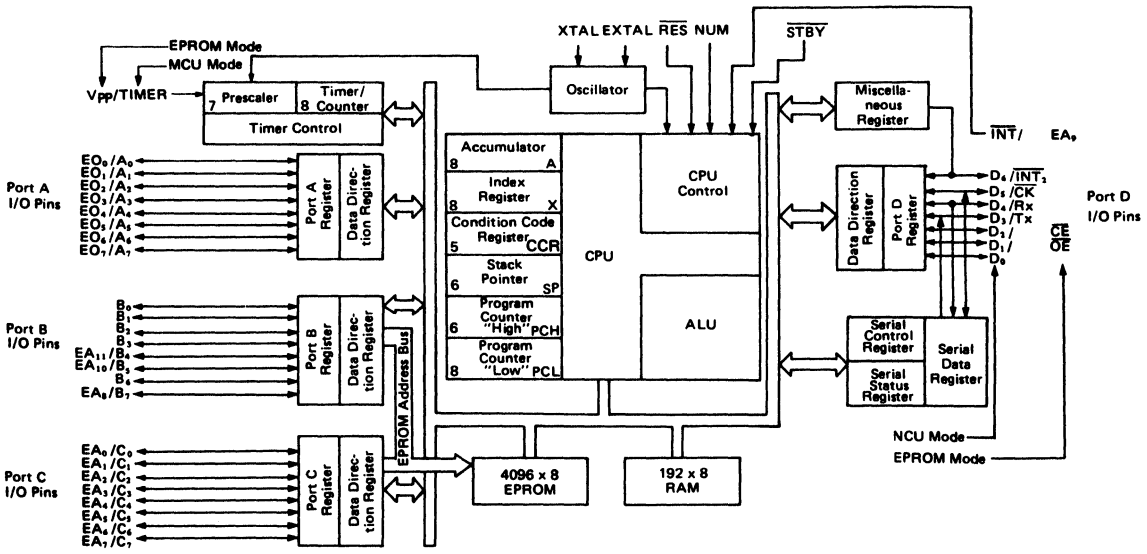


Fig. 1-2 HD63705V0 Block Diagram

Precaution for using the MCU in the ceramic package with a window

(1) The data stored in EPROM may be lost or the MCU may be malfunctioned by photocurrent if the MCU is exposed to strong light like a fluorescent lamp or the sunlight. Therefore, it is recommended to cover the window with an opaque label.

(2) Users should be particularly careful of static charge on the window, as it affects the MCU function to malfunction the LSI. The charge will be caused by rubbing the window with plastics or dry cloths, or touching a charged body on it. The opaque label that we recommended in (1) will be effective to distribute the charge evenly, if it is conductive.

1.3 Terminal Functions

(1) Terminal functions of HD6305U0 and HD6305V0 are described below.

- V_{CC}, V_{SS}

  Terminals for applying voltage. V_{CC} provides 5.0V±10% power supply, while V_{SS} is grounded.

- $\overline{INT}$, $\overline{INT_2}$; INPUT, INPUT/OUTPUT

  Terminals for external interrupt input to HD6305U0 and HD6305V0. Refer to "2.7 Interrupt" for details. $\overline{INT_2}$ is multiplexed with the D_6.

- XTAL, EXTAL

  Input terminals to the internal clock circuit. Connect a crystal oscillator (AT cut, 2.0 ∿8.0MHz) or ceramic filter to these terminals. Refer to "2.6 Internal Oscillator Options" for details.

- TIMER; INPUT

  An external input terminal for controling event counter. Refer to "2.3 Timer" for details.

- $\overline{RES}$; INPUT

  Resets the MCU. Refer to "2.5 Reset" for details.

- NUM

  NUM is not for user application. It can be used by connecting to V_{SS}.

- PORT A, B, C, D; INPUT/OUTPUT

  31 terminals consist of three 8-bit I/O ports (A, B, C) and one 7-bit I/O port (D). Each bit of these terminals function as input or output terminals by programming the Data Direction Register. Refer to "2.8 Input/Output" for details.

  D_6 of port D is multiplexed with $\overline{INT_2}$. When D_6 functions as a port, set MR_6 in Miscellaneous Register to "1" in order to mask $\overline{INT_2}$ interrupt.

- $\overline{STBY}$; INPUT

  A terminal for permitting the MCU to enter into the Standby Mode. When $\overline{STBY}$ goes to "Low" level, the oscillation stops and the MCU enters into the Reset Mode. Refer to "Standby Mode" in "2.9 Low Power Consumption Mode" for details.

- $\overline{CK}$ (D_5); INPUT/OUTPUT

  Terminal to transmit or receive SCI clock for serial data transfer. Refer to "2.4 Serial Communication Interface" for details.

- T_x (D_3); INPUT/OUTPUT

  This terminal used to transmit serial data. Refer to "2.4 Serial Communication Interface" for details.

- R_x (D_4); INPUT/OUTPUT

  This terminal is used to receive serial data. Refer to "2.4 Serial Communication Interface" for details.

(2) The followings describs HD63705V0 MCU input and output terminals.

- V_{CC}, V_{SS}

  These are the power supply inputs. V_{CC} = 5.0V ± 10%, V_{SS} = 0V (ground).

- $\overline{INT}/EA_9$, $\overline{INT_2}$; INPUT, INPUT/OUTPUT

  The MCU receives an external interrupt through these terminals. For details, refer to "2.7 Interrupts". $\overline{INT_2}$ is multiplexed with Port D_6. In the EPROM mode, $\overline{INT}$ is used as input of EA9.

- XTAL, EXTAL

  These pins are inputs of the internal oscillator circuit. Connect crystal (AT cut, 2.0 ∿ 8.0 MHz) or ceramic resonator to them. Refer to "2.6 Internal Oscillator Options" for using these inputs.

- TIMER/V_{PP}; INPUT

  This is an external input to control the internal timer circuit. Refer to "2.3 Timer" for details.

  In the EPROM mode, this terminal is used when programming EPROM. The programming voltage V_{PP} is applied to this terminal.

  When the voltage of 12.5V ± 0.3V is applied to this terminal and $\overline{CE}$ and $\overline{OE}$ are set "Low" and "High" respectively, datas are written into EPROM through Port A (EO_0 ∿ EO_7). EPROM addresses are input through the Port C (EA_0 ∿ EA_7), Port B (EA_8, EA_{10} ∿ EA_{11}) and $\overline{INT}$ (EA_9).

- $\overline{RES}$; INPUT

  This is used to reset the MCU. Refer to "2.5 Reset" for details.

- NUM

  This is not for user application. Connect this pin to the V_{SS} in the MCU mode and to the V_{CC} in the EPROM mode.

- PORT A, B, C, D; INPUT/OUTPUT

  These 31 terminals consist of three 8-bit I/O ports (A, B, C) and a 7-bit I/O port (D). Each bit of these pins can be programmed as an input or output by programming the Data Direction Register. D_6 is multiplexed with the $\overline{INT_2}$. When using the D_6 as a port, set the $\overline{INT_2}$ interrupt mask bit in the miscellaneous register to "1" to prevent the $\overline{INT_2}$ interrupt. Refer to "2.8 Input/Output" for details.

- $\overline{STBY}$; INPUT

  This is used to put the MCU into the standby mode. Setting this pin to "low" level stops the internal oscillator and resets the MCU internal state. Refer to "Standby Mode" in "2.9 Low Power Consumption Mode" for details.

  The followings are input/output pins for serial communication interface (SCI). These are multiplexed with D_3, D_4, or D_5. Refer to "2.4 Serial Communication Interface" for details.

- $\overline{CK}$ (D_5); INPUT/OUTPUT

  This is used to input or output clocks when receiving or transmitting serial data.

- R_x (D_4); INPUT/OUTPUT

  This is used to receive serial data.

- T_x (D_3); INPUT/OUTPUT

  This is used to transmit serial data.

2. INTERNAL HARDWARE AND OPERATIONS

2.1 Memory

Fig. 2-1 shows the memory map of HD6305U0, HD6305V0 and HD63705V0.

During the processing of the interrupt, the register contents are pushed onto the stack in the order shown in Fig. 2-2. The stack pointer decrements. The low order byte (PCL) of the program counter is stacked first and the high order byte (PCH) of the program, the index register (X), the accumulator (A) and the condition code register (CCR) are stacked in that order. For subroutine calls, program counter (PCH, PCL) contents are pushed onto the stack.

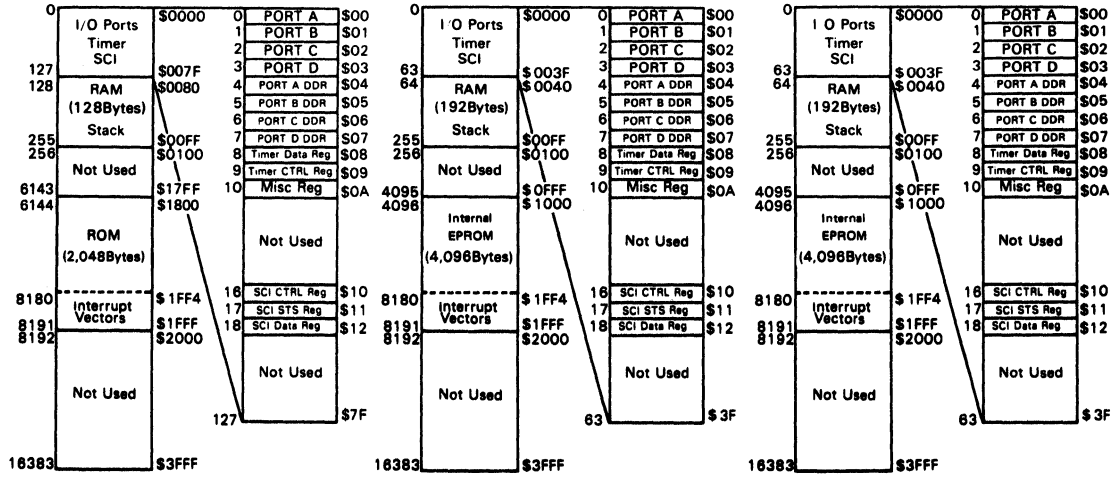


Fig. 2-1 Memory Map

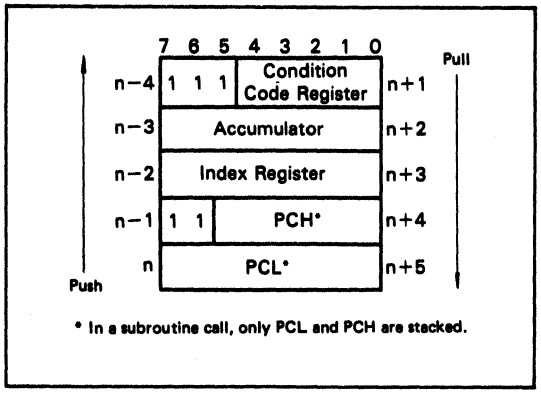


Fig. 2-2 Sequence of Interrupt Stacking

2.2 Registers

This CPU contains five registers available to the programmer. They are shown in Fig. 2-3.

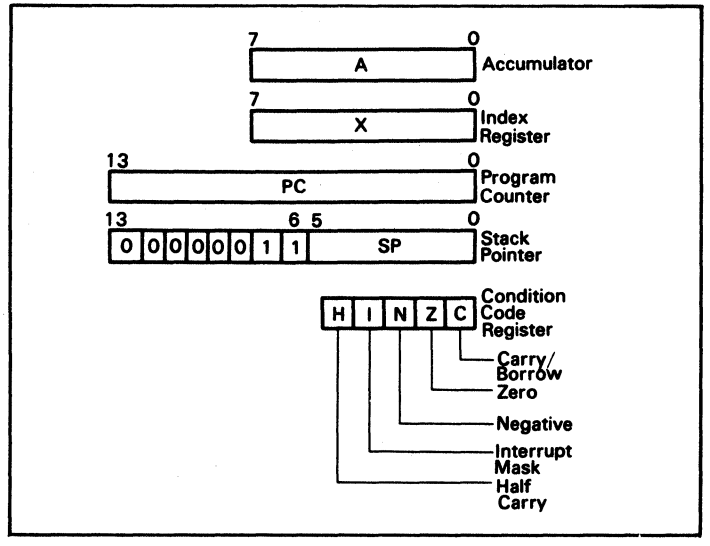


Fig. 2-3 Programming Model

(1) Accumulator (A)

The accumulator is an 8-bit general purpose register which holds operands and results of the arithmetic operations or data manipulations.

(2) Index Register (X)

The index register is an 8-bit register used for the indexed addressing mode. It contains an 8-bit value which is added to an offset to create an effective address.

The index register can also be used for data manipulations with read-modify-write instructions.

When not performing addressing operations, the register can be used as a temporary storage area.

(3) Program Counter (PC)

The program counter is a 14-bit register which contains the address of the next instruction to be executed.

(4) Stack Pointer (SP)

The stack pointer is a 14-bit register containing the address of the next free location of the stack. Initially, the stack

pointer is set to location $00FF. It is decremented as a data is pushed on to the stack and incremented as data is then pulled out of the stack. The 8 high-order bits of the stack pointer are fixed to 00000011. During an MCU reset or a reset stack pointer (RSP) instruction, the pointer is set to location $00FF. Subroutines and interrupts may be nested down to $00C1, which allows programmers to use up to 31 levels of subroutine calls and 12 levels of interrupt responses.

(5) Condition Code Register (CCR)

The condition code register is a 5-bit register indicating the results of the instruction just executed. These bits can be individually tested by conditional branch instructions. Each bit is described in the following paragraphs.

(a) Half Carry (H)

When set, this bit indicates that a carry occurred between bit 3 and 4 during an arithmetic operation (ADD, ADC).

(b) Interrupt (I)

Setting this bit masks all interrupts except for software ones. If an interrupt occurs while this bit is set, the interrupt is latched and is processed as soon as the interrupt bit (I) is cleared. (More precisely the interrupt enters the servicing routine after the instruction next to the CLI is executed.)

(c) Negative (N)

When set, this bit indicates that the result of the last arithmetic, logical, or data manipulation is negative. (Bit 7 in the result is a logical "1".)

(d) Zero (Z)

When set, this bit indicates that the result of the last arithmetic, logical, or data manipulation is zero.

(e) Carry/Borrow (C)

When set, this bit indicates that a carry or borrow occurred during the last arithmetic operation. This bit is also affected by bit test and branch instructions, shifts and rotates.

2.3 Timer

Fig. 2-4 contains a block diagram of the timer. The 8-bit counter may be loaded under program control and is decremented towards zero by the clock input. When the counter, that is, the timer data register (TDR) reaches zero, the timer interrupt request bit (bit 7) of the timer control register is set. When recognizing the interrupt request, the MCU proceeds to store the current CPU state on the stack, and then fetches the timer vector address from locations $1FF8 and $1FF9 (or $1FF6 and $1FF7 if in the wait mode) in order to execute the interrupt service routine. The timer interrupt can be masked by setting the interrupt mask bit (bit 6) of the timer control register. The mask bit (I) of the condition code register can also disable the timer interrupt.

The clock input to the timer can be from an external source applied to the TIMER input pin, or it can be the internal E signal (which is a clock obtained by dividing the oscillator clock by four). When the E signal is used as the source, it can be gated by an input applied to the TIMER pin.

The counter start counting down from "$FF" after it reaches zero. The counter may be monitored at any time by reading the contents of the timer data register. This allows a program to determine the length of the time since the occurrence of a timer interrupt without distarbing the counter contents.

At reset, the prescaler and counter are initialized to "$7F" and "$F0", respectively. The timer interrupt request bit (bit 7) is cleared and the timer interrupt request mask (bit 6) is set.

The timer interrupt request bit must be cleared by software.

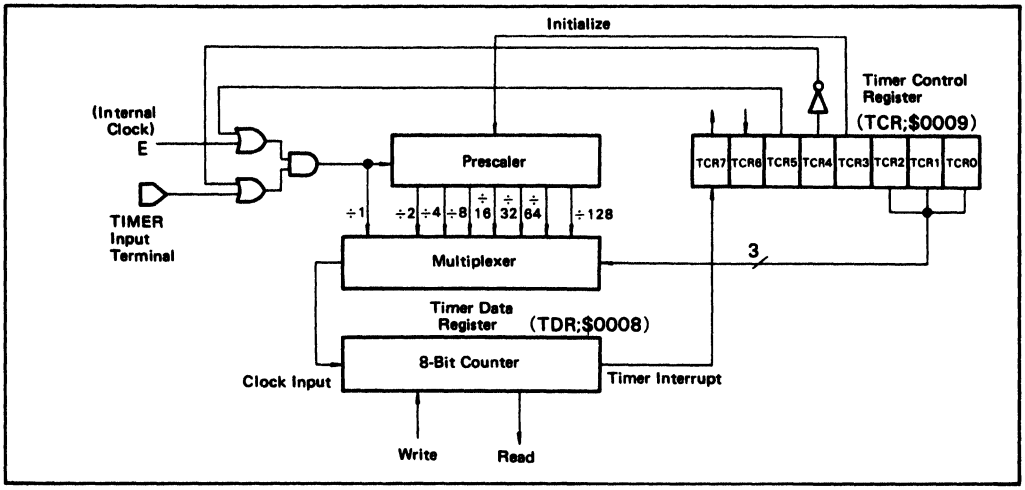


Fig. 2-4 Timer Block Diagram

Table 2-1

TCR7	Timer interrupt request
0	Absent
1	Present

TCR6	Timer interrupt mask
0	Enabled
1	Disabled

(1) Timer Control Register (TCR; $0009)

Selection of a clock source, selection of a prescaler frequency division ratio, and a timer interrupt can be controlled by the timer control register (TCR; $0009). (Refer to Fig. 2-5).

For the selection of a clock source, any one of the four modes (Refer to Table 2-2) can be selected by bits 5 and 4 of the timer control register (TCR).

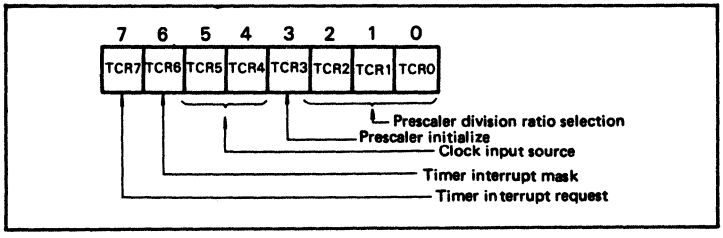


Fig. 2-5 Timer Control Register (TCR; $0009)

After reset, the TCR is initialized to E under timer terminal control (bit 5 = 0, bit 4 = 1). If the timer terminal is "1", the counter starts counting down with "$F0" immediately after reset.

When "1" is written in bit 3, the prescaler is initialized. This bit always shows "0" when read.

A prescaler division ratio is selected by the combination of three bits (bits 0, 1 and 2) of the timer control register (Refer to Table 2-3). There are eight different division ratios: ÷1, ÷2, ÷4, ÷8, ÷16, ÷32, ÷64 and ÷128. After reset, the TCR is set to the ÷1 mode.

A timer interrupt is enabled when the timer interrupt mask bit is "0", and disabled when the bit is "1". When a timer interrupt occurs, "1" is set in the timer interrupt request bit. This bit can be cleared by writing "0" in that bit.

Table 2-2 Clock Source Selection

TCR		Clock input source
Bit 5	Bit 4	
0	0	Internal clock E
0	1	E under timer terminal control
1	0	No clock input (counting stopped)
1	1	Event input from timer terminal

Table 2-3 Prescaler Division Ratio Selection

TCR			Prescaler division ratio
Bit 2	Bit 1	Bit 0	
0	0	0	÷1
0	0	1	÷2
0	1	0	÷4
0	1	1	÷8
1	0	0	÷16
1	0	1	÷32
1	1	0	÷64
1	1	1	÷128

2.4 Serial Communication Interface (SCI)

The SCI is used for transferring 8-bit data in serial. The transfer clock width ranges from 1 μs to about 32 ms (with a 4 MHz oscillator), and sixteen types of transfer clock widths are available. The SCI consists of three registers, an octal counter and a prescaler as shown in Fig. 2-6. It communicates with CPU through the data bus lines and with peripherals through bits 3, 4 and 5 of Port D. The followings describe the registers and the data transfer operations.

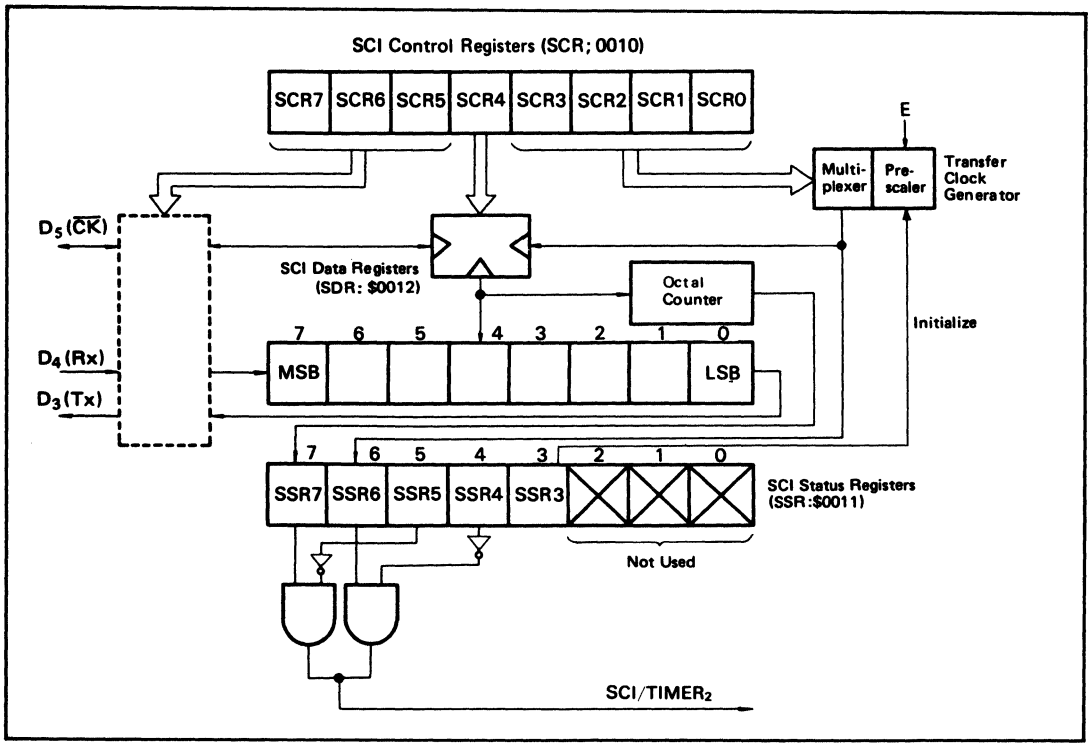


Fig. 2-6 SCI Block Diagram

(1) SCI Control Register (SCR: $0010)

Fig. 2-7 shows SCI Control Register configulation.

Bit 7 (SCR7)

When this bit is set to "1", the DDR corresponding to D_3 is set to "1" and D_3 functions as output of SCI data. After reset, the bit is initialized to "0".

Bit 6 (SCR6)

When this bit is set to "1", the DDR corresponding to D_4 is set to "0" and D_4 functions as input of SCI data. After reset, the bit is initialized to "0".

Bits 5 and 4 (SCR5, SCR4)

These bits select a clock source. After reset, the bits are initialized to "0".

Bits 3 ∿ 0 (SCR3 ∿ SCR0)

These bits select a transfer clock rate. After reset, the bits are initialized to "0".

7	6	5	4	3	2	1	0
SCR7	SCR6	SCR5	SCR4	SCR3	SCR2	SCR1	SCR0

Fig. 2-7 SCI Control Register

SCR3	SCR2	SCR1	SCR0	Transfer clock rate	
				4.00 MHz	4.194 MHz
0	0	0	0	1 µs	0.95 µs
0	0	0	1	2 µs	1.91 µs
0	0	1	0	4 µs	3.82 µs
0	0	1	1	8 µs	7.64 µs
≀	≀	≀	≀	≀	≀
1	1	1	1	32768 µs	1/32s

SCR7	D_3 terminal
0	Used as I/O terminal (by DDR).
1	Serial data output (DDR output)

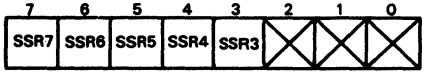
SCR6	D_4 terminal
0	Used as I/O terminal (by DDR).
1	Serial data input (DDR input)

SCR5	SCR4	Clock source	D_5 terminal
0	0	-	Used as I/O terminal (by DDR).
0	1	-	
1	0	Internal	Clock output (DDR output)
1	1	External	Clock input (DDR input)

(2) SCI Data Register (SDR: $0012)

A serial-parallel conversion register used for data transfer.

(3) SCI Status Register (SSR: $0011)



Bit 7 (SSR7)

This is the SCI interrupt request bit which is set on the completion of transmitting or receiving 8-bit data. It is cleared when the MCU is reset or data is written to or read from the SCI data register with the SCR5 = "1". This bit can also be cleared by writing "0" into it.

Bit 6(SSR6)

This is the $TIMER_2$ interrupt request bit. The $TIMER_2$ is also used as the serial clock generator, and this bit is set on the every negative edge of the internal transfer clock. When the MCU is reset, the bit is cleared. It can also be cleared by writing "0" into it. (Refer to "2.3 Timer" for details).

Bit 5 (SSR5)

This is the SCI interrupt mask bit which can be set or cleared by software. When this bit is "1", the SCI interrupt (SSR7) is masked. Resetting the MCU sets this bit to "1".

Bit 4 (SSR4)

This is the $TIMER_2$ interrupt mask bit which can be set or cleared by software. When this bit is "1", the $TIMER_2$ interrupt (SSR6) is masked. Resetting the MCU sets this bit to "1".

Bit 3 (SSR3)

Writing "1" into this bit initializes the prescaler of the transfer clock generator. A read of this bit always indicates "0".

Bit 2 ∿ Bit 0

Not used.

SSR7	SCI Interrupt Request
0	Not Requested
1	Requested

SSR6	$TIMER_2$ Interrupt Request
0	Not Requested
1	Requested

SSR5	SCI Interrupt Mask
0	Interrupt Allowed
1	Interrupt Inhibited

SSR4	$TIMER_2$ Interrupt Mask
0	Interrupt Allowed
1	Interrupt Inhibited

(4) Transmit Operations

The transfer clock width and the clock source are determined by setting the corresponding SCI control register bits, and then D_3 and D_5 are set to serial data output pin and serial clock pin respectively. The transmit data should be moved from the accumulator or the index register into the SCI data register. Then, the data moved into the SCI data register is output through the D_3/Tx pin, starting with the LSB, synchronously with the negative edge of the serial clock. Refer to Fig. 2-8. When 8 bits of data have been transmitted, the bit 7 of the SCI status register (interrupt request bit) is set on the positive edge of the last serial clock. This interrupt request can be masked by setting bit 5 of the SCI status register. After completion of the data transmission, the 8th bit of data (MSB) stays at the D_3/Tx pin. If the external clock source is selected, the transfer clock width determined by bit 0 to bit 3 of the SCI control register is ignored and the $D_5/\overline{CK}$ pin is set as input. If the internal clock source is selected, the $D_5/\overline{CK}$ is set as output and clocks are generated with the transfer clock width selected by bit 0 to 3 of the SCI control register.

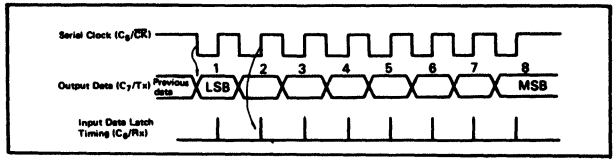


Fig. 2-8 SCI Timing

(5) Receive Operations

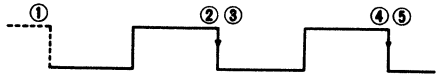
The transfer clock width and the clock source are determined by setting the corresponding SCI control register bits, and the D_4 pin and the D_5 pin are set to serial data input pin and serial clock pin respectively. Then the receive operation is enabled by dummy-reading or -writing the SCI data register. (This procedure is not needed after a data is received. It is needed after reset or when no data is received yet.) The received data through the D_4/Rx pin is input to the SCI data register synchronously with the positive edge of the serial clock. (Refer to Fig. 2-8.) At completion of 8-bit data reception, the bit 7 in the SCI status register (interrupt request bit) is set. This interrupt request can be masked by setting bit 5 of the SCI status register. If the external clock source is selected, the transfer clock width determined by bit 0 to bit 3 of the SCI control register is ignored and data is received synchronously with the clock input through the $D_5/\overline{CK}$ pin. If the internal clock source is selected, the $D_5/\overline{CK}$ acts as an output and clocks are output with the transfer clock selected by bit 0 to bit 3 of the SCI control register.

(6) $TIMER_2$

The SCI transfer clock generator functions as a timer. The SCI clock selected by bits 3 to 0 of the SCI Control Register (4 μs to approx -32 ms in 4 MHz operation) is transmitted to bit 6 of the SCI Status Register, and the timer 2 interrupt request bit is set at each falling edge of the SCI clock. Since this interrupt occurs periodically, $Timer_2$ is available for a reload counter or a timer.

$Timer_2$ is multiplexed with the SCI transfer clock generator. When using $Timer_2$ independently of the SCI, external clock should be selected as SCI clock source by setting both SC5 and SC4 to "1".

If Internal clock is selected as a SCI clock source, reading from or writing to the SDR initializes the prescalor of the SCI transfer clock generator.



① : Transfer clock generator is reset and mask bit (bit 4 of SCI status register) is cleared.
②,④ : TIMER2 interrupt request
③,⑤ : TIMER2 interrupt request bit cleared

2.5 Reset

The MCU can be reset either by external reset ($\overline{RES}$) or power-on reset (Refer to Fig. 2-9). On power up, the reset input must be held "Low" for at least t_{OSC} to assure that the internal oscillator is stabilized. A sufficient delay time can be obtained by connecting a capacitance to the $\overline{RES}$ input as shown in Fig. 2-10.

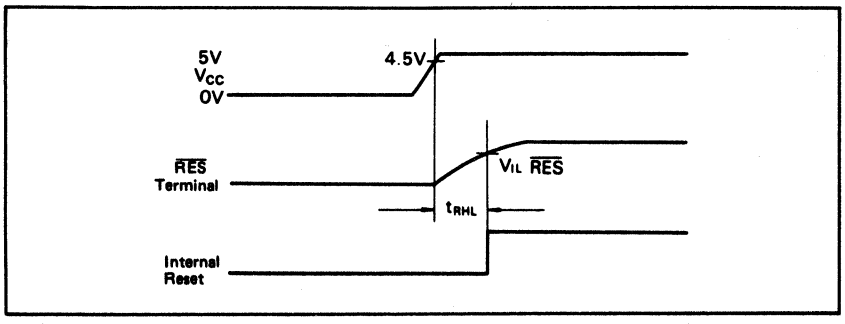


Fig. 2-9 Power On and Reset Timing

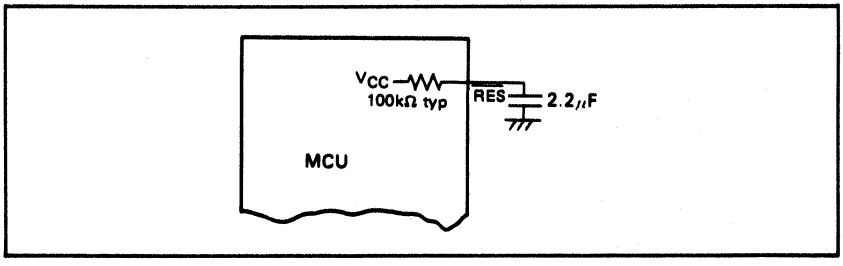


Fig. 2-10 Input Reset Delay Circuit

2.6 Internal Oscillator Options

The internal oscillator circuit is designed to meet the requirement for minimum external configurations. It can be operated by connecting a crystal (AT cut, 2.0 ∿ 8.0 MHz) or ceramic oscillator between pins 38 and 39 according to the required oscillation frequency stability.

Fig. 2-11 shows three types of terminal connections, while Fig. 2-12 and 2-13 illustrate the specifications and typical arrangement of the crystal, respectively.

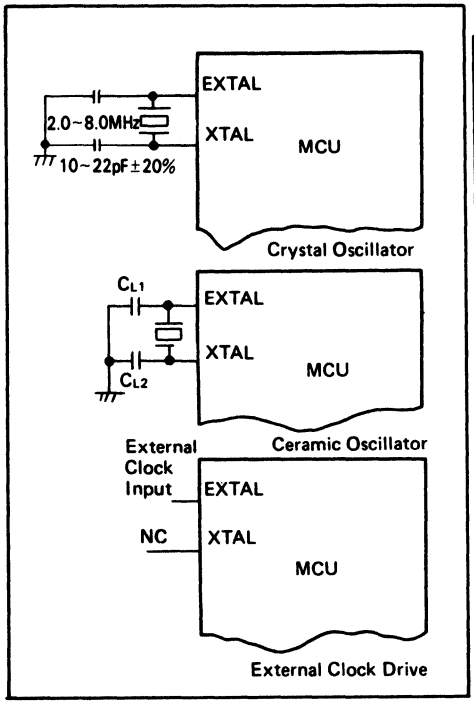


Fig. 2-11 Internal Oscillator Options

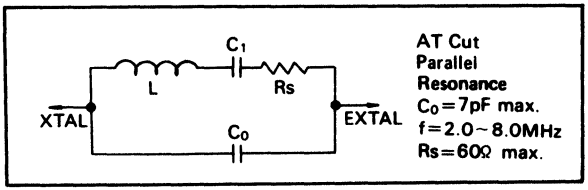


Fig. 2-12 Crystal Parameters

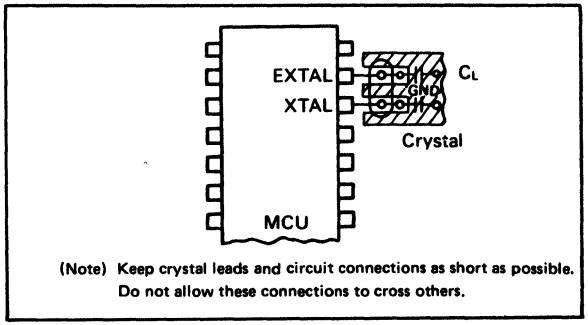


Fig. 2-13 Typical Arrangement of Crystal

2.7 Interrupts

HD6305U0 and HD6305V0 have six interrupt sources; external interrupt ($\overline{INT}$, $\overline{INT_2}$), internal timer interrupt (TIMER, $TIMER_2$), serial interrupt (SCI) and interrupt by an instruction (SWI). $\overline{INT_2}$, TIMER, SCI, and $TIMER_2$ interrupts generate vector addresses.

When an interrupt occurs, the progressing program stops and the current CPU status is pushed onto the stack. Then Interrupt Mask bit (I) of Condition Code Register is set and the start address of the interrupt service routine is obtained from a particular interrupt vector address. Then the interrupt service routine starts from the start address. System can return from the interrupt service routine by RTI instruction. When RTI instruction is executed, the CPU status before the interrupt (saved onto the stack) is pulled and the CPU restarts the sequence with the instruction next to the one at which the interrupt occurred. Table 2-4 lists the priority of interrupts and their vector address.

A flowchart of the interrupt sequence is shown in 2-14. A block diagram of the interrupt request source is shown in Fig. 2-15.

(1) External Interrupt

$\overline{INT}$ input can be selected as level and edge combination sense, or edge sense by programming bit 5 of Miscellaneous Register. (If bit 5 = 1, $\overline{INT}$ is level and edge sensitive. If bit 5 = 0, $\overline{INT}$ is edge sensitive.) $\overline{INT_2}$ input is an edge triggered input. If the falling edge of $\overline{INT}$, or $\overline{INT_2}$ input is detected, it is internally latched and the interrupt request is accepted. Internally latched edge sensitive $\overline{INT}$ request is automatically cleared when the CPU initiates $\overline{INT}$ interrupt service routine. The $\overline{INT_2}$ interrupt request is cleared when "0" is written to bit 7 of the Miscellaneous Register.

If the I bit of the Condition Code Register is set to "1", interrupt requests of the external interrupts ($\overline{INT}$, $\overline{INT_2}$) are retained, but not serviced. Immediately after the I bit is cleared, the CPU jumps to the corresponding interrupt routine. The $\overline{INT_2}$ interrupt can be masked by setting bit 6 of the Miscellaneous Register.

The $\overline{INT}$ status can be tested by a BIL or BIH instruction. The $\overline{INT}$ falling edge detector circuit and its latching circuit, and $\overline{INT_2}$ are not affected by executing BIL and BIH instructions.

(2) Internal Interrupt and SCI Interrupt

When I bit of the Condition Code Register is set to "1", internal timer interrupts (TIMER, $TIMER_2$) and SCI interrupt requests are retained, without being serviced. Immediately after the I bit is cleared, the CPU initiates the corresponding interrupt service routine. Timer interrupt, SCI interrupt, $Timer_2$ interrupt can be masked by setting bit 6 of Timer Control Register, bit 5 of SCI Status Register, and bit 4 of SCI Status Register, respectively.

Table 2-4 Interrupt Execution Priority

Interrupt	Priority	Vector Address
$\overline{RES}$	1	$1FFE, $1FFF
SWI	2	$1FFC, $1FFD
$\overline{INT}$	3	$1FFA, $1FFB
TIMER/$\overline{INT_2}$	4	$1FF8, $1FF9
TIMER (Wait Mode)	5	$1FF6, $1FF7
SCI/$TIMER_2$	6	$1FF4, $1FF5

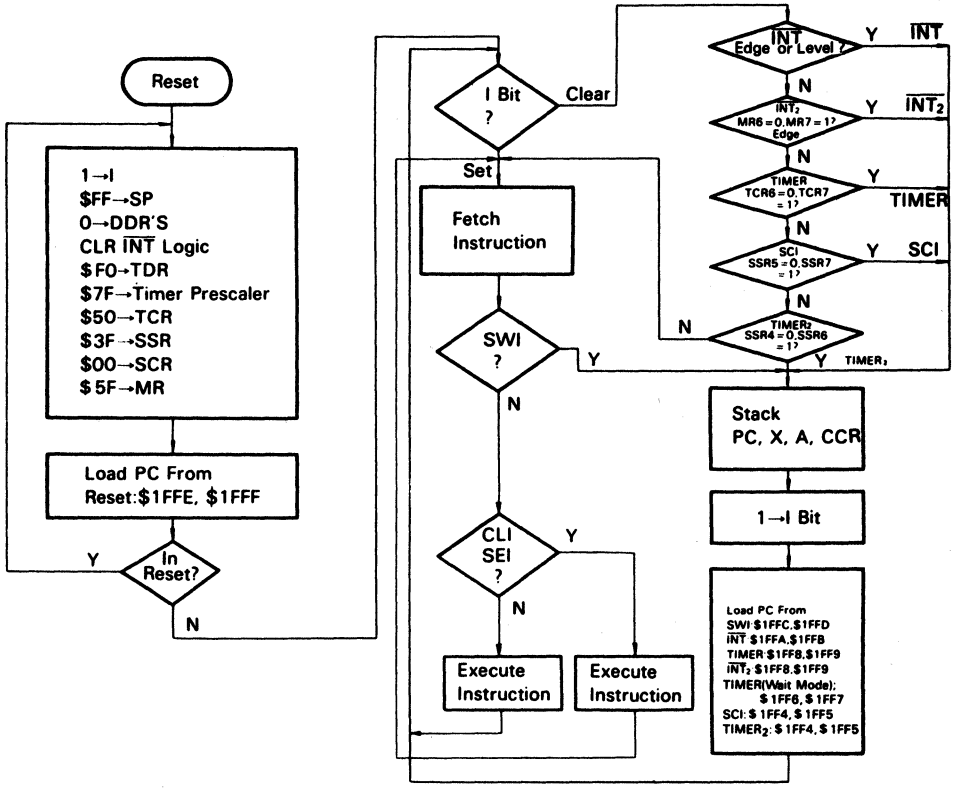


Fig. 2-14 Interrupt Flowchart

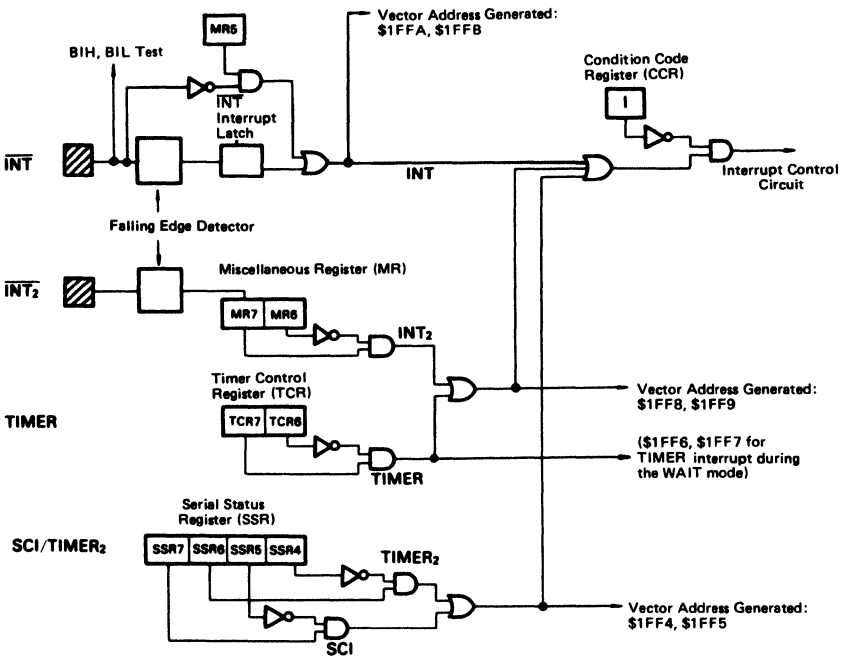
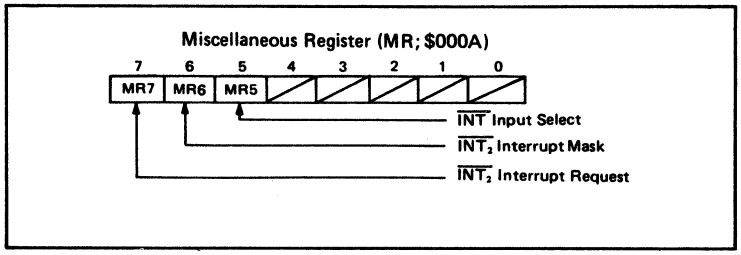


Fig. 2-15 Interrupt Request Generation Circuitry

(3) Miscellaneous Register (MR: $000A)

Miscellaneous Register (MR: $000A) specifies $\overline{INT}$ request sense (edge sense or level sense) and controls $\overline{INT_2}$ interrupt. Bit 7 of Miscellaneous Register is an $\overline{INT_2}$ interrupt request flag. When the falling edge of $\overline{INT_2}$ is detected, MR7 is set to "1". In the interrupt service routine (vector address: $1FF8, $1FF9), bit 7 is checked by software if it is $\overline{INT_2}$ interrupt or not. Bit 7 can be reset by software. Bit 6 is the $\overline{INT_2}$ interrupt mask bit. If this bit is set to "1", the $\overline{INT_2}$ interrupt is disabled. Bit 7 is softwarely readable and writable, but cannot be programmed to "1". This means that an interrupt cannot be requested by software.

During reset, bit 7 and bit 5 are initialized to "0" and bit 6 is initialized to "1".



2.8 Input/Output

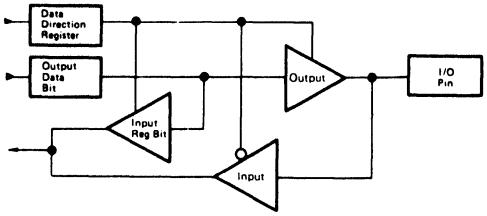
(1) I/O port

31 Input/Output ports (port A, B, C, D) are provided. Each of them can function as either input or output by programming the Data Direction Register. If corresponding bit of Data Direction Register is programmed to "0", the I/O port functions as an input. While corresponding bit of Data Direction Register is programmed to "1", the I/O port functions as an output (Refer to Fig. 2-16(a)).

During reset, all the Data Direction Registers are initialized to "0" and all I/O ports function as input.

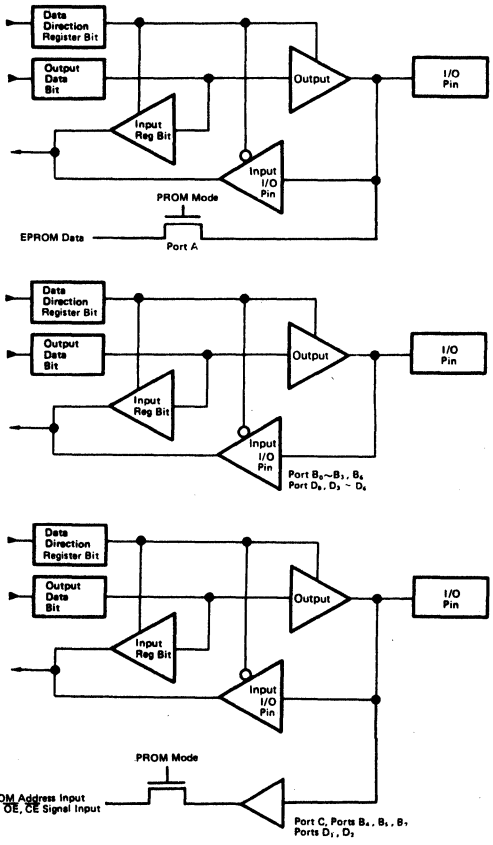
I/O ports are compatible with TTL and CMOS in respect to both input and output.

If I/O ports are not used, they should be connected to V_{SS} through resistors. If these terminals are left open, they may consume extra power.



Bit of data direction register	Bit of output data	Status of output	Input to MCU
1	0	0	0
1	1	1	1
0	×	3-state	Pin

a. HD6305U0, HD6305V0 (Ports A, B, C and D)



b. HD63705V0

Fig. 2-16 Input/Output Common Port Circuit Diagram

2.9 Low Power Consumption Mode

The HD 6305U0, HD6305V0, HD63705V0 have 3 Low Power Consumption Modes; Wait mode, Standby mode, Stop mode.

(1) Wait Mode

When WAIT instruction is executed, the MCU enters into the WAIT mode, the oscillator does not stop, but the internal clock stops. The CPU stops but the internal peripherals; Timer and SCI; continue their current operations. (NOTE: Once the system enters into the wait mode, the serial communication interface can no longer be retriggered.) During the Wait mode, CPU registers, except I bit of the Condition Code Register, RAM and I/O terminals hold their conditions just before entering into the wait mode. I bit of the Condition Code Register is cleared to "0".

The MCU is released from wait mode by an interrupt (INT, TIMER/$\overline{INT_2}$, or SCI/$TIMER_2$), $\overline{RES}$ or $\overline{STBY}$. The $\overline{RES}$ resets the MCU, and the $\overline{STBY}$ brings it into the standby mode. (Refer to Standby mode for details).

When an interrupt request is accepted, the CPU is released from the Wait mode to start interrupt response sequence for vectoring to the interrupt service routine. If TIMER/$\overline{INT_2}$ or SCI/$\overline{TIMER_2}$ interrupt is masked by programming the Timer Control Register, Miscellaneous Register or SCI Status Register, the wait mode cannot be released by interrupt.

Fig. 2-17 shows a flowchart when executing $\overline{WAIT}$ instruction.

(2) Stop Mode

When STOP instruction is executed, the MCU enters into the stop mode. In this mode, the oscillator stops, and the CPU and internal peripherals stop operating. The RAM, registers (except bit 6 and 7 of the timer control register and the I bit of the condition code register) and I/O terminals retain their condition just before entering into the stop mode.

The MCU is recovered from the stop mode by the external interrupt ($\overline{INT}$, or $\overline{INT_2}$), $\overline{RES}$ or $\overline{STBY}$. The $\overline{RES}$ resets the MCU, and the $\overline{STBY}$ brings it into the standby mode.

When an interrupt request is accepted, the stop mode is exited and the CPU begins the interrupt response sequence for vectoring to the interrupt service routine. If the $\overline{INT_2}$ interrupt is masked by programming Miscellaneous Register, the stop mode cannot be exited

by $\overline{INT_2}$ interrupt.

Fig. 2-18 shows a flowchart when executing STOP instruction. And Fig. 2-19 shows a timing chart when recovering from stop mode. When an interrupt ($\overline{INT}$ or $\overline{INT_2}$) occurs during stop mode, the oscillation starts and 20 ms max later the stop mode is exited and the interrupt response sequence starts. When $\overline{RES}$ input is held"Low" for at least 20 ms during stop mode, the oscillator starts and the MCU restarts operation by asserting $\overline{RES}$ "High". Note that $\overline{RES}$ input should be held "Low" for at least 20 ms (t_{osc}) to assure oscillator stabilization time.

(3) Standby Mode

The MCU goes into the standby mode when the $\overline{STBY}$ goes "Low". In this mode, all MCU operations stop and the internal condition is reset, holding the RAM contents. All I/O terminals enters into high-impedance state. The standby mode can be exited by asserting $\overline{STBY}$ "High", and $\overline{RES}$ "Low" simultaneously. Then, $\overline{RES}$ should be held "Low" for at least 20 ms (t_{osc}) to assure oscillation stabilization time. Fig. 2-20 shows $\overline{RES}$ and $\overline{STBY}$ timing.

Table 2-5 lists the status of each part of the MCU in each low power dissipation modes. Transitions between each mode are shown in Fig. 2-21.

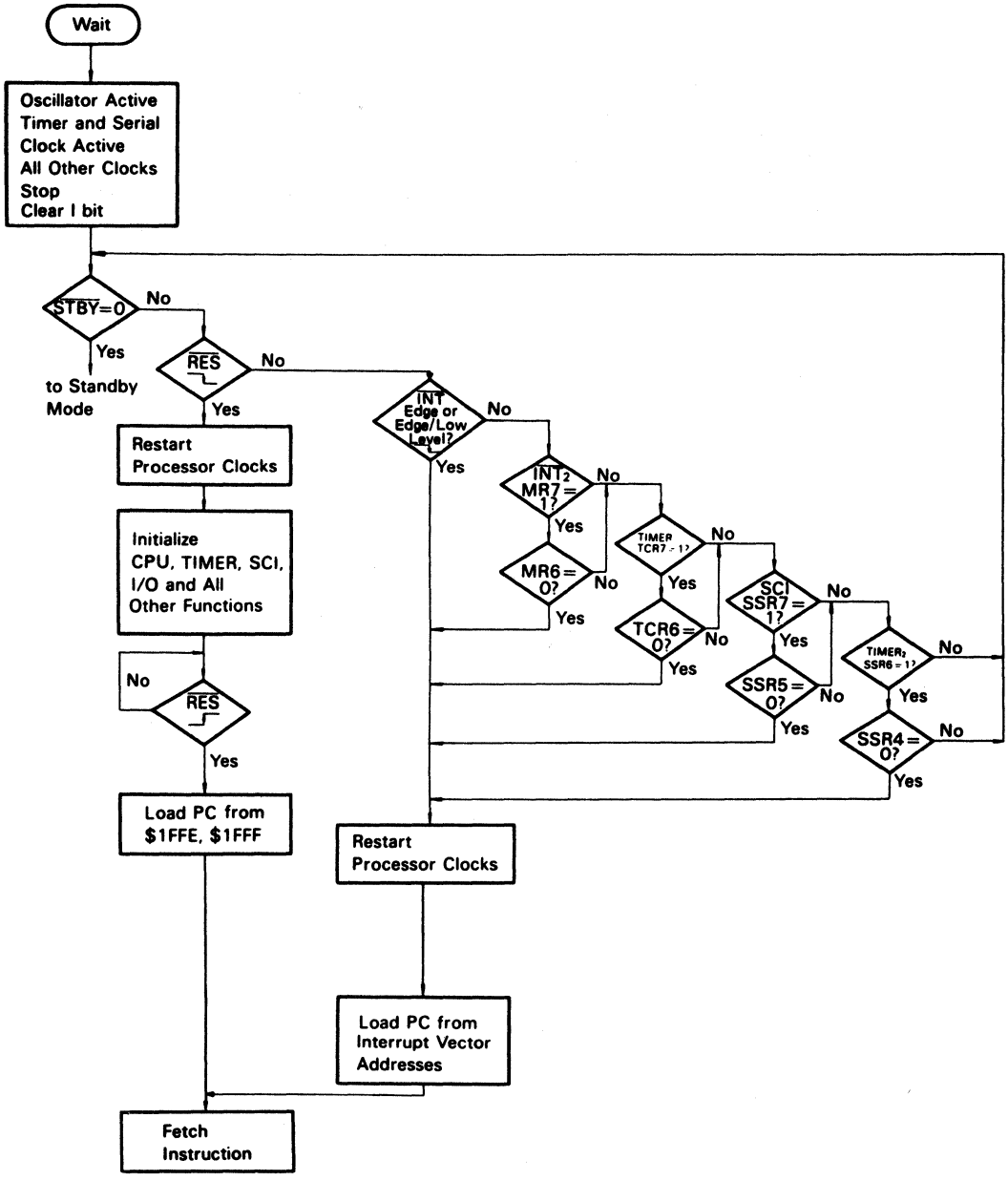


Fig. 2-17 Wait Mode Flowchart

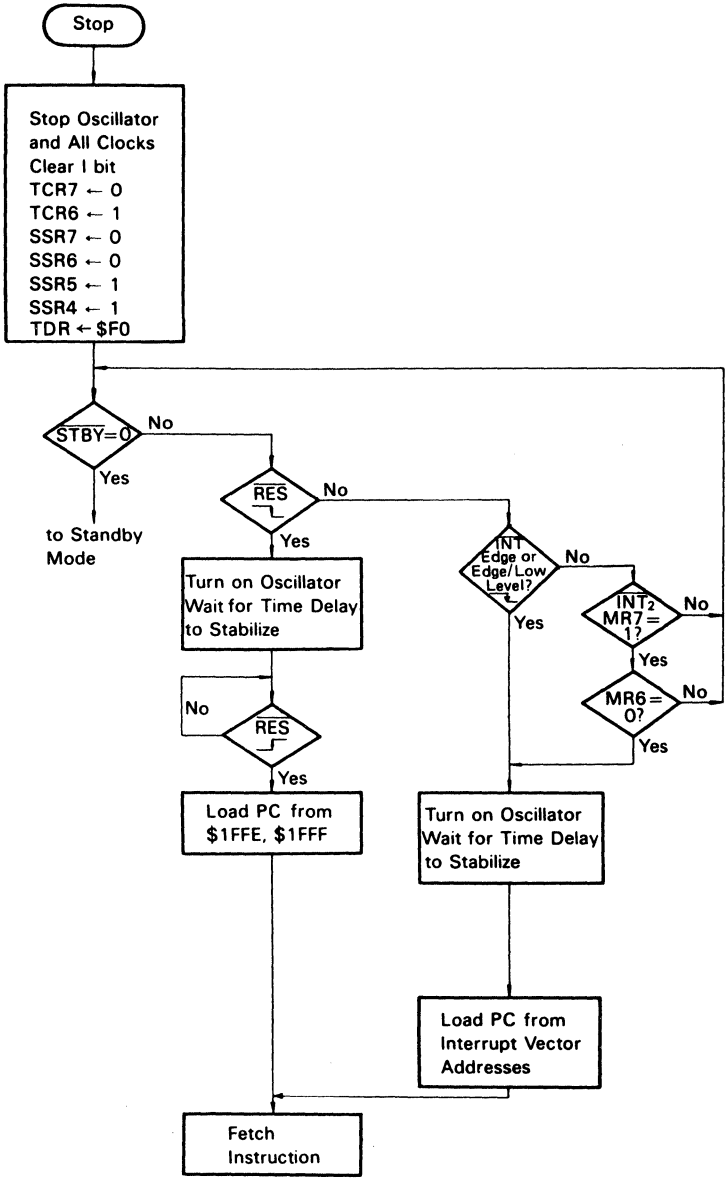


Fig. 2-18 Stop Mode Flowchart

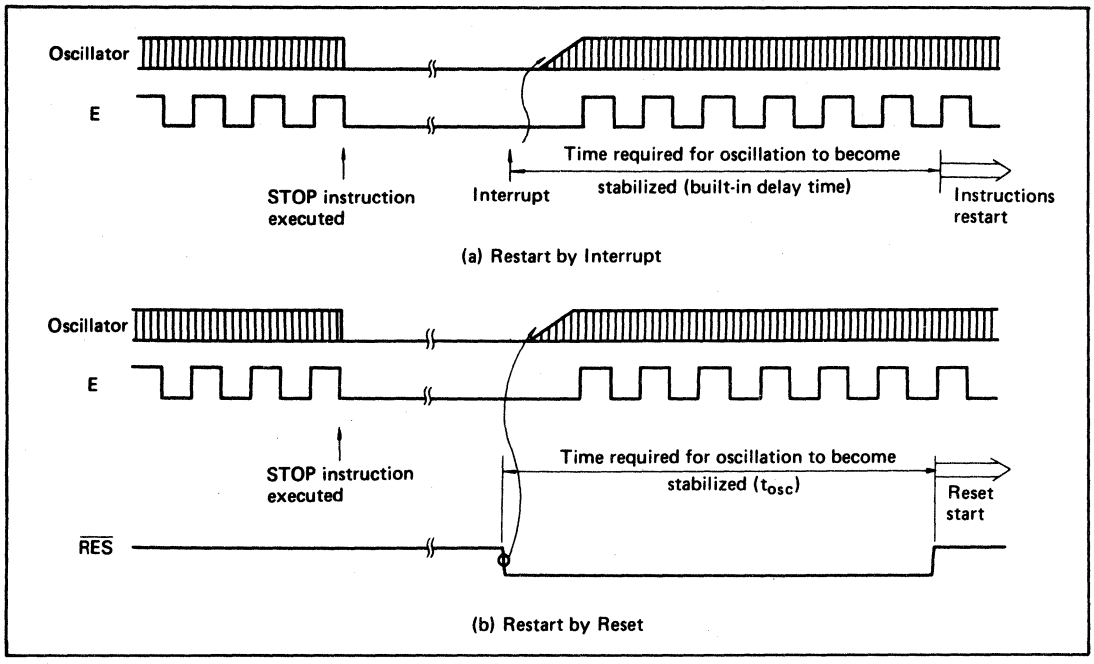


Fig. 2-19 Timing Chart of Releasing from Stop Mode

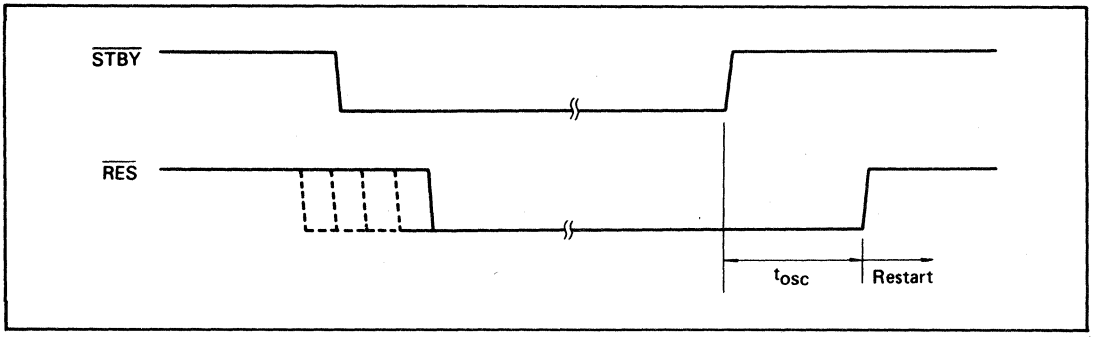


Fig. 2-20 Timing Chart of Releasing from Standby Mode

Table 2-5 Status of Each Part of MCU in Low Power Dissipation Modes

Mode	Start		Condition						Escape
			Oscillator	CPU	Timer, Serial	Register*	RAM	I/O terminal	
WAIT	Software	WAIT instruction	Active	Stop	Active	Keep	Keep	Keep	$\overline{STBY}$, $\overline{RES}$, $\overline{INT}$, $\overline{INT_2}$, each interrupt request of TIMER, $TIMER_2$, SCI
STOP		STOP instruction	Stop	Stop	Stop	Keep**	Keep	Keep	$\overline{STBY}$, $\overline{RES}$, $\overline{INT}$, $\overline{INT_2}$
Standby	Hardware	$\overline{STBY}$="Low"	Stop	Stop	Stop	Reset	Keep	High impedance	$\overline{STBY}$="High"

* Internal Register (except I-bit of Condition Code Register).
** Refer to Fig. 2-20.

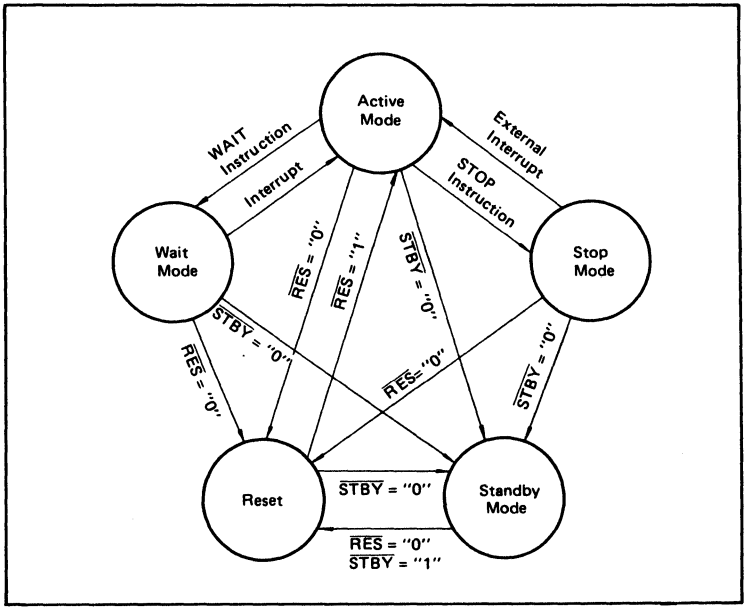


Fig. 2-21 Transitions among Active Mode Wait Mode, Stop Mode, Standby Mode and Reset

2.10 EPROM Mode (HD63705V0)

(1) Mode Selection

The HD63705V0 has two operation modes; MCU mode and EPROM mode. Each mode can be determined according to the NUM and $\overline{STBY}$ states as Table 2-6 shows.

(a) MCU Mode

All ports are available in this mode. (Refer to Fig. 2-22).

Table 2-6 Mode Selection

Mode	NUM	$\overline{STBY}$	EPROM	RAN	Interrupt Vector	Operation Mode
MCU Mode	"L"	*	I	I	I	Single-chip mode
EPROM Mode	"H"	"L"	I	*	*	EPROM programming mode

"L" = logic "0", "H" = logic "1", I; Internal *; Don't care

(b) EPROM Mode

EPROM can be programmed in this mode. Refer to "Internal EPROM Programming" for details.

(c) Overview of Mode and Port

Table 2-7 shows MCU signals in each mode.

Table 2-7 Overview of Mode and Port

Mode	MCU Mode	EPROM Mode
Port A	I/O port	Data bus ($EO_0 \sim EO_7$)
Port B	I/O port	Data Address Input EA_8 (B_7), EA_{10}, EA_{11}(B_5, B_4) (Note 1)
Port C	I/O port	Address Input ($EA_0 \sim EA_7$)
Port D	I/O port	$\overline{OE}$ (D_1), $\overline{CE}$ (D_2) (Note 2)
$\overline{INT}$	Input	Address Input (EA_9)
TIMER	Input	Program Voltage (VPP)

(Note 1) $B_0 \sim B_2$ are not used. Ground B_3 to GND. B_6 is not used.

(Note 2) D_0, $D_3 \sim D_6$ are not used. Ground D_6 to GND.

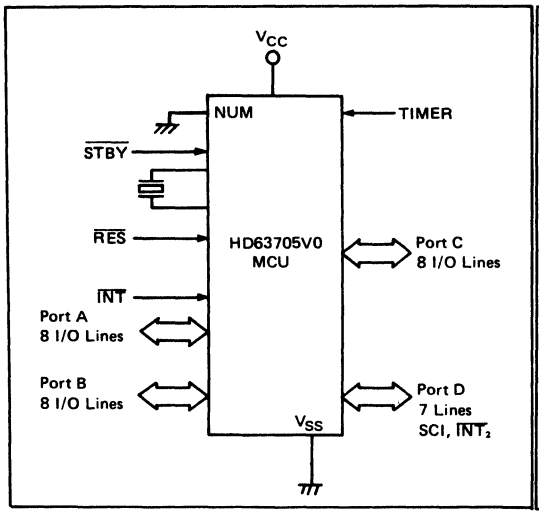


Fig. 2-22 MCU Mode

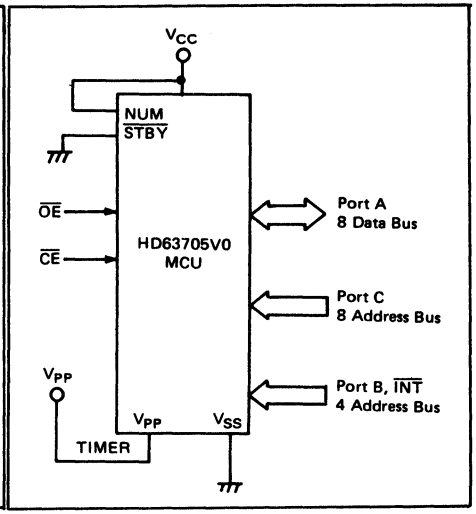


Fig. 2-23 EPROM Mode

(2) Memory Map

Fig. 2-24 shows a memory map in each mode. Internal registers addresses are located in $0000 to $001F as shown in Table 2-8.

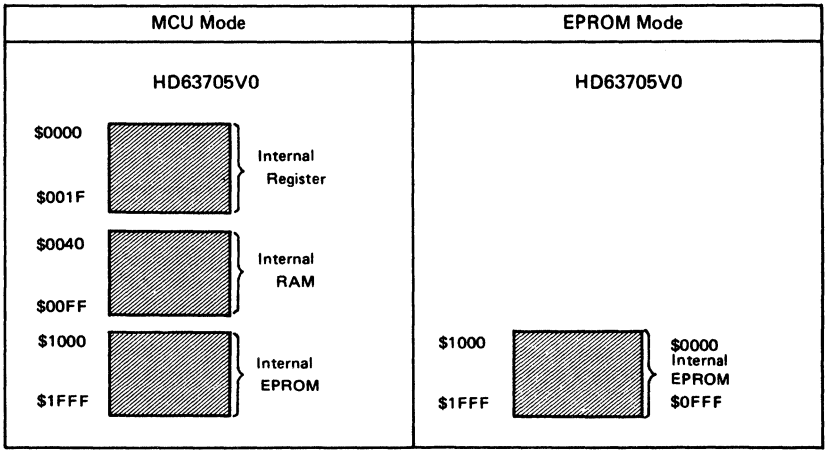


Fig. 2-24 Memory Map

Table 2-8 Internal Register

(R/W) : Read/Write Register
(R) : Read Only Register
(W) : Write Only Register

Register	Address	Read/Write*1/Initial Value after Reset							
		7	6	5	4	3	2	1	0
Port A Data Register	$00	R/W							
		$00							
Port B Data Register	$01	R/W							
		$00							
Port C Data Register	$02	R/W							
		$00							
Port D Data Register	$03	Not Used	R/W						
		1	$00						
Port A Data Direction Register	$04	R/W							
		$00							
Port B Data Direction Register	$05	R/W							
		$00							
Port C Data Direction Register	$06	R/W							
		$00							
Port D Data Direction Register	$07	Not Used	R/W						
		1	0	0	0	0	0	0	0
Timer Data Register	$08	R/W							
		$F0							
Timer Control Register	$09	R/W							
		0	1	0	1	0	0	0	0
Miscellaneous Register	$0A	R/W			Not Used				
		0*2	1	0	1	1	1	1	1
Not Used	$0B ~ $0F								
SCI Control Register	$10	R/W							
		$00							
SCI Status Register	$11	R/W							
		0*2	0*2	1	1	1	1	1	1
SCI Data Register	$12	R/W							
		Undefined							

*1 R: Read Only W: Write Only R/W: Readable/Writeable
*2 Only "0" can be written.

(3) Internal EPROM Programming

When the HD63705V0 enters into EPROM mode, the CPU and internal peripherals stop operations, and the MCU comes to be equivalent to 27256 type EPROM. Then, the 4k bytes of internal EPROM can be programmed in the same way as 27256 type EPROM. The HD63705V0 enters into the EPROM mode by pulling $\overline{STBY}$ "Low" and NUM "High" as shown in Table 2-6. In this mode $A_0 \sim A_7$ are used as Data input/output, $C_0 \sim C_7$ and B_4, B_5 and B_7 are used as address input, D is used as $\overline{OE}$ input, and D_2 is used as $\overline{CE}$ input. Note that the address range must be $0000 through $00FF because the on-chip EPROM is 4k bytes. Fill remainder of EPROM with FFFF for PROM programmer to correctly verify.

(a) Program/Verify

Data is input and output through Port A in EPROM mode. Programming voltage should be applied to TIMER/V_{PP} for EPROM programming and verification.

If $\overline{CE}$ is asserted "Low" and $\overline{OE}$ is pulled "High", EPROM can be programmed through Port A.

If $\overline{OE}$ is asserted "Low" after programming, the program can be verified through Port A.

If both $\overline{CE}$ and $\overline{OE}$ are pulled "High", Port A enters into high-impedance state to inhibit EPROM programming and verification.

Table 2-9 lists terminal state in EPROM mode.

Table 2-9 Pin Conditions in EPROM Mode

MODE	V_{CC}	V_{SS}	TIMER (V_{PP})	$\overline{CE}$	$\overline{OE}$	Port $A_0 \sim A_7$	Port $C0 \sim C7$, B_4, B_5, B_7	NUM	$\overline{STBY}$	Port B_3, D_6
Programming	+6	GND	V_{PP}	"L"	"H"	Data input	Address input	"H"	"L"	GND
Verification	+6	GND	V_{PP}	Don't care	"L"	Data output	Address input	"H"	"L"	GND
Programming/ Verification Disable	+6	GND	V_{PP}	"H"	"H"	High impedance	Don't care	"H"	"L"	GND
Read	+5	GND	+5	"L"	"L"	Data output	Address input	"H"	"L"	GND
Output Disable	+5	GND	+5	"L"	"H"	High impedance	Don't care	"H"	"L"	GND

"H"; V_{IH} level, "L"; V_{IL} level

3. APPLICATION AND PRECAUTIONS

3.1 Watch-Dog Timer

(1) Purpose

A simple method of implementing the watch-dog timer, which is generally known as a means of recovery from a system upset, will be described below.

(2) Basic circuit

Fig. 3-1 shows the basic circuit for recovery from a system upset.

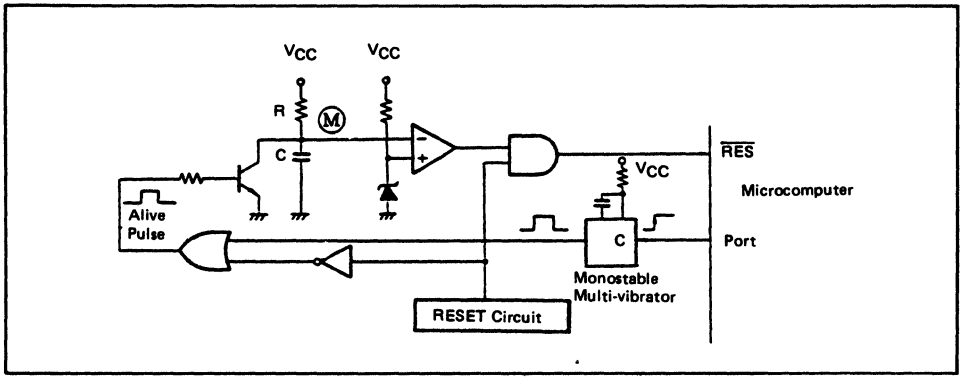


Fig. 3-1 Basic Circuit for Recovery from a System Upset

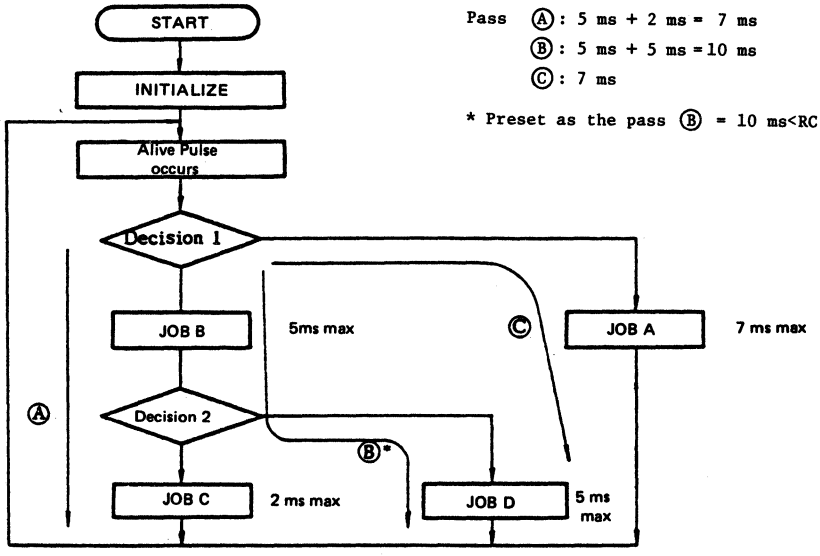
(3) Operation outline

(i) In structuring system software, design the whole system so that a pulse is generated through the port within a predetermined period (T<RC) if the program is normally executed.

(ii) When the system enters into the loop and is deadlocked or when the system upsets the system presents no change in the port output (e.g. it does not generate alive pulse), so that the Ⓜ point potential increases and the microcomputer is reset.

(iii) Preset the RC at above the maximum value among all of the routes that can exist in normal operation. RC is set at a value greater than 10 ms in the example below.

(Example)


START
INITIALIZE
Alive Pulse occurs
Decision 1
JOB B
5ms max
JOB A
7 ms max
Decision 2
JOB C
2 ms max
JOB D
5 ms max
Pass Ⓐ: 5 ms + 2 ms = 7 ms
Ⓑ: 5 ms + 5 ms = 10 ms
Ⓒ: 7 ms
* Preset as the pass Ⓑ = 10 ms<RC

3.2 Auto Reset Circuit

(1) Purpose

The following describes how to implement the reset circuit for power-on reset or auto reset at $V_{CC}<V_{LM}$. (V_{LM}; Threshold level.)

(2) Basic circuit

Fig. 3-2 shows the basic circuit of auto reset.

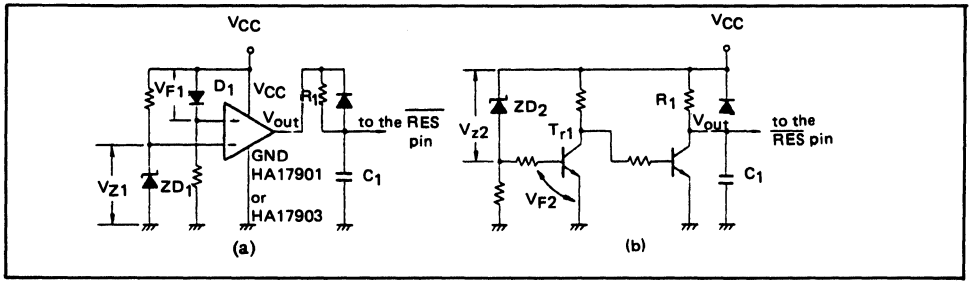
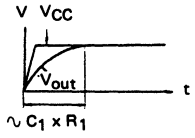


Fig. 3-2 Basic Circuit of Auto Reset

(3) Operation principle

(i) Power-on reset



Since the $\overline{RES}$ pin is pulled up by R_1, the $\overline{RES}$ rising time is delayed by $C_1 \cdot R_1$ during power up, so that the MCU can be reset normally.

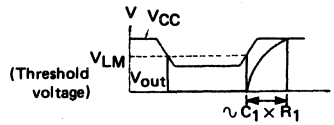
(ii) Auto-reset

The auto-reset operation of the basic circuits (a, b) in Fig. 3-4 is performed as follows.

(a) Since the voltages V_{Z1} and V_{F1}, applied to ZD_1 and D_1, are constant even if V_{CC} changes. The followings describes the $\overline{RES}$ signals.

$$\begin{cases} V_{out} = V_{CC} \text{ when } V_{CC} > V_{Z1} + V_{F1} \\ V_{out} = GND \text{ when } V_{CC} < V_{Z1} + V_{F1} \end{cases}$$

(b) Since the voltage V_{Z2}, applied to ZD_2, and the ON level V_{F2} of Tr_2 are constant, even if the V_{CC} changes. The followings describe the $\overline{RES}$ signals.



$$\begin{cases} V_{out} = V_{CC} \text{ when } V_{CC} > V_{Z2} + V_{F2} \\ V_{out} = GND \text{ when } V_{CC} < V_{Z2} + V_{F2} \end{cases}$$

When the V_{CC} changes, threshold level, V_{LM} is affected by replacing the Zener diodes ZD_1 and ZD_2.

(iii) The Auto-reset function will be more stabilized by feeding back the output signal to the $\overline{RES}$ pin, tuning finely the reference voltage, and providing hysteresis with the V_{LM1} and the V_{LM2}. (V_{LM1} is a voltage during the shifting from operation to resetting, and V_{LM2} is the voltage in recovering.)

3.3 Manual Reset Circuit

(1) Purpose

When the MCU is reset by an external switch, it is necessary to prevent chattering. The following describes a reset circuit in which chattering prevention and power-on reset functions are provided.

(2) Basic circuit

Fig. 3-3 shows the basic circuit of manual reset.

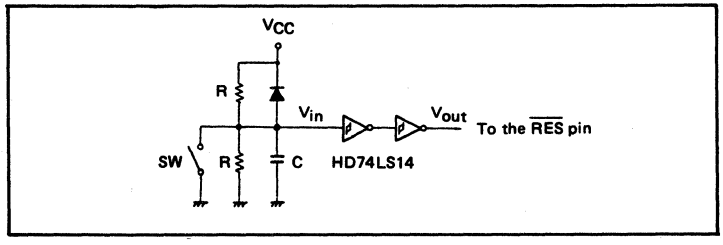
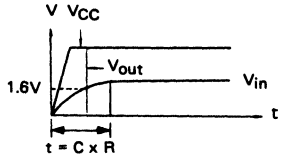


Fig. 3-3 Basic Circuit of Manual Reset

(3) Operation principle

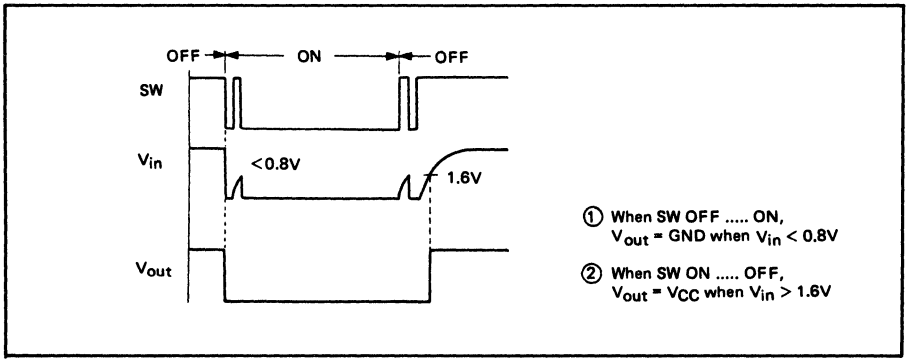
(i) Power-on reset

During power-up, the capacitor C will be charged and the V_{in} rising time will be delayed by the CR charge time, so that power-on would reset normally.



(ii) Manual reset

When SW is on, the V_{out} goes "0" and the MCU is reset. When SW is off, the capacitor C is charged, so that the rising time of V_{in} is delayed by the CR charge time. Chattering is prevented by the capacitor C and the Schmidt trigger HD74LS14.



3.4 A/D Converter Circuit (1) ... High Speed

(1) Purpose

The following describes how to implement a simple A/D conversion (analog-digital conversion) using the resistance radder (R-2R) system.

(2) Basic circuit

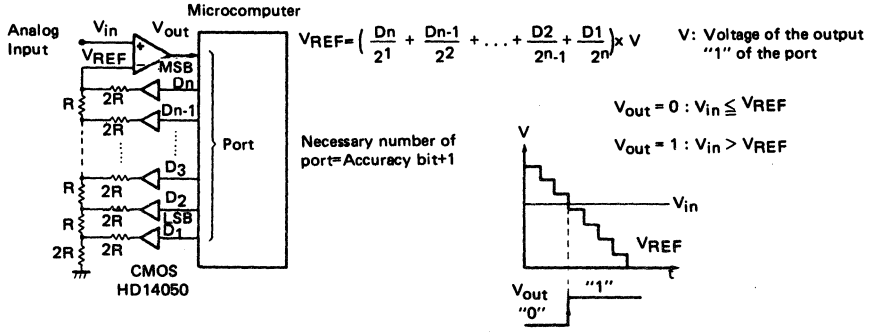


Fig. 3-4 A/D Converter Basic Circuit

Fig. 3-5 Timing Chart

(3) Operation outline

The followings describe the operation outline of 3-bits A/D converter circuit.

(i) The digital outputs (D_3, D_2, D_1) are changed from (1, 1, 1) to (0, 0, 0) at the port. V_{REF} is reduced from 7/8 V to 0 by every 1/8 V according to Table 3-1.

(ii) The output V_{out}, which is the result of comparison between the analog input V_{in} and V_{REF}, is reversed from "0" to "1" when V_{in} is greater than V_{REF} as shown in Fig 3-5.

(iii) The value of (D_3, D_2, D_1), when V_{out} is reversed from "0" to "1", is a digital value to the analog input V_{in}.

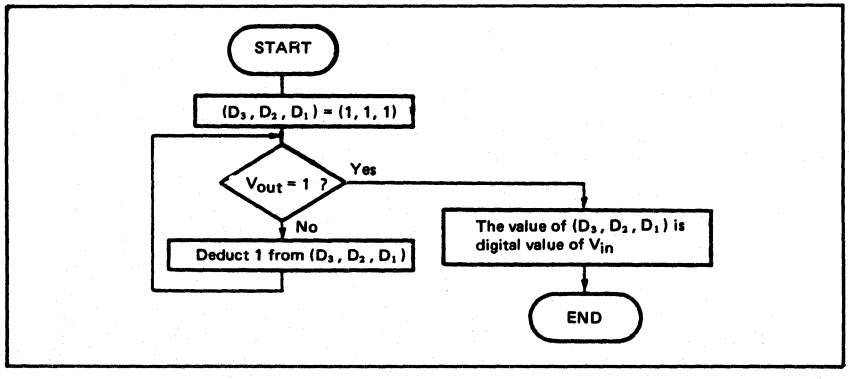
Table 3-1 Value of V_{REF}

D_3	D_2	D_1	V_{REF}
0	0	0	0
0	0	1	1/8 × V
0	1	0	2/8 × V
0	1	1	3/8 × V
1	0	0	4/8 × V
1	0	1	5/8 × V
1	1	0	6/8 × V
1	1	1	7/8 × V

$$V_{REF} = \left(\frac{D_3}{2^1} + \frac{D_2}{2^2} + \frac{D_1}{2^3}\right) \times V$$

D_1 to D_3 are denoted by "1" or "0"

V: Voltage of the port output "1"



3.5 A/D Converter Circuit (2) ... Low Speed

(1) Purpose

The following describes the A/D conversion system utilizing time for charge/discharge to/from the CR circuit. This conversion system is available even if the A/D conversion time is slow.

(2) Basic circuit

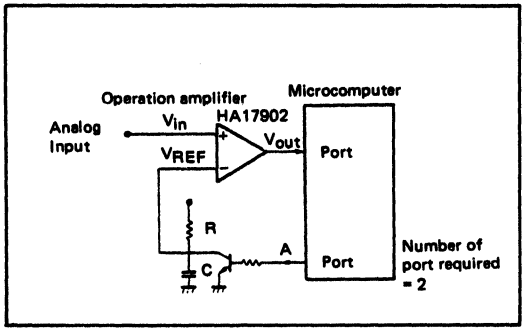


Fig. 3-6 A/D Converter Basic Circuit

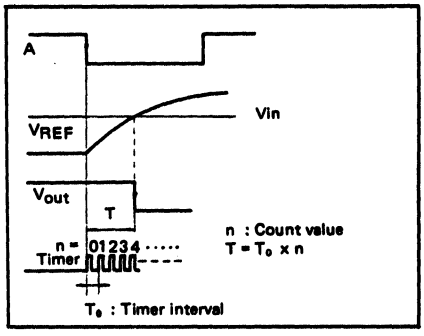


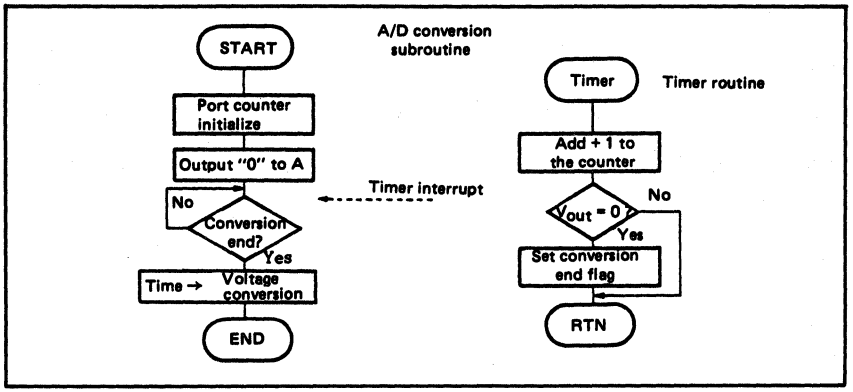
Fig. 3-7 Timing Chart

(3) Operation outline

(i) Set the output terminal A of the port to "1", discharge the external condenser C for a sufficiently long period, decline A from "1" to "0" synchronously with the timer interrupt and charge C.

(ii) Check V_{out} for each timer interrupt and observe the time T in which V_{out} declines from "1" to "0". V_{out} becomes "0" when V_{in} is less than V_{REF}. T is obtained as $T=T_0 \times n$, where "n" represents the number of timer interrupts and T_0 represents the timer interval.

(iii) The obtained time T, which is voltage converted, is a digital value of the analog input V_{in}.



3.6 Precaution;- Board Design of Oscillation Circuit

When connecting crystal and ceramic resonator with the XTAL and EXTAL pins to oscillate, observe the followings in designing the board.

(1) Locate crystal, ceramic resonator, and load capacity C_1 and C_2 as near the LSI as possible. (Induction of noise from outside to the XTAL and EXTAL pins may cause trouble in oscillation.)

(2) Wire the signal lines to the neighbouring XTAL and EXTAL pins as far apart as possible.

(3) Board design of situating signal lines or power supply lines near the oscillator circuit as shown in Fig. 3-9, should not be used because of trouble in oscillation by induction. The resistor between the XTAL and EXTAL, and pins close to them should be 10MΩ or more.

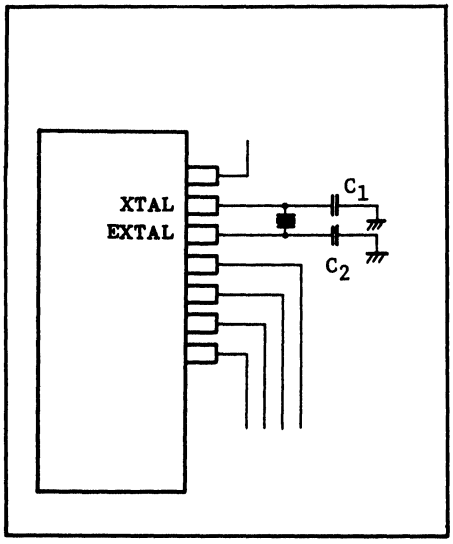


Fig. 3-8 Design of Oscillation Circuit Board

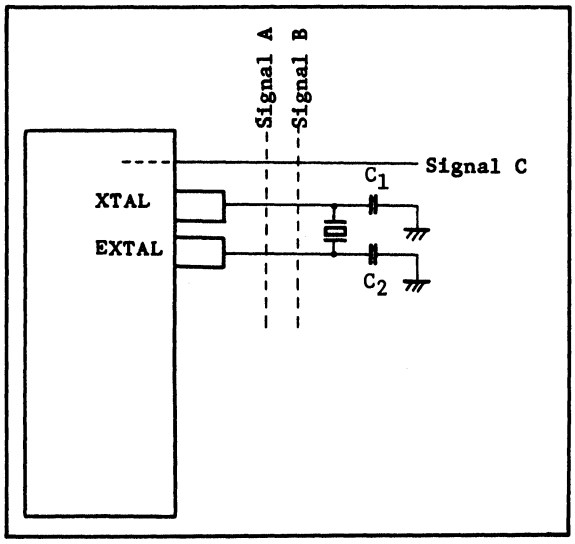


Fig. 3-9 Example of Circuit Causing Trouble in Oscillation

3.7 Precaution;- Sending/Receiving Program of Serial Data

Reading from or Writing into the SCI data register (SDR: $0012) during sending/receiving of serial data may make sending/receiving operation of SCI out of order.

3.8 Precaution;- WAIT/STOP Instructions Program

When I bit of condition code register is "1" and an interrupt ($\overline{INT}$, TIMER/$\overline{INT_2}$, SCI/$TIMER_2$) is held, the MCU does not enter into WAIT mode by executing the WAIT instruction.

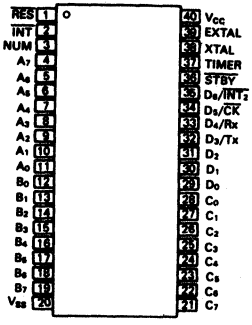
In that case, the MCU executes the corresponding interrupt processing routine after the 4 dummy cycles.

When external interrupts ($\overline{INT}$, $\overline{INT_2}$) are held at the I bit mask, the MCU does not enter into the STOP mode by the STOP instruction execution. The MCU executes the corresponding interrupt processing routine after the 4 dummy cycles, in this case, either.

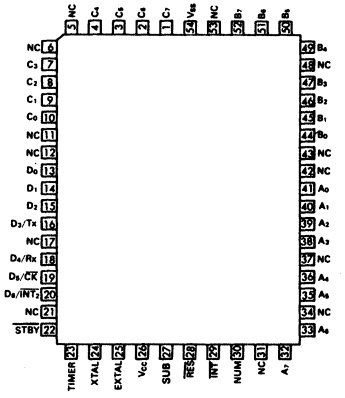
4. PIN ARRANGEMENT AND DIMENSIONAL OUTLINE

(1) HD6305U0, HD6305V0

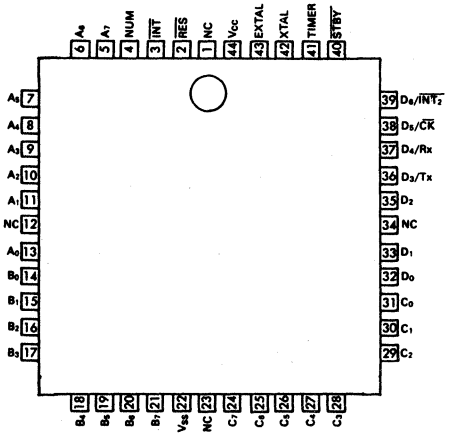
- HD6305U0P, HD6305V0P (DP-40)



- HD6305U0F, HD6305V0F (FP-54)



- HD6305U0CP, HD6305V0CP (CP-44)



- HD63705V0C (DC-40)

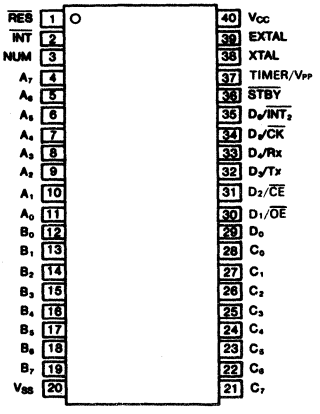
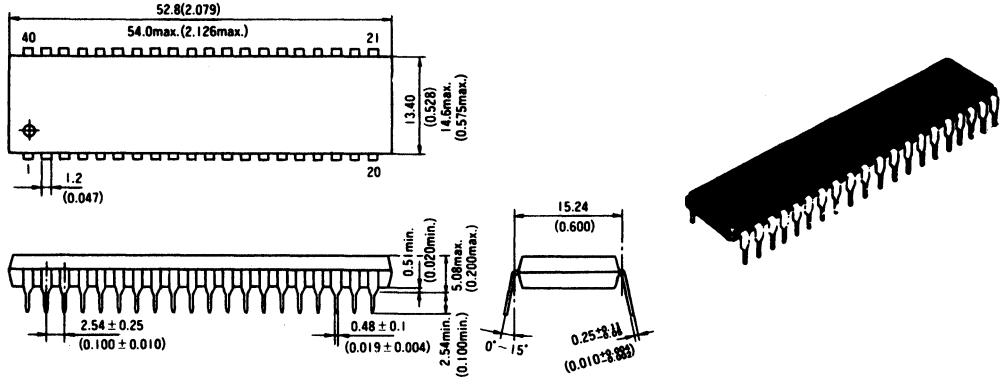
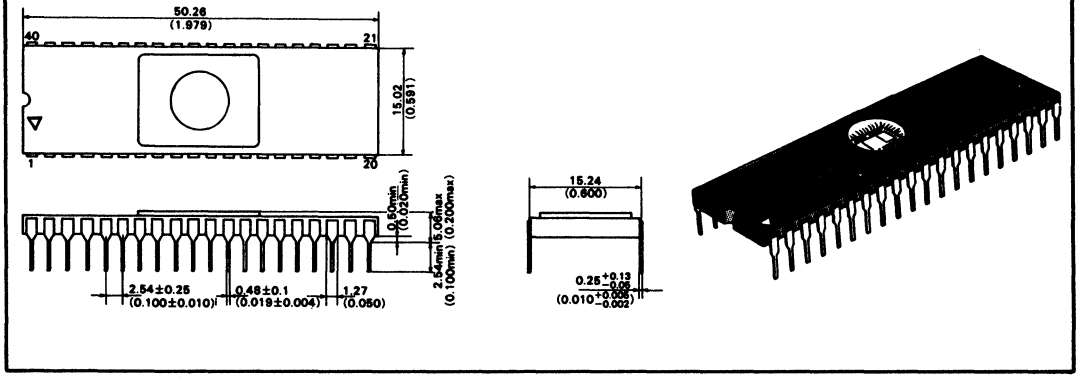


Fig. 4-1 Pin Arrangement (Top View)

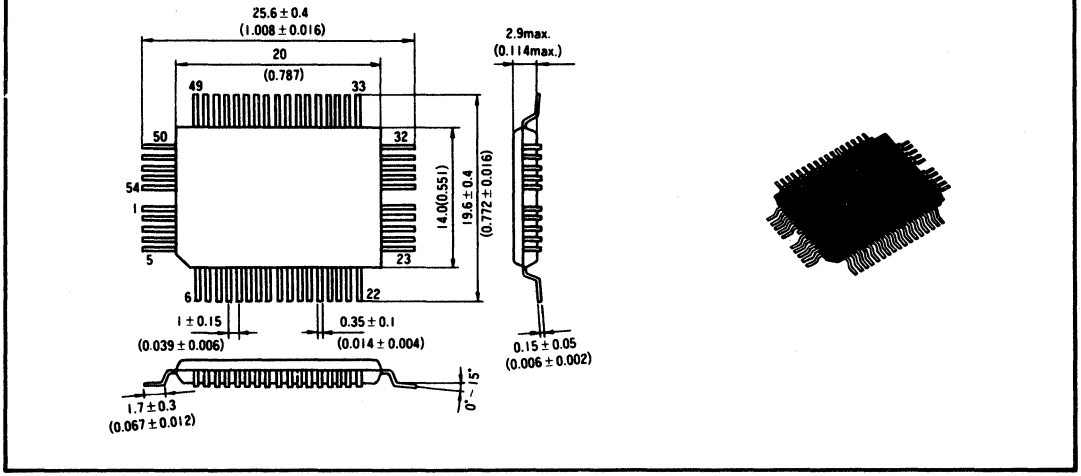
● DP-40



● DC-40



● FP-54



(NOTE) Inch value indicated for your reference

Fig. 4-2 Dimensional Outline

Unit: mm(inch)

● CP-44

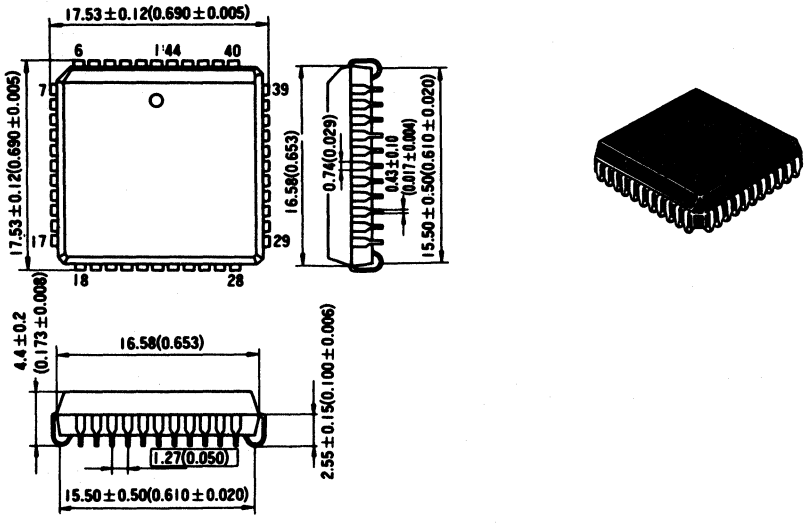


Fig. 4-2 Dimensional Outline

5. ELECTRICAL CHARACTERISTICS

5.1 Electrical Characteristics for HD6305U0, HD6305V0

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 ∿ +7.0	V
Input Voltage	V_{in}	-0.3 ∿ V_{CC} + 0.3	V
Operating Voltage	T_{opr}	0 ∿ +70	°C
Storage Voltage	T_{stg}	-55 ∿ +150	°C

(Note)

These devices contain circuits to protect the inputs against high static voltages or high electric fields. Be careful not to apply any voltage higher than the absolute maximum rating to these high input impedance circuits. For normal operation, we recommend the V_{in} and V_{out} be constrained to the range $V_{SS} \leq (V_{in}$ or $V_{out}) \leq V_{CC}$.

■ MCU MODE ELECTRICAL CHARACTERISTICS

● DC Characteristics (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C unless otherwise specified)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}-0.5	-	V_{CC}+0.3	V
	EXTAL			V_{CC}×0.7	-	V_{CC}+0.3	
	All Others			2.0	-	V_{CC}+0.3	
Input "Low" Voltage	All Inputs	V_{IL}		-0.3	-	0.8	V
Current ** Dissipation	Active	I_{CC}	f=1MHz*	-	5	10	mA
	Wait			-	2	5	mA
	Stop			-	2	10	µA
	Standby			-	2	10	µA
Input Leakage Current	TIMER/V_{PP} $\overline{INT}$, $\overline{STBY}$	$\lvert I_{IL} \rvert$	V_{in}=0.5 ∿ V_{CC}-0.5V	-	-	1	µA
Three State Leakage Current	A_0∿A_7, B_0∿B_7, C_0∿C_7, D_0∿D_6	$\lvert I_{TSI} \rvert$		-	-	1	
Input Capacitance	All Inputs	C_{in}	f=1MHz, V_{in}=0V	-	-	12	pF

* The value at f=x MHz is gain by I_{CC}=(f=xMHz) = I_{CC}(f=1MHz) × X.

** V_{IH} min = V_{CC} -1.0V, V_{IL} max = 0.8V Penetrate current is not included.

- AC Characteristics (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C, unless otherwise specified)

Item	Symbol	Test Condition	HD6305U0/V0			HD63A05U0/V0			HD63B05U0/V0			Unit
			min	typ	max	min	typ	max	min	typ	max	
Clock Frequency	f_{cl}		0.4	-	4	0.4	-	6	0.4	-	8	MHz
Cycle Time	t_{cyc}		1.0	-	10	0.666	-	10	0.5	-	10	µs
$\overline{INT}$ Pulse Width	t_{IWL}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
$\overline{INT_2}$ Pulse Width	t_{IWL2}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
$\overline{RES}$ Pulse Width	t_{RWL}		5	-	-	5	-	-	5	-	-	t_{cyc}
TIMER Pulse Width	t_{TWL}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
Oscillation Start Time (Crystal)	t_{osc}	C_L = 22pF± 20% R_s = 60Ω max	-	-	20	-	-	20	-	-	20	ms
Reset Delay Time	t_{RHL}	external capacitance 2.2 µF	80	-	-	80	-	-	80	-	-	ms

- Port Characteristics (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C unless otherwise specified)

Item		Symbol	Test Condition	min	typ	max	Unit
Output "High" Voltage	Port A, B, C, D	V_{OH}	I_{OH} = -200µA	2.4	-	-	V
			I_{OH} = -10µA	V_{CC}-0.7	-	-	V
Output "Low" Voltage		V_{OL}	I_{OL} = 1.6mA	-	-	0.55	V
Input "High" Voltage	Port A, B, C, D	V_{IH}		2.0	-	V_{CC}+0.3	V
Input "Low" Voltage		V_{IL}		-0.3	-	0.8	V
Input Leakage Current		$\|I_{IL}\|$	V_{in} = 0.5∿V_{CC} -0.5V	-	-	1	µA

● SCI Timing (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C) unless otherwise specified)

Item	Symbol	HD6305U0/V0			HD63A05U0/V0			HD63B05U0/V0			Unit
		min	typ	max	min	typ	max	min	typ	max	
Clock Cycle Time	t_{Scyc}	1	-	32768	0.67	-	21845	0.5	-	16384	µs
Data Output Delay Time	t_{TXD}	-	-	250	-	-	250	-	-	250	ns
Data Set-up Time	t_{SRX}	200	-	-	200	-	-	200	-	-	ns
Data Hold Time	t_{HRX}	100	-	-	100	-	-	100	-	-	ns

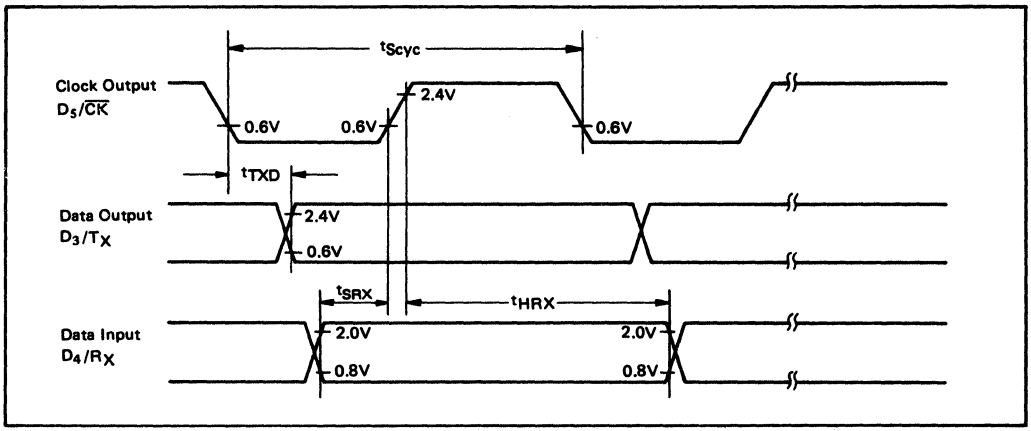


Fig. 5-1 SCI Timing (Internal Clock)

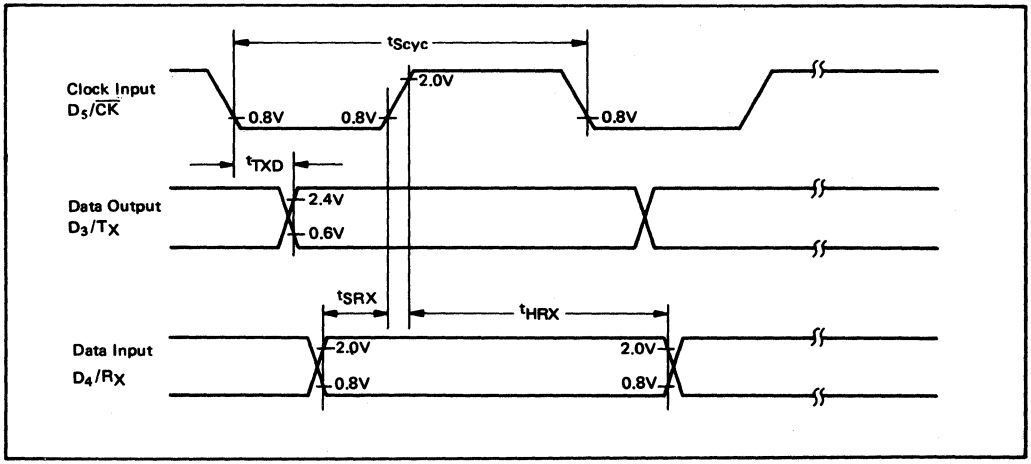


Fig. 5-2 SCI Timing (External Clock)

5.2 Electrical Characteristics for HD63705V0

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit	Note
Supply Voltage	V_{CC}	-0.3∿+7.0	V	
Program Voltage	V_{PP}	-0.3∿+15.0	V	1
Input Voltage	V_{in}	-0.3∿V_{CC}+0.3	V	2
Operating Temperature	T_{opr}	0∿+70	°C	
Storage Temperature	T_{stg}	-55 ∿+125	°C	

Note 1. Applied to TIMER/V_{PP}

Note 2. Applied to all terminals except TIMER/V_{PP}

(Note) These products have a protection circuit in their input terminals against high electrostatic voltage or high electric fields. Notwithstanding, be careful not to apply any voltage higher than the absolute maximum rating to these high input impedance circuits. To assure normal operation, we recommended V_{in}, V_{out}; $V_{SS} \leqq (V_{in}$ or $V_{out}) \leqq V_{CC}$.

■ ELECTRICAL CHARACTERISTICS

● DC Characteristics (V_{CC}=5.0V ± 10%, V_{SS} = GND and T_a = 0 ∿+70°C unless otherwise specified)

Item		Symbol	Test condition	min	typ	max	Unit
Input voltage "High"	$\overline{RES}$,$\overline{STBY}$	V_{IH}		V_{CC}-0.5	-	V_{CC}+0.3	V
	EXTAL			V_{CC}×0.7	-	V_{CC}+0.3	V
	Others			2.0	-	V_{CC}+0.3	V
Input voltage "Low"	All Input	V_{IL}		-0.3	-	0.8	V
*** Current dissipation	Operating	I_{CC}	f=1MHz*	-	5	10	mA
	Wait			-	2	5	mA
	Stop			-	2	10	µA
	Standby			-	2	10	µA
Input leakage current	TIMER/V_{PP}	$\lvert I_{IL} \rvert$	V_{in}=0.5∿ V_{CC} - 0.5V	-	1	100	µA
	$\overline{INT}$,$\overline{STBY}$			-	-	1.0	
Three-state current	A_0 ∿A_7, B_0 ∿B_7, C_0 ∿C_7, D_0 ∿D_6	$\lvert I_{TSI} \rvert$		-	-	1.0	µA
Input capacity	TIMER/V_{PP}	Cin	f=1MHz, V_{in}= 0V	-	-	100	pF
	All terminals except TIMER/V_{PP}			-	-	15	

* The value at f = ×MHz can be calculated by the following equation: I_{CC}(f = ×MHz) = I_{CC}(f = 1MHz) multiplied by ×

** At standby mode

*** V_{IH} min = V_{CC} - 1.0V, V_{IL} max = 0.8V, Penetrate current is not included.

● AC Characteristics (V_{CC} = 5.0V ±10%, V_{SS} = GND and T_a = 0 ∿ +70°C unless otherwise specified)

Item	Symbol	Test condition	HD63705V0			HD637A05V0			HD637B05V0			Unit
			min	typ	max	min	typ	max	min	typ	max	
Clock frequency	f_{cl}		0.4	-	4	0.4	-	6	0.4	-	8	MHz
Cycle time	t_{cyc}		1.0	-	10	0.666	-	10	0.5	-	10	µs
$\overline{INT}$ pulse width	t_{IWL}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
$\overline{INT_2}$ pulse width	t_{IWL2}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
$\overline{RES}$ pulse width	t_{RWL}		5	-	-	5	-	-	5	-	-	t_{cyc}
TIMER pulse width	t_{TWL}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
Oscillation start time (crystal)	t_{OSC}	C_L = 22pF± 20% R_S = 60Ω max	-	-	20	-	-	20	-	-	20	ms
Reset delay time	t_{RHL}	External cap. 2.2µF	80	-	-	80	-	-	80	-	-	ms

■ PROM MODE ELECTRICAL CHARACTERISTICS

● Programming Operation

DC Characteristics (V_{CC} = 6V ± 0.25V, V_{PP} = 12.5V ± 0.3V, T_a = 25°C ± 5°C)

Item	Symbol	Test Condition	min	typ	max	Unit
Input Leakage Current	I_{LI}	V_{in}=6.25V/0.45V	-	-	2	µA
Output Voltage	V_{OL}	I_{OL} = 2.1 mA	-	-	0.45	V
	V_{OH}	I_{OH} = -400 µA	2.4	-	-	V
Power Supply Current (Active)	I_{CC}		-	-	30	mA
Input Voltage	V_{IL}		-0.1	-	0.8	V
	V_{IH}		2.2	-	V_{CC}+0.3	V
Programming Current	I_{PP}	$\overline{CE}$ = V_{IL}	-	-	40	mA

(NOTE) 1. V_{PP} (+12.5V) must be applied after V_{CC} (6V) is settled and must be removed before V_{CC}.

2. Vpp must not exceed +15V. Be careful to prevent overshoot of the V_{PP} when switching to 12.5V.

3. The device must not be inserted into a board with V_{PP} at 12.5V to prevent damage to the reliability of the device.

4. When $\overline{CE}$ = V_{IL}, V_{PP} must not be changed from V_{CC} to 12.5V or from 12.5V to V_{CC}.

- AC Characteristics (V_{CC} = 6V ± 0.25V, V_{PP} = 12.5V ± 0.3V, T_a = 25°C ± 5°C)

Item	Symbol	Test Condition	min	typ	max	Unit
Address Set-up Time	t_{AS}	Fig. 6-3	2	-	-	µs
$\overline{OE}$ Set-up Time	t_{OES}		2	-	-	µs
Data Set-up Time	t_{DS}		2	-	-	µs
Address Hold Time	t_{AH}		0	-	-	µs
Data Hold Time	t_{DH}		2	-	-	µs
Output Disable Delay Time	t_{DF}*		0	-	130	ns
Vpp Set-up Time	t_{VPS}		2	-	-	µs
Program Pulse Width	t_{PW}		0.95	1.0	1.05	ms
V_{CC} Set-up Time	t_{VCS}		2	-	-	µs
$\overline{OE}$ Output Delay Time	t_{OE}		0	-	150	ns
Overprogramming $\overline{CE}$ Pulse Width	t_{OPW}		2.85	-	78.75	ms

* t_{DF} is defined when any lines are not connected to output and the output level can not be referred.

Switching Characteristics

Test Condition;

Input pulse level .... 0.8V to 2.2V

Input rise/fall time.. ≤ 20 ns

I/O timing reference level .... input 1V, 2V
output 0.8V, 2V

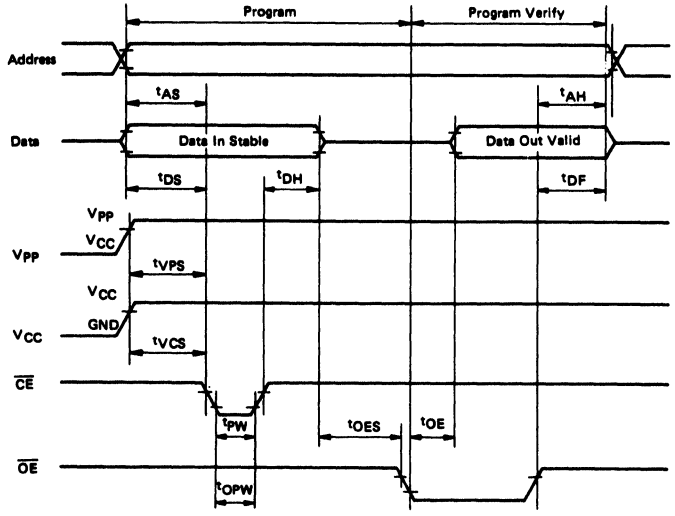


Fig. 5-3 PROM Program/Verify Timing

● Read Operation

DC Characteristics (V_{CC} = 5V ± 10%, V_{PP} = V_{CC} ± 0.6V, T_a = 25°C ± 5°C)

Item	Symbol	Test Condition	min	typ	max	Unit
Input Leakage Current	I_{LI}	V_{CC}= 5.5V, V_{in}= GND ∿ V_{CC}	-	-	1	µA
Output Leakage Current	I_{LO}	V_{CC}= 5.5V, V_{out}=GND ∿ V_{CC}	-	-	1	µA
Programming Current	I_{PP}	V_{PP}=V_{CC}	-	1	100	µA
Power Supply Current (Active)	I_{CC}*	$\overline{CE}$=$\overline{OE}$=V_{IL}	-	-	30	mA
Input Voltage	V_{IL}		-0.3	-	0.8	V
	V_{IH}		2.2	-	V_{CC}+0.3	V
Output Voltage	V_{OL}	I_{OL} = 2.1 mA	-	-	0.40	V
	V_{OH}	I_{OH} = -400 µA	2.4	-	-	V

* Excepts straight current through input.

AC Characteristics (V_{CC} = 5V ± 10%, V_{PP} = V_{CC} ± 0.6V, T_a = 25°C ± 5°C)

Item	Symbol	Test Condition	min	max	Unit
Access Time	t_{ACC}	$\overline{CE}$=$\overline{OE}$=V_{IL}	-	500	ns
$\overline{CE}$ Output Delay Time	t_{CE}	$\overline{OE}$=V_{IL}	-	500	ns
$\overline{OE}$ Output Delay Time	t_{OE}	$\overline{CE}$=V_{IL}	10	150	ns
Output Disable Delay Time	t_{DF}*	$\overline{CE}$=V_{IL}	0	105	ns
Data Output Hold Time	t_{OH}	$\overline{CE}$=$\overline{OE}$=V_{IL}	0	-	ns

* t_{DF} is defined when any lines are not connected to output and the output level can not be referred.

Switching Characteristics

Test Condition;

Input pulse level ..... 0.8 to 2.2V

Input rise/fall time .... ≦20 ns

Output load ..... 1TTL Gate+100 pF

I/O timing reference level ..... input ; 1V, 2V

output; 0.8V, 2V

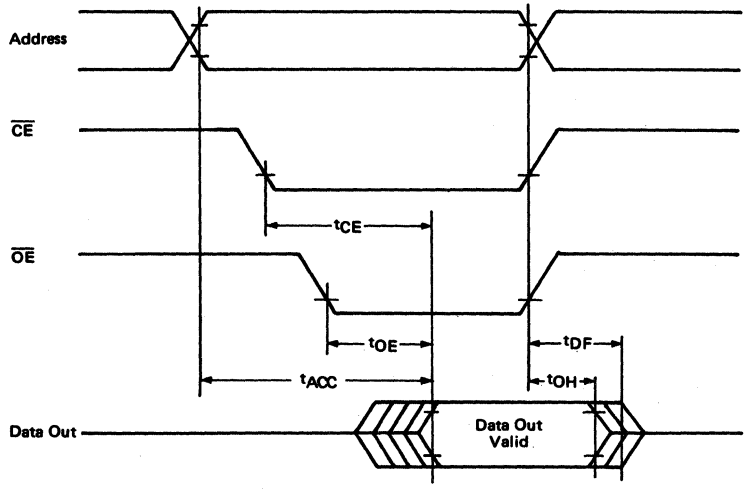


Fig. 5-4 PROM Read Timing

(1) HIGH-SPEED PROGRAMMING

The HD63705V0 provides the high-speed programming method shown in the following flowchart. This method realizes faster programming time without any voltage stress to the device nor deterioration in reliability of programmed data.

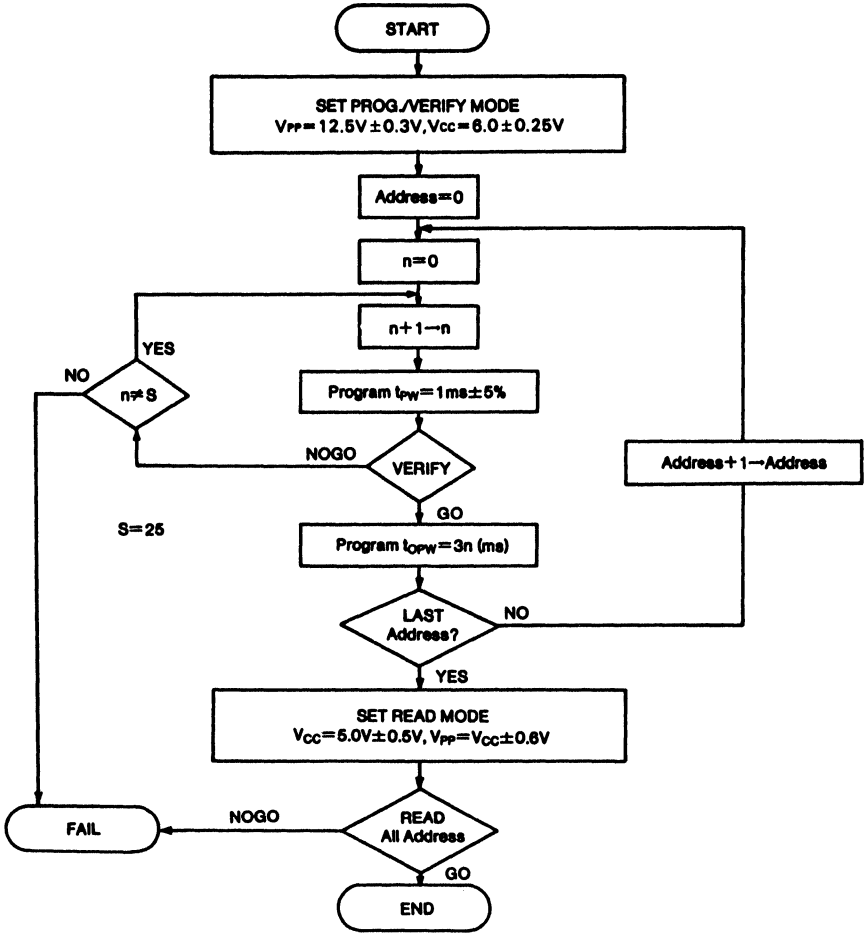
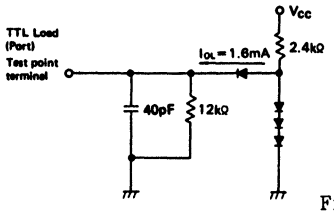


Fig. 5-5 High-Speed Programming Flowchart

Fig. 5-6 shows a port test.



(NOTE)

1. The load capacitance includes stray capacitance caused by the probe, jig, etc.
2. All diodes are 1S2074Ⓗ.

Fig. 5-6 Test Load

6. PROGRAMMABLE ROM

HD63705V0 MCU stops functioning in PROM mode and can be programmed as EPROM equivalent to 27256 type. (Refer to "2.10 EPROM Mode" about PROM).

HD63705V0 MCU can be programmed with commercially available PROM programmer by using socket adapter in order to convert the pin arrangement into that equivalent to single EPROM.

Table 6-1 show the recommended combination of PROM programmer and socket adapter in programming HD63705V0 MCU.

Note that socket adapter selects 24 pins from 40 pins of MCU, then converts from 40-pin socket to 24-pin socket with ordinary EPROM (27256 type).

Table 6-1 PROM Programmer and Socket Adapter

PROM Programmer		Socket Adapter	
Maker	Type No.	Maker	Type No.
DATA I/O	121B	DATA I/O	HD63705V0
	29B	HITACHI	H35VSA01A

6.1 Programming/Verification

When applying programming voltage (V_{PP}) to V_{PP}, asserting $\overline{OE}$ to "High" and $\overline{CE}$ to "Low", HD63705V0 MCU write the data transmitted from Port A by asserting both $\overline{OE}$ and $\overline{CE}$ to "Low". Refer to Fig. 5-3 and 5-4 about data transmit timing.

When both $\overline{OE}$ and $\overline{CE}$ are asserted to "Low", Port A will enter into high impedance. This masks programming/verification of the PROM.

(1) Precaution of the PROM Programming

The PROM memory cell should be programmed with the specified voltage and timing. The higher program voltage V_{PP} or the longer program pulse width t_{PW} is applied, the more will be the quantity of electrons injected to the floating gate. However, a p-n junction will be broken permanently if V_{PP} is applied to more than maximum ratings. Especially V_{PP} overshoot of an EPROM programmer should be checked.

Negative-noise to device pins may cause a parasitic transistor effect and reduce the breakdown voltage.

Users should also be careful of the following, since PROM is electrically connected to PROM programmer through HD63705V0 socket adapter.

① Confirm that socket adapter is fixed to PROM programmer before programming.

② Don't touch the socket adapter and the MCU while programming. If you do, sometimes you cannot program because of malconnection.

6.2 Erasure (MCU with a window)

Internal PROM data can be erased if the memory elements are exposed to ultraviolet light. The erased data is set to "1".

When erasing the data, the recommended conditions are as follows; wavelength: 2537Å, an integrated does of at least: 1.5W•sec/cm)

Exposing the LSI to an ultraviolet lamp of 12000 $\mu W/cm^2$ for about 20 minutes, at a distance of about 1 inch, would be sufficient.

(1) The PROM Programming and Data Retention

HD63705V0's memory cell is the same as an EPROM device, and it is programmed by hot electrons injected to the floating gate with applying high voltage at the control gate and the drain. The electrons have been trapped by the potential barrier at the polysilicon-oxide (SiO_2) by which the floating gate is completely sorrounded. The programmed cell becomes a "0".

The memory cell will be discharged by;

① Exposure to ultraviolet light; discharged by photo emitting electrons (erasure principle).

② Heat; discharged by thermal emitting electrons.

③ Applied with high voltage; discharged by high electric field applied to control gate or drain.

If there is something wrong with the oxidation membrance around the floating gate, the LSI will lose more electrons. However, we usually eliminate these defective products, and a LSI rarely loses electrons in the memory cell.

Memory elements without electrons in the floating gate is usually set to "1".

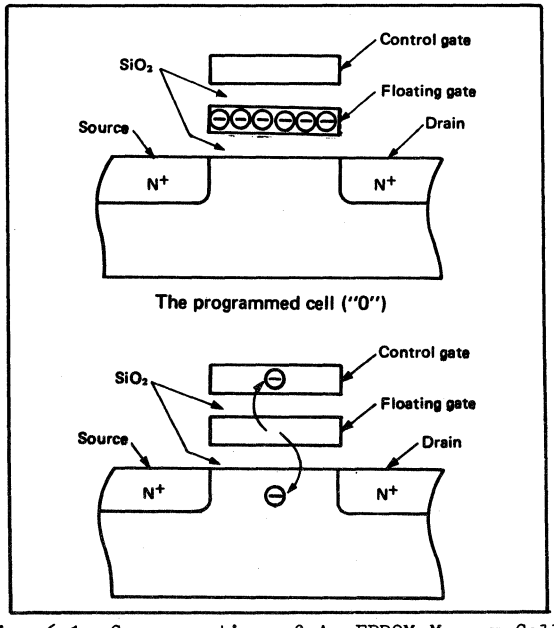


Fig. 6-1 Cross-section of An EPROM Memory Cell

(2) Precaution for using the MCU in the ceramic package with a window

(a) Window

Static charge on the window surface may adversely affect the function of the MCU. The charge will be caused by rubbing the window with plastics or dry cloths, or touching a charged body on it. They can be discharged by exposure to ultraviolet light for a short time. It is recommended to program the memory cell again after exposure, since the electrons trapped at the floating gate will reduce. The methods to prevent static charge on the window are follows.

① Connect the body of an operator to the ground.

② Do not rub the window with plastics or dry cloths.

③ Do not use coolant sprays which contain some ions.

④ Use a conductive opaque label.

(b) Precaution after programming EPROM

The data stored in EPROM may be lost or the MCU may malfunction by photocurrent if the MCU is exposed to strong light like a fluorescent lamp or the sunlight. Therefore, it is recommended to cover the window with an opaque label.

Special labels are commercially available for this purpose. Labels containing metal are effective in that they absorb ultraviolet light.

Users should be careful to the followings in choosing opaque labels.

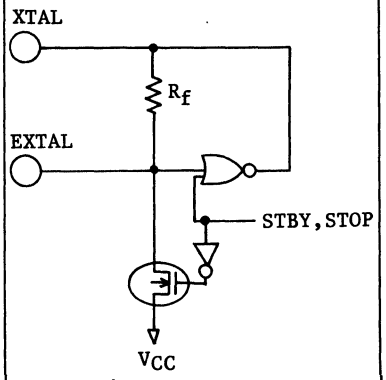
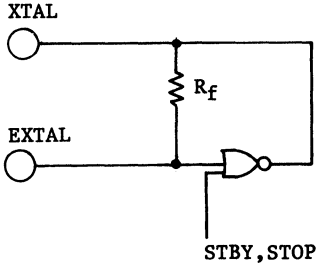
① Adhesion; Opaque label will lose its adhesion in these cases: Static electricity may be charged in removing the opaque label, therefore it is recommended to erase or rewrite the data by ultraviolet light after removing the opaque label.

② Available range of temperature tolerance; The window should be stained with an opaque label within available range of temperature tolerance and operating environment temperature. Otherwise, paste will be harden or cling to the window, causing the label easily to be removed or the paste to remain on the back glass.

③ Humidity; Opaque label covering tolerance of temperature and operating environment condition should be used.

It is difficult to find an opaque label covering all environmental conditions available to MCU. Therefore, users should make a good choice according to the purpose.

Table 6-2 Difference between HD6305V0 and HD63705V0

Item \ Type	HD6305V0	HD63705V0
Operating Voltage	V_{CC} = 3 ∿ 6 V	V_{CC} = 4.5 ∿ 5.5 V
EXTAL (OSC Buffer Circuit at STANDBY or STOP Mode)	 EXTAL is connected to V_{CC} through pull up MOS at STANDBY or STOP Mode.	 There is no pull up MOS. EXTAL is at V_{SS} by R_f at STANDBY or STOP Mode.

7. ROM CODE ORDER METHOD

Users' programs are mask programmed into ROM by Hitachi, and are shipped as LSI. We request users to hand in three EPROMs in which the same contents are written, ordering specifications, and list of the ROM contents.

Relationship between the mask ROM address and the EPROM address is shown in Table 7-1. Program $FF for the unused address data of the EPROM.

Table 7-1 Relationship between the Address of Mask ROM and that of EPROM

Type Name	Address of Mask ROM	Address of EPROM	Remarks
HD6305U0	$1800 ∿ $1FF3 $1FF4 ∿ $1FFF	$1800 ∿ $1FF3 $1FF4 ∿ $1FFF	HN482764 HN27C64 or their equivalent
HD6305V0	$1000 ∿ $1FF3 $1FF4 ∿ $1FFF	$1000 ∿ $1FF3 $1FF4 ∿ $1FFF	HN482764 HN27C64 or their equivalent

APPENDIX

I. DESIGN PROCEDURES AND SUPPORT TOOLS

Cross assembler and hardware emulator, containing various kinds of computers, are available as supporting systems to develop users' programs. Users' programs are mask programmed into ROM and shipped by Hitachi.

Fig. I-1 shows a typical program design procedure and Table I-1 summarizes a set of system development support tool for HD6305U0 and HD6305V0.

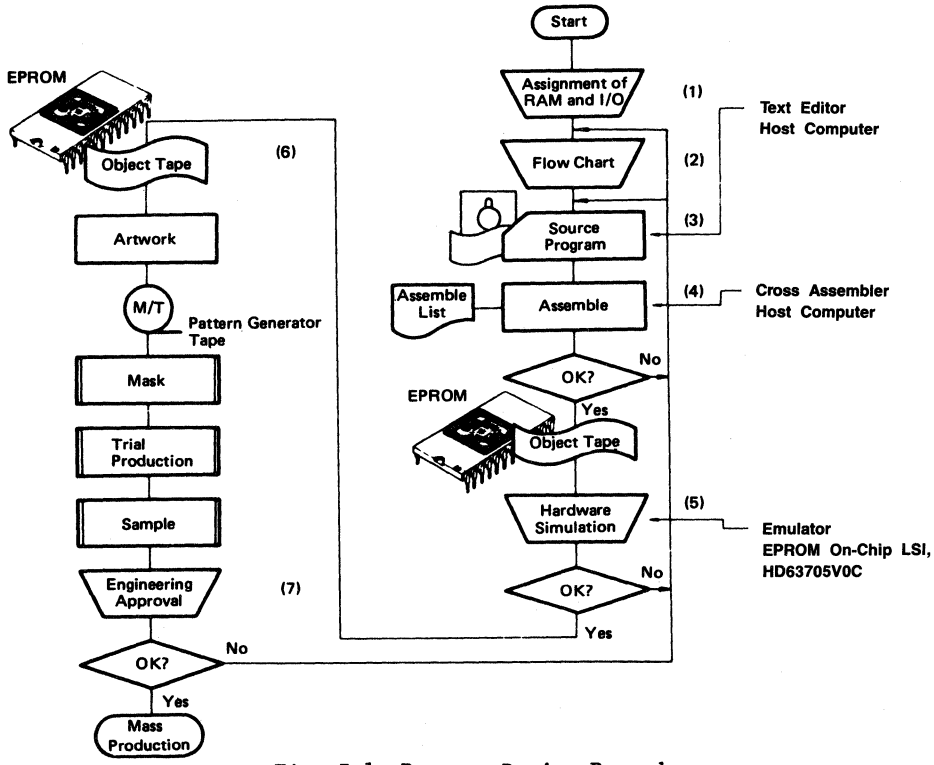


Fig. I-1 Program Design Procedure

The following explains the system development procedure.

1. Specify functional assignment of I/O pins and allocation of RAM area before starting programming.
2. Design a flowchart to implement the functions and encode this flowchart with mnemonic codes.
3. Punch the coded format onto cards or write it on a floppy disk. This set of coded form is a source program.
4. Assemble the source program to generage an object with a cross system. And check errors out here.
5. Verify the program through hardware simulation with emulator or EPROM on-chip microcomputer.
6. Send the completed program in EPROM to Hitachi.
7. After Hitachi received user's specified ROM pattern, Hitachi fabricate sample LSI for users' evaluation for the functions. If a user does not find any problem in the sample LSI, Hitachi will start mass production of the LSI.

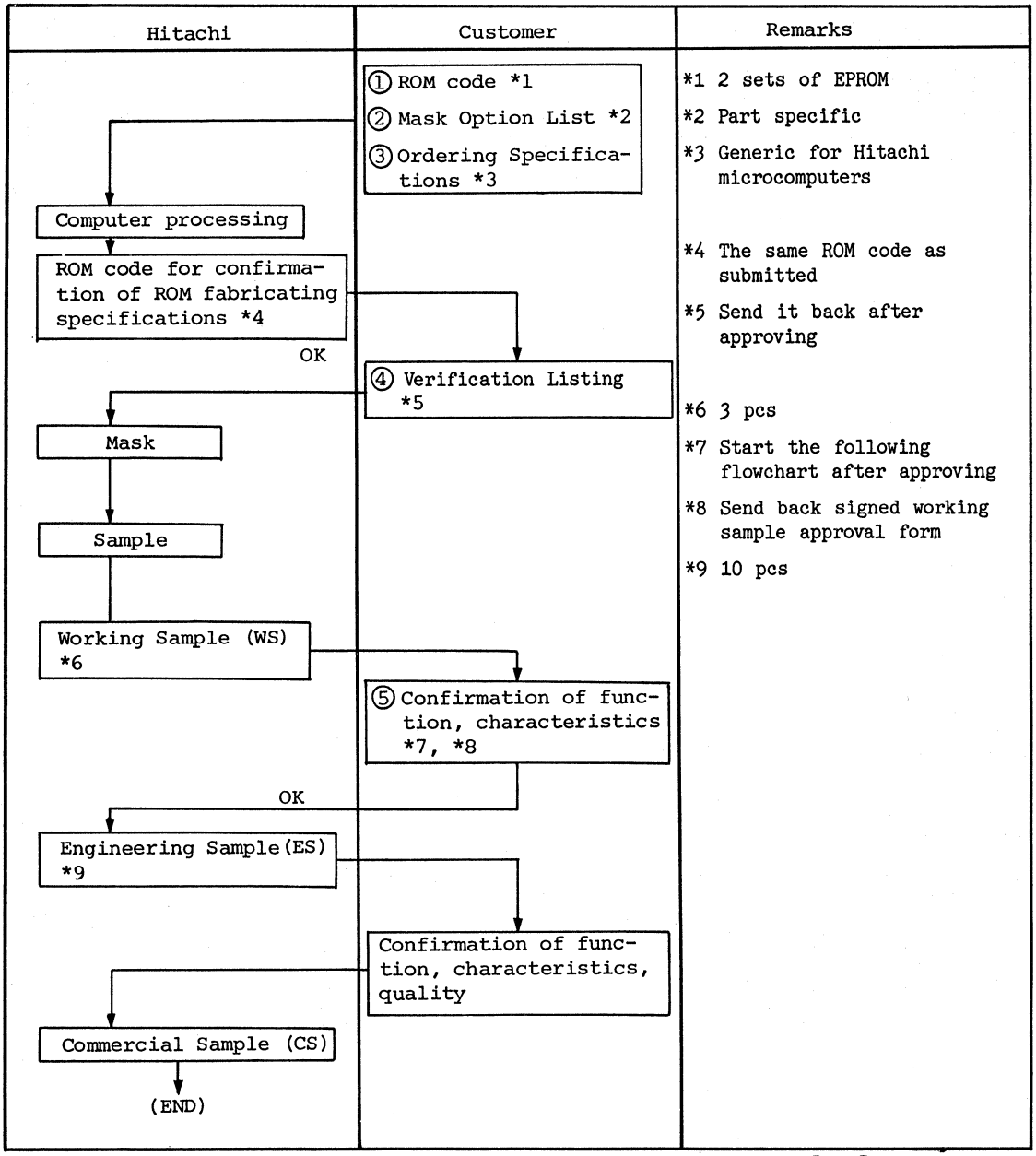
Table I-1 System Development Support Tools

Type No.	Emulator	EPROM On-Chip LSI	IBM PC Cross Assembler
HD6305U0 HD6305V0	H35MIX3 (HS35VEML03H)	HD63705V0C	S35IBMPC

Single Chip Microcomputer ROM Ordering Procedure

(1) Development Flowchart

Single chip microcomputer device is developed according to the following flowchart after program development.



(Note) Please send in ①, ②, and ③ at ROM ordering, and send back ④, ⑤ after approving.

(2) Data you send and precautions

(a) Ordering specifications ----- Common style for all Hitachi single chip microcomputer devices. Please enter as for the followings. The format is shown in the next page.

- Basic ITEM
- Environment Check List
- Check List of attached data
- Customer

(b) ROM code ----- Please send in the ordering ROM code by 2 sets of EPROM the same contents are written. Enter ROM code No. in them. It is desirable to send in program list for easy confirmation of the program contents.

(3) Change of ROM code

Note that if you change the ROM code once sended in or other specification, the ROM must be developed from the beginning. The cost of mask charge should be provided again in this case.

(4) Samples and Mass production

(Working Sample) ---------- Sample for confirmation of the contents of ROM code and that of mask option. Normally 3 samples are sent, but not guaranteed as for reliability. Please evaluate and approve immediately because the following sample making and mass production are set about after obtaining your evaluation.

(Engineering Sample) ------- Sample for evaluating also reliability. 10 pcs are included in mask charge.

(Commercial Sample) -------- Samples for pre-production which maybe purchased separately.

(Mass Product) ------------ Products for actual mass production. Please enter the plan of mass production in full.

HD6305U0, HD6305V0 ORDERING SPECIFICATIONS

(1) GENERAL CHARACTERISTICS (Fill in blank space or appropriate box ☐).

Device Type		Package Outline (See Section) 3,4.1	☐ DP-40 ☐ CP-44 ☐ FP-54
Application (be specific)		Options/Remarks:	
Customer ROM Code ID			
ROM Code Media	☐ EPROM ☐ ZTAT™	Must Specify:	Customer Programmed Start Address _____ Customer Programmed Stop Address _____
Operating Temperature	☐ Standard	☐ J Specification (-40°C to +85°C), if offered	
Remask	☐ Yes ☐ No	Previous Hitachi P/N_____	

(2) OPERATING CHARACTERISTICS

LSI Ambient Temperature	Average	°C	Target Level Of Reliability	☐ 500 Fit ☐ (_____) ☐ 1000 Fit	
	Range	°C- °C			
LSI Ambient Humidity	Average	%	Acceptable Quality Level	☐ 1.0% ☐ 0.4% ☐ 0.65% ☐ (_____)	
	Range	%- %			
Power On Duration	Average	Hours/Day	Remarks:		
Maximum Applied Voltage To LSI	Power Supply	Max V			
	I/O	Max V			

(3) ELECTRICAL CHARACTERISTICS

☐ Purchasing Specifications _____	☐ Hitachi's Standard Specifications Refer To Data Sheet: _____

For Hitachi Use Only

(4) CUSTOMER APPROVAL

Customer Name_____

PO#_____

Accepted By (print) _____

Accepted By (signature)_____

Date _____

(5) ROM CODE VERIFICATION

LSI Type No.	
Shipping Date of ROM To Customer	
Approved Date of ROM From Customer	

Section Four

• HD6305X
• HD6305Y
• HD63P05Y
User's Manual

Section 4
HD6305X, HD6305Y, HD63P05Y User's Manual
Table of Contents

4

1. OVERVIEW

1.1 Features

The HD6305X0, HD6305X1, HD6305X2 MCU are CMOS 8-bit single chip Microcomputers containing CPU, Clock Oscillator, ROM (HD6305X2 does not contain Internal ROM), RAM, I/O port, Timer and Serial Communication Interface (SCI) on a single chip. The HD6305X1 and HD6305X2 MCU configurations are equivalent to the HD6305X0 MCU configuration except the number of I/O terminals.

The HD6305X0 and HD6305X1 MCU contain 4096 bytes of Internal ROM. The HD6305X1 MCU expands its memory up to 12k-bytes externally. The HD6305X2 MCU, without Internal ROM, expands its memory space up to 16k bytes.

Features of these Microcomputers are as follows.

- Upward Compatible with the HD6805 Family Instructions
- 8-bit Architecture
- 4096-Bytes of Internal ROM (HD6305X0)
- 4096-Bytes of Internal ROM and 12k-Bytes of External Memory (HD6305X1)
- 16K-Bytes of External Memory (HD6305X2)
- 128-Bytes of RAM
- I/O Terminal × 32, Input Only Terminal × 7, Output Only Terminal × 16 (HD6305X0)

  I/O Terminal × 24, and Input Only Terminal × 7 (HD6305X1, HD6305X2)
- 2 Internal Timers

  An 8-Bit Timer with 7-Bit Prescaler × 1

  A 15-Bit Timer (Multiplexed with SCI Clock Driver) × 1
- Internal Serial Communication Interface
- Interrupts; External Interrupt × 2, Timer Interrupt × 2, SCI Interrupt × 1, Soft Interrupt × 1
- Low Power Dissipation Modes

  Wait Mode; Continues clock oscillation, stops CPU, permits Timer/Serial/Interrupt to function

  Stop Mode; Stops clock, holds RAM data I/O Status, and Register contents

  Standby Mode; Stops clock, holds RAM data, resets internal options.

- Minimum Instruction Cycle Time
  HD6305X0/X1/X2 .... 1 µs (f= 1MHz)
  HD63A05X0/X1/X2 ... 0.67 µs (f= 1.5MHz)
  HD63B05X0/X1/X2 ... 0.5 µs (f= 2MHz)
- Wide Operation Range
  V_{CC} = 3 ∿ 6V (f=0.1 ∿ 0.5MHz)
  HD6305X0/X1/X2 f=0.1 ∿ 1MHz (V_{CC}=5V±10%)
  HD63A05X0/X1/X2 f=0.1 ∿ 1.5MHz (V_{CC}=5V±10%)
  HD63B05X0/X1/X2 f=0.1 ∿ 2MHz (V_{CC}=5V±10%)
- Sufficient Supporting System by Evaluation Kit
  3 Additional Instructions ... DAA, WAIT, STOP

The HD6305Y0, HD6305Y1, HD6305Y2 MCU are CMOS 8-bit single chip Microcomputers containing CPU, Clock Oscillator, ROM (HD6305Y2 does not contain Internal ROM), RAM, I/O port, Timer, and Serial Communication Interface (SCI) on a single chip. The HD6305Y1 and HD6305Y2 MCU configurations are equivalent to the HD6305Y0 MCU configuration except the number of I/O terminals.

The HD6305Y0 and HD6305Y1 MCU contain 7872 bytes of Internal ROM. The HD6305Y1 MCU expands its memory space up to 8k-bytes externally. The HD6305Y2 MCU, without Internal ROM, expands its memory space up to 16k bytes externally.

Features of these Microcomputers are as follows.

- Upward Compatible with the HD6805 Family Instructions
- 8-bit Architecture
- 7872-Bytes of Internal ROM (HD6305Y0)
- 7872-Bytes of Internal ROM and 8k-Bytes of External Memory (HD6305Y1)
  16k-Bytes of External Memory (HD6305Y2)
- 256-Bytes of RAM
- I/O Terminal × 32, Input Only Terminal × 7, Output Only Terminal × 16 (HD6305Y0)
  I/O Terminal × 24, and Input Only Terminal × 7 (HD6305Y1, HD6305Y2)
- 2 Internal Timers
  An 8-Bits Timer with 7-Bits Prescaler × 1
  A 15-Bits Timer (Multiplexed with SCI Clock Driver) × 1
- Internal Serial Communication Interface

- Interrupts; External Interrupt × 2, Timer Interrupt × 2,
  SCI Interrupt × 1, Soft Interrupt × 1.
- Low Power Dissipation Modes

  Wait Mode; Continues clock oscillation, stops CPU, permits
  Timer/Serial/Interrupt to function.

  Stop Mode; Stops clock, holds RAM data, I/O status, and
  Register contents.

  Standby Mode; Stops clock, holds RAM data, resets internal options.
- Minimum Instruction Cycle Time

  HD6305Y0/Y1/Y2 .... 1 μs (f= 1MHz)

  HD63A05Y0/Y1/Y2 ... 0.67 μs (f= 1.5MHz)

  HD63B05Y0/Y1/Y2 ... 0.5 μs (f= 2MHz)
- Wide Operation Range

  V_{CC} = 3 ∿ 6V (f=0.1 ∿ 0.5MHz)

  HD6305Y0/Y1/Y2 f=0.1 ∿ 1MHz (V_{CC}=5V±10%)

  HD63A05Y0/Y1/Y2 f=0.1 ∿ 1.5MHz (V_{CC}=5V±10%)

  HD63B05Y0/Y1/Y2 f=0.1 ∿ 2MHz (V_{CC}=5V±10%)
- Sufficient Supporting System by Evaluation Kit

  3 Additional Instructions ... DAA, WAIT, STOP

The HD6305 family instruction sets, which are completely compatible, are available for system expansion.

Table 1-1 and 1-2 summarizes the features of each LSI.

Table 1-1 Features of HD6305X, HD6305Y

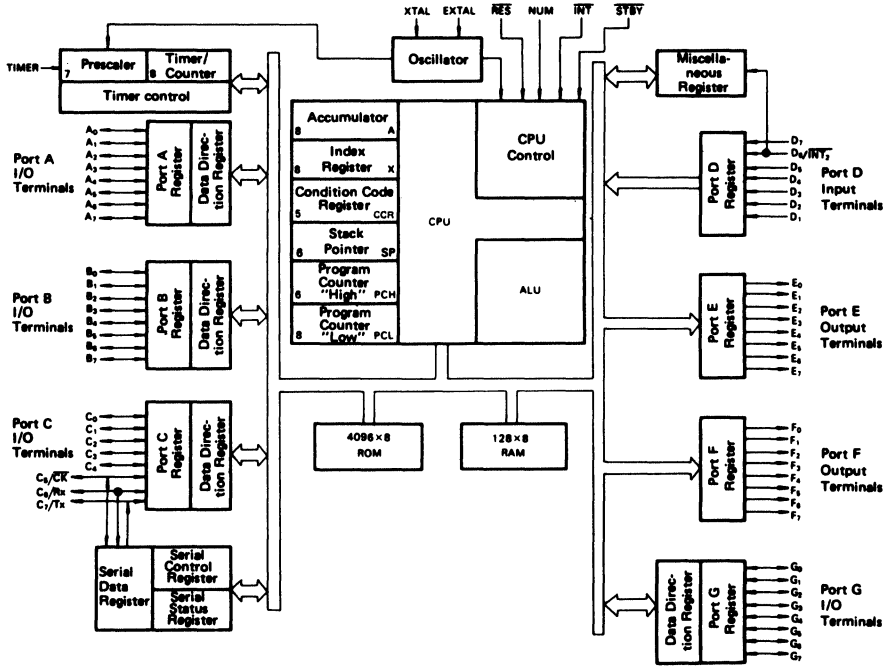
Type No.		HD6305X0 HD63A05X0 HD63B05X0	HD6305X1 HD63A05X1 HD63B05X1	HD6305X2 HD63A05X2 HD63B05X2	HD6305Y0 HD63A05Y0 HD63B05Y0	HD6305Y1 HD63A05Y1 HD63B05Y1	HD6305Y2 HD63A05Y2 HD63B05Y2
Package		DP-64S/FP-64	DP-64S/FP-64	DP-64S/FP-64	DP-64S/FP-64	DP-64S/FP-64	DP-64S/FP-64
Internal Memory	ROM(k bytes)	4	4	-	7.9	7.9	-
	RAM (bytes)	128	128	128	256	256	256
External Memory	(Expanded) (K bytes)	-	12	16	-	8	16
I/O Port	Input Output	55 / 32	31 / 24	31 / 24	55 / 32	31 / 24	31 / 24
	Input	7	7	7	7	7	7
	Output	16	-	-	16	-	-
Interrupt	Nestings	12	12	12	12	12	12
	External Interrupt	2	2	2	2	2	2
	Soft Interrupt	1	1	1	1	1	1
	Timer Interrupt	2	2	2	2	2	2
	Serial Interrupt	1	1	1	1	1	1
Timer	8 bits (with 7 bits prescaler)	1	1	1	1	1	1
	15 bits	1	1	1	1	1	1
Oscillator	Crystal	Possible	Possible	Possible	Possible	Possible	Possible
	Ceramic Oscillator	Possible	Possible	Possible	Possible	Possible	Possible
EPROM on-package type		HD63P05Y0 HD63PA05Y0 HD63PB05Y0	HD63P05Y1 HD63PA05Y1 HD63PB05Y1	—	HD63P05Y0 HD63PA05Y0 HD63PB05Y0	HD63P05Y1 HD63PA05Y1 HD63PB05Y1	—

Table 1-2 Features of EPROM On-Package Type Family

Type No.		HD63P05Y0	HD63PA05Y0	HD63PB05Y0	HD63P05Y1	HD63PA05Y1	HD63PB05Y1
Package		DC-64SP	DC-64SP	DC-64SP	DC-64SP	DC-64SP	DC-64SP
Equivalent Device		HD6305X0 HD6305Y0	HD63A05X0 HD63A05Y0	HD63B05X0 HD63B05Y0	HD6305X1 HD6305Y1	HD63A05X1 HD63A05Y1	HD63B05X1 HD63B05Y1
EPROM	4 k bytes	HN482732A-30	HN482732A-30	HN482732A-25	HN482732A-30	HN482732A-30	HN482732A-25
	8 k bytes	HN482764-3 HN27C64-30	HN482764-3 HN27C64-30	HN482764 HN27C64-25	HN482764-3 HN27C64-30	HN482764-3 HN27C64-30	HN482764 HN27C64-25

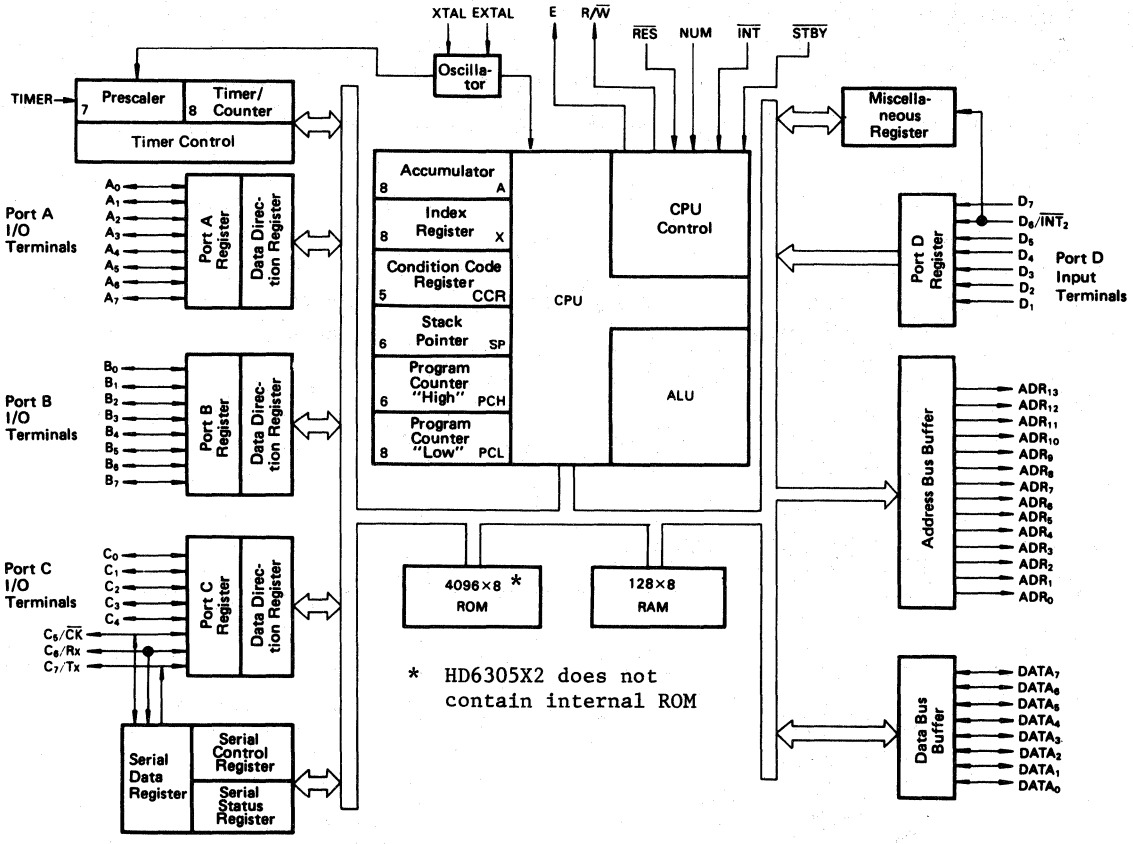
1.2 Block Diagram

Fig. 1-1 gives block diagrams of the HD6305X, HD6305Y, and HD63P05Y MCU. Each terminal function is described in Table 1-3.



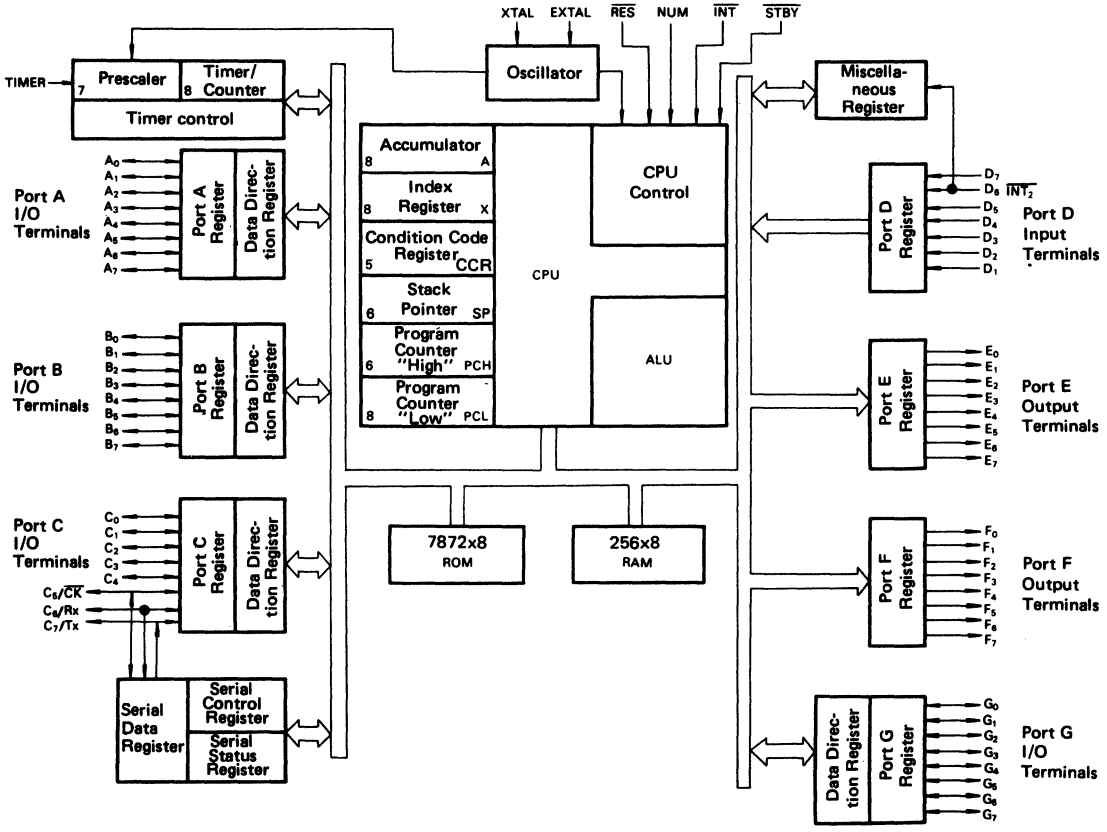
(a) HD6305X0

Fig. 1-1 Block Diagram (to be continued)



(b) HD6305X1, HD6305X2

Fig. 1-1 Block Diagram (to be continued)



(c) HD6305YO

Fig. 1-1 Block Diagram(to be continued)

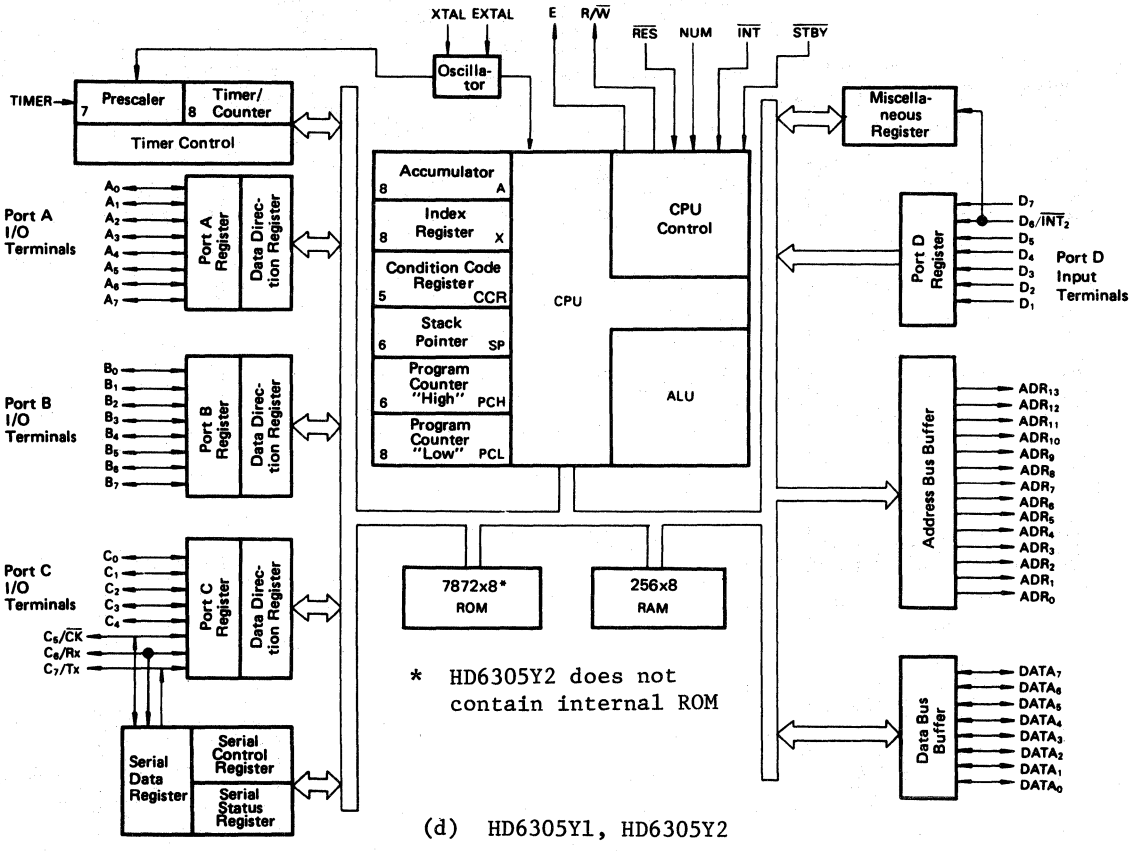
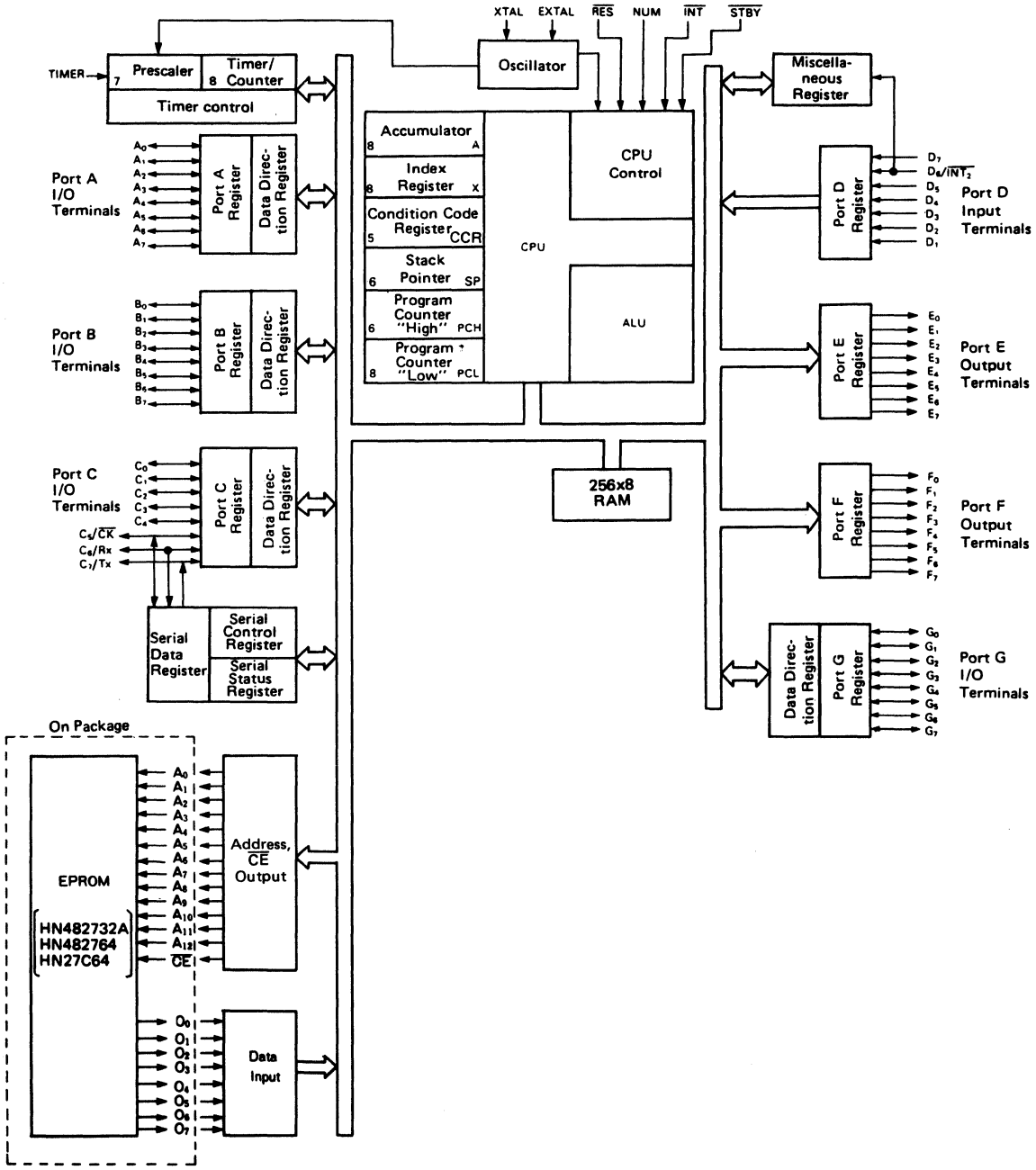
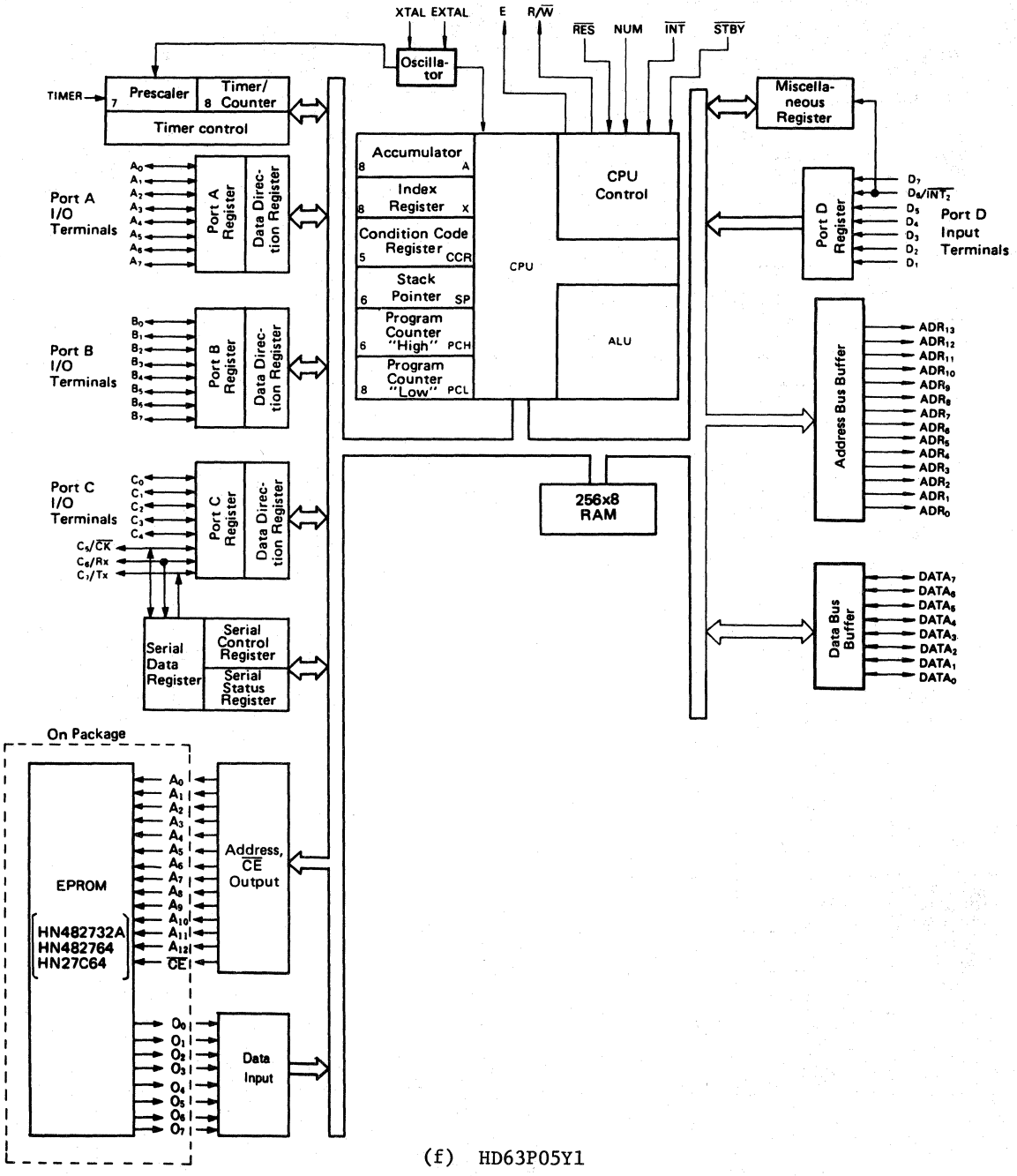


Fig. 1-1 Block Diagram (to be continued)



(e) HD63P05Y0

Fig. 1-1 Block Diagram (to be continued)



(f) HD63P05Y1

Fig. 1-1 Block Diagram

Table 1-3 Terminal Functions

Terminals	Functions	Corresponding Devices
V_{CC}, Vss	Voltage is applied to the MCU through these terminals. V_{CC} provides 5.0V±10% power supply, while Vss is grounded.	HD6305X0/Y0 HD6305X1/Y1 HD6305X2/Y2 HD63P05Y0/Y1
$\overline{INT}$, $\overline{INT_2}$	These terminals function as external interrupt request inputs to the MCU. Refer to "2.7 INTERRUPT" for details. The $\overline{INT_2}$ is multiplexed with the D_6.	
XTAL, EXTAL	These terminals are inputs to the internal clock circuit. Connect a crystal oscillator (AT cut, 2.0∿8.0 MHz) or ceramic filter to these terminals. Refer to "2.6 INTERNAL OSCILLATOR" for further informations.	
TIMER	This external input terminal controls internal timer circuit. Refer to "2.3 TIMER" for details.	
$\overline{RES}$	This terminal resets the MCU. Refer to "2.5 RESET" for details.	
NUM	This terminal is not for user application. Connect the HD6305X1, HD6305Y1 and HD63P05Y1 MCU to V_{CC} through resistance 10kΩ. Ground HD6305X0, HD6305X2, HD6305Y0, HD6305Y2 and HD63P05Y0 MCU to Vss.	
Enable(E)	This terminal transmits E clock. This output is of single phase and TTL compatible, and has a clock frequency which is a quarter of crystal oscillation frequency or external clock frequency. This terminal drives one TTL load and 90pF capacitance.	HD6305X1/Y1 HD6305X2/Y2 HD63P05Y1

(to be continued)

Terminals	Functions	Corresponding Devices
Port A I/O terminals (A_0~A_7) Port B I/O terminals (B_0~B_7) Port C I/O terminals (C_0~C_7) Port G I/O terminals (G_0~G_7)	Each terminal can be individually programmed as an input or as an output by programming the Data Direction Register. Refer to "2.8 I/O Port" for details.	HD6305X0/Y0 HD6305X1/Y1 HD6305X2/Y2 HD63P05Y0/Y1 (HD6305X1/Y1, HD6305X2/Y2, HD63P05Y1 don't contain G Port I/O terminal)
Port D Input terminals (D_1~D_7)	These 7 terminals are TTL/CMOS compatible Input Only terminals. D_6 of Port D is multiplexed with $\overline{INT_2}$. When D_6 functions as a port, set $\overline{INT_2}$ interrupt mask bit of Miscellaneous Register to "1" to mask $\overline{INT_2}$ interrupt.	HD6305X0/Y0 HD6305X1/Y1 HD6305X2/Y2 HD63P05Y0/Y1
Port E Output terminals (E_0~E_7) Port F Output terminals (F_0~F_7)	These 16 terminals are TTL/CMOS compatible Output only terminals.	HD6305X0/Y0 HD63P05Y0
Data bus ($DATA_0$~$DATA_7$)	These 8 I/O terminals are for Data Bus. They drive one TTL load and 90pF capacitance.	HD6305X1/Y1 HD6305X2/Y2 HD63P05Y1
Address bus (ADR_0~ADR_{13})	These 14 output only terminals are for Address Bus. They drive one TTL load and 90pF capacitance.	
$\overline{STBY}$	This terminal permits the MCU to enter into the Standby Mode. When $\overline{STBY}$ goes into "Low" level, the oscillation stops and the MCU enters into the Reset Mode. Refer to "Standby Mode" in "2.9 LOW POWER CONSUMPTION MODE" for details.	HD6305X0/Y0 HD6305X1/Y1 HD6305X2/Y2 HD63P05Y0/Y1
$\overline{CK}$ (C_5)	This terminal transmits or receives SCI clock for SCI operation. Refer to "2.4 SERIAL COMMUNICATION INTERFACE" for details.	
Rx (C_6)	This terminal receives serial data. Refer to "2.4 SERIAL COMMUNICATION INTERFACE" for details.	
Tx (C_7)	This terminal transmits serial data. Refer to "2.4 SERIAL COMMUNICATION INTERFACE" for details.	

2. INTERNAL HARDWARE AND OPERATIONS

2.1 Memory

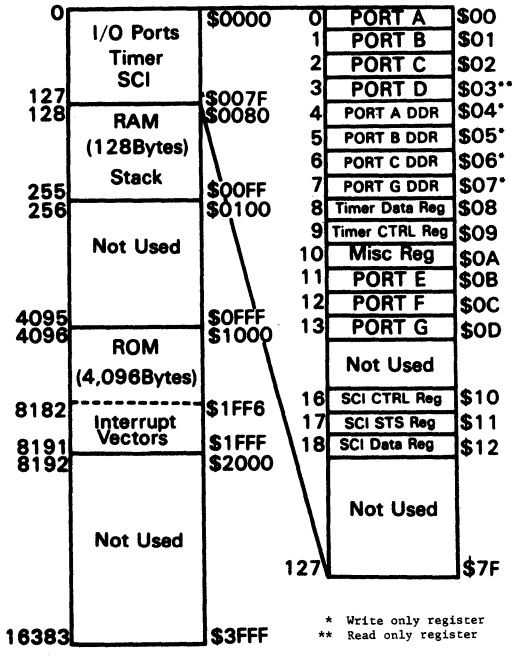
Fig. 2-1 shows the memory map for the HD6305X, HD6305Y, HD63P05Y MCU.

When an interrupt occurs, the CPU register contents are pushed onto the stack in the order as Fig. 2-2 shows, because stack pointer is decreased during pushes.

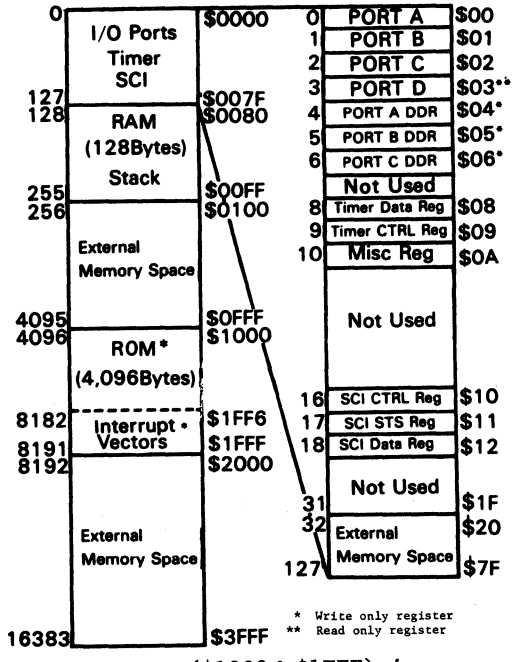
The lower byte (PCL) of the Program Counter, the higher byte (PCH) of the Program Counter, Index Register (IX), Accumulator (A) and Condition Code Register are stacked in this order.

In a subroutine call, only the Program Counter (PCL, PCH) is pushed onto the stack.

Note that the Stack Pointer specifies the next stack area to store data and decreased after stacking 2-byte data. Meanwhile, the Stack Pointer is increased before pulling 1-byte from stack.



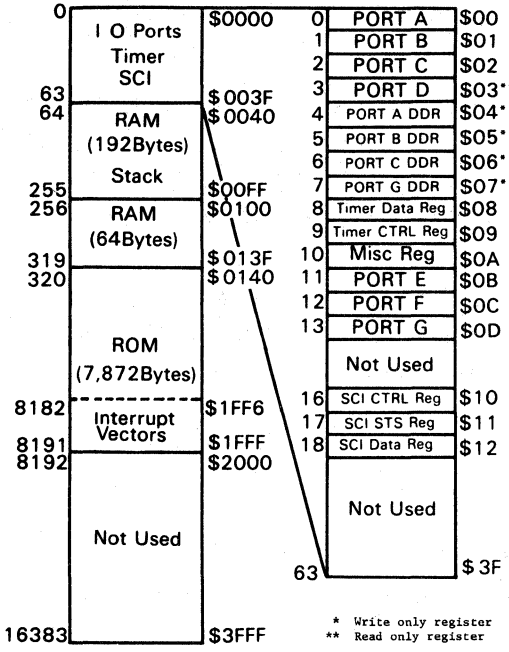
(a) HD6305X0



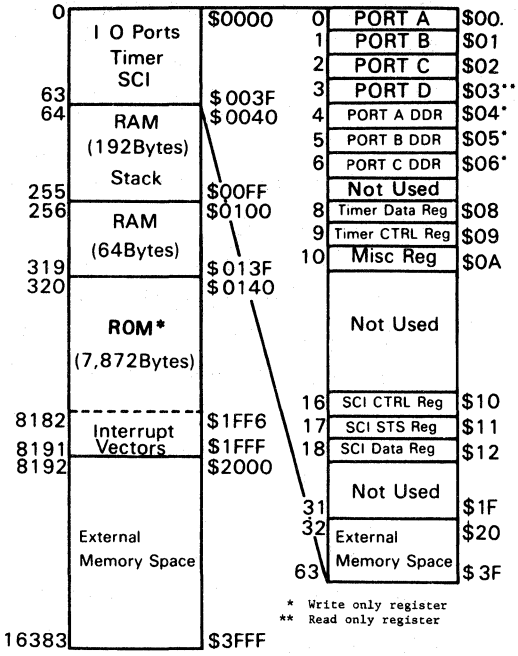
* ROM area ($1000 ∿ $1FFF) in the HD6305X2 changes into External Memory Space.

(b) HD6305X1, HD6305X2

Fig. 2-1 Memory Map

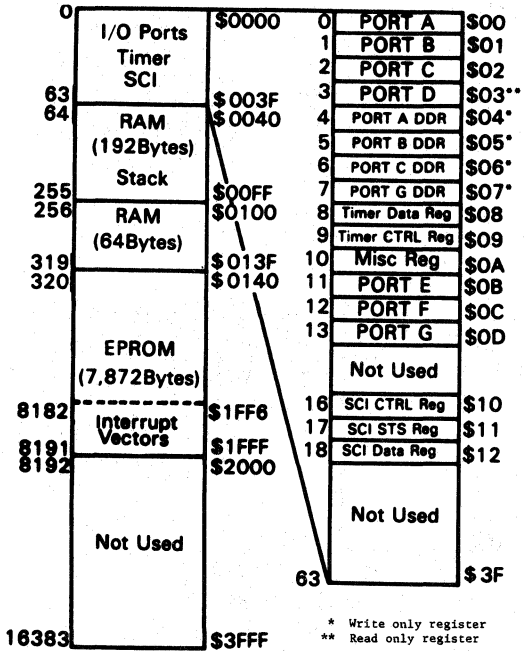


(c) HD6305Y0

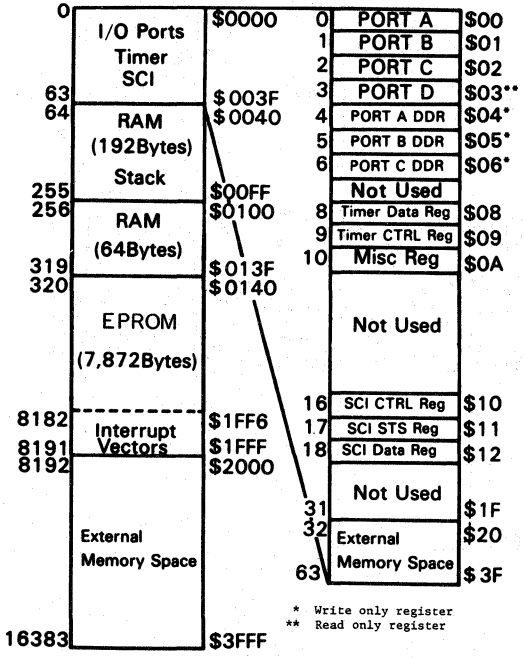


* ROM area ($0140 ∿ $1FFF) in the HD6305Y2 changes into External Memory Space.

(d) HD6305Y1, HD6305Y2



(e) HD63P05Y0



(f) HD63P05Y1

Fig. 2-1 Memory Map

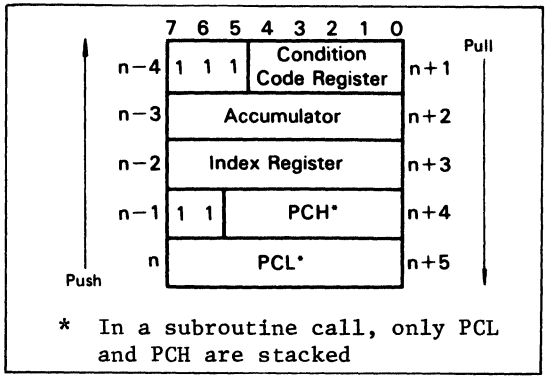


Fig. 2-2 Sequence of Interrupt Stacking

2.2 Registers

CPU has 5 registers available to programmers. Fig. 2-3 shows these.

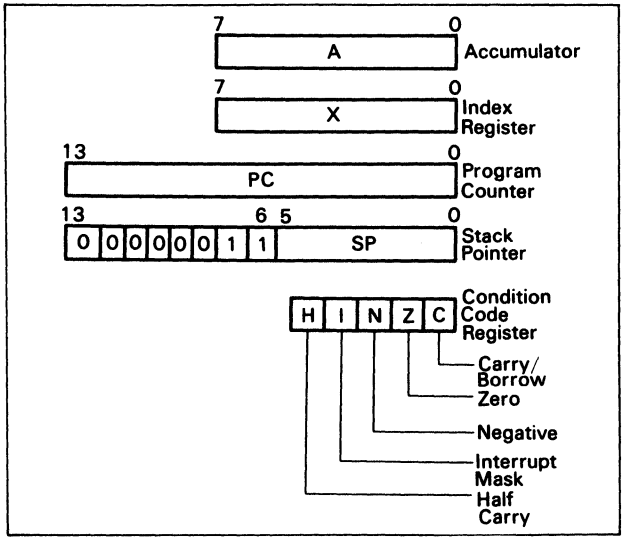


Fig. 2-3 Programming Model

(1) Accumulator (A)

An 8-bit general purpose register to hold operands, and results of arithmetic operations and data processings.

(2) Index Register (X)

An 8-bit register for the Indexed Addressing Mode. An 8-bit address contained in the Index Register will create an effective address if an offset value is added to it.

When executing a read/modify/write instruction, the Index Register is available for holding operands or the result of operation instead of the Accumulator.

The Index Register is also available for storing data temporarily.

(3) Program Counter (PC)

A 14-bit register which contains the address of the next instruction to be executed.

(4) Stack Pointer (SP)

A 14-bit register that indicates the address of the next stack space. Just after reset, the Stack Pointer is initialized to $00FF. It is decreased when data is pushed onto the stack, and is increased when data is pulled from the stack. The 8 most significant bits of the Stack Pointer are fixed to 00000011. During the MCU is reset or executing Reset Stack Pointer (RSP) instruction, the Stack Pointer is set to $00FF. Since subroutines and interrupts are designed to use memory space up to address $00C1 for stacking, programmers can use up to 31 levels for subroutine, and up to 12 levels for interrupts.

(5) Condition Code Register (CCR)

A 5-bit register, each bit indicates the result of the instruction execution just executed. These bits can be individually tested by conditional branch instructions. Condition Code Register bits (H, I, N, Z, C) are described as follows.

Half Carry (H)

Set if a carry occurs between bit 3 and bit 4 during an arithmetic operation (ADD, ADC).

Interrupt (I)

If this bit is set, all of the interrupts, except software interrupts, will be masked.

If an interrupt occurs while this bit is set, the interrupt will be held, and will be processed as soon as this interrupt mask bit is reset.

After executing an instruction following to the CLI instruction, the CPU will enter into the interrupt service routine.

Negative (N)

Set if the result of the arithmetic operation, logical operation or data processing is negative (bit 7 is set to "1"); otherwise reset.

Zero (Z)

Set if the result of the arithmetic operation, logical operation, or data processing is "0".

Carry/Borrow (C)

Set if a carry or borrow occurs in the last arithmetic operation. This bit is affected by Bit Test and Branch Instruction, Shift Instruction, and Rotate Instruction.

2.3 Timer

Fig. 2-4 shows the MCU Timer Block Diagram.

8-bit Timer Data Register (TDR) is loaded by program control. When it receives the clock input, it starts counting down.

When TDR counts down to "0", the timer interrupt request bit (TCR7) in the Timer Control Register (TCR) is set to "1".

In response to the interrupt request, the CPU saves its status into the stack and fetches timer interrupt routine address from addresses $1FF8 and $1FF9 and execute the interrupt routine. The timer interrupt can be masked by setting the timer interrupt mask bit (bit 6) in the timer control register. The mask bit (I) in the condition code register can also mask the timer interrupt.

The source clock to the timer can be either an external signal from the timer input terminal or the internal E signal (the oscillator clock divided by 4). If the E signal is used as the source, the clock input can be gated by the input to the timer input terminal.

Once the timer count has reached 0, it starts counting down with "$FF". The count can be monitored whenever desired by reading the timer data register. This permits the program to know the length of time having passed after the occurrence of a timer interrupt, without disturbing the contents of the counter.

When the MCU is reset, the prescaler and counter are both initialized to logic "1". The timer interrupt request bit (bit 7) is cleared and the timer interrupt mask bit (bit 6) is set.

Write "0" in that bit to clear the timer interrupt request bit (bit 7). (Refer to Table 2-1)

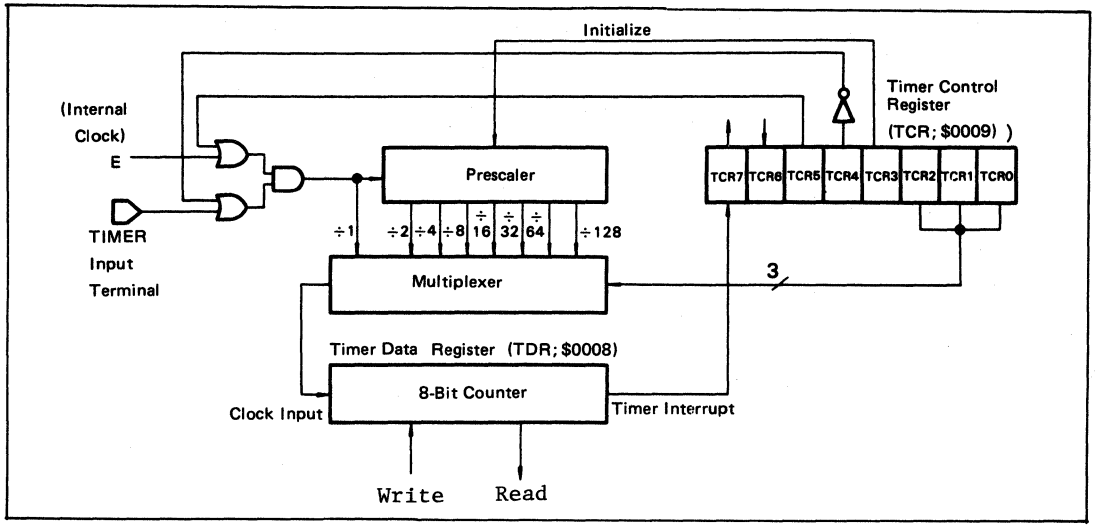


Fig. 2-4 Timer Block Diagram

Table 2-1

TCR7	Timer interrupt request
0	Absent
1	Present

TCR6	Timer interrupt mask
0	Enabled
1	Disabled

(1) Timer Control Register (TCR; $0009)

Selection of a clock source, selection of a prescaler frequency division ratio, and a timer interrupt can be controlled by the timer control register (TCR; $0009).

For the selection of a clock source, any one of the four modes (Refer to Table 2-2) can be selected by bits 5 and 4 of the timer control register (TCR).

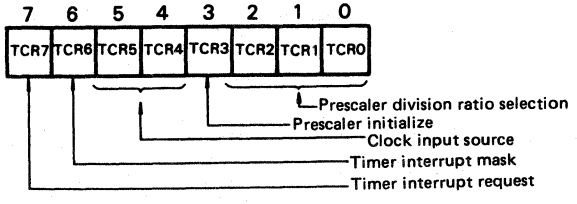


Fig. 2-5 Timer Control Register (TCR; $0009)

After reset, the TCR is initialized to "E under timer terminal control" (bit 5 = 0, bit 4 = 1). If the TIMER is set to "1", the counter starts counting down with "$FF" immediately after reset.

When "1" is written in bit 3, the prescaler is initialized. This bit is always initialized to "0" when read.

A prescaler division ratio is selected by the combination of three bits (bits 0, 1 and 2) of the timer control register (Refer to Table 2-3). There are eight different division ratios: ÷1, ÷2, ÷4, ÷8, ÷16, ÷32, ÷64 and ÷128. After reset, the TCR is set to the ÷1 mode.

A timer interrupt is enabled when the timer interrupt mask bit is "0", and disabled when the bit is "1". When a timer interrupt occurs, "1" is set in the timer interrupt request bit. This bit can be cleared by writing "0" in that bit.

Table 2-2 Clock Source Selection

TCR		Clock input source
Bit 5	Bit 4	
0	0	Internal clock E
0	1	E under TIMER terminal control
1	0	No clock input (counting stopped)
1	1	Event input from TIMER terminal

Table 2-3 Prescaler Division Ratio Selection

TCR			Prescaler division ratio
Bit 2	Bit 1	Bit 0	
0	0	0	÷1
0	0	1	÷2
0	1	0	÷4
0	1	1	÷8
1	0	0	÷16
1	0	1	÷32
1	1	0	÷64
1	1	1	÷128

2.4 Serial Communication Interface (SCI)

The SCI is used to transmit or receive 8-bit data in serial 16 types of transfer rates, in the range from 1 μs to approx. 32 ms in 4MHz operation, are available.

The SCI is consisted of 3 registers, 1 eighth counter, and 1 prescaler. (Refer to Fig. 2-6.)

The SCI communicates with the CPU through data bus, and external I/O devices through bit 3, 4 and 5 of Port C.

The following explains each register function and the SCI operation.

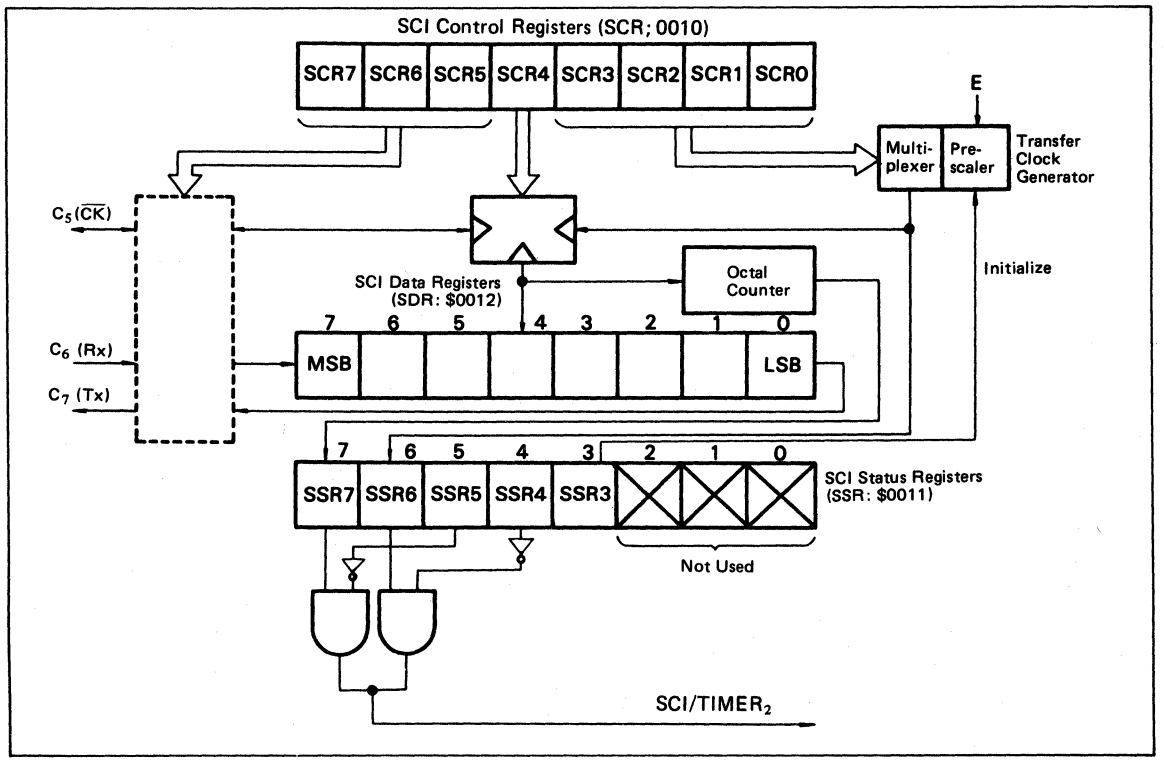


Fig. 2-6 SCI Block Diagram

(1) SCI Control Register (SCR: $0010)

Fig. 2-7 shows SCI Control Register configuration.

Bit 7 (SCR7)

When this bit is set to "1", the DDR corresponding to C_7 is set to "1" and C_7 transmits SCI data. After reset, the bit is initialized to "0".

Bit 6 (SCR6)

When this bit is set to "1", the DDR corresponding to C_6 is set to "0" and C_6 transmits SCI data. After reset, the bit is initialized to "0".

Bits 5 and 4 (SCR5, SCR4)

These bits select a clock source. After reset, the bits are initialized to "0".

Bits 3 ∿ 0 (SCR3 ∿ SCR0)

These bits select a transfer clock rate. After reset, the bits are initialized to "0".

7	6	5	4	3	2	1	0
SCR7	SCR6	SCR5	SCR4	SCR3	SCR2	SCR1	SCR0

Fig. 2-7 SCI Control Register

SCR3	SCR2	SCR1	SCR0	Transfer clock rate	
				4.00 MHz	4.194 MHz
0	0	0	0	1 µs	0.95 µs
0	0	0	1	2 µs	1.91 µs
0	0	1	0	4 µs	3.82 µs
0	0	1	1	8 µs	7.64 µs
∿	∿	∿	∿	∿	∿
1	1	1	1	32768 µs	1/32s

SCR7	C_7 terminal
0	Used as I/O terminal (by DDR).
1	Serial data output (DDR output)

SCR6	C_6 terminal
0	Used as I/O terminal (by DDR).
1	Serial data input (DDR input)

SCR5	SCR4	Clock source	C_5 terminal
0	0	-	Used as I/O terminal (by DDR).
0	1	-	
1	0	Internal	Clock output (DDR output)
1	1	External	Clock input (DDR input)

(2) SCI Data Register (SDR; $0012)

A serial-parallel conversion register that is used for transferring data.

(3) SCI Status Register (SSR; $0011)

Bit 7 (SSR7)

Bit 7 is the SCI interrupt request bit which is set upon completion of transmitting or receiving 8-bit data. It is cleared when reset or data is written to or read from the SCI data register with the SCR5="1". The bit can also be cleared by writing "0" into it.

Bit 6 (SSR6)

Bit 6 is the $TIMER_2$ interrupt request bit. $TIMER_2$ is multiplexed with the serial clock generator, and SSR6 is set each time the internal transfer clock falls. When reset, the bit is cleared. It also be cleared by writing "0" in it. (For details, see $TIMER_2$.)

Bit 5 (SSR5)

Bit 5 is the SCI interrupt mask bit which can be set or cleared by software. When this bit is set to "1", the SCI interrupt (SSR7) is masked. When reset, it is set to "1".

Bit 4 (SSR4)

Bit 4 is the $TIMER_2$ interrupt mask bit which can be set or cleared by software. When the bit is set to "1", the $TIMER_2$ interrupt (SSR6) is masked. When reset, it is set to "1".

Bit 3 (SSR3)

When "1" is written into this bit, the prescaler of the transfer clock generator is initialized. When read, the bit is always "0".

Bits 2 ∿ 0

Not used.

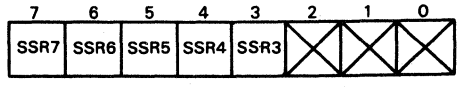


Fig. 2-8 SCI Status Register (SSR; $0011)

SSR7	SCI interrupt request
0	Absent
1	Present

SSR6	$TIMER_2$ interrupt request
0	Absent
1	Present

SSR5	SCI interrupt mask
0	Enabled
1	Disabled

SSR4	$TIMER_2$ interrupt mask
0	Enabled
1	Disabled

(4) Data Transmission

By writing the desired control bits into the SCI control registers, a transfer rate and a source of transfer clock are determined and bits 7 and 5 of port C are set at the serial data output terminal and the serial clock terminal, respectively. The transmit data should be stored from the accumulator or index register into the SCI data register. The data written in the SCI data register is transmitted from the C_7/Tx terminal, starting with the LSB, synchronously with the falling edge of the serial clock. (Refer to Fig. 2-9.) When 8 bits of data have been transmitted, the interrupt request bit is set in bit 7 of the SCI status register with the rising edge of the last serial clock. This request can be masked by setting bit 5 of the SCI status register. Once the data has been transmitted, the 8th bit data (MSB) stays at the C_7/Tx terminal. If an external clock source has been selected, the transfer rate determined by bits 0 ∿ 3 of the SCI control register is ignored, and the $C_5/\overline{CK}$ terminal is set as input. If the internal clock has been selected, the $C_5/\overline{CK}$ terminal is set as output and clocks are transmitted at the transfer rate selected by bits 0 ∿ 3 of the SCI control register.

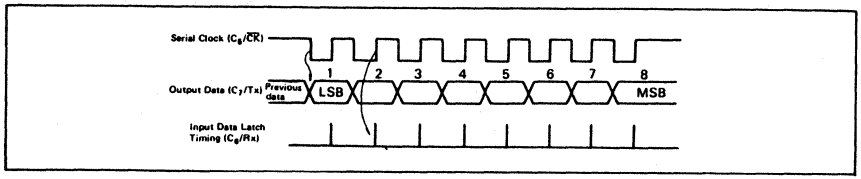


Fig. 2-9 SCI Timing Chart

(5) Data Reception

By writing the desired control bits into the SCI control register, a transfer rate and a source of transfer clock are determined and bit 6 and 5 of port C are set at the serial data input terminal and the serial clock terminal, respectively. Then dummy-writing or reading the SCI data register, the system is ready for receiving data. (This procedure is not needed after reading a subsequent received data. It must be taken after reset and after not reading a subsequent received data.)

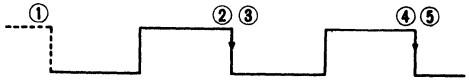
The data from the C_6/Rx terminal is input to the SCI data register synchronously with the leading edge of the serial clock (Refer to Fig. 2-9). When 8 bits of data have been received, the interrupt request bit is set in bit 7 of the SCI status register. This request can be masked by setting bit 5 of the SCI status register. If an external clock source has been selected, the transfer rate determined by bits 0 ∿ 3 of the SCI control register is ignored and the data is received synchronously with the clock from the $C_5/\overline{CK}$ terminal. If the internal clock has been selected, the $C_5/\overline{CK}$ terminal is set as output and clocks are output at the transfer rate selected by bits 0 ∿ 3 of the SCI control register.

(6) $TIMER_2$

The SCI transfer clock generator can be used as a timer. The SCI clock selected by bits 3 ∿ 0 of the SCI Control Register (4 μs ∿ approx. 32ms in 4MHz operation) is received by bit 6 of the SCI Status Register, and the timer 2 interrupt request bit is set at each falling edge of the SCI clock. Since this interrupt occurs periodically, $Timer_2$ is available for a reload counter or a timer.

$Timer_2$ is multiplexed with the SCI transfer clock generator. When using $Timer_2$ independently of the SCI, external clock should be selected as SCI clock source by setting both SCR5 and SCR4 to "1".

If Internal clock is selected as a SCI clock source, reading from or writing to the SDR initializes the prescalor of the SCI transfer clock generator.



① : Transfer clock generator is reset and mask bit (bit 4 of SCI status registor) is cleared

②,④ : $TIMER_2$ interrupt request

③,⑤ : $TIMER_2$ interrupt request bit cleared

2.5 Reset

The MCU can be reset either by external reset input (RES) or power-on reset. (Refer to Fig. 2-10.) On power up, the reset input must be held "Low" for at least t_{osc} to assure that the internal oscillator is stabilized. A sufficient time of delay can be obtained by connecting a capacitance to the $\overline{RES}$ inputs as shown in Fig. 2-11.

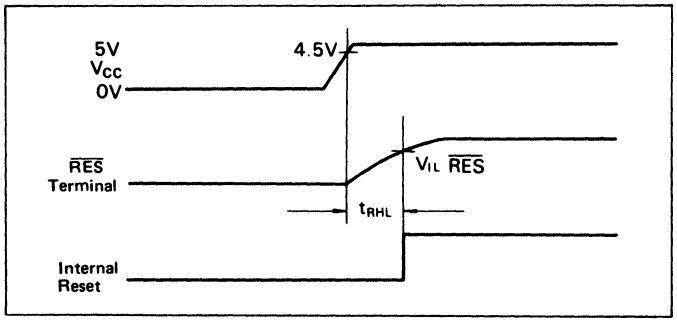


Fig. 2-10 Power On and Reset Timing

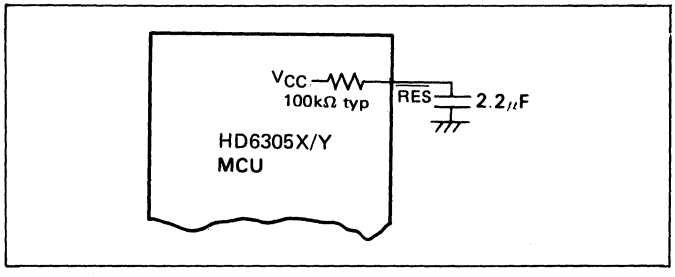


Fig. 2-11 Input Reset Delay Circuit

2.6 Internal Oscillator Options

The internal oscillator circuit is designed to meet the requirement for minimum external configurations. It can be driven by connecting a crystal (AT cut 2.0 ∿ 8.0MHz) or ceramic oscillator between pins 5 and 6 depending on the required oscillation frequency stability.

Three different terminal connections are shown in Fig. 2-12. Figs. 2-13 and 2-14 illustrate the specifications and typical arrangement of the crystal, respectively.

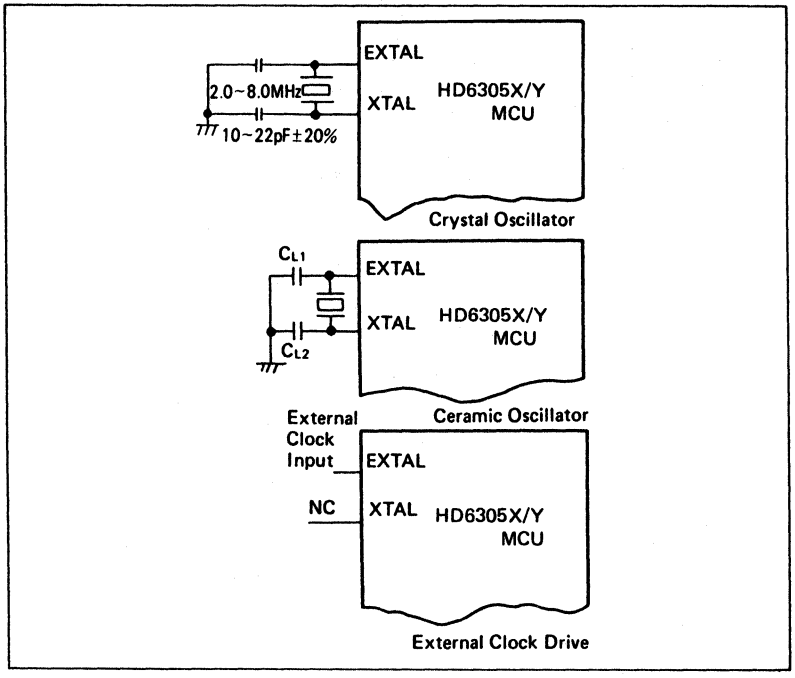


Fig. 2-12 Internal Oscillator Circuit

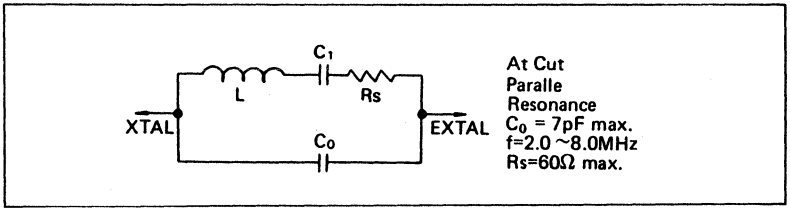
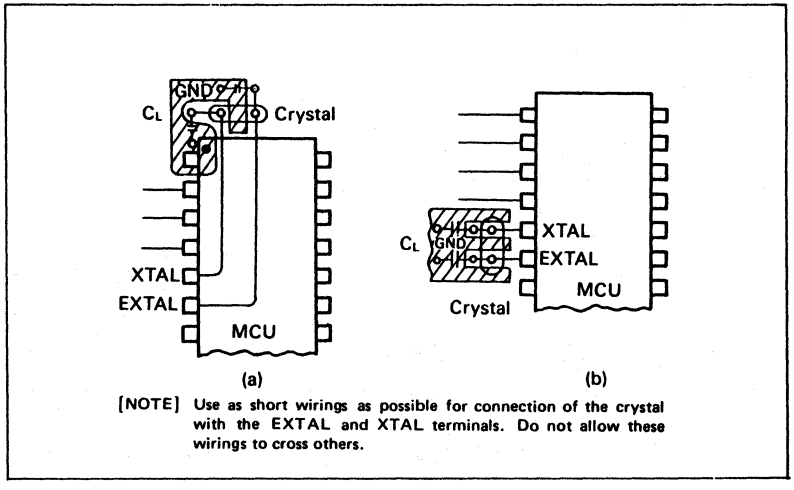


Fig. 2-13 Parameters of Crystal



[NOTE] Use as short wirings as possible for connection of the crystal with the EXTAL and XTAL terminals. Do not allow these wirings to cross others.

Fig. 2-14 Typical Crystal Arrangement

2.7 Interrupts

There are six interrupts: external interrupts ($\overline{INT}$, $\overline{INT_2}$), internal timer interrupts (TIMER, $TIMER_2$), serial interrupt (SCI) and interrupt by an instruction (SWI).

Of these six interrupts, the $\overline{INT_2}$ and TIMER interrupt or the SCI and $TIMER_2$ interrupt generate the same vector address, respectively.

When an interrupt occurs, the program in progress stops and the then CPU status is saved onto the stack. Then, the interrupt mask bit (I) of the condition code register is set and the start address of the interrupt processing routine is obtained from a particular interrupt vector address. Then the interrupt routine starts from the start address. System can exit from the interrupt routine by an RTI instruction. When this instruction is executed, the CPU status before the interrupt (saved onto the stack) is pulled and the CPU restarts the sequence with the instruction next to the one at which the interrupt occurred. Table 2-4 lists the priority of interrupts and their vector addresses.

A flowchart of the interrupt sequence is shown in Fig. 2-15. A block diagram of the interrupt request source is shown in Fig. 2-16.

(1) External Interrupt

In the block diagram, both the external interrupts $\overline{INT}$ and $\overline{INT_2}$ are edge trigger inputs. At the falling edge of each input, an interrupt request is generated and latched. The $\overline{INT}$ interrupt request is automatically cleared if jumping is made to the $\overline{INT}$ processing routine. The $\overline{INT_2}$ interrupt request is masked when "0" is written to bit 7 of the Miscellaneous Register.

If the I bit of the Condition Code Register is set to "1", interrupt requests of the external interrupt ($\overline{INT}$, $\overline{INT_2}$) are retained, but not processed. Immediately after the I bit is cleared, the CPU jumps to the corresponding interrupt routine. The $\overline{INT_2}$ interrupt can be masked by setting bit 6 of the Miscellaneous Register.

The $\overline{INT}$ terminal status can be tested by a BIL or BIH instruction. The $\overline{INT}$ falling edge detector circuit and its latching

circuit, and $\overline{INT_2}$ terminal are unaffected by executing BIL and BIH instructions.

(2) Internal Interrupt and SCI Interrupt

When I bit of the Condition Code Register is set to "1", internal timer interrupts (TIMER, $TIMER_2$) and SCI interrupt requests are retained without being processed. Immediately after I bit is cleared, the CPU initiates the corresponding interrupt service routine. Timer interrupt, SCI interrupt, $Timer_2$ interrupt can be masked by setting bit 6 of Timer Control Register, bit 5 of SCI Status Register, and bit 4 of SCI Status Register, respectively.

Table 2-4 Priority of Interrupts

Interrupt	Priority	Vector Address
$\overline{RES}$	1	$1FFE, $1FFF
SWI	2	$1FFC, $1FFD
$\overline{INT}$	3	$1FFA, $1FFB
TIMER/$\overline{INT_2}$	4	$1FF8, $1FF9
SCI/$TIMER_2$	5	$1FF6, $1FF7

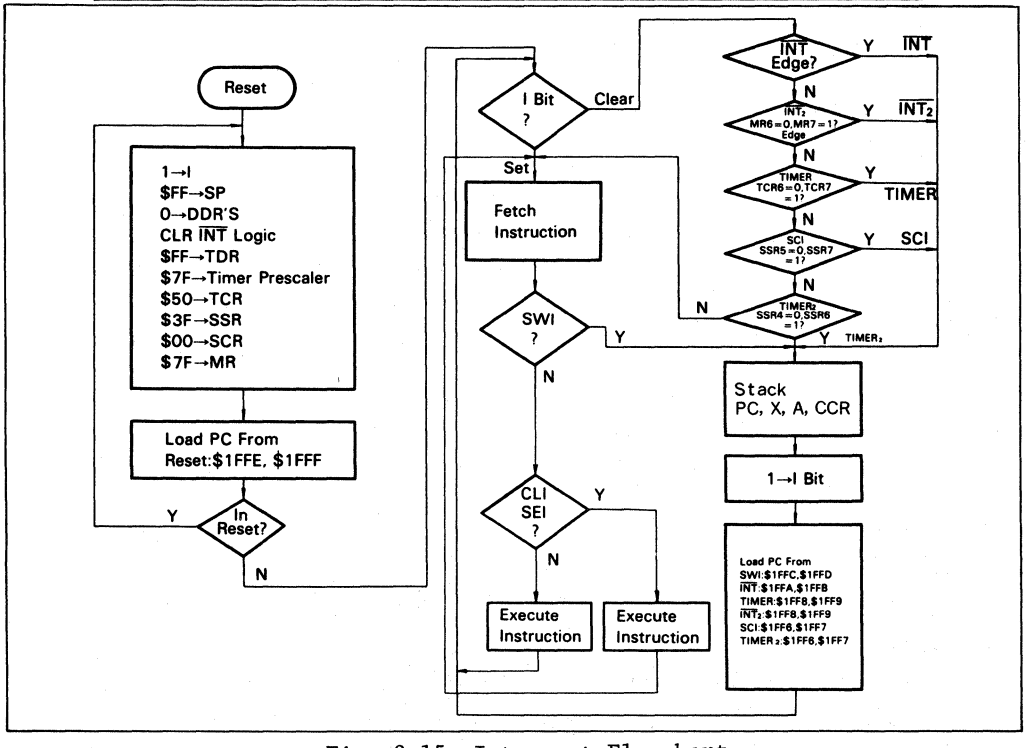


Fig. 2-15 Interrupt Flowchart

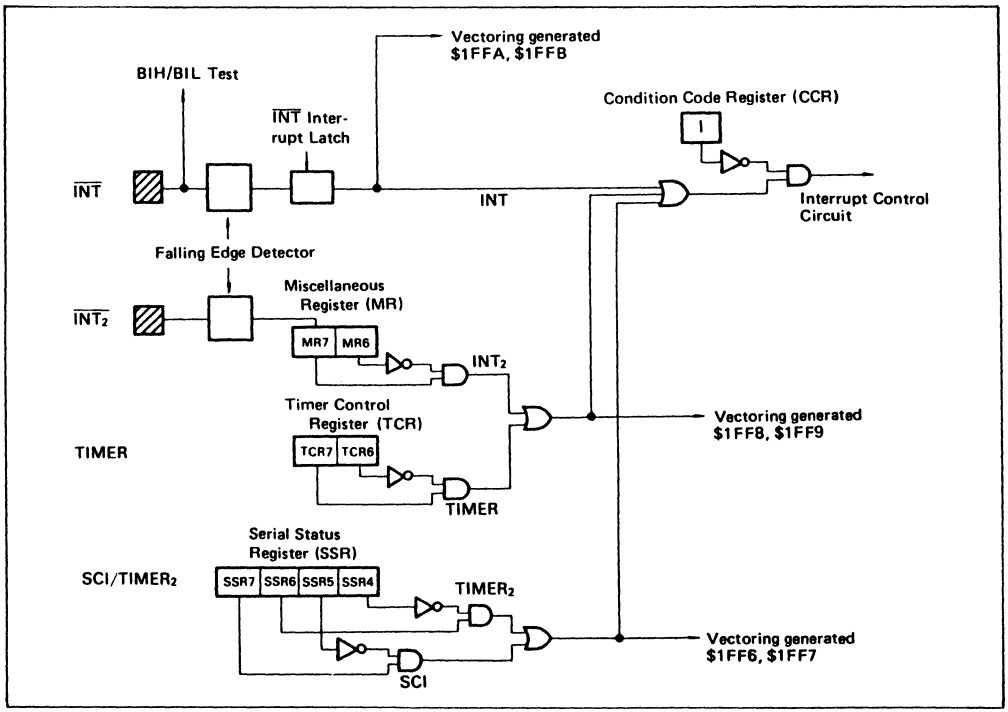


Fig. 2-16 Interrupt Request Generation Circuitry

(3) Miscellaneous Register (MR: $000A)

Miscellaneous Register (MR: $000A) specifies $\overline{INT}$ request sense (edge sense or level sense) and controls $\overline{INT_2}$ interrupt.

Bit 7 of Miscellaneous Register (MR7) is an $\overline{INT_2}$ interrupt request flag. When the falling edge is detected in $\overline{INT_2}$, MR7 is set to "1". MR7 can be checked by software if it is $\overline{INT_2}$ interrupt or not in the interrupt service routine (vector address: $1FF8, $1FF9). This bit can be reset by software.

Bit 6 (MR6) is the $\overline{INT_2}$ interrupt mask bit. If this bit is set to "1", the $\overline{INT_2}$ interrupt is masked. Both read and write are possible with bit 7, but "1" cannot be written in this bit by software. Thus an interrupt cannot be requested by software.

During reset, bit 7 is initialized to "0", while bit 6 is initialized to "1".

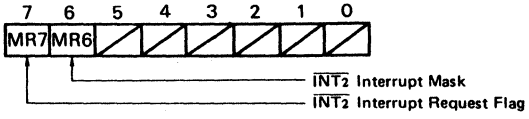


Fig. 2-17 Miscellaneous Register (MR:$000A)

2.8 Input/Output Ports

(1) I/O Ports

HD6305X0/Y0 and HD63P05Y0 have 32 I/O ports (Port A, B, C, G), HD6305X1/Y1, HD6305X2/Y2 provide 24 I/O ports (Port A, B, C). Each terminal can individually function either as input or output by programming the Data Direction Register. If a corresponding bit of Data Direction Register is set to "0", the I/O port functions as an input. While corresponding bit of Data Direction Register is programmed to "1", the I/O port functions as an output. Port A, B or C, when functioning as output and reading I/O port data, reads latched data, even if the output level is flunched by the output load (Refer to Fig. 2-18(a)). In this case, Port G always reads the pin level (Refer to Fig. 2-18(b)). Thus even when "1" is transmitted, Port G sometimes reads "0" if the load condition causes the output voltage to decrease less than 2.0V.

During reset, all Data Direction Registers are initialized to "0" and all I/O ports function as inputs.

I/O ports are compatible with TTL and CMOS in respect to both input and output.

If I/O ports are not used, they should be connected to Vss through resistors. If these terminals are left open, they may consume extra power.

(2) Output Ports (for HD6305X0/Y0, HD63P05Y0 only)

There are 16 output-only terminals (ports E and F). Each of them can also read. In this case, latched data is read even with the output terminal level being fluctuated by the output load (as with ports A, B and C).

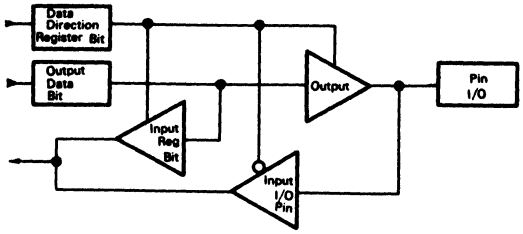
When reset, "Low" level is transmitted from each output terminal.

(3) Input Ports

Seven input-only terminals are available (port D). Writing to an input terminal is invalid.

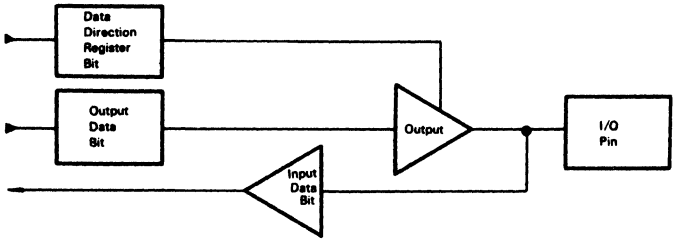
All input/output terminals, output terminals and input terminals are TLL compatible and CMOS compatible in respect of both input and output.

If I/O ports or input ports are not used, they should be connected to Vss through resistors. If these terminals are left open, they may consume extra power.



Bit of data direction register	Bit of output data	Status of output	Input to MCU
1	0	0	0
1	1	1	1
0	×	3-state	Pin

a. Ports A, B and C



b. Port G

Fig. 2-18 Input/Output Port Diagram

2.9 Low Power Dissipation Mode

HD6305X family and HD6305Y family have three types of low power dissipation modes: Wait mode, Stop mode, and Stop mode.

(1) Wait Mode

When Wait instruction is executed, the MCU enters into the WAIT mode; the oscillator does not stop, but the internal clock stops. The CPU stops but the internal peripherals (e.g. Timer and SCI) continue their current operations. (Note: Once the system entered into the Wait mode, SCI can no longer be retriggered.) During the Wait mode, RAM and I/O terminals hold their conditions just before entering into the Wait mode, while I bit of the Condition Code Register is cleared to "0".

The MCU can be recovered from the Wait Mode by an interrupt ($\overline{INT}$, TIMER/$\overline{INT_2}$, or SCI/$TIMER_2$), $\overline{RES}$ or $\overline{STBY}$. The $\overline{RES}$ resets the MCU, and the $\overline{STBY}$ brings it into the standby mode (Refer to Standby mode for details).

When the CPU accepts interrupt request, the wait mode escapes, then the CPU is brought to the operation mode and vectors to the interrupt routine. If the interrupt is masked by the I bit of the condition code register, after releasing from the wait mode, the MCU executes the instruction next to the WAIT. If an interrupt other than the $\overline{INT}$ (e.g., TIMER/$\overline{INT_2}$ or SCI/$TIMER_2$) is masked by the timer control register, miscellaneous register or serial status register, the CPU does not receive any interrupt request, Thus the wait mode cannot be released.

Fig. 2-19 shows a flowchart for the wait function.

(2) Stop Mode

When STOP instruction is being executed, the MCU enters into the stop mode. In this mode, the oscillator stops and the CPU and peripheral stops functioning but the RAM, registers and I/O terminals hold their condition just before entering into the stop mode.

The escape from this mode can be done by an external interrupt ($\overline{INT}$ or $\overline{INT_2}$), $\overline{RES}$ or $\overline{STBY}$. The $\overline{RES}$ resets the MCU and the $\overline{STBY}$ brings it into the standby mode.

When the CPU accepts interrupt request, the stop mode escapes, then the CPU is brought to the operation mode and vectors to the interrupt routine. If the interrupt is masked by the I bit of the condition code register, after releasing from the stop mode, the MCU executes the instruction next to the STOP. The MCU cannot be released from the STOP mode if the $\overline{INT_2}$ interrupt is masked by the miscellaneous register.

Fig. 2-20 shows a flowchart for the stop function. Fig. 2-21 shows a timing chart when the MCU is released from the stop mode.

For releasing from the stop mode by an interrupt, oscillation starts upon input of the interrupt and, after the internal delay time for stabilized oscillation, the CPU becomes active. When restarting by $\overline{RES}$, oscillation starts when the $\overline{RES}$ goes "0" and the CPU restarts when the $\overline{RES}$ goes "1". The duration of $\overline{RES}$="0" must exceed t_{osc}=20ms to assure stabilized oscillation.

(3) Standby Mode

The MCU enters into the standby mode when the $\overline{STBY}$ goes "Low". In this mode, all operations stop and the internal condition is reset holding the RAM contents. The I/O terminals go into High-impedance state. The standby mode should escape by bringing $\overline{STBY}$ "High". The CPU must be restarted by reset. The timing of input signals at the $\overline{RES}$ and $\overline{STBY}$ is shown in Fig. 2-22.

Table 2-5 lists the status of each parts of the MCU in each low power dissipation modes. Transitions between each mode are shown in Fig. 2-23.

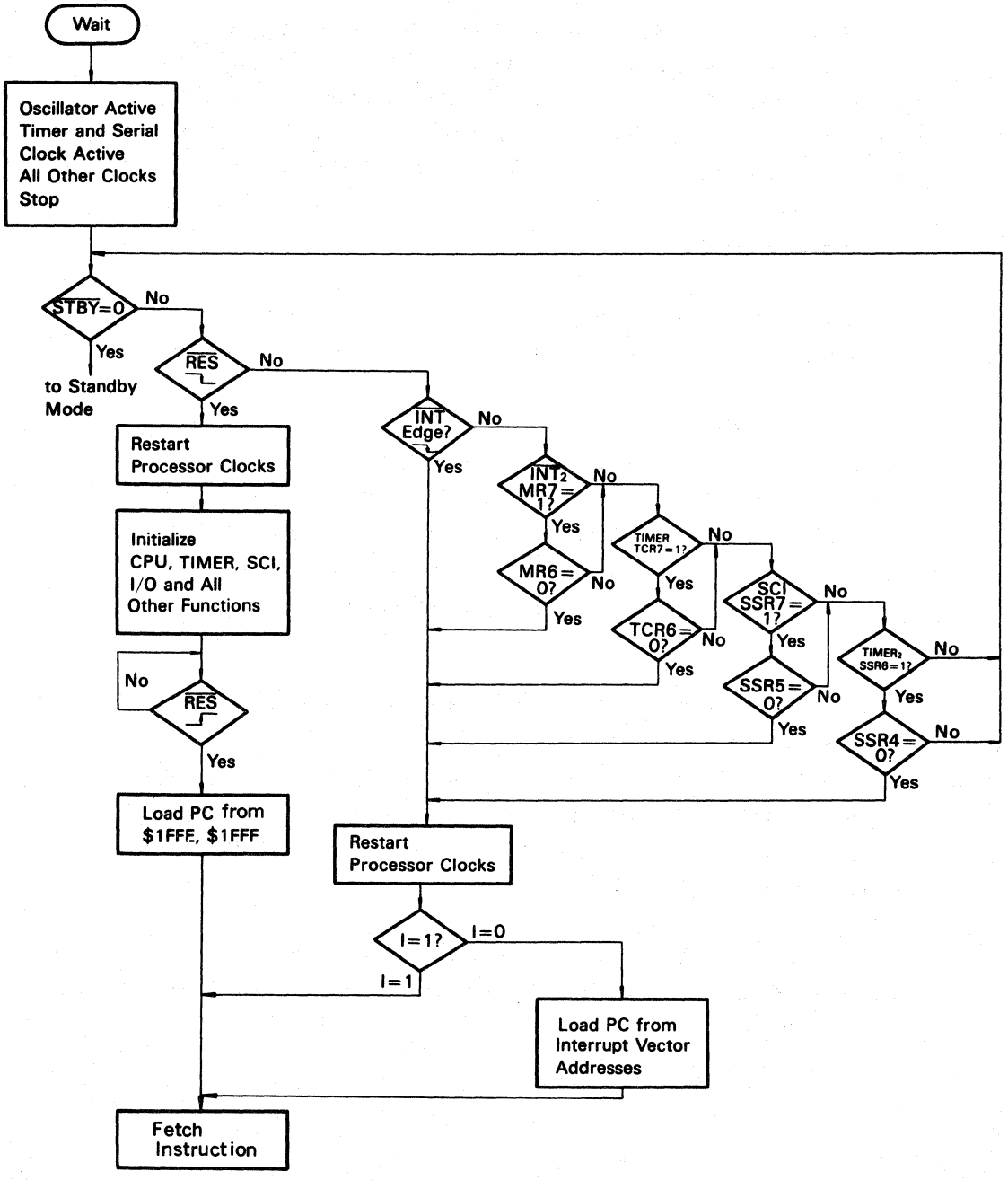


Fig. 2-19 Wait Mode Flowchart

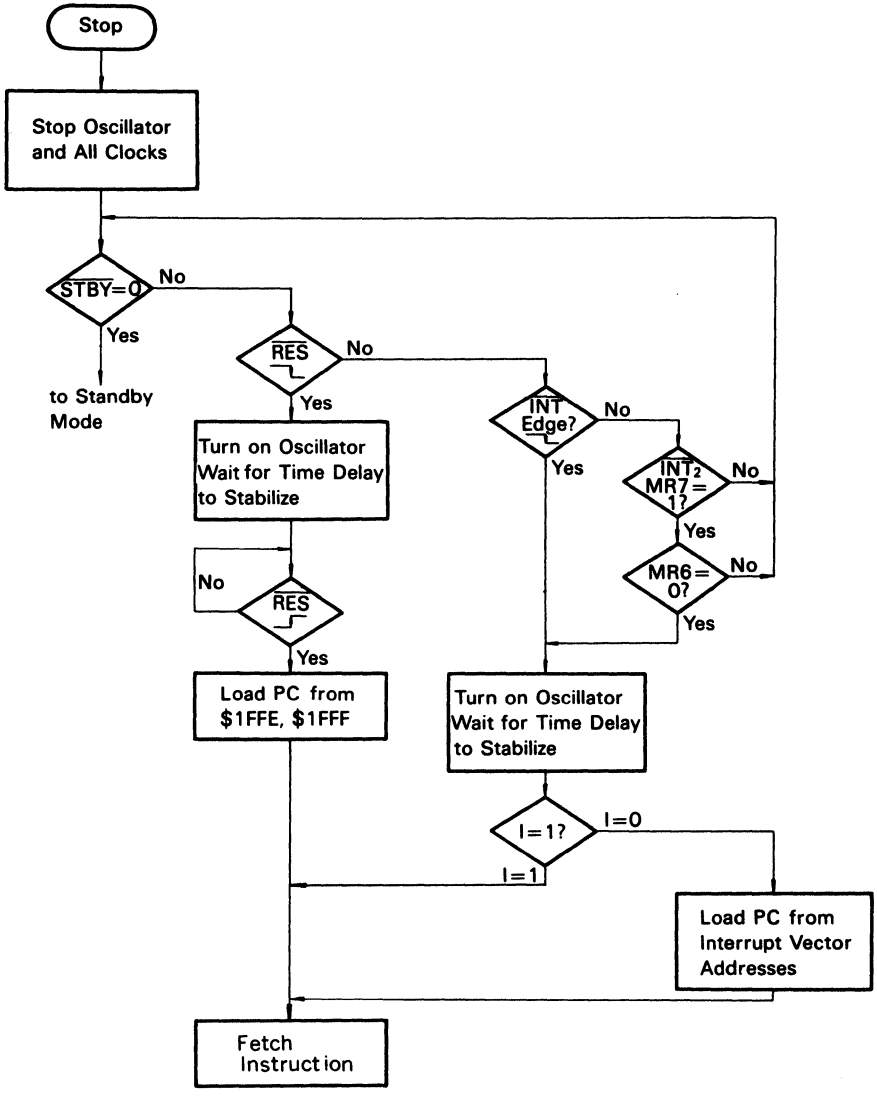


Fig. 2-20 Stop Mode Flowchart

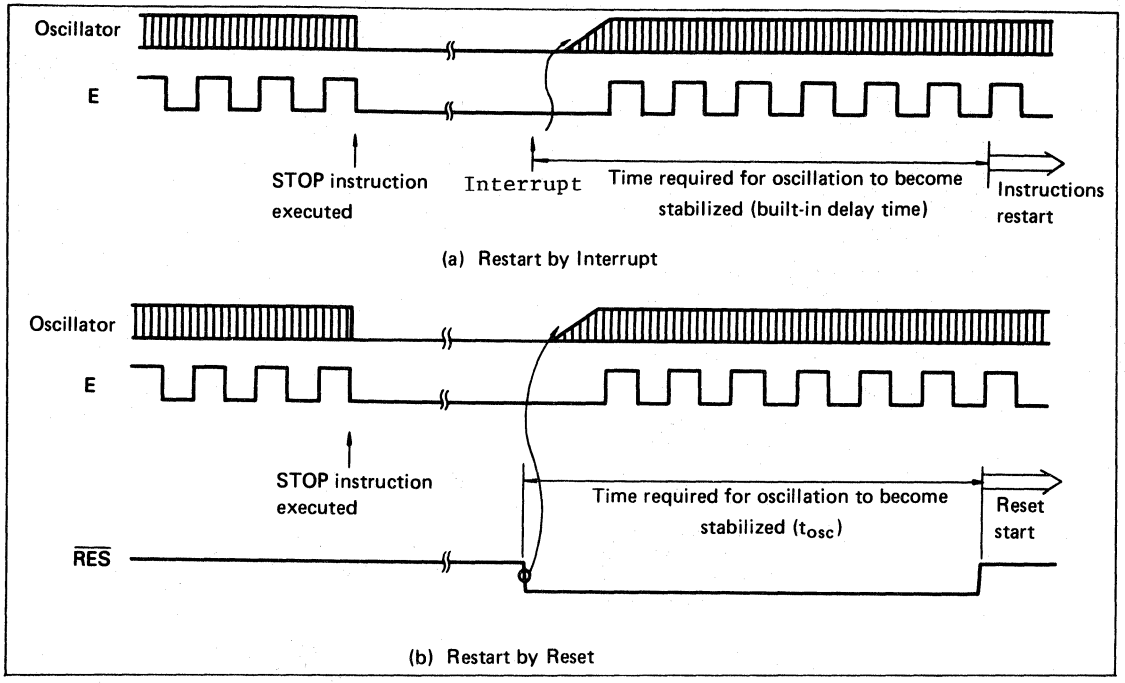


Fig. 2-21 Timing Chart of Releasing from Stop Mode

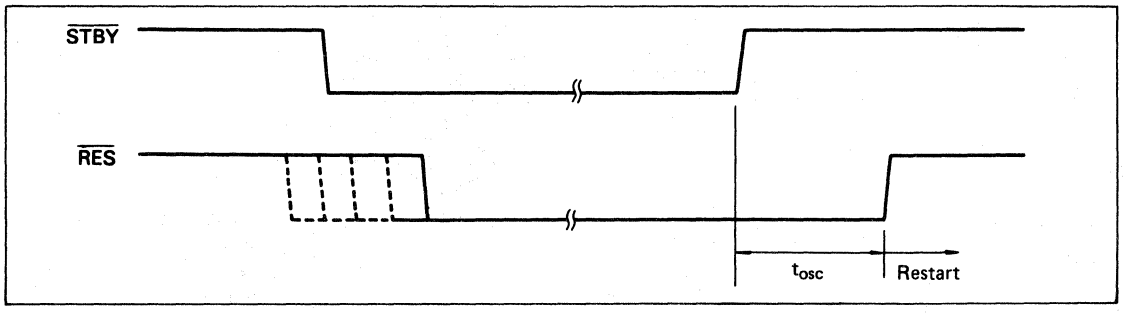


Fig. 2-22 Timing Chart of Releasing from Standby Mode

Table 2-5 Status of Each Part of MCU in Low Power Dissipation Modes

Mode	Start		Condition						Escape
			Oscillator	CPU	Timer, Serial	Register	RAM	I/O terminal	
WAIT	Software	WAIT instruction	Active	Stop	Active	Keep	Keep	Keep	$\overline{STBY}$, $\overline{RES}$, $\overline{INT}$, $\overline{INT_2}$, each interrupt request of TIMER, $TIMER_2$, SCI
STOP		STOP instruction	Stop	Stop	Stop	Keep	Keep	Keep	$\overline{STBY}$, $\overline{RES}$, $\overline{INT}$, $\overline{INT_2}$
Standby	Hardware	$\overline{STBY}$="Low"	Stop	Stop	Stop	Reset	Keep	High impedance	$\overline{STBY}$="High"

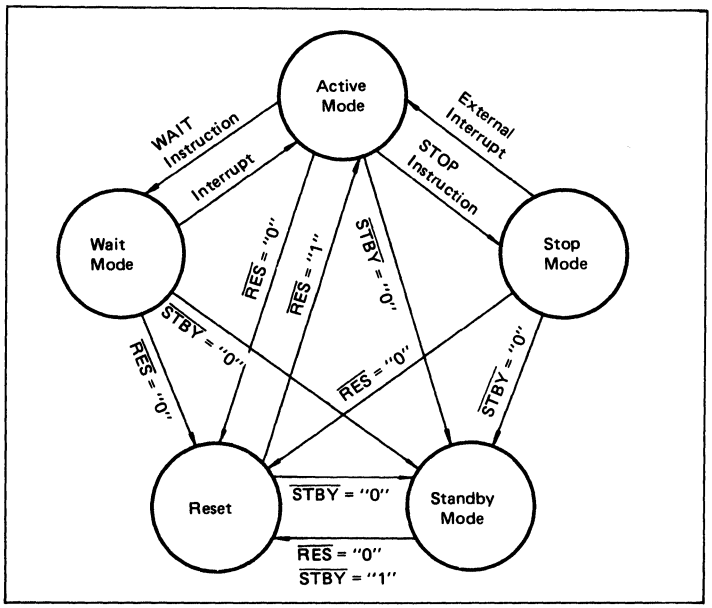


Fig. 2-23 Transitions among Active Mode, Wait Mode, Stop Mode, Standby Mode and Reset

3. APPLICATIONS AND PRECAUTIONS

3.1 Memory Space Expansion of HD6305X1/Y1, HD6305X2/Y2, HD63P05Y1

(1) ROM

ROM space can be expanded by externally connecting EPROM or Mask ROM to the MCU. Fig. 3-1 shows an example of using the HN482732A.

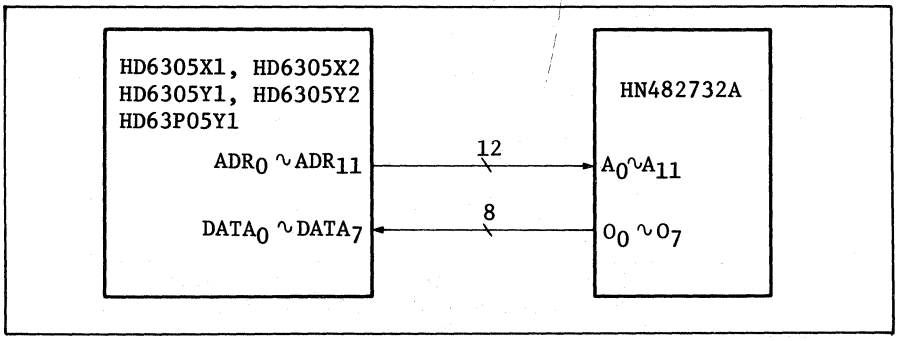


Fig. 3-1 Memory Space Expansion for ROM

(2) RAM

Fig. 3-2 shows a system configuration by using the HM6116.

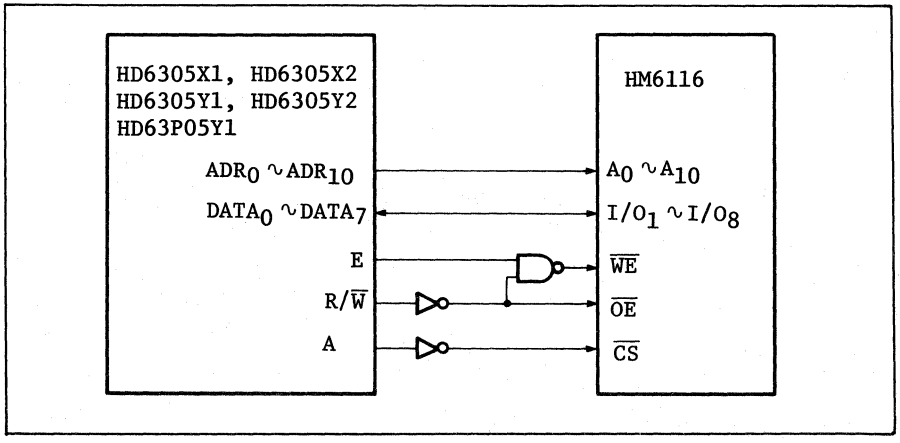


Fig. 3-2 Memory Space Expansion for RAM

3.2 Watch-Dog Timer

(1) Purpose

A simple method of implementing the watch dog timer, which is generally known as a means of recovery from a system upset, will be described below.

(2) Basic circuit

Fig. 3-3 shows the basic circuit for recovery from a system upset.

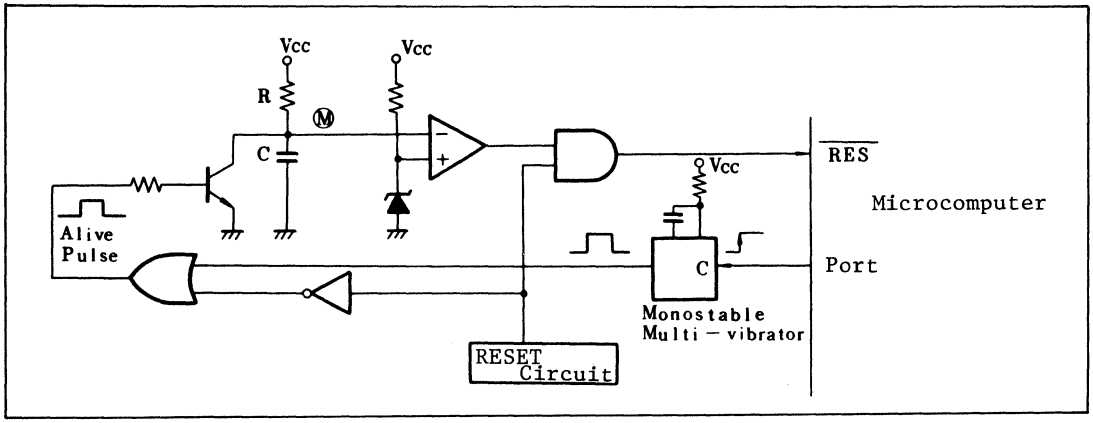


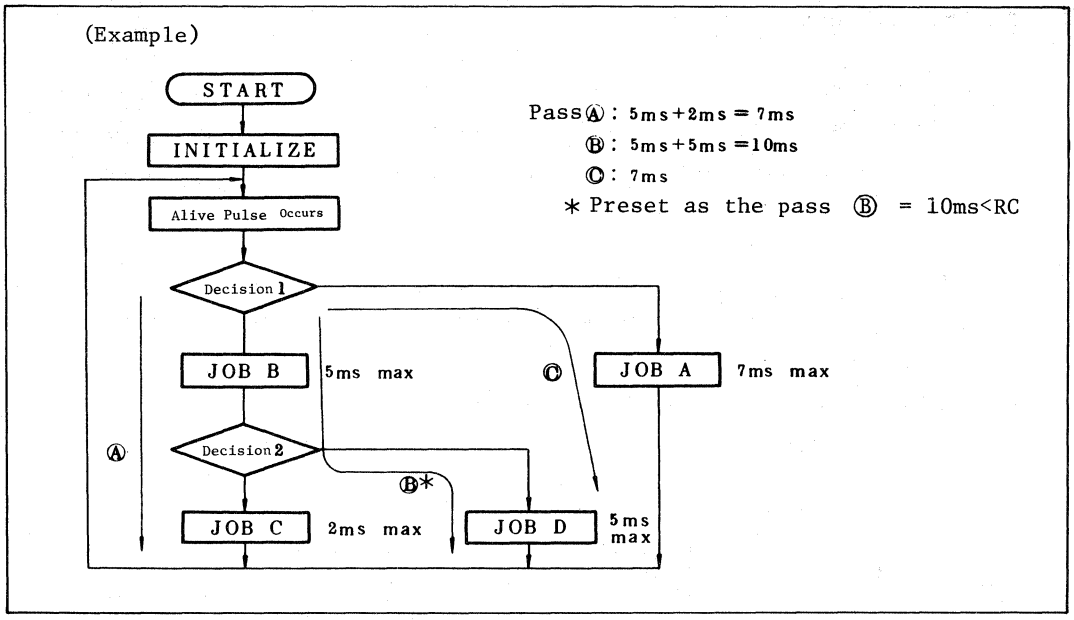
Fig. 3-3 Basic Circuit for Recovery from a System Upset

(3) Operation outline

(i) In structuring system software, design the whole system so that a pulse is generated through the port within a predetermined period (T<RC) if the program is normally executed.

(ii) When the system enters into the loop and is deadlocked or when the system is upset the system presents no change in the port output (e.g., it does not generate alive pulse), which increases the Ⓜ point potential and resets the MCU.

(iii) Preset the RC at above the maximum value of all existing routes in normal operation. RC is set to a value greater than 10ms in the example below.



3.3 Auto Reset Circuit

(1) Purpose

The following describes how to implement the reset circuit for power-on reset or auto reset at $V_{CC}<V_{LM}$. (V_{LM}; Treshold level.)

(2) Basic Circuit

Fig. 3-4 shows the basic circuit of auto reset.

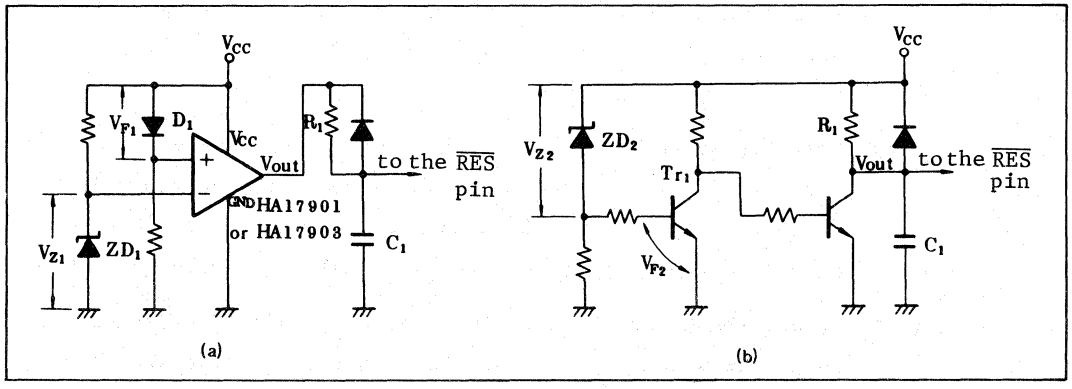
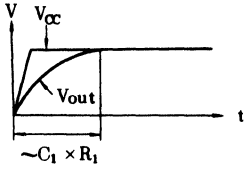


Fig. 3-4 Basic Circuit of Auto Reset

(3) Operation Principle

(i) Power-on Reset

Since the $\overline{RES}$ terminal is pulled up by R_1, the $\overline{RES}$ rising time is delayed by C_1 R_1 during power up, so that the MCU can be reset normally.



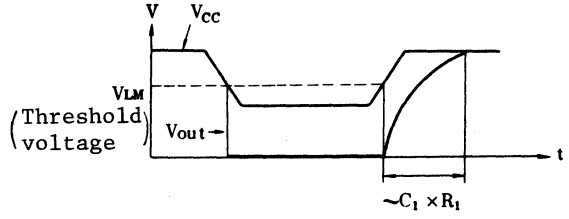
(ii) Auto-reset

The auto-reset operation of the basic circuits (a, b) in Fig. 4-4 is performed as follows.

(a) The voltages V_{Z1} and V_{F1}, applying to ZD_1 and D_1, are constant even if V_{CC} changes. The $\overline{RES}$ signals are as follows.

$$\begin{cases} V_{out} = V_{CC} > V_{Z1} + V_{F1} \\ V_{out} = GND \text{ when } V_{CC} < V_{Z1} + V_{F1} \end{cases}$$

(b) The voltage V_{Z2}, applying to ZD_2, and the ON level V_{F2} of Tr_1 are constant, even if the V_{CC} changes. The followings describe the $\overline{RES}$ signals.



$$\begin{cases} V_{out} = V_{CC} \text{ when } V_{CC} > V_{Z2} + V_{F2} \\ V_{out} = GND \text{ when } V_{CC} < V_{Z2} + V_{F2} \end{cases}$$

When the V_{CC} changes, threshold level V_{LM} is affected by replacing the Zener diodes ZD_1 and ZD_2.

(iii) The Auto-reset function will be more stable by feeding back the output signal to the $\overline{RES}$ terminal, fine tuning the reference voltage, and providing hysteresis with the V_{LM1} and the V_{LM2}. (V_{LM1} is a voltage during the shifting

from operation to resetting, and V_{LM2} is the voltage in recovering.)

3.4 Manual Reset Circuit

(1) Purpose

When the MCU is reset by an external switch, it is necessary to prevent the MCU from chattering. The following describes a reset circuit in which chattering prevention and power-on reset functions are provided.

(2) Basic Circuit

Fig. 3-5 shows the basic circuit of manual reset.

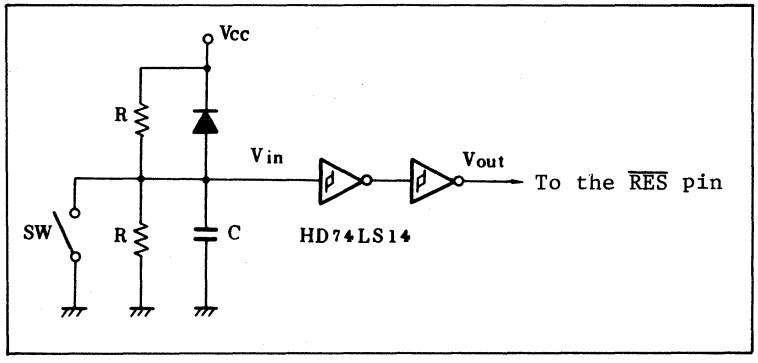
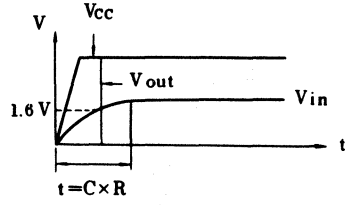


Fig. 3-5 Basic Circuit of Manual Reset

(3) Operation Principle

(i) Power-on reset

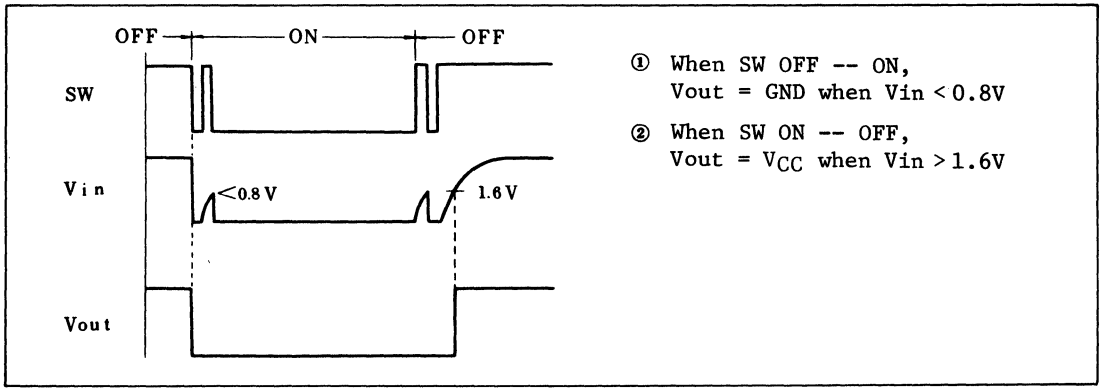
During power-up, the capacitor C will be charged and the V_{in} rising time will be delayed by the CR charge time, so that power-on would be reset normally.



(ii) Manual reset

When SW is on, the V_{out} goes "0" and the MCU is reset
When SW is off, the capacitor C is charged, so that the rising time of V_{in} is delayed by the CR charge time.

Chattering is prevented by the capacitor C and the schmidt trigger HD74LS14.



3.5 A/D Converter Circuit (1) ... High Speed

(1) Purpose

The following describes how to implement a simple A/D conversion (analog-digital conversion) using the resistance radder (R-2R system).

(2) Basic Circuit

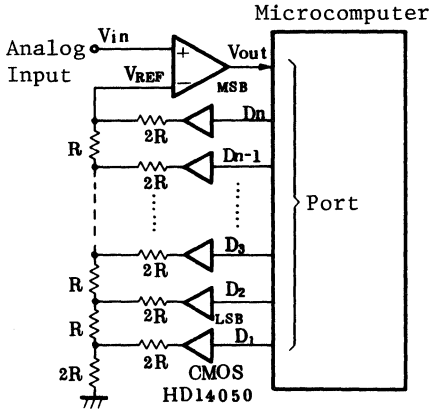


Fig. 3-6 A/D Converter Basic Circuit

$$V_{REF}=\left(\frac{Dn}{2^1}+\frac{Dn-1}{2^2}+\cdots+\frac{D_2}{2^{n-1}}+\frac{D_1}{2^n}\right)\times V$$

V : Voltage of the output "1" of the port

Vout = 0 : Vin ≦ VREF

Vout = 1 : Vin > VREF

Necessary number of port=Accuracy bit+1

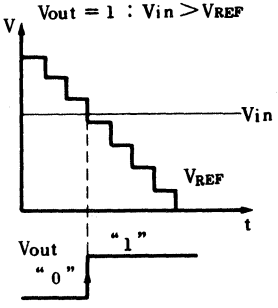


Fig. 3-7 Timing Chart

(3) Operation Outline

The following describes the operation outline of 3-bits A/D converter circuit.

(i) The digital outputs (D_3, D_2, D_1) are changed from (1, 1, 1) to (0, 0, 0) at the port. V_{REF} is reduced from 7/8 V to 0 by every 1/8 V according to Table 3-1.

(ii) The output V_{out}, which is the result of comparison between the analog input V_{in} and V_{REF}, is reversed from "0" to "1" when V_{in} is greater than V_{REF} as shown in Fig. 3-7.

(iii) The value of (D_3, D_2, D_1), when V_{out} is reversed from "0" to "1", is a digital value to the analog input V_{in}.

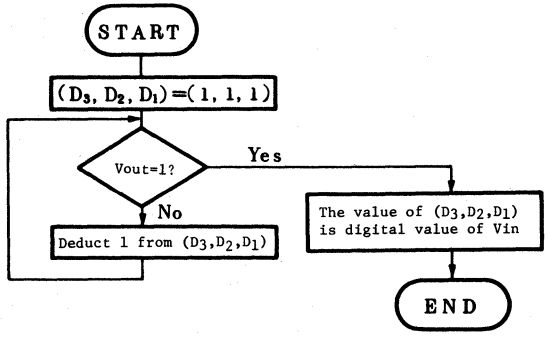
Table 3-1 Value of V_{REF}

D_3	D_2	D_1	V_{REF}
0	0	0	0
0	0	1	1/8 × V
0	1	0	2/8 × V
0	1	1	3/8 × V
1	0	0	4/8 × V
1	0	1	5/8 × V
1	1	0	6/8 × V
1	1	1	7/8 × V

$$V_{REF} = \left(\frac{D_3}{2^1} + \frac{D_2}{2^2} + \frac{D_1}{2^3}\right) \times V$$

D_1 ∿ D_3 are denoted by "1" or "0"

V: Voltage of the port output "1"



3.6 A/D Converter Circuit (2) ... Low Speed

(1) Purpose

The following describes the A/D conversion system utilizing time for charge/discharge to/from the CR circuit. This conversion system is available even if the A/D conversion time is slow.

(2) Basic Circuit

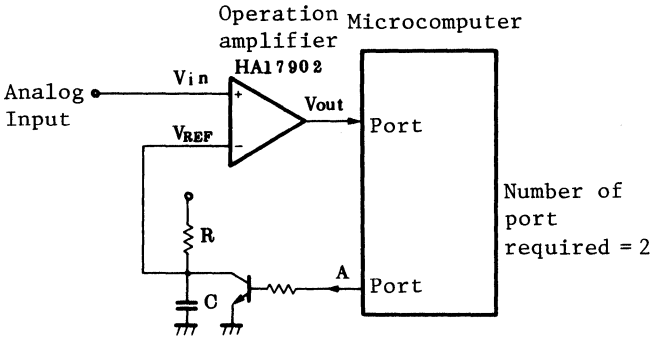


Fig. 3-8 A/D Converter Basic Circuit

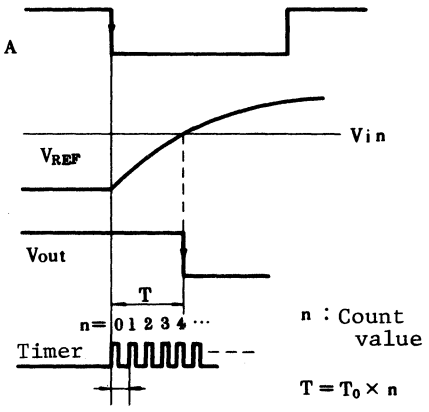


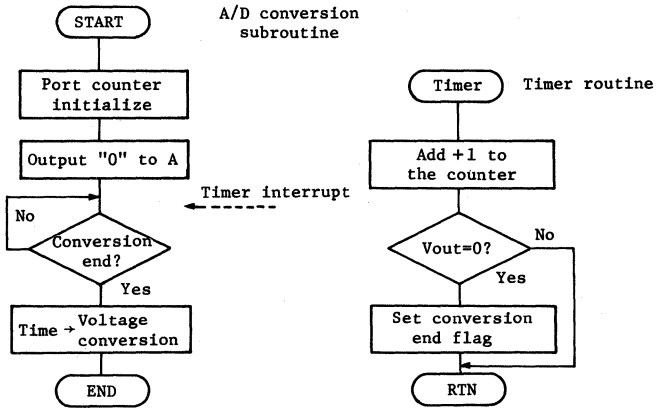
Fig. 3-9 Timing Chart

(3) Operation Outline

(i) Set the output terminal A of the port to "1", discharge the external condenser C for a sufficiently long period, decline A from "1" to "0" synchronously with the timer interrupt and charge C.

(ii) Check Vout for each timer interrupt and observe the time T in which Vout declines from "1" to "0". Vout becomes "0" when Vin is less than V_{REF}. Obtained as $T=T_0 \times n$.

(iii) The obtained time T which is voltage converted is a digital value of the analog input Vin.



3.7 Precaution;- Board Design of Oscillation Circuit

When connecting crystal and ceramic resonator with the XTAL and EXTAL pins to oscillate, observe the followings in designing the board.

(1) Locate crystal, ceramic resonator, and load capacity C1 and C2 as near the LSI as possible. (Induction of noise from outside to the XTAL and EXTAL pins may cause trouble in oscillation.)

(2) Wire the signal lines to the neighbouring XTAL and EXTAL pins as far apart as possible.

(3) Board design of situating signal lines or power supply lines near the oscillator circuit as shown in Fig. 3-11, should not be used because of trouble in oscillation by induction. The resistor between the XTAL and EXTAL, and pins close to them should be 10MΩ or more.

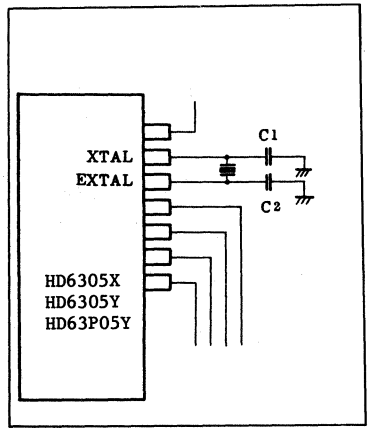


Fig. 3-10 Design of Oscillation Circuit Board

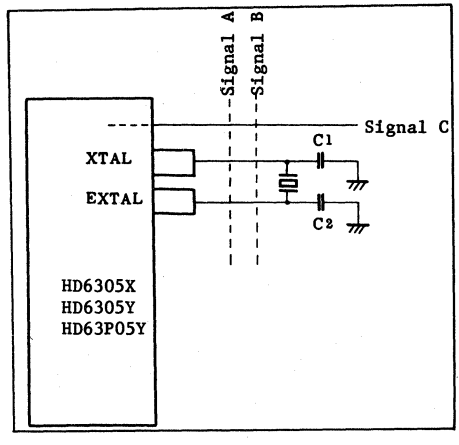


Fig. 3-11 Example of Circuit Causing Trouble in Oscillation

3.8 Precaution; - Program of Write Only Register

Read/Modify/Write instructions are unavailable for changing the contents of Write Only Register (e.g. DDR; Data Direction Register of I/O port) of HD6305X, HD6305Y and HD63P05Y.

(1) Data cannot be read from Write Only Register.
(e.g. DDR of I/O port)

While Read/Modify/Write instructions are executed in the following sequence.

(i) Reads the contents from appointed address.

(ii) Changes the data which has been read.

(iii) Turn the data back to the original address.

Thus, Read/Modify/Write instructions cannot be applied to Write Only Register such as DDR.

(2) For the same reason, do not set DDR of I/O port using BSET and BCLR instructions.

(3) Stored instructions (e.g. STA and STX, etc.) are available for writting into the Write Only Register.

3.9 Precaution. - Sending/Receiving Program of Serial Data

Reading from or Writing into the SCI data register (SDR:$0012) during sending/receiving of serial data may make sending/receiving operation of SCI out of order.

3.10 Precaution; - WAIT/STOP Instructions Program

When I bit of condition code register is "1" and interrupt ($\overline{INT}$, TIMER/$\overline{INT_2}$, SCI/TIMER2) is held, the MCU does not enter into WAIT mode by executing WAIT instruction.

In that case, after the 4 dummy cycles, the MCU executes the next instruction.

In the same way, when external interrupts ($\overline{INT}$, $\overline{INT_2}$) are held at the bit I set, the MCU does not enter into the STOP mode by executing STOP instruction. In that case the MCU executes the next instruction after the 4 dummy cycles.

3.11 Precaution; - To use the ERPOM ON-PACKAGE 8-bit Single-chip Microcomputer

Please be careful of the following, since this MCU has a special structure with pin socket on the package.

(1) Do not apply high static voltage or surge voltage over MAXIMUM RATINGS to the socket pins as well as the LSI pins. Otherwise permanent damage to the device would be caused.

(2) When using 32k EPROM (24-pin), insert it leaving the four pins above open.

(3) When inserting this into system products like mask ROM type single chip microcomputer, be careful of the following to give effective contact between the EPROM pins and socket pins.

(a) When soldering the LSI onto a printed circuit board, the recommended condition is:

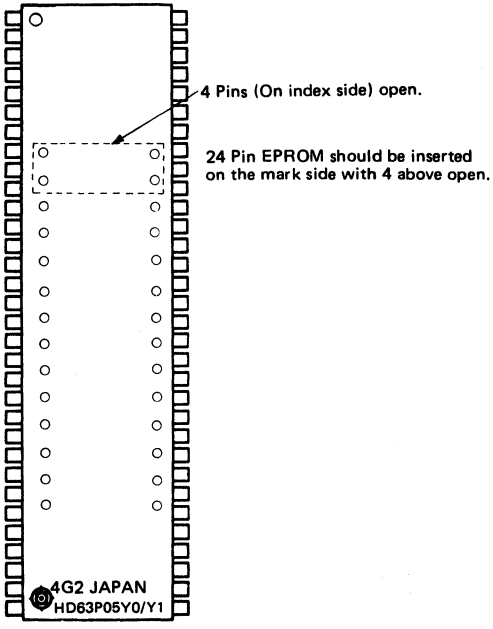
Temperature: lower than 250°C

Time: within 10 sec.

(b) Be careful that detergent or coating does not get into the socket during flux washing or board coating after soldering, because that may adversely affect the socket contact.

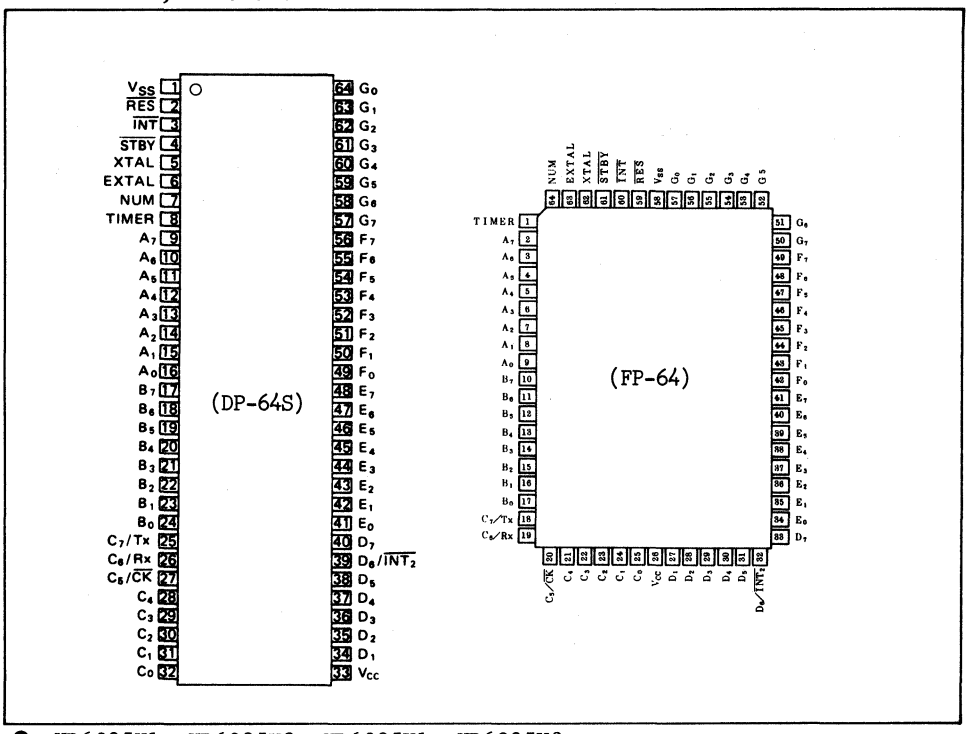
(c) Avoid permanent application of this during continuous vibration.

(d) The socket, when repeatedly inserted and removed, loses its contactability. It is recommended to use a new one when used in production.

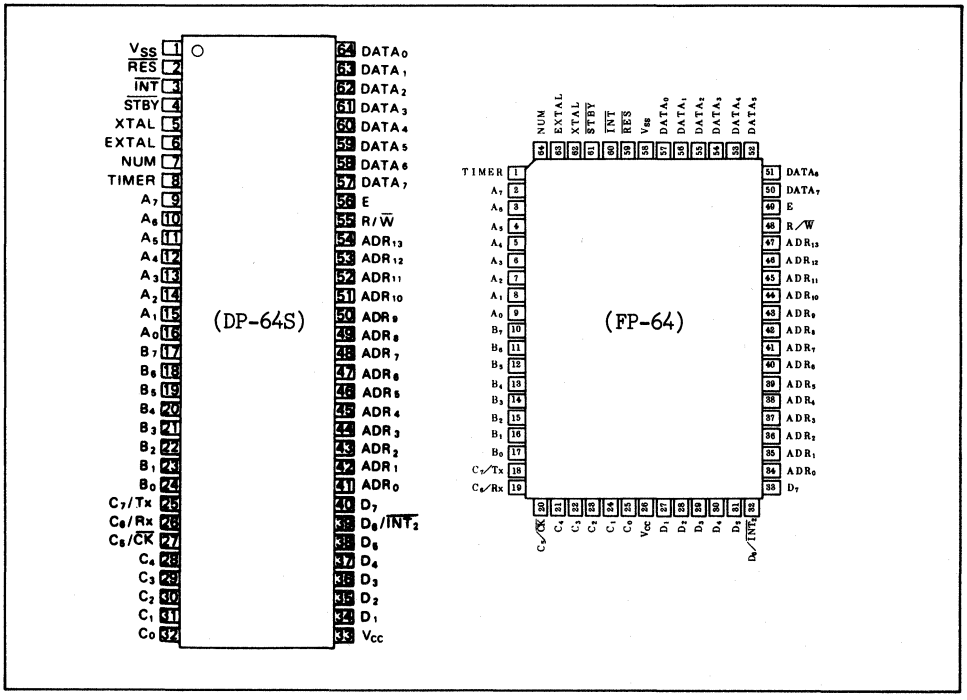


4. PIN ARRANGEMENT AND DIMENSIONAL OUTLINE

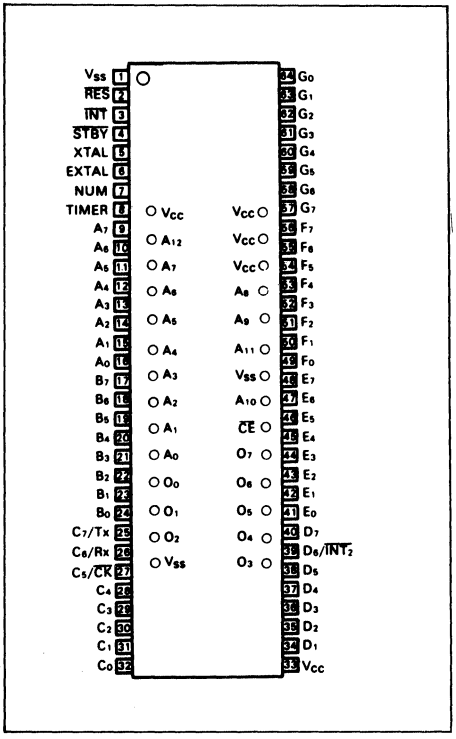
- HD6305X0, HD6305Y0



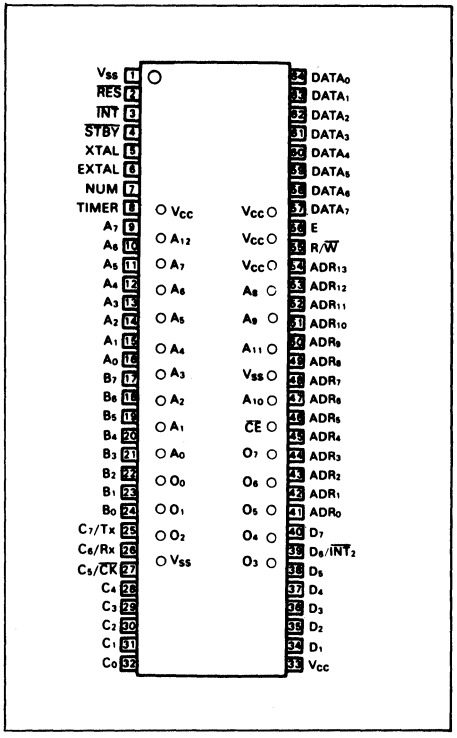
- HD6305X1, HD6305X2, HD6305Y1, HD6305Y2



● HD63P05Y0



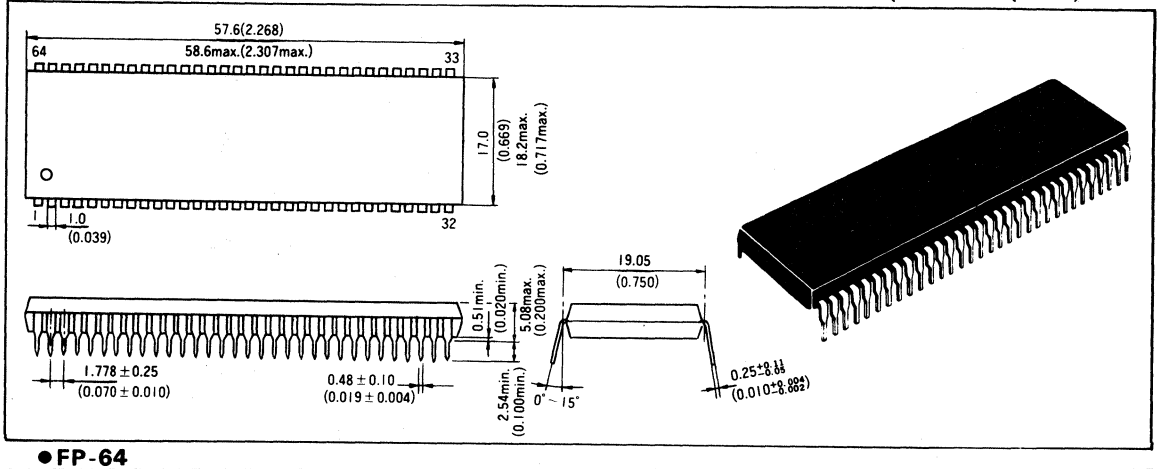
● HD63P05Y1



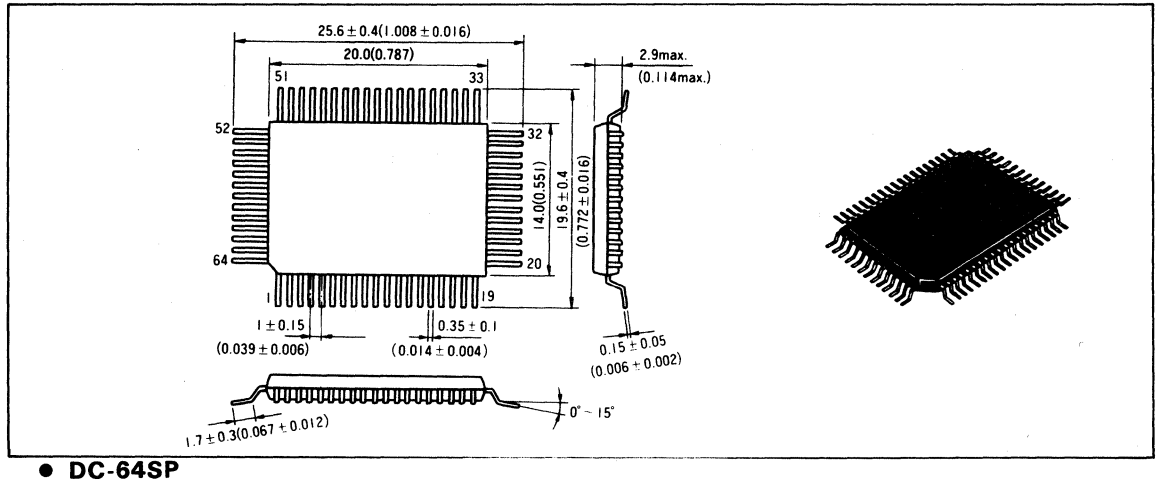
4

● **DP-64S**

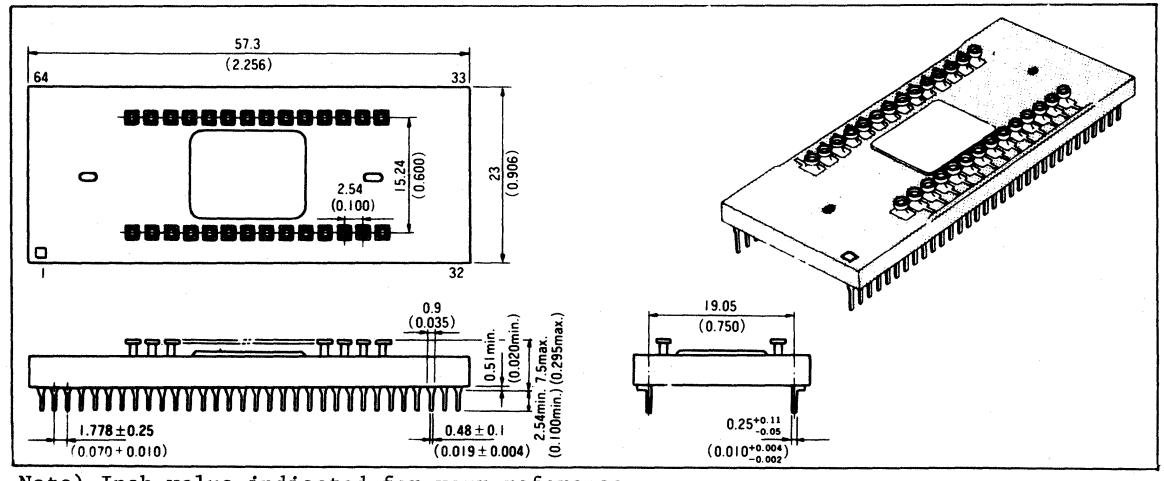
(Unit: mm(inch)



● **FP-64**



● **DC-64SP**



Note) Inch value indicated for your reference.

Fig. 4-2 Dimensional Outline

5. ELECTRICAL CHARACTERISTICS

5.1 Electrical Characteristics of HD6305X0, HD6305Y0, HD63P05Y0

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply voltage	V_{CC}	-0.3 ∿ +7.0	V
Input voltage	V_{in}	-0.3 ∿ V_{CC} + 0.3	V
Operating temperature	T_{opr}	0 ∿ +70	°C
Storage temperature	T_{stg}	-55 ∿ +150	°C

[NOTE] These products have a protection circuit in their input terminals against high electrostatic voltage or high electric fields. Neverthless, be careful not to apply any voltage higher than the absolute maximum rating to these high input impedance circuits. To assure normal operation, we recommende V_{in}, V_{out}; $V_{SS} \leqq (V_{in}$ or $V_{out}) \leqq V_{CC}$

4

■ ELECTRICAL CHARACTERISTICS

● DC Characteristics (V_{CC} = 5.0V ± 10%, V_{SS} = GND and T_a = 0 ∿ +70°C unless otherwise specified)

Item		Symbol	Test condition	min	typ	max	Unit
Input voltage "High"	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}-0.5	-	V_{CC}+0.3	V
	EXTAL			V_{CC}×0.7	-	V_{CC}+0.3	V
	Others			2.0	-	V_{CC}+0.3	V
Input voltage "Low"	All Inputs	V_{IL}		-0.3	-	0.8	V
Current * dissipation	Operating	I_{CC}	f=1MHz**	-	5	10	mA
	Wait			-	2	5	mA
	Stop			-	2	10	μA
	Standby			-	2	10	μA
Input leakage current	TIMER, $\overline{INT}$, $\overline{STBY}$, D_1∿D_7	$\lvert I_{IL} \rvert$	V_{in}=0.5∿ V_{CC}-0.5V	-	-	1	μA
Three-state current	A_0 ∿ A_7, B_0 ∿ B_7, C_0 ∿ C_7, G_0 ∿ G_7, E_0 ∿ E_7,** F_0 ∿ F_7,**	$\lvert I_{TSI} \rvert$		-	-	1	μA
Input *** capacity	All terminals	C_{in}	f=1MHz, V_{in}=0V	-	-	12	pF

* The value at f=XMHz is given by

I_{CC} (f=XMHz) = I_{CC}(f=1MHz)×X. V_{IH} min = V_{CC} - 1.0V, V_{IL} max = 0.8V.

For HD63P05Y0, I_{CC} of EPROM is not included.

** In Standby Mode

*** HD63P05Y0 is MAX 15pF

● AC Characteristics (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C, unless otherwise noted)

Item	Symbol	Test Condition	HD6305X0/Y0 HD63P05Y0			HD63A05X0/Y0 HD63PA05Y0			HD63B05X0/Y0 HD63PB05Y0			Unit
			min	typ	max	min	typ	max	min	typ	max	
Clock Frequency	f_{cl}		0.4	-	4	0.4	-	6	0.4	-	8	MHz
Cycle Time	t_{cyc}		1.0	-	10	0.666	-	10	0.5	-	10	µs
$\overline{INT}$ Pulse Width	t_{IWL}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
$\overline{INT_2}$ Pulse Width	t_{IWL2}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
$\overline{RES}$ Pulse Width	t_{RWL}		5	-	-	5	-	-	5	-	-	t_{cyc}
TIMER Pulse Width	t_{TWL}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
Oscillation Start Time (Crystal)	t_{osc}	CL=22pF±20% R_S=60Ω max	-	-	20	-	-	20	-	-	20	ms
Reset Delay Time	t_{RHL}	external capacitance 2.2µF	80	-	-	80	-	-	80	-	-	ms

● Port Characteristics (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ + 70°C, unless otherwise noted)

Item		Symbol	Test Condition	min	typ	max	Unit
Output "High" Voltage	Port A, B, C, G, E, F	V_{OH}	I_{OH} = -200µA	2.4	-	-	V
			I_{OH} = -10µA	V_{CC}-0.7	-	-	V
Output "Low" Voltage		V_{OL}	I_{OL} = 1.6mA	-	-	0.55	V
Input "High" Voltage	Port A, B, C, D, G	V_{IH}		2.0	-	V_{CC}+0.3	V
Input "Low" Voltage		V_{IL}		-0.3	-	0.8	V
Input Leakage Current		$\|I_{IL}\|$	V_{in} = 0.5 ∿ V_{CC}-0.5V	-1	-	1	µA

- SCI Timing (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305X0/Y0 HD63P05Y0			HD63A05X0/Y0 HD63PA05Y0			HD63B05X0/Y0 HD63PB05Y0			Unit
			min	typ	max	min	typ	max	min	typ	max	
Clock Cylce Time	t_{Scyc}	Fig. 6-1 Fig. 6-2	1	-	32768	0.67	-	21845	0.5	-	16384	µs
Data Output Delay Time	t_{TXD}		-	-	250	-	-	250	-	-	250	ns
Data Set-up Time	t_{SRX}		200	-	-	200	-	-	200	-	-	ns
Data Hold Time	t_{HRX}		100	-	-	100	-	-	100	-	-	ns

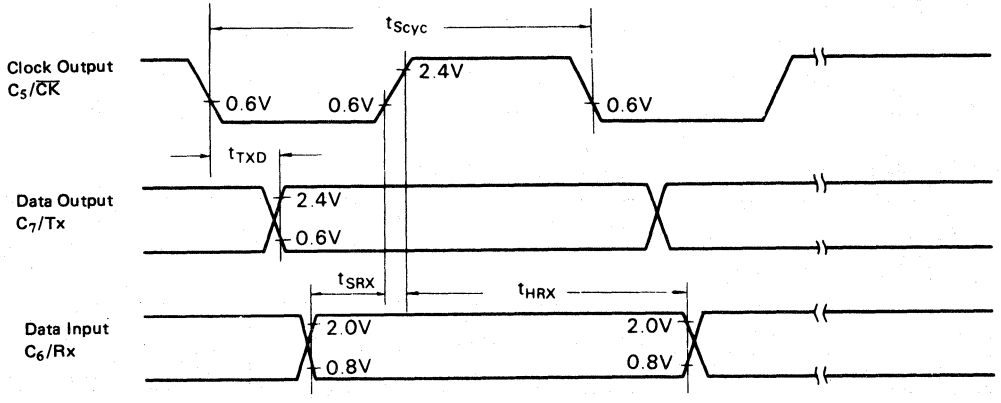


Fig. 5-1 SCI Timing (Internal Clock)

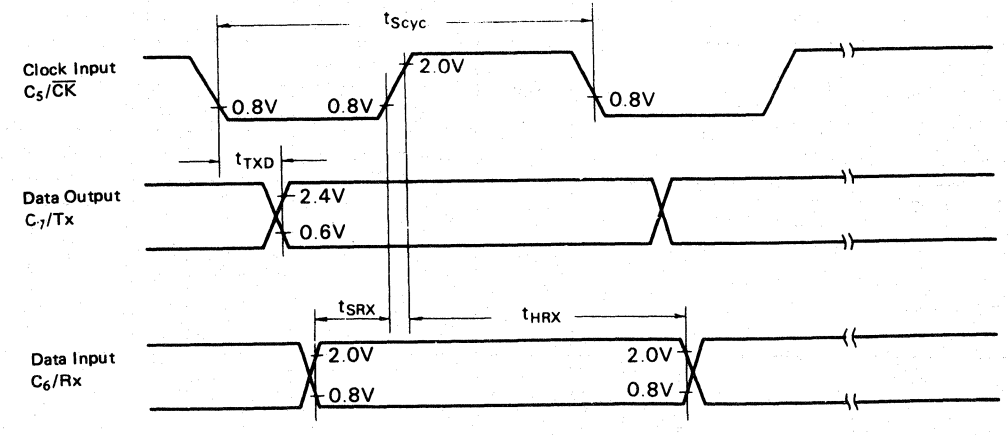
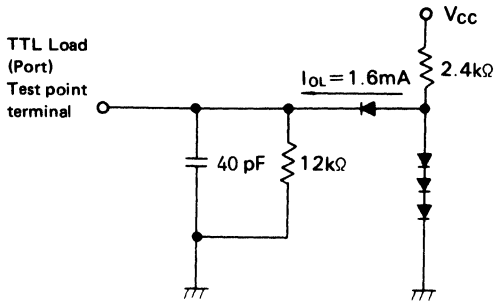


Fig. 5-2 SCI Timing (External Clock)



[NOTES] 1. The load capacitance includes stary capacitance caused by the probe, etc.
2. All diodes are 1S2074 Ⓗ.

Fig. 5-3 Test Load

5.2 Electrical Characteristics of HD6305X1/X2, HD6305Y1/Y2, HD63P05Y1

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply voltage	V_{CC}	-0.3 ∿ +7.0	V
Input voltage	V_{in}	-0.3 ∿ V_{CC} + 0.3	V
Operating temperature	T_{opr}	0 ∿ +70	°C
Storage temperature	T_{stg}	-55 ∿ +150	°C

[NOTE] These products have a protection circuit in their input terminals against high electrostatic voltage or high electric fields. Notwithstanding, be careful not to apply any voltage higher than the absolute maximum rating to these high input impedance circuits. To assure normal operation, we recommended V_{in}, V_{out}; $V_{SS} \leqq (V_{in}$ or $V_{out}) \leqq V_{CC}$

■ ELECTRICAL CHARACTERISTICS

● DC Characteristics (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C, unless otherwise specified.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input voltage "High"	$\overline{RES}$, $\overline{STBY}$	V_{IH}		V_{CC}-0.5	-	V_{CC}+0.3	V
	EXTAL			V_{CC}×0.7	-	V_{CC}+0.3	V
	Others			2.0	-	V_{CC}+0.3	V
Input voltage "Low"	All Input	V_{IL}		-0.3	-	0.8	V
Output voltage "High"	All Output	V_{OH}	I_{OH}=-200μA	2.4	-	-	V
			I_{OH}=-10μA	V_{CC}-0.7	-	-	
Output voltage "Low"	All Output	V_{OL}	I_{OL}=1.6mA	-	-	0.55	V
Input leakage current	$\overline{TIMER}$, $\overline{INT}$, D_1 ∿ D_7, $\overline{STBY}$	$\|I_{IL}\|$	V_{in}=0.5 ∿ V_{CC} - 0.5V	-	-	1.0	μA
Three-state current	A_0 ∿ A_7, B_0 ∿ B_7, C_0 ∿ C_7, ADR_0 ∿ ADR_{13},* $DATA_0$ ∿$DATA_7$, E*, R/$\overline{W}$*	$\|I_{TSI}\|$		-	-	1.0	μA
Current** dissipation	Operating	I_{CC}	f = 1MHz***	-	5	10	mA
	Wait			-	2	5	mA
	Stop			-	2	10	μA
	Standby			-	2	10	μA
Input**** capacity	All terminals	C_{in}	f = 1MHz, V_{in} = 0V	-	-	12	pF

* At standby mode

** V_{IH} min = V_{CC} - 1.0V. V_{IL} max = 0.8 V. For HD63P05Y1, I_{CC} of EPROM is not included.

*** The value at f = xMHz can be calculated by the following equation: I_{CC} (f = xMHz) = I_{CC} (f = 1MHz) multiplied by X.

**** HD63P05Y1 is MAX. 15pF

● AC Characteristics (V_{CC} = 5.0V ± 10%, V_{SS} =GND, T_a = 0 ∿ +70°C, unless otherwise specified.)

Item	Symbol	Test Condition	HD6305X1/X2/Y1/Y2 HD63P05Y1			HD63A05X1/X2/Y1/Y2 HD63PA05Y1			HD63B05X1/X2/Y1/Y2 HD63PB05Y1			Unit
			min	typ	max	min	typ	max	min	typ	max	
Cycle Time	t_{cyc}	Fig. 6-4	1	-	10	0.666	-	10	0.5	-	10	µs
Enable Rise Time	t_{Er}		-	-	20	-	-	20	-	-	20	ns
Enable Fall Time	t_{Ef}		-	-	20	-	-	20	-	-	20	ns
Enable Pulse Width ("High" Level)	PW_{EH}		450	-	-	300	-	-	220	-	-	ns
Enable Pulse Width ("Low" Level)	PW_{EL}		450	-	-	300	-	-	220	-	-	ns
Address Delay Time	t_{AD}		-	-	250	-	-	190	-	-	180	ns
Address Hold Time	t_{AH}		40	-	-	30	-	-	20	-	-	ns
Data Delay Time	t_{DW}		-	-	200	-	-	160	-	-	120	ns
Data Hold Time (Write)	t_{HW}		40	-	-	30	-	-	20	-	-	ns
Data Set-up Time (Read)	t_{DSR}		80	-	-	60	-	-	50	-	-	ns
Data Hold Time (Read)	t_{HR}		0	-	-	0	-	-	0	-	-	ns

● Port Timing (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305X1/X2/Y1/Y2 HD63P05Y1			HD63A05X1/X2/Y1/Y2 HD63PA05Y1			HD63B05X1/X2/Y1/Y2 HD63PB05Y1			Unit
			min	typ	max	min	typ	max	min	typ	max	
Port Data Set-up Time (Port A, B, C, D)	t_{PDS}	Fig. 6-5	200	-	-	200	-	-	200	-	-	ns
Port Data Hold Time (Port A, B, C, D)	t_{PDH}		200	-	-	200	-	-	200	-	-	ns
Port Data Delay Time (Port A, B, C)	t_{PDW}	Fig. 6-6	-	-	300	-	-	300	-	-	300	ns

● Control Signal Timing (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C, unless otherwise noted.)

Item	Symbol	Test Condition	HD6305X1/X2/Y1/Y2 HD63P05Y1			HD63A05X1/X2/Y1/Y2 HD63PA05Y1			HD63B05X1/X2/Y1/Y2 HD63PB05Y1			Unit
			min	typ	max	min	typ	max	min	typ	max	
$\overline{INT}$ Pulse Width	t_{IWL}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
$\overline{INT_2}$ Pulse Width	t_{IWL2}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
$\overline{RES}$ Pulse Width	t_{RWL}		5	-	-	5	-	-	5	-	-	t_{cyc}
Control Set-up Time	t_{CS}	Fig. 6-8	250	-	-	250	-	-	250	-	-	ns
Timer Pulse Width	t_{TWL}		t_{cyc} +250	-	-	t_{cyc} +200	-	-	t_{cyc} +200	-	-	ns
Oscillation Start Time (Crystal)	t_{OSC}	Fig. 6-8*	-	-	20	-	-	20	-	-	20	ms
Reset Delay Time	t_{RHL}	**	80	-	-	80	-	-	80	-	-	ms

* C_L = 22pF±20%, R_S = 60Ω max.

** 2.2 μF

● SCI Timing (V_{CC} = 5.0V ± 10%, V_{SS} = GND, T_a = 0 ∿ +70°C, unless otherwise noted)

Item	Symbol	Test Condition	HD6305X1/X2/Y1/Y2 HD63P05Y1			HD63A05X1/X2/Y1/Y2 HD63PA05Y1			HD63B05X1/X2/Y1/Y2 HD63PB05Y1			Unit
			min	typ	max	min	typ	max	min	typ	min	
Clock Cycle Time	t_{Scyc}	Fig. 6-9 Fig. 6-10	1	-	32768	0.67	-	21845	0.5	-	16384	µs
Data Output Delay Time	t_{TXD}		-	-	250	-	-	250	-	-	250	ns
Data Set-up Time	t_{SRX}		200	-	-	200	-	-	200	-	-	ns
Data Hold Time	t_{HRX}		100	-	-	100	-	-	100	-	-	ns

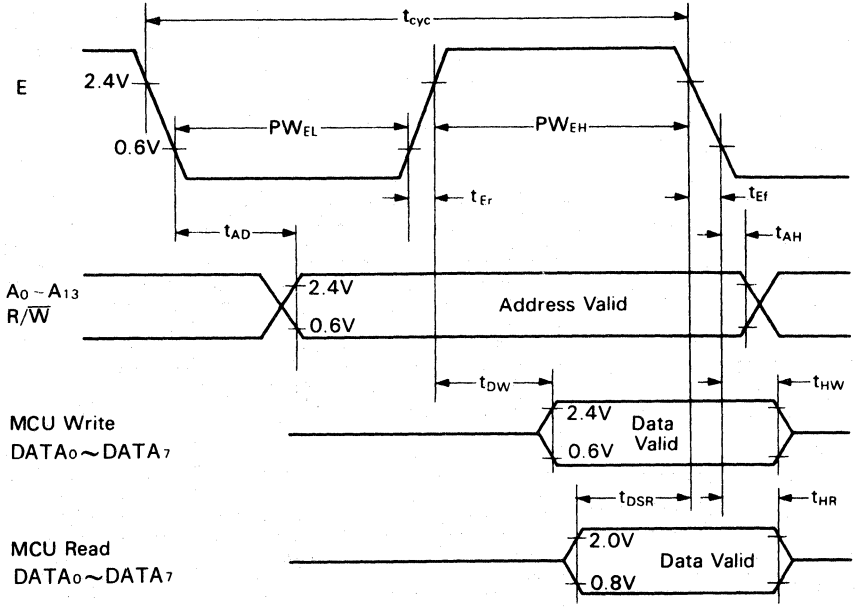


Fig. 5-4 Bus Timing

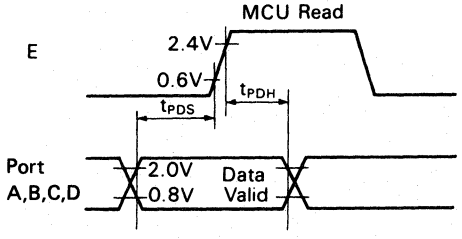


Fig. 5-5 Port Data Set-up and Hold Times (MCU Read)

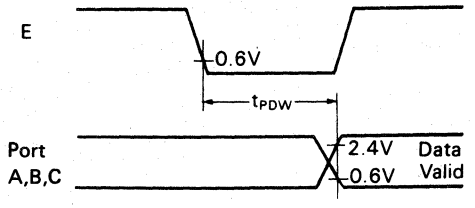


Fig. 5-6 Port Data Delay Time (MCU Write)

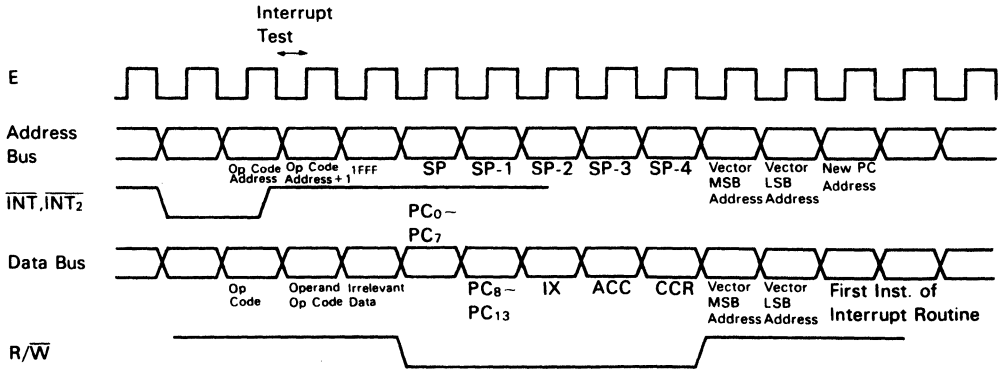


Fig. 5-7 Interrupt Sequence

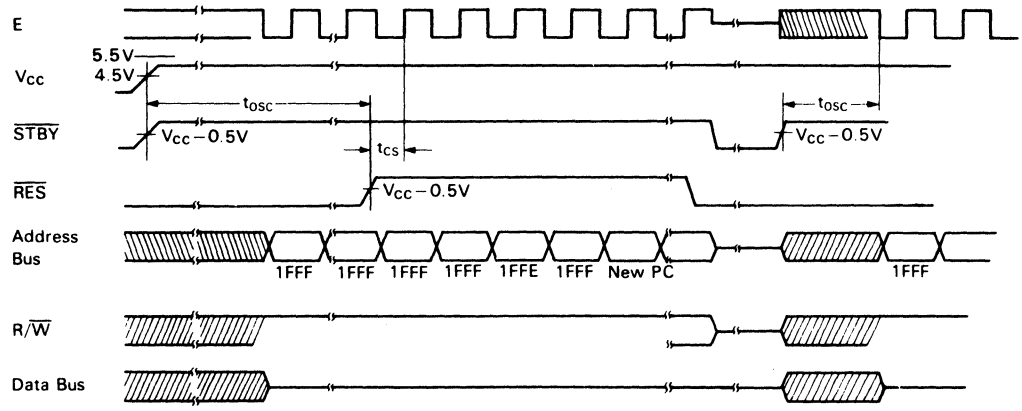


Fig. 5-8 Reset Timing

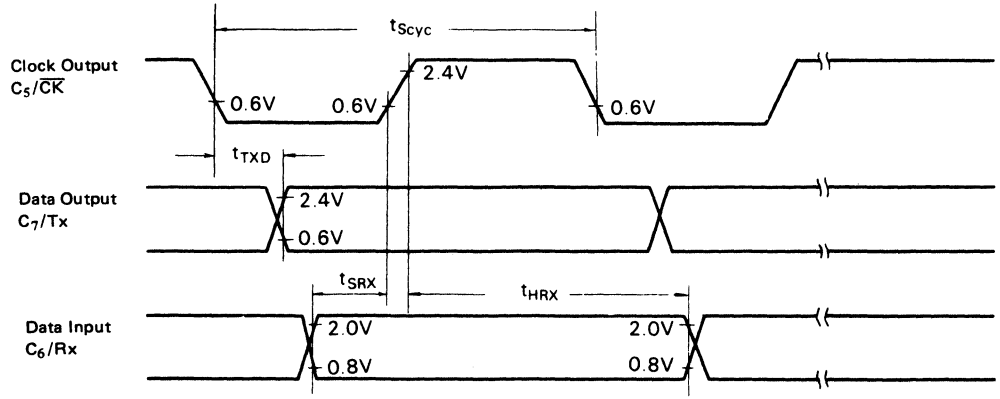


Fig. 5-9 SCI Timing (Internal Clock)

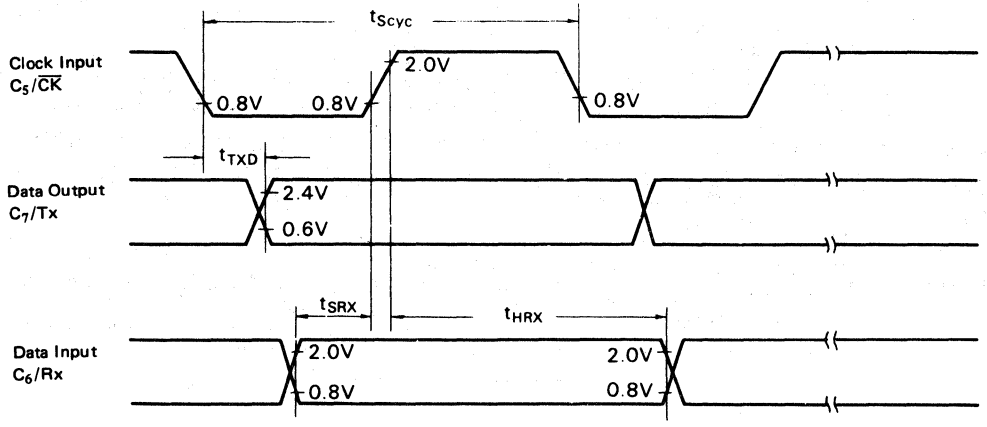
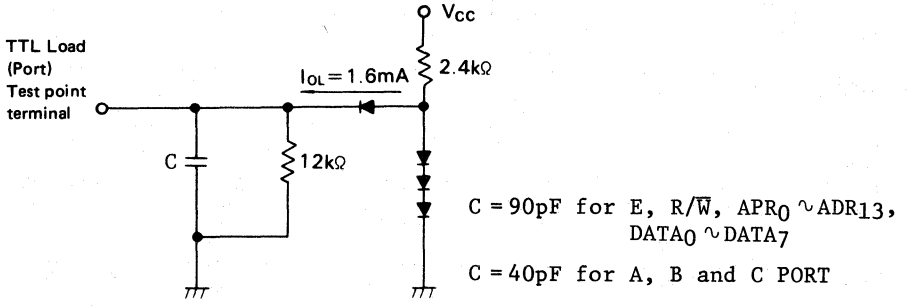


Fig. 5-10 SCI Timing (External Clock)



[NOTES] 1. The load capacitance includes stary capacitance caused by the probe, etc.
2. All diodes are 1S2074 Ⓗ.

Fig. 5-11 Test Load

6. ROM CODE ORDER METHOD

User's programs are mask programmed into ROM by Hitachi to be shipped as LSI. The users are requested to hand in three EPROMs in which the same contents are written, ordering specifications and list of the ROM contents.

Relationship between the address of the mask ROM and that of the EPROM is shown in Table 6-1. Write $FF for the unused address data of EPROM.

Table 6-1 Relationship between the Address of Mask ROM and that of EPROM

Type Name	Address of Mask ROM	Address of EPROM	Remarks
HD6305X0 HD6305X1	$1000 ~ $1FFF	$000 ~ $FFF	HN482732A or their equivalent
HD6305Y0 HD6305Y1	$0140 ~ $0FFF	$140 ~ $FFF	HN482732A or their equivalent
	$1000 ~ $1FFF	$000 ~ $FFF	HN482732A or their equivalent
HD6305Y0 HD6305Y1	$0140 ~ $1FFF	$0140 ~ $1FFF	HN482764 or their equivalent

APPENDIX

I. DESIGN PROCEDURE AND SUPPORT TOOL

Cross assembler and Hardware emulator, containing various kinds of computers, are available as supporting systems to develop user's programs. Hitachi will mask program user's programs into ROM to ship them as LSI.

Fig. I-1 shows a typical program design procedure and Table I-1 summarizes a set of system development supporting tool for the HD6305 MCU.

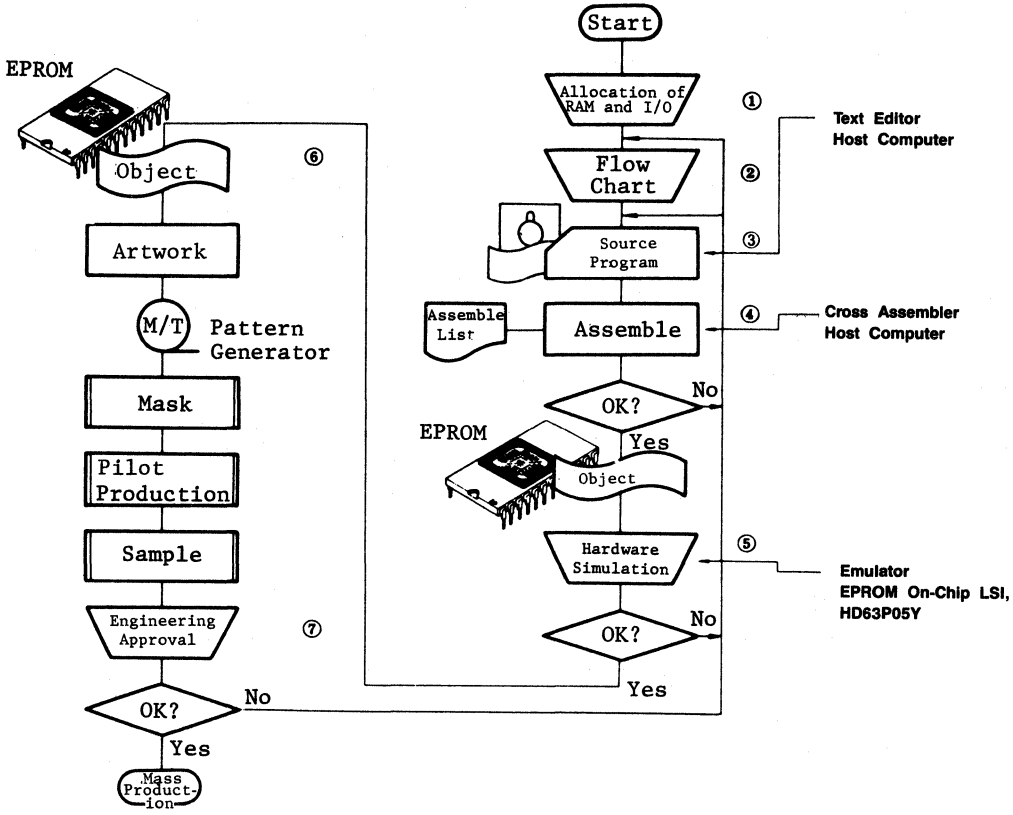


Fig. I-1 Program Design Procedure

The following explains the system development procedure.

1. Specify functional assignment of I/O pins and allocation of RAM area before starting programming.
2. Design flowchart to implement the functions and encode this flowchart with mnenic codes.
3. Punch the coded format onto cards or paper tape, or write it on a floppy disk. This set of coded form is a source program.
4. Assemble the source program to form an object program with cross system. Then check errors out.
5. Verify the program through hardware simulation with an emulator or EPROM on-package microcomputer.
6. Send the completed program in EPROM to Hitachi.
7. After Hitachi received user's specified ROM pattern and options, Hitachi will fabricate sample LSI for user's evaluation for the functions. If a user finds no problem in the sample LSI, Hitachi will start mass production of the LSI.

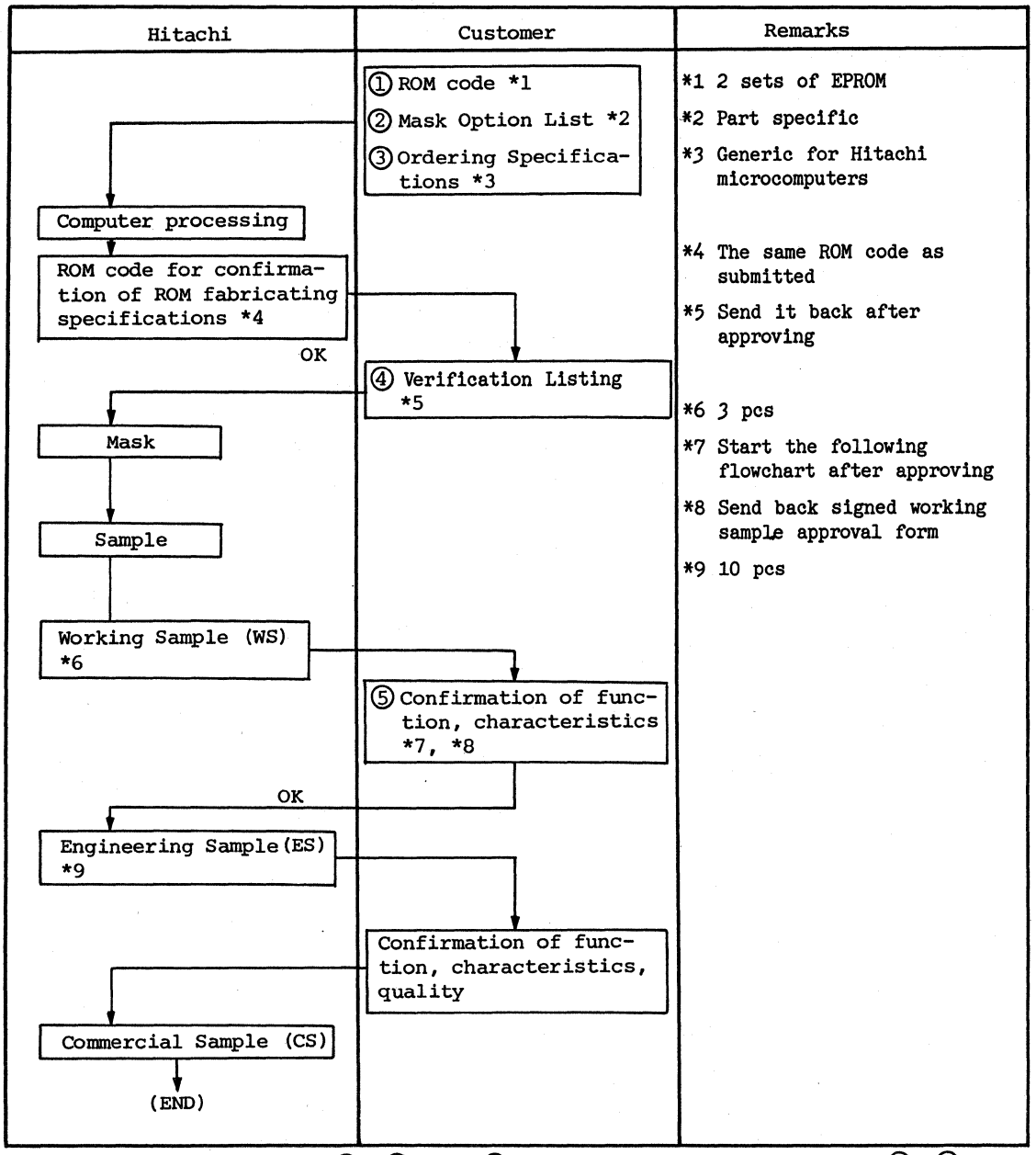
Table I-1 System Development Support Tool

Type No.	Emulator	EPROM On-Package LSI	IBM PC Cross Assembler
HD6305X0/X1/X2 HD6305Y0/Y1/Y2	H35MIX5 (HS35YEML05H)	HD63P05Y0/Y1	S35IBMPC

Single Chip Microcomputer ROM Ordering Procedure

(1) Development Flowchart

Single chip microcomputer device is developed according to the following flowchart after program development.



(Note) Please send in ①, ②, and ③ at ROM ordering, and send back ④, ⑤ after approving.

Device Development Flowchart

(2) Data you send and precautions

(a) Ordering specifications ----- Common style for all Hitachi single chip microcomputer devices. Please enter as for the followings. The format is shown in the next page.

- Basic ITEM
- Environment Check List
- Check List of attached data
- Customer

(b) ROM code ----- Please send in the ordering ROM code by 2 sets of EPROM the same contents are written. Enter ROM code No. in them. It is desirable to send in program list for easy confirmation of the program contents.

(3) Change of ROM code

Note that if you change the ROM code once sended in or other specification, the ROM must be developed from the beginning. The cost of mask charge should be provided again in this case.

(4) Samples and Mass production

(Working Sample) ---------- Sample for confirmation of the contents of ROM code and that of mask option. Normally 3 samples are sent, but not guaranteed as for reliability. Please evaluate and approve immediately because the following sample making and mass production are set about after obtaining your evaluation.

(Engineering Sample) ------- Sample for evaluating also reliability. 10 pcs are included

(Commercial Sample) -------- Samples for pre-production which maybe purchased separately.

(Mass Product) ------------ Products for actual mass production. Please enter the plan of mass production in full.

HD6305X0/X1, HD6305Y0/Y1
ORDERING SPECIFICATIONS

(1) GENERAL CHARACTERISTICS (Fill in blank space or appropriate box☐).

Device Type		Package Outline (See Section) 3,4.1	☐ DP-40 ☐ CP-44 ☐ FP-54
Application (be specific)		Options/Remarks:	
Customer ROM Code ID			
ROM Code Media	☐ EPROM ☐ ZTAT™	Must Specify:	Customer Programmed Start Address _____ Customer Programmed Stop Address _____
Operating Temperature	☐ Standard	☐ J Specification (-40°C to +85°C), if offered	
Remask	☐ Yes ☐ No	Previous Hitachi P/N_____	

(2) OPERATING CHARACTERISTICS

LSI Ambient Temperature	Average	°C	Target Level Of Reliability	☐ 500 Fit ☐ (_____) ☐ 1000 Fit	
	Range	°C- °C			
LSI Ambient Humidity	Average	%	Acceptable Quality Level	☐ 1.0% ☐ 0.4% ☐ 0.65% ☐ (_____)	
	Range	%- %			
Power On Duration	Average	Hours/Day	Remarks:		
Maximum Applied Voltage To LSI	Power Supply	Max V			
	I/O	Max V			

(3) ELECTRICAL CHARACTERISTICS

☐ Purchasing Specifications _____	☐ Hitachi's Standard Specifications Refer To Data Sheet: _____

For Hitachi Use Only

(4) CUSTOMER APPROVAL

Customer Name_____

PO#_____

Accepted By (print) _____

Accepted By (signature)_____

Date _____

(5) ROM CODE VERIFICATION

LSI Type No.	
Shipping Date of ROM To Customer	
Approved Date of ROM From Customer	

Section Five

• HD63L05 User's Manual

Foreword

The HD63L05 MCU is a 3V CMOS single-chip microcomputer for battery-operated systems with the same instruction set as the HD6805 and abundant internal functions with extremely low power consumption.

The HD63L05 MCU has an architecture that is suitable for the controller field in which bit input / output and status flag processing on software are important. This device contains the LCD drivers, analog-to-digital converter, etc, which are effective in the reduction of the number of system parts.

Section 5
HD63L05 User's Manual
Table of Contents

1. Overview

1.1 Features of the HD63L05 MCU

The HD63L05 MCU is a 3V CMOS single-chip microcomputer for battery-operated systems with the same instruction set as the HD6805S MCU and abundant internal functions with extremely low power consumption.

This device operates from a 3V power supply with extremely low power consumption and has two lower power consumption modes; a software-controlled half mode and a standby mode which is controlled through the input pin.

The HD63L05 MCU contains two internal oscillators, an 8-bit timer with a programmable 7-bit prescaler, 20 input/output ports, 4k byte ROM, 96 byte RAM, LCD drivers and an 8-bit analog-to-digital converter. By mask-option, the functions of the limited pins can be substituted with other functions such as LCD drivers, analog inputs, or digital outputs.

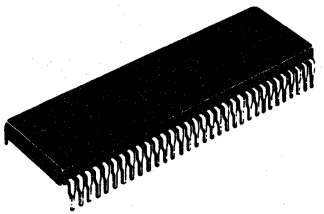
The 63L05 instruction set is compatible with 6805 code at the operation code level. The processor keeps the advanced features of the HD6805 family's instruction set, such as powerful bit manipulation.

Table 1-1 Features of the HD63L05 MCU

Function		HD63L05F1 MCU
Package		FP-80/DP-64S
ROM (Bytes)		3760
RAM (Bytes)		96
Input/Output (I/O)	I/O	20
	O	17 (mask-option)
LCD driver		static, 1/3 bias-1/3 duty max 17 segments
Analog-to-digital converter		8-bit A/D converter max 8 channels.
Timer		8-bit timer (7-bit prescaler)
Interruption		External × 1 Timer × 1 A/D × 1 Time Base × 1
Register	Accumulator A	8 bits × 1
	Index X	8 bits × 1
	Stack pointer SP	5 bits × 1
	Program counter PC	12 bits × 1
Oscillator	OSC1	RC, crystal
	OSC2	crystal, Internal

Package

(FP-80)
HD63L05F1F



(DP-64S)
HD63L05F1P

1.2 Block Diagram

Input signals and output signals of the MCU are described below.

- V_{CC}, V_{SS}
  Power is supplied to the MCU by using these two pins. V_{CC} has a voltage of 3.0V ± 0.8V, and V_{SS} is grounded.

- $\overline{INT}$
  This pin functions as an external interruption request input to the MCU. Refer to "2.8 Interruptions" for details.

- XTAL, EXTAL
  These pins are control input pins to the internal clock circuit. A crystal or resistor is connected to these pins in accordance with the stability of internal oscillation. Refer to "2.7 Internal Oscillator Options" for details.

- XIN, XOUT
  A crystal (32.768kHz) is connected for OSC2 oscillation.
  When OSC2 is not used, connect XIN to V_{CC}.

- TIMER
  This pin is an external input pin to count down the internal timer circuit. Refer to "2.4 Timer" for details. When timer is not used, connect to V_{CC}.

- $\overline{RES}$
  This pin resets the MCU. Refer to "2.5 Resets" for details.

- STANDBY (SB)
  This external input pin stops the MCU operation and holds data. Refer to "2.7 Internal Oscillator Options" for details.

- A/D Input Pins ($CH_1 \sim CH_8$)
  These input pins are for analog voltages necessary for A/D conversion. These can also be used as level check inputs under program control. Refer to "2.10 A/D Converter" for details.

- V_{RH}, V_{RL}
  Reference voltages for A/D conversion are applied to these two pins. Refer to "2.10 A/D Converter" for details.

- CC1, CC2

  These pins are not for user application.
  They should be left open.

- NUM

  This pin is not for user application. Connect it to V_{CC}.

- Input/Output Pins ($A_0 \sim A_7$, $B_0 \sim B_7$, $C_0 \sim C_3$)

  These 20 pins, consisted of two 8-bit ports and one 4-bit port, can be individually used as an input or as an output by programming the Data Direction Register. Refer to "2.9 Input/Output" for details.

- Liquid Crystal Driver Pins ($COM_1 \sim COM_3$, $SEG_1 \sim SEG_{17}$)

  $COM_1 \sim COM_3$ are for driving common electrodes, while $SEG_1 \sim SEG_{17}$ are for driving segments. $SEG_1 \sim SEG_{17}$ can also function as outputs. Refer to "2.11 LCD Circuit" for details.

- V_1, V_2

  These pins are for LCD driver. V_1 and V_2 are connected to V_{CC} through capacitors (0.1μF each).

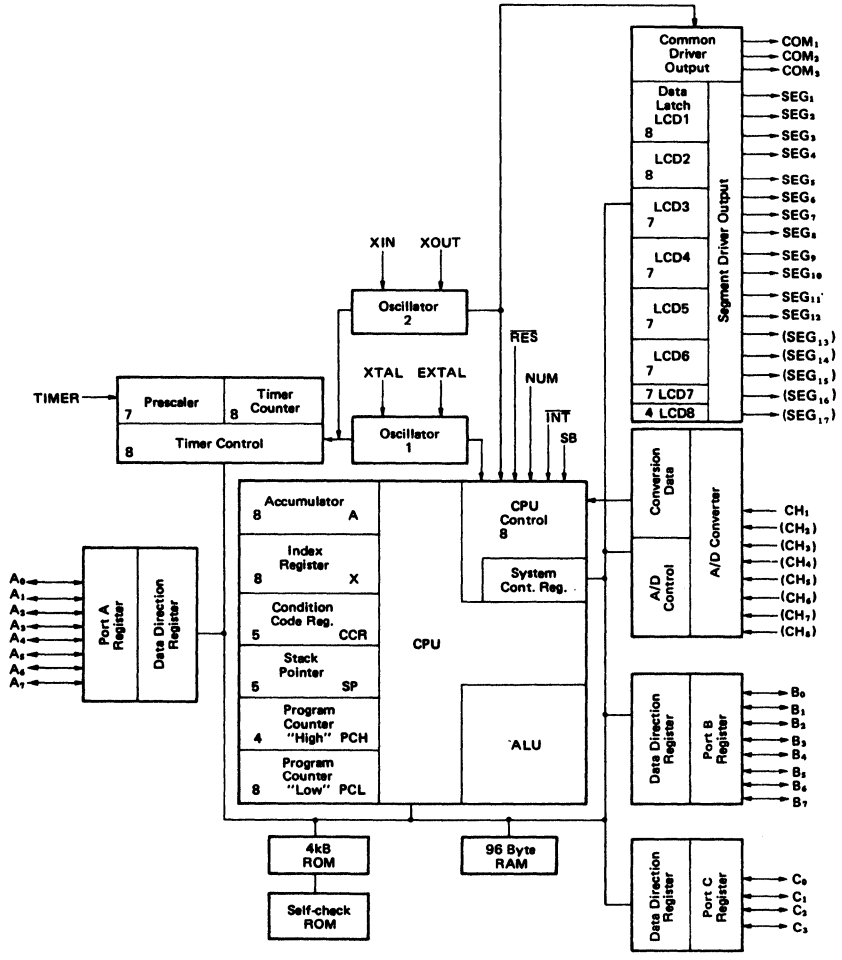
- V_{CH}

  This is an output pin from internal voltage regulator. A capacitor ($0.5\mu F \pm 10\%$) is connected between V_{CH} and V_{CC}.

- E

  System clock output (100kHz typ.)
  This NMOS open-drain output stays at "Low" level when the MCU is in halt status or standby. This pin should be left open.

HD63L05 Block Diagram is shown in Figure 1-1.



( ) Mask Option

Figure 1-1 HD63L05 Block Diagram

2. Architecture

2.1 Memory

The MCU memory is configured as shown in Fig. 2-1. During interruption processing, the contents of the MCU registers are pushed onto the stack in the order shown in Fig. 2-2. Since the stack pointer decreases during pushes, the low order byte (PCL) of the program counter is stacked first; then the high order four bits (PCH) are stacked. This ensures that the program counter is loaded correctly as the stack pointer increases when it pulls data from the stack. A subroutine call will cause only the program counter (PCH, PCL) contents to be pushed onto the stack.

Cautions:

(1) It is not possible to change the contents of the Write Only Register (e.g. the Data Direction Register of the I/O port) of the HD63L05 MCU by applying the Read/Modify/Write instructions, BSET, or BCLR.

(2) For prevention of the system from running wild, do not address the Not Used area of the memory map.

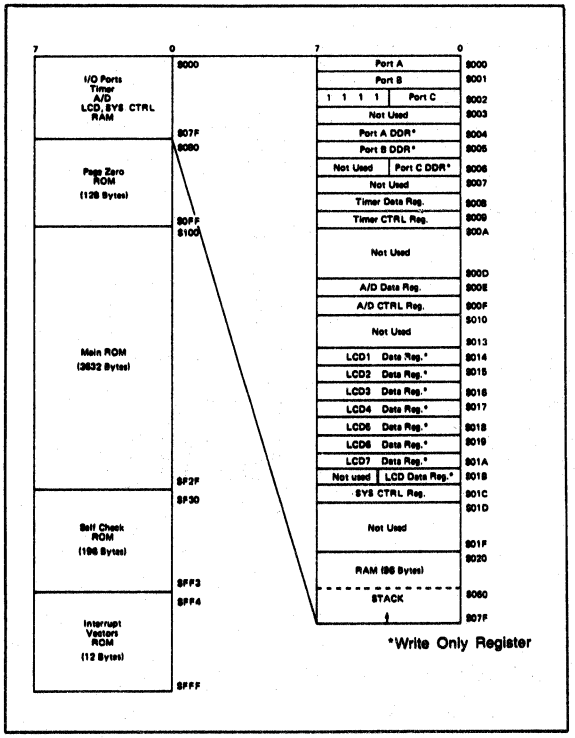


Figure 2-1 Memory Map

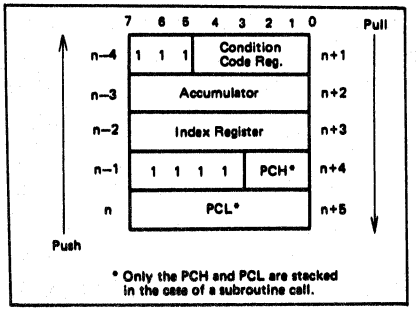


Figure 2-2 Interrupt Stacking Order

2.2 Registers

The MCU has five registers available to programmers. They are shown in Figure 2-3 and are explained in the following paragraphs.

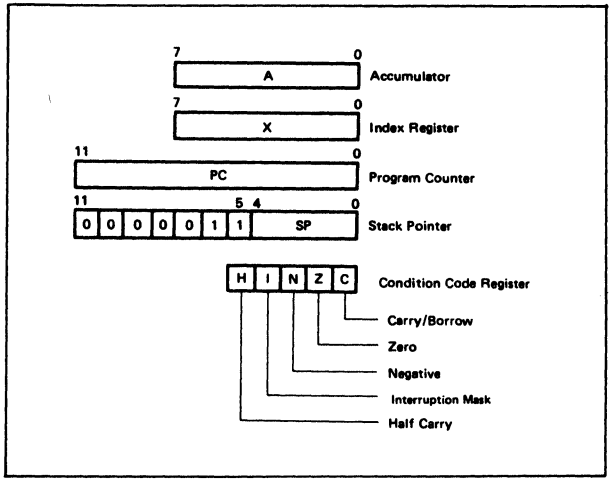


Figure 2-3 Programming Model

(1) Accumulator (A)

An 8-bit register of general purpose to hold operands, results of arithmetic calculations, or data manipulations.

(2) Index Register (X)

An 8-bit register for the indexed addressing mode. It contains an 8-bit address that may be added to an offset value to create an effective address. The index register is available for limited calculations and data manipulations when executing read/modify/write instructions. When not required by a code sequence being executed, the index register can be used as a temporary storage register.

(3) Program Counter (PC)

A 12-bit register that contains the address of the next instruction to be executed.

(4) Stack Pointer (SP)

A 12-bit register that indicates the address of the next stack area on the stack. Initially, the stack pointer is set to address $07F. It is decreased as data is pushed onto the stack and increased as data is pulled from the stack. The most significant bits of the Stack Pointer are fixed to 0000011.

During the MCU is reset or executing the reset stack pointer (RSP) instruction, the Stack Pointer is set to location $07F. Since subroutines and interruptions are designed to use memory area up to address $061 for stacking, programmers can use area up to 15 levels for subroutine.

(5) Condition Code Register (CCR)

A 5-bit register in which each bit is used to indicate the results of the instruction just executed. These bits can be individually tested by branch instruction with condition. Each condition code register bit (H, I, N, Z, C) is explained in the following paragraphs.

(a) Half Carry (H)

This bit is set if a carry occurs between bit 3 and bit 4 during arithmetic operations (ADD, ADC).

(b) Interruption (I)

This bit is set to mask the internal interruption (INT) and external interruption ($\overline{INT}$). If an interruption occurs while this bit is set, the interruption is latched and will be processed as soon as the interruption mask bit (I) is reset.

NOTE: CLI (clear interrupt mask bit) is used to allow the interruption from the instruction after next. SEI (set interrupt mask bit) masks the interruption from next instruction.

(c) Negative (N)

This bit indicates that the result of the last arithmetic, logical of data manipulation is negative (bit7 in result logical "1").

(d) Zero (Z)

This bit indicates that the result of the last arithmetic, logical of data manipulation is "0".

(e) Carry/Borrow (C)

This bit indicates that a carry or borrow has occurred during the last arithmetic operation. This bit is also affected by bit test and branch instructions, shift instruction, and rotates instruction.

2.3 System Control Register

In addition to the registers for program operation explained above, there is a register that controls system operation. It's configuration is shown in Figure 2-4.

(1) Time Base Interruption Request Flag (TB INT)
Stores an interruption request from the time base which is selected by the TB select bit.
If the TB MASK bit or I (Interruption bit in the CCR) is set, the interruption request is not acknowledged. Only logical "0" can be written into this bit by program.

(2) Time Base Interruption Mask (TB MASK)
If this bit is set, any interruption request from the time base is masked.

(3) Time Base Select Bit (TB SELECT)
Selects the time base. In logical "1", 1-second cycle time base is acknowledged. In logical "0", 1/16 second cycle time base is acknowledged.

(4) Time Base Reset Bit (TB RESET)
Resets the frequency divider behind the 32kHz oscillator. When this bit is set, one shot reset pulse is generated by the hardware. Then it resets the frequency divider and, after that, the frequency divider restarts. When this bit is read, the CPU always reads this bit as logical "0".
The frequency divider also provides the system clocks to the A/D converter and LCD drivers. Thus, it is needed to pay attention when "TB RESET" is used. Whenever this bit is read, "0" is transmitted.

(5) Halt (HALT)
Halts the CPU operation. When this bit is set, the registers are saved onto the stack in the same sequence as interruption processing. After all registers have been saved, the CPU operation halts to prepare for processing interruption. When the CPU acknowledges interruption, such as external interruption or time base interruption, the bit is reset and the CPU restarts operating. By using the Halt function with Time Base Interruption, the CPU can operate intermittently.

(6) EXT

In selecting the form of output port by DUTY select bit or mask-option, ϕWRITE is available every time data is written into LCD register in the case that this bit is initialized to "1". ϕWRITE functions as the clock for writing data when transferring the data of LCD register to the outside. Normally, EXT is reset.

(7) Duty Select Bit (DUTY)

The MCU contains switching circuits for static drive signal and output ports in order to select driving form of LCD drive circuit. The LCD drive circuit is based on 1/3 bias - 1/3 duty. Refer to "2.11 LCD Circuit" for details.

NOTE: The EXT bit and the DUTY bits have to be initiallized in a 1 m second from the beginning of the system reset when the static drive signal or output port is selected.

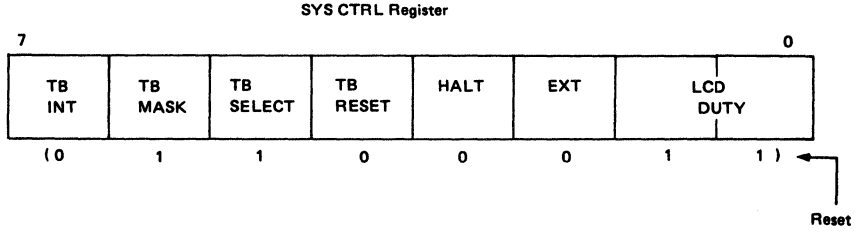


Figure 2-4 System Control Register Configulation

2.4 Timer

The MCU timer block diagram is shown in Figure 2-5. The 8-bit counter is loaded under program control and counts down toward "0" as soon as the clock input is applied. When the timer reaches "0", the timer interruption request bit (bit 7) in the Timer Control Register is set. The MCU responds to this interruption by saving the present MCU state onto the stack, fetching the timer interruption vector from locations $FF8 and $FF9, and executing the interruption routine. The timer interruption can be masked by setting the timer interruption mask bit (bit 6) in the Timer Control Register. The interruption bit (I bit) in the Condition Code Register also masks a timer interruption.

Either an external clock signal or the internal clock signal function as the clock input to the timer (Note: External clock signal is applied to timer input pin for the external source). When the internal clock functions as the source, the clock input is gated by the input applied to the timer input pin; this permits the MCU to measure its pulse width with ease. There are two types of internal clock signals (E and ϕ_{32k}) to allow timer operation when the CPU is halted. (E is active when OSC1 is not stopped.) These clock signals are under program control.

A 7-bit prescaler is provided to extend the timing of the timer. The number of prescaling counts can be program selected by the lower 3 bits within the Timer Control Register. The timer continues to count when it reaches "0" and its present count can be monitored at any time by reading the Timer Data Register.

When reset, the prescaler and the counter are all initialized to logical "1". The timer interruption request bit is cleared and the timer interruption mask bit is set. Only logical "0" can be written into this bit by program.

2.5 Resets

The MCU can be reset by pulling the external reset input ($\overline{RES}$) "LOW".

Immediately after power up, a minimum of 150 milliseconds is needed before allowing the reset input to go "High" to allow the internal oscillator (OSC1) to be stabilized. Connecting a capacitor to the $\overline{RES}$ input as shown in Figure 2-7 will provide sufficient delay.

2.6 Self Check

The self check capability of the MCU provides an internal check to determine if the LSI is functional. Connect the MCU as shown in Figure 2-9, and the LSI function can be checked by an oscillation of port C bit 3 (approximately 0.5Hz). This self check capability also provides the internal state specification of the MCU to measure the LSI current. After a system reset, the MCU goes into each current measurement mode by the combination of the control switches. The LSI current can be measured when the NUM is returned to V_{CC} after setting the current mode.

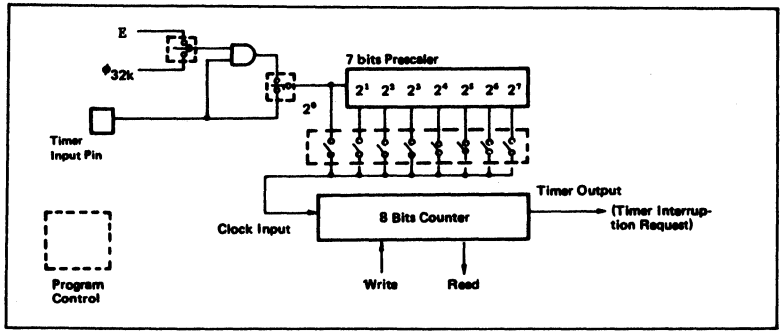


Figure 2-5 Timer Block Diagram

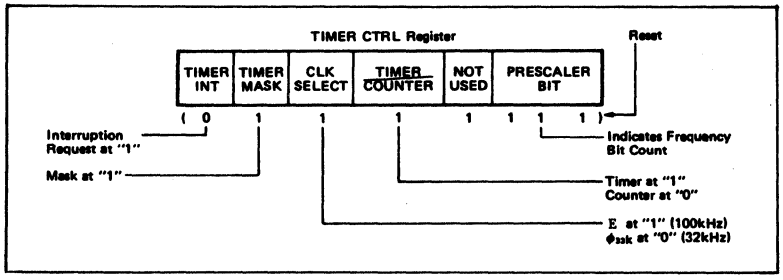


Figure 2-6 Timer Control Register Configuration

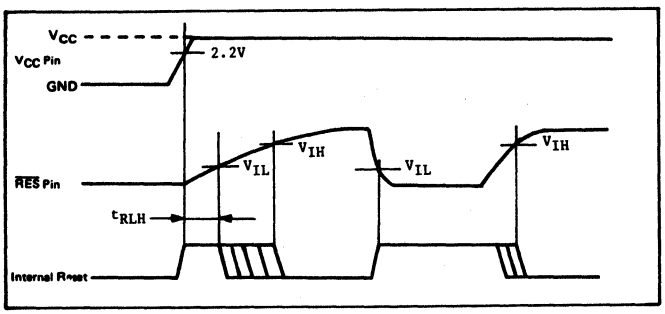


Figure 2-7 Power and Rest Timing

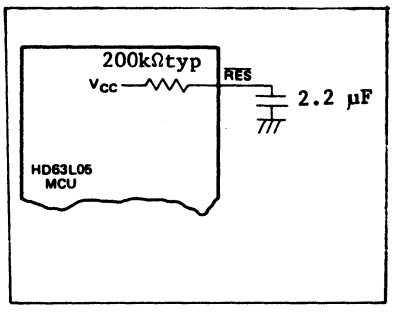
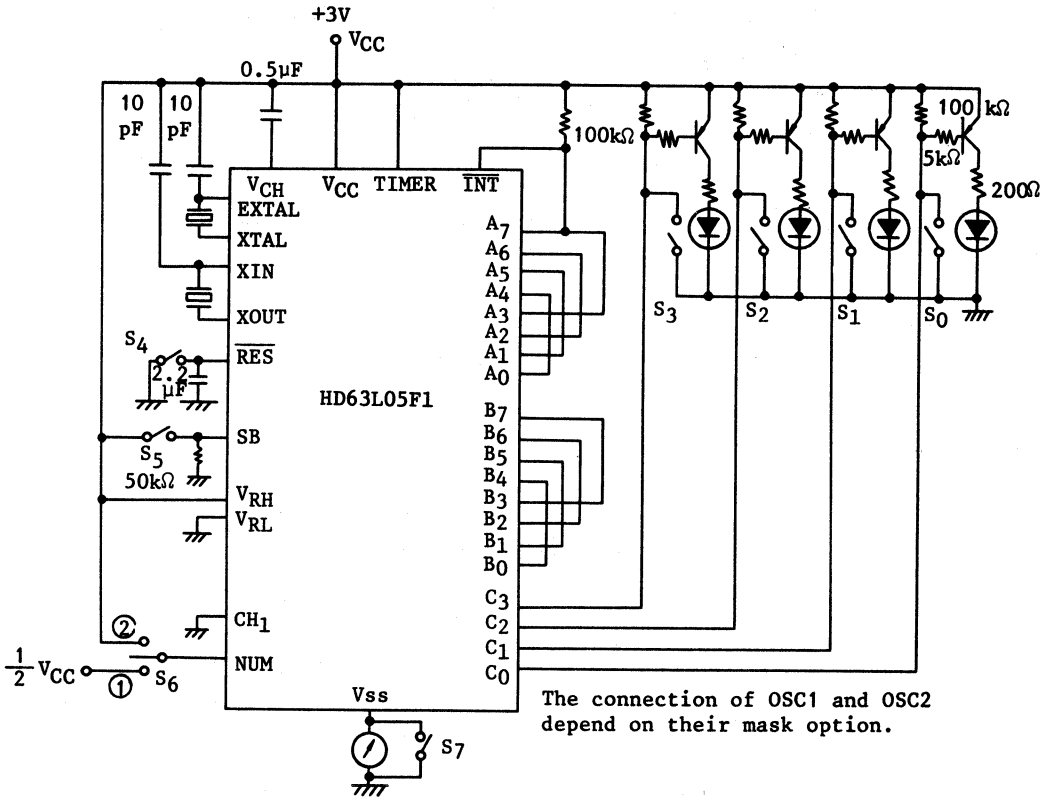


Figure 2-8
Input Reset Delay Circuit

When the MCU is reset, all I/O ports go into "High" impedance state. Peripheral circuits, such as Timer, A/D converter, or Time Base, are set to predetermined value.



		Selection of Switch							
		S_0	S_1	S_2	S_3	S_4	S_5	S_6	S_7
LSI Function		×	×	×	×	○→×	×	①	○
LSI Current	During operation	○	×	×	×	○→×	×	①→②	×
	Halt	○	○	○	×	○→×	×	①→②	×
	A/D	○	○	×	×	○→×	×	①→②	×
	Stand-by	○	○	○	×	○→×	×→○	①→②	×

× : OFF ○ : ON → : Change the state

Figure 2-9 Self Check Connections

2.7 Internal Oscillator Options

The MCU contains two oscillators: oscillator 1 for system clock supply, and oscillator 2 for time base interruption and LCD drivers.

(1) Oscillator 1 (OSC1)

The internal oscillator circuit can be driven by connecting a crystal or a resistor between XTAL and EXTAL. A manufacturing mask option is available for selecting better matching between the external components and the internal oscillator at start up.

Figure 2-10 shows the connection of oscillator circuit.

A resistor selection graph to determine oscillation frequency of RC oscillator is given in Figure 2-11.

When RC oscillation, the oscillation can be stopped by mask option in halt status. This saves consuming power when halting.

(2) Oscillator 2 (OSC2)

Clocks are supplied to Timer, Time Base, A/D Converter, and LCD Drive Circuit from OSC2 by connecting 32.768 kHz of crystal to XIN and XOUT. OSC2 operates even in the halt status, which permits the MCU to implement intermittent action indicating LCD status. In standby status, this frequency divider stops, while OSC2 keeps on operating to maintain restart time after it has been relieazed from the instruction.

Figure 2-12 shows the connection between OSC1 and 2, while Figure 2-13 and 2-14 shows the relations between them.

Mask options are available for deciding if OSC2 is used or not. When OSC2 is not used, OSC1/12 will be provided to the peripheral circuit as ϕ_{32k}.

In this case, fix Xin pin to V_{CC}.

Figure 2-14 shows the combinations of selectable mask-options and the corresponding system operation statuses. Refer to this figure in selecting mask-options on oscillation circuit.

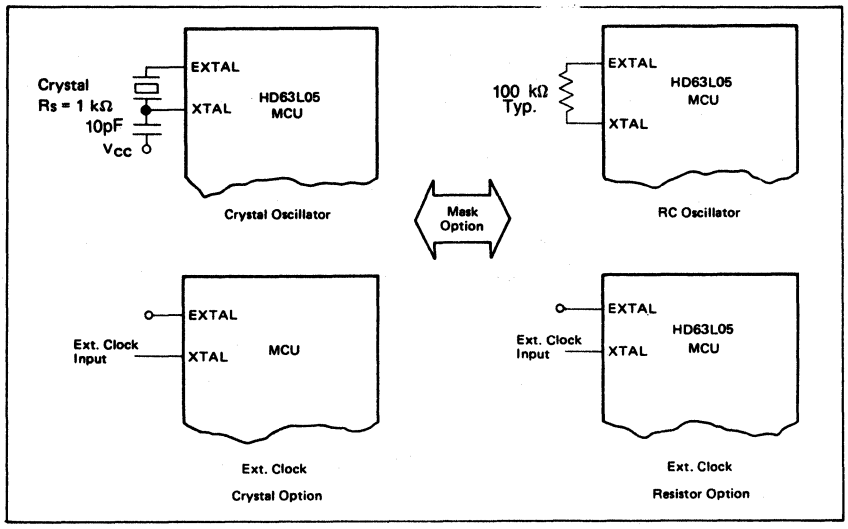


Figure 2-10 Mask Option for Oscillator1

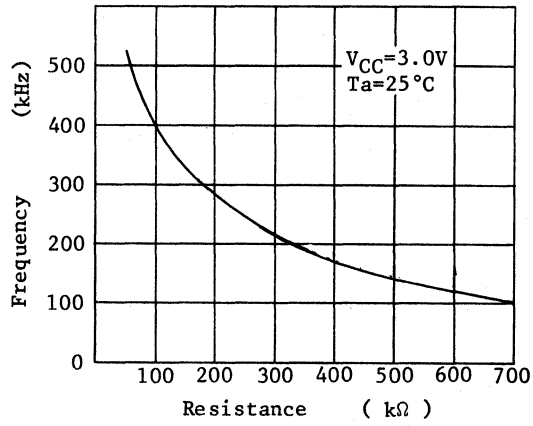


Figure 2-11 Typical Resistor Selection Graph

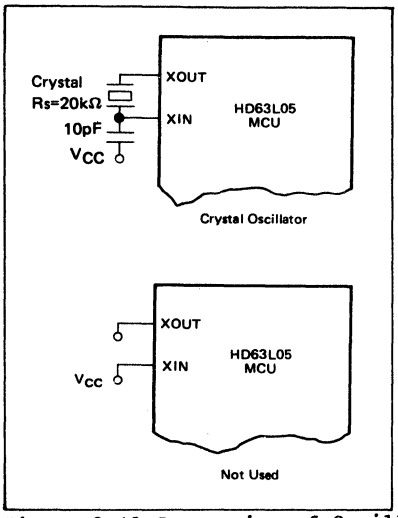


Figure 2-12 Connection of Oscillator2

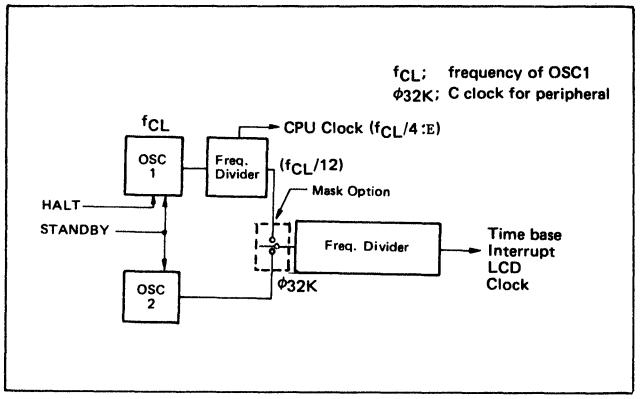


Figure 2-13 Relation between Oscillator1 and Oscillator2

Combinations of Mask Option				Delay Time(s)				System Operation		
OSC1 Type	OSC2	Other Options		0	1/16	1/2	1	Fault / System	At Halt	At Stand-by
*1 Crystal Oscillation	*2 Absent	Stand-by	Used	×	×	o	o	OSC1	o	×
								CPU	×	×
								Peripheral	o	×
			Not Used	o	o	o	o	OSC1	o	×
								CPU	×	×
								Peripheral	o	×
	Present	Stand-by	Used	×	×	o	o	OSC1	o	×
								CPU	×	×
								Peripheral	o	×
			Not Used	o	o	o	o	OSC1	o	×
								CPU	×	×
								Peripheral	o	×
*3 CR Oscillation	*2 Absent	OSC1 Oscillation at Stand-by	OSC1 Stops	o	×	×	×	OSC1	×	×
								CPU	×	×
								Peripheral	o	×
			OSC1 runs	o	o	o	o	OSC1	o	×
								CPU	×	×
								Peripheral	o	×
	Present	OSC1 Oscillation at Stand-by	OSC1 Stops	×	×	×	×	-	-	-
			OSC1 runs	o	o	o	o	OSC1	o	×
								CPU	×	×
								Peripheral	o	×
o ... Selectable × ... Unselectable								o ... run × ... stop		

Note *1 In selecting crystal oscillation, OSC1 cannot be stopped in halt status.

*2 Delay time cannot be accurate without OSC2.

*3 In selecting CR oscillation, stand-by operation is available for any options.

Figure 2-14 Oscillator2 Mask-option and System Operation

(3) Stand-by

The MCU goes into stand-by mode when STAND-BY is pulled "High". In this status, all circuits, except OSC2, stop functioning holding current status. (Time Base, Timer, A/D Converting Data are not held). OSC1 restarts oscillating by pulling STAND-BY "Low". Then the CPU restarts its operating sequence after delay time for oscillation stability has passed.

CPU operation must be masked after the CPU has been released from stand-by mode until OSC1 oscillation is stabilized. This is because OSC1 stops functioning in the stand-by mode. For this reason, the HD63L05 MCU contains a frequency divider as a delay circuit, which permits the CPU to gain the accurate delay time if OSC2 is provided. Users should be careful when OSC2 is not provided, because the CPU starts functioning after ϕ_{32k} has been divided by the internal divider corresponding to the selected delay time. This frequency circuit also operates just after reset. Don't input STAND-BY again during the delay time, because it will cause the system to malfunction. As a result, delay time is calculated as:

$$\left[(32768\ \text{Hz})/\left(\frac{\text{OSC1}}{12}\right)\right] \times \text{delay time.}$$

2.8 Interruptions

There are six different interruptions to the MCU: external interruption through external interruption pin ($\overline{\text{INT}}$), internal timer interruption, interruption by termination of A/D conversion, time base interruption (2 types), and software interruption by an instruction (SWI).

When any interruption occurs, processing is suspended and the present MCU status is pushed onto the stack. The interruption mask bit (I) in the Condition Code Register is set, the address of the interruption routine is obtained from the appropriate interruption vector address, then the interruption routine is executed. When RTI instruction has been completed, the MCU continues processing recovering the MCU status by RTI instruction. Table 2-1 provides a listing of the interruptions, their priority, and the vector addresses. In case a number of interruptions occurred simultaneously the MCU processes them according to the priority.

Figure 2-15 shows the system operation flow, in which the portion surrounded with dot-dash line shows an interruption execution sequence.

Note: A clear interruption bit instruction (CLI) allows to suspend the processing of the program by an interruption after execution of the next instruction while a set interruption bit instruction (SEI) inhibits any interruptions before execution of the next instruction.

When a mask bit of a control register is cleared by an instruction, interruption is allowed before execution of the next instruction.

(1) Acknowledging interruptions in HALT Status

In HALT status, the CPU stops functioning, while the peripheral circuit are operating. When an interruption is acknowledged, the CPU reads the head address of the interruption routine from the vector address corresponding to the interruption condition. Then the CPU executes interruption service to meet the interruption condition.

(2) Acknowledging interruptions in Stand-by Status

In Stand-by status, the system stops with power is supplied to it. Any interruption request (including $\overline{\text{RES}}$) is therefore, ignored.

Table 2-1 Interruption Priority

Interruption	Priority	Vector Address
$\overline{\text{RES}}$	1	$FFE, $FFF
SWI	2	$FFC, $FFD
$\overline{\text{INT}}$	3	$FFA, $FFB
TIMER	4	$FF8, $FF9
A/D	5	$FF6, $FF7
TIME BASE	6	$FF4, $FF5

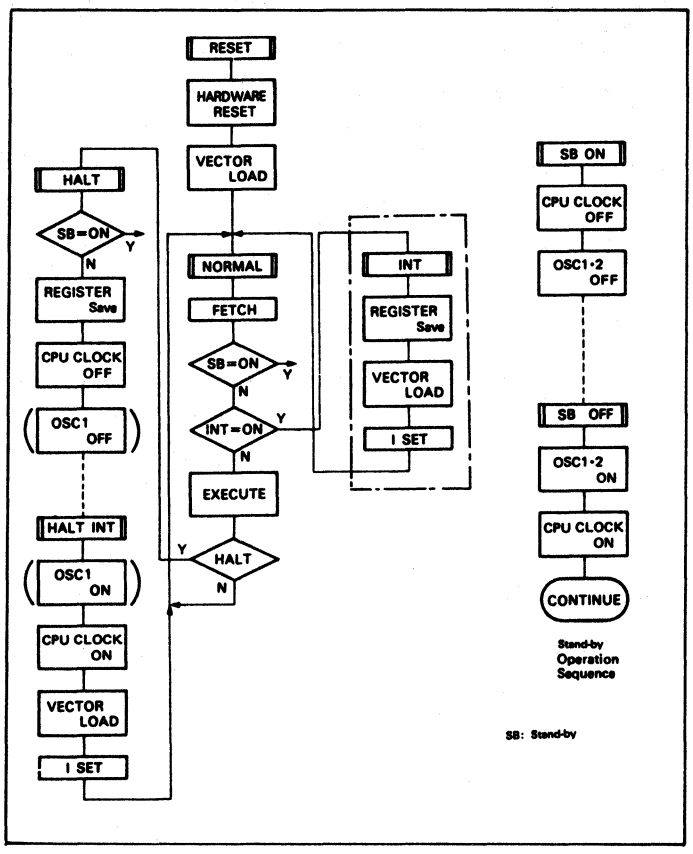


Figure 2-15 System Operation Flowchart

2.9 Input/Output (Port A,B,C)

There are 20 input/output pins. All pins can be programmed either as inputs or outputs under software control of the Data Direction Register. When programmed as outputs, all I/O pins read output data of the programmed logic level even if the actual output level is affected by output load (Refer to Figure 2-16).

(1) Port Configuration

Figure 2-17 shows the port configuration.

Each pin can individually select four types of configurations by mask option. The following explains these.

A: This port configuration is of general type.

B: This configuration contains pull-up resistance to prevent input from floating when using the port as an input. Select this configuration in such cases as connecting the switch to port directly.

C: This port configulation is for driving key matrix. If key matrix is driven under this port configuration, the CPU can read the necessary data precisely, even if more than two keys are pushed simultaneously.

D: This port configuration is NMOS open drain output, which is convenient for removing wired OR or driving transistor circuit. Users should be careful of the maximum output voltage of open drain output. Output voltage of open drain output covers from "0V" to "V_{CC}".

2.10 A/D Converter

The MCU contains an 8-bit A/D converter based on the resistor ladder system. Figure 2-18 shows its block diagram.

The "High" side of reference voltage is applied to V_{RH}, while the "Low" side of reference voltage is applied to V_{RL}.

The reference voltage, which is divided by resistors into voltages matching each bit, is compared with the analog input voltage in order to implement A/D conversion.

As the analog input voltage is applied to the MOS gate of the comparator through the analog multiplexer, this voltage comparison system achieves high input impedance.

A/D converter provides two modes; A/D conversion mode, and program comparison mode.

(1) A/D Conversion Mode

This mode allows the MCU to perform A/D conversion automatically. A/D conversion starts when A/D operating mode selection bit is initialized to "0", and "1" is programmed to A/D conversion flag. When A/D conversion has been completed, A/D conversion flag is automatically reset while A/D conversion interruption request flag is set. In response to this interruption request, the MCU loads the head address of A/D conversion interruption routine, then executes the interruption routine. A/D interruption can be masked by setting A/D interruption mask bit to "1".

The MCU requires about 2 ms to perform A/D conversion itself. However, the MCU requires 4 ms at most to complete the whole A/D conversion process, because clock generator is multiplexed with Time Base and LCD drive circuit.

(2) Program Comparison Mode

This mode allows the MCU to compare the predetermined standard level programmed into the A/D data register with the analog input level. When ''1'' is programmed in A/D operating mode selection bit, the MCU enters into program comparison mode. Result of the comparison is transmitted to comparison result output bit by programming the digital value corresponding to standard level and by reading A/D control register. ''1'' is transmitted when analog input level is higher than standard level, while ''0'' is transmitted when analog input level is lower. Note that the MCU wastes more consuming power in this mode, because supply voltage is supplied with the comparator constantly.

A/D converter continues functioning even when halting. So halt power does not decrease when A/D converter is functioning. In stand-by mode, conversion is suspended and comparator source power is off.

A/D converter sometimes reperforms A/D conversion after being recovered from stand-by mode. However, its result cannot be assured.

(3) A/D Interruption Request Flag (A/D INT)

The A/D INT bit is set to logical "1" after A/D conversion has been completed, and is cleared by system reset. A/D INT bit should be reset by program under interruption subroutine. Only logical "0" can be programmed into this bit.

(4) A/D Interruption Mask (A/D MASK)

If this bit is set, interruption request from the A/D converter is masked.

(5) A/D Converter Flag (CNV)

Set this bit to logical "1" in order to implement A/D conversion. During conversion, data of this bit holds "1". The bit is automatically reset to "0" when the A/D conversion has been completed. In A/D conversion, supply voltage is applied to the comparator only when CNV="1". The digital data, which is obtained by A/D conversion, is held in the A/D data register. This data is reset when the CNV is set to "1" again.

(6) A/D Operation Mode Selection Bit (Auto/Program)

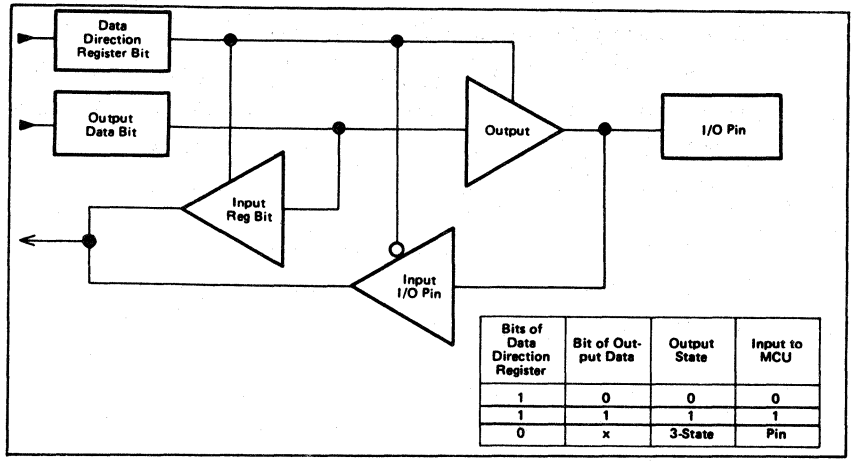
This bit selects either auto-run 8-bit A/D conversion or 8-bit programmed comparator operation (Auto 8 bits A/D conversion at "0"). When this bit is reset to "0", the MCU selects the 8-bit A/D conversion mode, then performs A/D conversion according to the value represented by the A/D conversion flag. When this bit is set to "1", the MCU selects Program Compare Mode, which allows the program to compare the value between A/D data register and analog input level.

(7) COMP OUT

This bit transmits the result of comparator operation between the standard level at A/D register and analog input value (Logical "1" means that input voltage is higher than programmed reference voltage).

(8) MPX

This bit selects 8-channel analog inputs. The multiplexer is an analog switch based on CMOS.



Bits of Data Direction Register	Bit of Output Data	Output State	Input to MCU
1	0	0	0
1	1	1	1
0	x	3-State	Pin

Figure 2-16 Port I/O Circuit

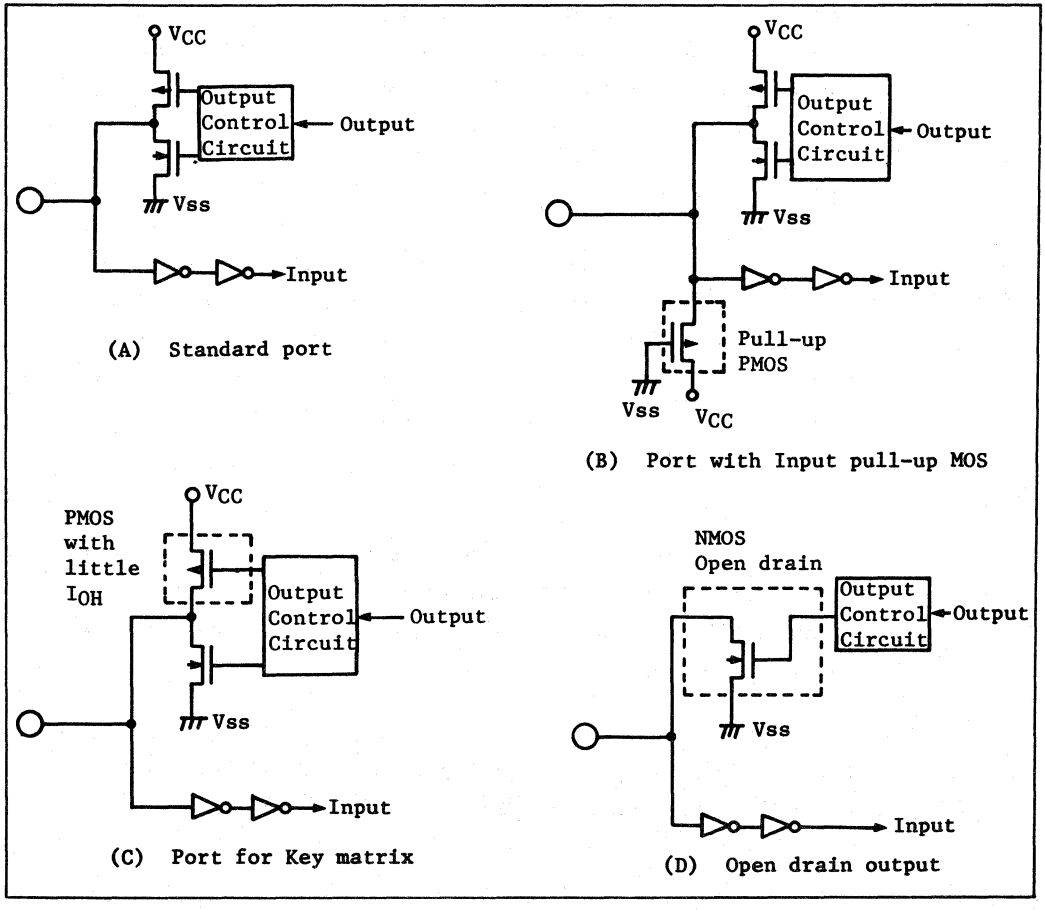


Fig. 2-17 I/O Port Configuration

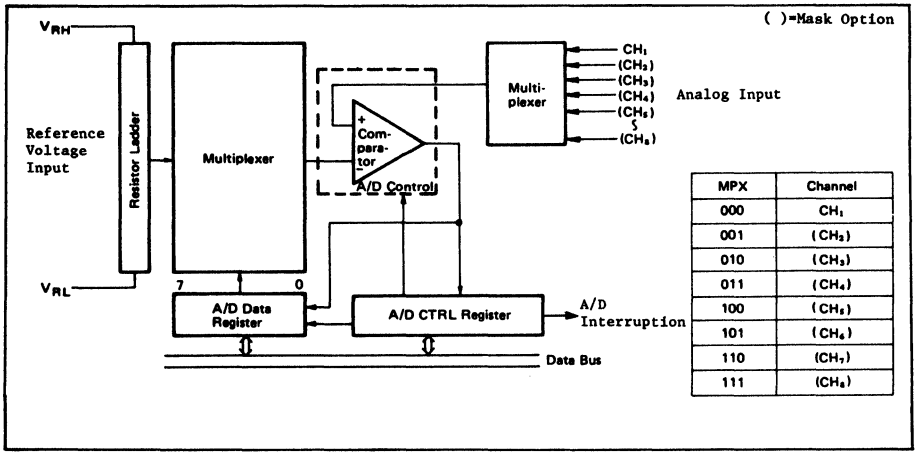


Figure 2-18 8 Bits A/D Converter Block Diagram

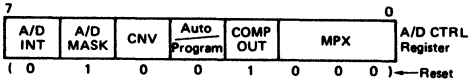


Figure 2-19 A/D Control Register Configuration

2.11 LCD CIRCUIT

The system configuration of the LCD circuits is shown in Figure 2-20. 1/3 bias-1/3 duty drive, static drive, or output port configuration can be selected in LCD circuit. This selection can be specified by both mask option and duty select bit of system control register.

Segment data for display are stored in data registers (LCD1 to LCD8). Since the circuits are connected to the output pins through pin location block, users can specify a combination of datas to be multiplexed with the segment output pins.

NOTE: In selecting LCD other than 1/3 bias-1/3 duty LCD, specify the Duty bit within 1 ms since reset has been vectored.

The bit data of the LCD register (LCD1 to LCD8) are combined with the timing clocks (ϕ_1, ϕ_2 or ϕ_3), and three combined bit data are gathered to make a segment output data in 1/3 bias - 1/3 duty drive. In case of static LCD drive or output port, timing is always fixed to ϕ_1 (always "High") and one bit data of the LCD register is assigned to segment pin.

Note that the output pins (SEG_{13} to SEG_{17}) are analog inputs and mask options.

When the form of output port is selected by Duty bit ("00"), ϕWRITE clock and $f_{CL}/4$ (system clock) can be transmitted every time data is written into LCD1 register by setting EXT bit to "1". This signal is also available for connecting the MCU to external circuit easily. All bits of LCD1, LCD2, LCD3 register are cleared by reset.

2.12 Liquid Crystal Driver Waveforms

The LCD circuit is based on 1/3 bias - 1/3 duty driving. Figure 2-21 shows the common electrode output signal waveforms (COM_1, COM_2, COM_3), segment signal waveforms (SEG_1 to SEG_{17}) and LCD bias waveforms (between COM and SEGMENT).

Assignment of segment pins to the bits of the LCD data register, including the case in which segment pins are used as output pins, is to be specified by the user when he orders masks.

(Note). . The pin function (V_1/CH_7 or V_2/CH_8) can be selected by mask options. In case of 1/3 bias - 1/3 duty drive, select V_1 and V_2 in order to reduce supply impedance. For your application, connect 0.1 μF condensors between V_1 pin and V_{CC} as well as between V_2 pin and V_{CC}, respectively.

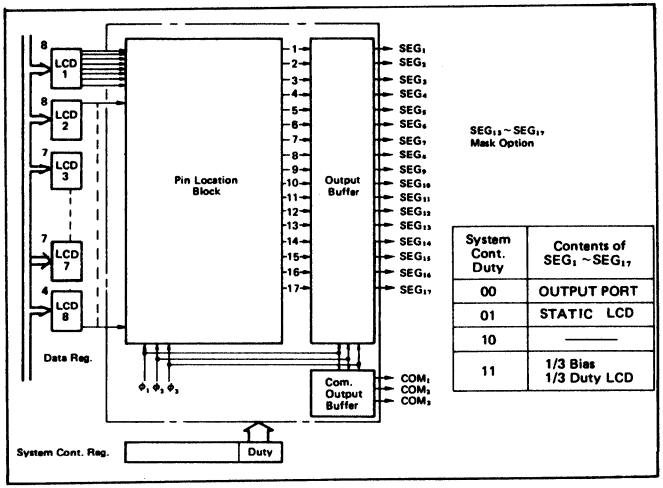


Figure 2-20 LCD Circuit System Configuration

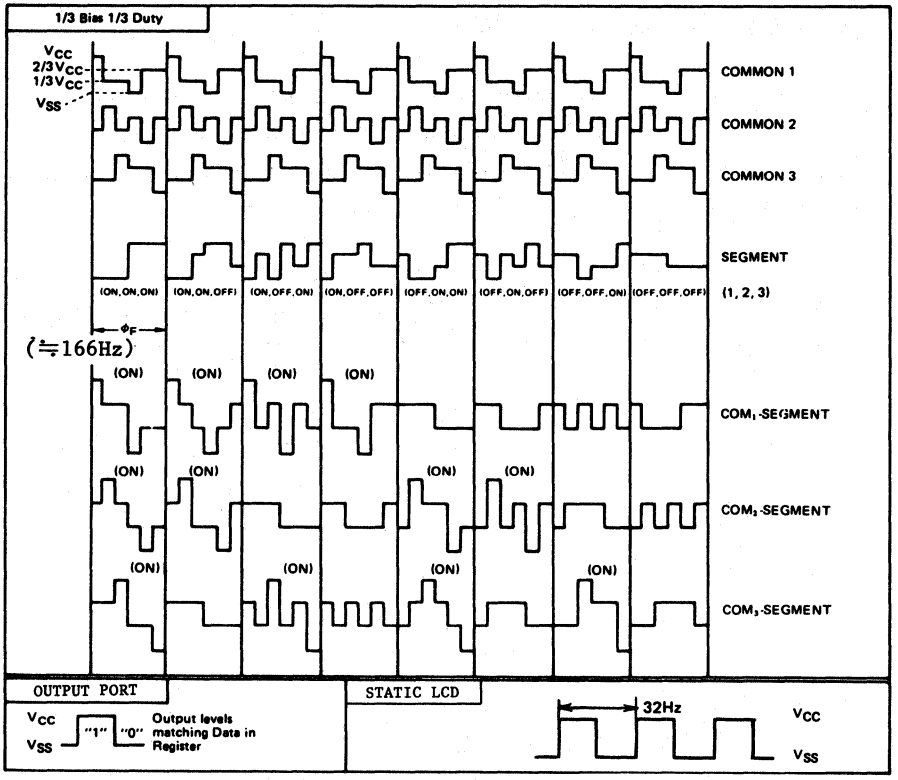


Figure 2-21 LCD Driving Waveforms

2.13 Bit Manipulation

The MCU has the ability to set or clear any single random access memory or input/output bit (except the data direction registers) with a single instruction (BSET,BCLR). Any bit in the page zero read only memory can be tested,using the BRSET and BRCLR instructions,and the program branches as a result of its state. This capability to work with any bit in RAM,ROM or I/O allows the user to have individual flags in RAM or to handle single I/O bits as control lines.

NOTE

It is necessary to pay attention to the system control register, the timer control register,and A/D control register when BSET,BCLR,or Read/Modify/Write instructions are applied to them. If own interruption request occured onto the interruption request bit (bit 7) of the control register between read cycle and write cycle of these instructions,the bit7 might be cleared in the write cycle and not acknowledged by CPU.

2.14 Addressing Modes

The MCU has ten addressing modes available for use by the programmer. They are explained and illustrated briefly in the fillowing paragraphs.

(1) Immediate
Refer to Figure 2-22. The immediate addressing mode accesses constants which do not change during program execution. Such instructions are two bytes long. The effective address (EA) is the PC and the operand is fetched from the byte following the opcode.

(2) Direct
Refer to Figure 2-23. In direct addressing,the address of the operand is contained in the second byte of the instruction. Direct addressing allows the user to directly address the lowest 256 bytes in memory. All RAM space,I/O registers and 128 bytes of ROM are located in page zero to take advantage of this efficient memory addressing.

(3) Extended
Refer to Figure 2-24. Extended addressing is used to reference any location in memory space. The EA is the contents of the two byte following the opcode.Extended addressing instructions are three bytes long.

(4) Relative
Refer to Figure 2-25. The relative addressing mode applies only to the branch instructions. In this mode the contents of the byte following the opcode are added to the program counter when the branch is taken. EA=(PC)+2+Rel. Rel is the contents of the location following the instruction opcode with bit 7 being the sign bit. If the branch is not taken,Rel=0. When a branch takes place,the program goes to somewhere within the range of +129 bytes to -127 of the present instruction. These instructions are two bytes long.

(5) Indexed (No Offset)
Refer to Figure 2-26. This mode of addressing accesses the lowest 256 bytes of memory. These instructions are one byte long and their EA is the contents of the index register.

(6) Index (8-bit Offset)
Refer to Figure 2-27. The EA is calculated by adding the contents of the byte following the opcode to the contents of the index register. In this mode,511 lowest memory locations are accessible. These instructions occupy two bytes.

(7) Indexed (16-bit Offset)
Refer to Figure 2-28. This addressing mode calculates the EA by adding the contents of two bytes following the opcode to the index register. Thus,the entire memory space may be accessed. Instructions which use this addressing mode are three bytes long.

(8) Bit Set/Bit Clear
Refer to Figure 2-29. This mode of addressing applies to instructions which can set or clear any bit on page zero. The lower three bits in the opcode specify the bit to be set or cleared while the byte following the opcode specifies the addresss in page zero.

(9) Bit Test and Branch

Refer to Figure 2-30. This mode of addressing applies to instructions which can test any bit in first 256 locations ($00-$FF) and branch to any location relative to the PC. The byte to be tested is addressed by the byte following the opcode . The individual bit within that byte to be tested is addressed by the lower three bits of the opcode. The third byte is the relative address to be added to the program counter if the branch condition is met. These instructions are three bytes long. The value of the bit tested is written to the carry bit in the condition code register.

(10) Implied

Refer to Figure 2-31. The implied mode of addressing has no EA. All the information necessary to execute an instruction is contained in the opcode. Direct operations on accumulator and the index register are included in this mode of addressing. In addition,control instructions such as SWI,RTI belong to this group. All implied addressing instructions are one byte long.

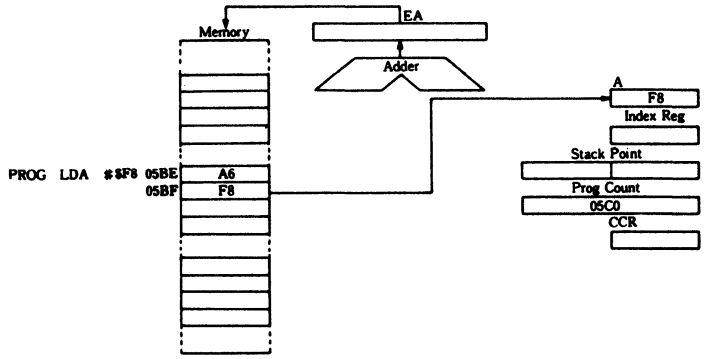


Figure 2-22 Immediate Addressing Example

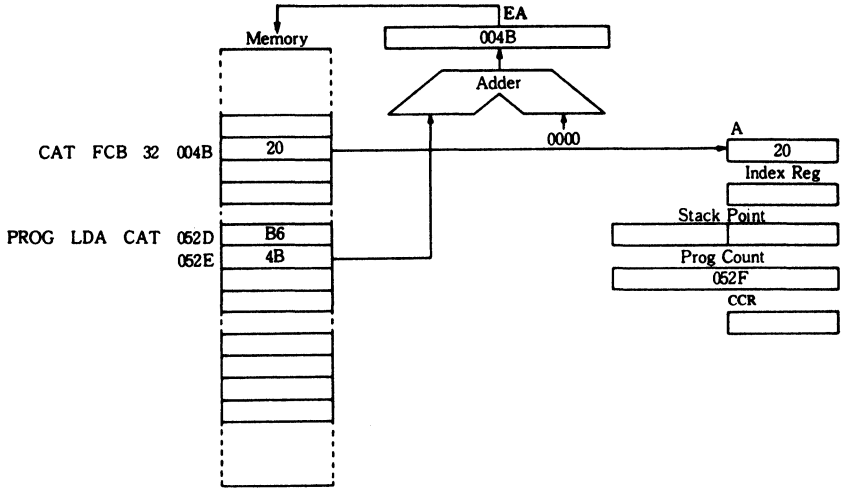


Figure 2-23 Direct Addressing Example

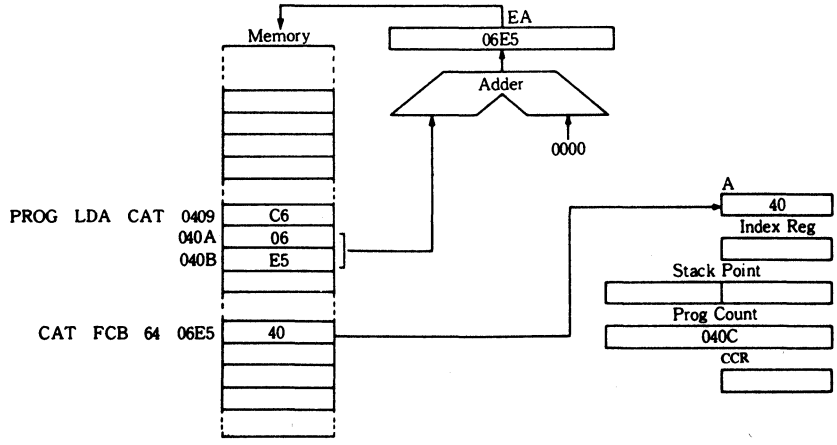


Figure 2-24 Extended Addressing Example

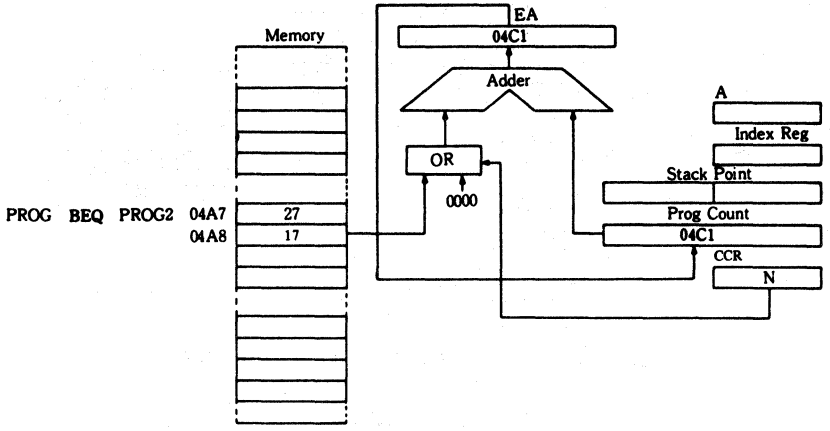


Figure 2-25 Relative Addressing Example

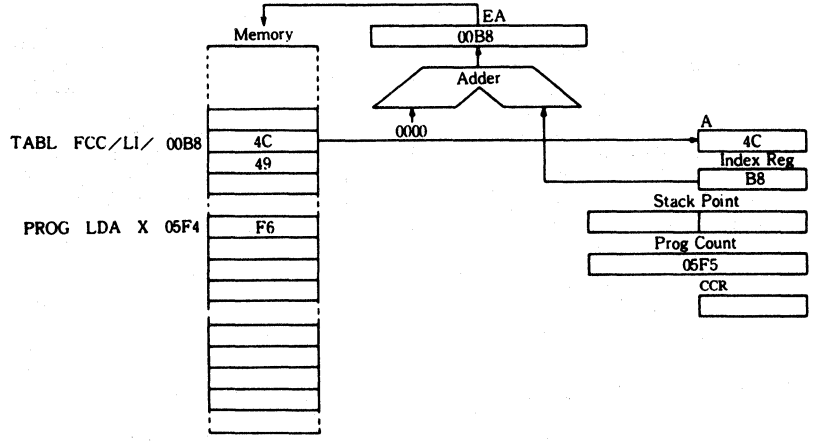


Figure 2-26 Indexed (No Offset) Addressing Example

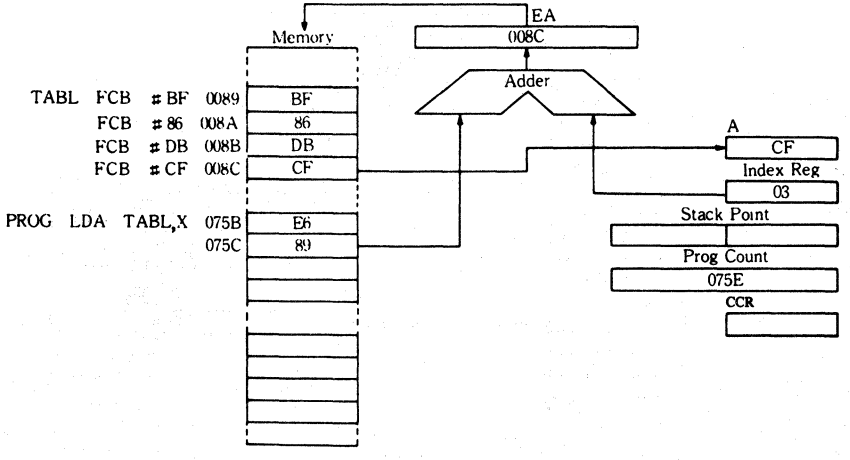


Figure 2-27 Indexed (8-Bit Offset) Addressing Example

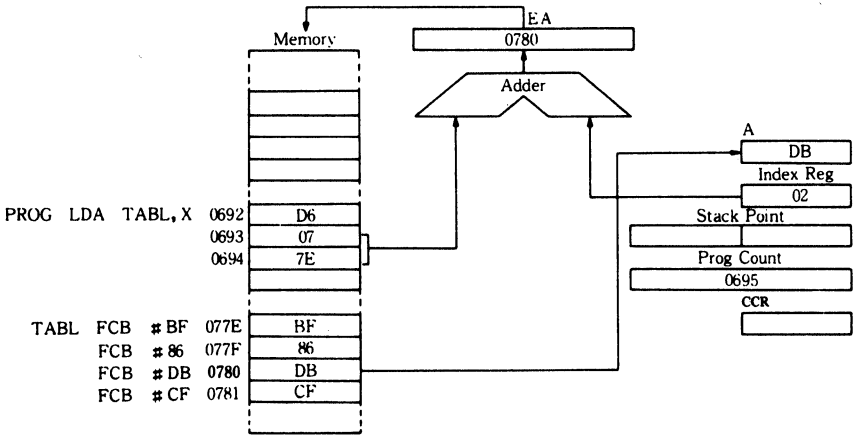


Figure 2-28 Indexed (16-Bit Offset) Addressing Example

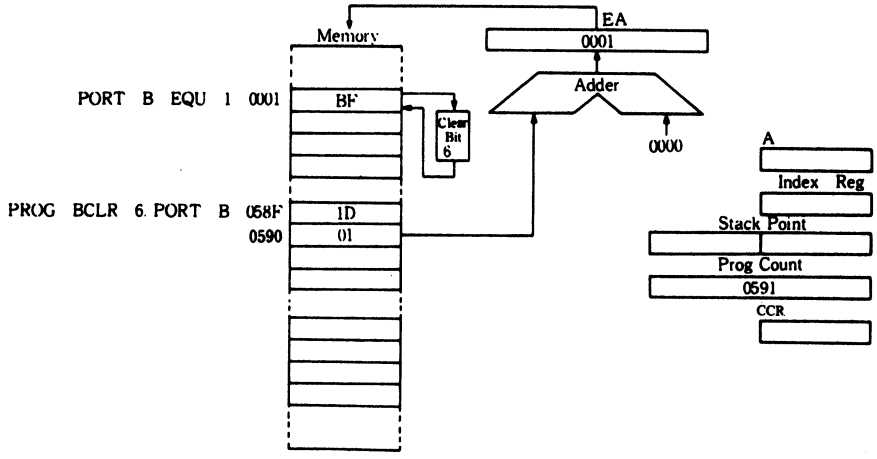


Figure 2-29 Bit Set/Bit Clear Addressing Example

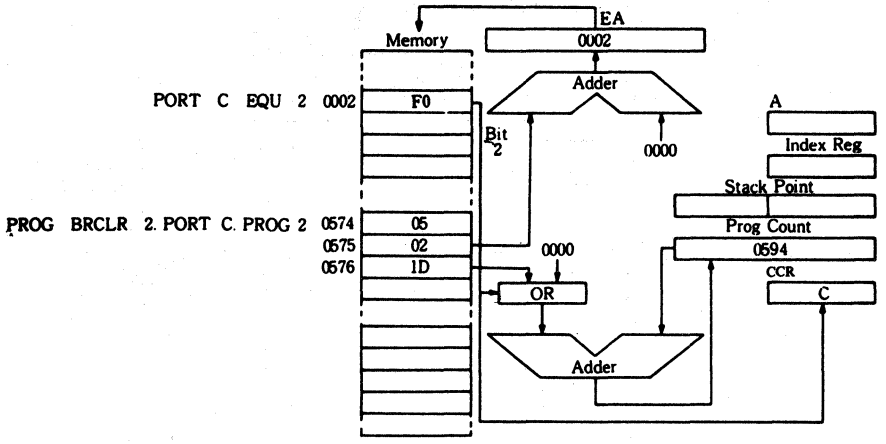


Figure 2-30 Bit Test and Branch Addressing Example

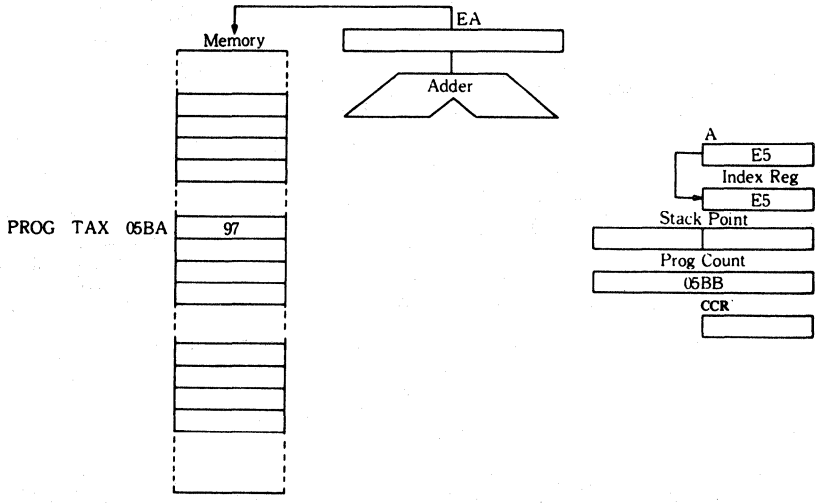


Figure 2-31 Implied Addressing Example

2.15 Instruction Set

The MCU has a set of 59 basic instructions. They can be divided into five different types: register/memory, read/modify/write, branch, bit manipulation, and control. The following paragraphs briefly explain each type. All the instructions within a given type are presented in individual tables.

(1) Register/Memory Instructions

Most of these instructions use two operands. One operand is either the accumulator or the index register. The other operand is obtained from memory using one of the addressing modes. The jump unconditional (JMP) and jump to subroutine (JSR) instructions have no register operand. Refer to Table 2-2.

(2) Read/Modify/Write Instructions

These instructions read a memory location or a register, modify or test its contents, and write the modified value back to memory or to the register. The test for negative or zero (TST) instruction is an exception to the read/modify/ write instructions as it does not perform the write. Refer to Table 2-3.

(3) Branch Instructions

The branch instructions cause a branch from the program when a certain condition is met. Refer to Table 2-4.

(4) Bit Manipulation Instructions

These instructions are used on any bit in the first 256 bytes of the memory. One group either sets or clears. The other group performs the bit test and branch operations. Refer to Table 2-5.

(5) Control Instructions

The control instructions control the MCU operations during program execution. Refer to Table 2-6.

(6) Alphabetical Listing

The complete instruction set is given in alphabetical order in Table 2-7.

(7) Op-code Map

Table 2-8 is an op-code map for the instructions used on the MCU.

Table 2-2 Register/Memory Instructions

		Addressing Mode																	
		Immediate			Direct			Extended			Indexed (No Offset)			Indexed (8-Bit Offset)			Indexed (16-Bit Offset)		
Operation	Mnemonic	Op Code	# Bytes	# Cycles	Op Code	# Bytes	# Cycles	Op Code	# Bytes	# Cycles	Op Code	# Bytes	# Cycles	Op Code	# Bytes	# Cycles	Op Code	# Bytes	# Cycles
Load A from Memory	LDA	A6	2	2	B6	2	3	C6	3	4	F6	1	2	E6	2	4	D6	3	5
Load X from Memory	LDX	AE	2	2	BE	2	3	CE	3	4	FE	1	2	EE	2	4	DE	3	5
Store A in Memory	STA	–	–	–	B7	2	4	C7	3	5	F7	1	3	E7	2	5	D7	3	6
Store X in Memory	STX	–	–	–	BF	2	4	CF	3	5	FF	1	3	EF	2	5	DF	3	6
Add Memory to A	ADD	AB	2	2	BB	2	3	CB	3	4	FB	1	2	EB	2	4	DB	3	5
Add Memory and Carry to A	ADC	A9	2	2	B9	2	3	C9	3	4	F9	1	2	E9	2	4	D9	3	5
Subtract Memory	SUB	A0	2	2	B0	2	3	C0	3	4	F0	1	2	E0	2	4	D0	3	5
Subtract Memory from A with Borrow	SBC	A2	2	2	B2	2	3	C2	3	4	F2	1	2	E2	2	4	D2	3	5
AND Memory to A	AND	A4	2	2	B4	2	3	C4	3	4	F4	1	2	E4	2	4	D4	3	5
OR Memory with A	ORA	AA	2	2	BA	2	3	CA	3	4	FA	1	2	EA	2	4	DA	3	5
Exclusive OR Memory with A	EOR	A8	2	2	B8	2	3	C8	3	4	F8	1	2	E8	2	4	D8	3	5
Arithmetic Compare A with Memory	CMP	A1	2	2	B1	2	3	C1	3	4	F1	1	2	E1	2	4	D1	3	5
Arithmetic Compare X with Memory	CPX	A3	2	2	B3	2	3	C3	3	4	F3	1	2	E3	2	4	D3	3	5
Bit Test Memory with A (Logical Compare)	BIT	A5	2	2	B5	2	3	C5	3	4	F5	1	2	E5	2	4	D5	3	5
Jump Unconditional	JMP	–	–	–	BC	2	2	CC	3	3	FC	1	2	EC	2	3	DC	3	4
Jump to Subroutine	JSR	–	–	–	BD	2	4	CD	3	5	FD	1	3	ED	2	4	DD	3	5

Symbols: Op = Operation # = Instruction

Table 2-3 Read/Modify/Write Instructions

		Addressing Mode														
		Implied (A)			Implied (X)			Direct			Indexed (No Offset)			Indexed (8-Bit Offset)		
Operation	Mnemonic	Op Code	# Bytes	# Cycles	Op Code	# Bytes	# Cycles	Op Code	# Bytes	# Cycles	Op Code	# Bytes	# Cycles	Op Code	# Bytes	# Cycles
Increment	INC	4C	1	1	5C	1	1	3C	2	4	7C	1	3	6C	2	5
Decrement	DEC	4A	1	1	5A	1	1	3A	2	4	7A	1	3	6A	2	5
Clear	CLR	4F	1	1	5F	1	1	3F	2	4	7F	1	3	6F	2	5
Complement	COM	43	1	1	53	1	1	33	2	4	73	1	3	63	2	5
Negate (2's Complement)	NEG	40	1	1	50	1	1	30	2	4	70	1	3	60	2	5
Rotate Left Thru Carry	ROL	49	1	1	59	1	1	39	2	4	79	1	3	69	2	5
Rotate Right Thru Carry	ROR	46	1	1	56	1	1	36	2	4	76	1	3	66	2	5
Logical Shift Left	LSL	48	1	1	58	1	1	38	2	4	78	1	3	68	2	5
Logical Shift Right	LSR	44	1	1	54	1	1	34	2	4	74	1	3	64	2	5
Arithmetic Shift Right	ASR	47	1	1	57	1	1	37	2	4	77	1	3	67	2	5
Arithmetic Shift Left	ASL	48	1	1	58	1	1	38	2	4	78	1	3	68	2	5
Test for Negative or Zero	TST	4D	1	1	5D	1	1	3D	2	4	7D	1	3	6D	2	5

Symbols: Op = Operation # = Instruction

Table 2-4 Branch Instructions

Operation	Mnemonic	Relative Addressing Mode		
		Op Code	# Bytes	# Cycles
Branch Always	BRA	20	2	3
Branch Never	BRN	21	2	2 or 3 *
Branch IF Higher	BHI	22	2	2 or 3 *
Branch IF Lower or Same	BLS	23	2	2 or 3 *
Branch IF Carry Clear	BCC	24	2	2 or 3 *
(Branch IF Higher or Same)	(BHS)	24	2	2 or 3 *
Branch IF Carry Set	BCS	25	2	2 or 3 *
(Branch IF Lower)	(BLO)	25	2	2 or 3 *
Branch IF Not Equal	BNE	26	2	2 or 3 *
Branch IF Equal	BEQ	27	2	2 or 3 *
Branch IF Half Carry Clear	BHCC	28	2	2 or 3 *
Branch IF Half Carry Set	BHCS	29	2	2 or 3 *
Branch IF Plus	BPL	2A	2	2 or 3 *
Branch IF Minus	BMI	2B	2	2 or 3 *
Branch IF Interrupt Mask Bit is Clear	BMC	2C	2	2 or 3 *
Branch IF Interrupt Mask Bit is Set	BMS	2D	2	2 or 3 *
Branch IF Interrupt Line is Low	BIL	2E	2	2 or 3 *
Branch IF Interrupt Line is High	BIH	2F	2	2 or 3 *
Branch to Subroutine	BSR	AD	2	4

Symbol: Op = Operation # = Instruction
* If branched, each instruction will be a 3-cycle instruction.

Table 2-5 Bit Processing Instructions

Operations	Mnemonic	Addressing Mode					
		Bit Set/Clear			Bit Test and Branch		
		Op Code	# Bytes	# Cycles	Op Code	# Bytes	# Cycles
Branch IF Bit n is Set	BRSET n (n = 0.....7)	—	—	—	2 • n	3	4 or 5 *
Branch IF Bit n is Clear	BRCLR n (n = 0.....7)	—	—	—	01 + 2 • n	3	4 or 5 *
Set Bit n	BSET n (n = 0.....7)	10 + 2 • n	2	4	—	—	—
Clear Bit n	BCLR n (n = 0.....7)	11 + 2 • n	2	4	—	—	—

Symbol: Op = Operation # = Instruction
* If Branched, each instruction will be a 5-cycle instruction.

Table 2-6 Control Instructions

Operation	Mnemonic	Implied		
		Op Code	# Bytes	# Cycles
Transfer A to X	TAX	97	1	1
Transfer X to A	TXA	9F	1	1
Set Carry Bit	SEC	99	1	1
Clear Carry Bit	CLC	98	1	1
Set Interrupt Mask Bit	SEI	9B	1	1
Clear Interrupt Mask Bit	CLI	9A	1	1
Software Interrupt	SWI	83	1	9
Return from Subroutine	RTS	81	1	4
Return from Interrupt	RTI	80	1	7
Reset Stack Pointer	RSP	9C	1	1
No-Operation	NOP	9D	1	1

Symbol: Op = Operation # = Instruction

Table 2-7 Instruction Set

Mnemonic	Addressing Modes										Condition Code				
	Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8 Bits)	Indexed (16 Bits)	Bit Set/ Clear	Bit Test & Branch	H	I	N	Z	C
ADC		x	x	x		x	x	x			∧	•	∧	∧	∧
ADD		x	x	x		x	x	x			∧	•	∧	∧	∧
AND		x	x	x		x	x	x			•	•	∧	∧	•
ASL	x		x			x	x				•	•	∧	∧	∧
ASR	x		x			x	x				•	•	∧	∧	∧
BCC					x						•	•	•	•	•
BCLR									x		•	•	•	•	•
BCS					x						•	•	•	•	•
BEQ					x						•	•	•	•	•
BHCC					x						•	•	•	•	•
BHCS					x						•	•	•	•	•
BHI					x						•	•	•	•	•
BHS					x						•	•	•	•	•
BIH					x						•	•	•	•	•
BIL					x						•	•	•	•	•
BIT		x	x	x		x	x	x			•	•	∧	∧	•
BLO					x						•	•	•	•	•
BLS					x						•	•	•	•	•
BMC					x						•	•	•	•	•
BMI					x						•	•	•	•	•
BMS					x						•	•	•	•	•
BNE					x						•	•	•	•	•
BPL					x						•	•	•	•	•
BRA					x						•	•	•	•	•

(Continued)

Symbols for condition code:
- H Half Carry (From Bit 3)
- I Interrupt Mask
- N Negative (Sign Bit)
- Z Zero
- C Carry/Borrow
- ∧ Test and Set if True, Cleared Otherwise
- • Not Affected

Table 2-7 Instruction Set (Continued)

Mnemonic	Addressing Modes										Condition Code				
	Implied	Immediate	Direct	Extended	Relative	Indexed (No Offset)	Indexed (8 Bits)	Indexed (16 Bits)	Bit Set/ Clear	Bit Test & Branch	H	I	N	Z	C
BRN					x						•	•	•	•	•
BRCLR										x	•	•	•	•	∧
BRSET										x	•	•	•	•	∧
BSET									x		•	•	•	•	•
BSR					x						•	•	•	•	•
CLC	x										•	•	•	•	0
CLI	x										•	0	•	•	•
CLR	x		x			x	x				•	•	0	1	•
CMP		x	x	x		x	x	x			•	•	∧	∧	∧
COM	x		x			x	x				•	•	∧	∧	1
CPX		x	x	x		x	x	x			•	•	∧	∧	∧
DEC	x		x			x	x				•	•	∧	∧	•
EOR		x	x	x		x	x	x			•	•	∧	∧	•
INC	x		x			x	x				•	•	∧	∧	•
JMP			x	x		x	x	x			•	•	•	•	•
JSR			x	x		x	x	x			•	•	•	•	•
LDA		x	x	x		x	x	x			•	•	∧	∧	•
LDX		x	x	x		x	x	x			•	•	∧	∧	•
LSL	x		x			x	x				•	•	∧	∧	∧
LSR	x		x			x	x				•	•	0	∧	∧
NEG	x		x			x	x				•	•	∧	∧	∧
NOP	x										•	•	•	•	•
ORA		x	x	x		x	x	x			•	•	∧	∧	•
ROL	x		x			x	x				•	•	∧	∧	∧
ROR	x		x			x	x				•	•	∧	∧	∧
RSP	x										•	•	•	•	•
RTI	x										?	?	?	?	?
RTS	x										•	•	•	•	•
SBC		x	x	x		x	x	x			•	•	∧	∧	∧
SEC	x										•	•	•	•	1
SEI	x										•	1	•	•	•
STA			x	x		x	x	x			•	•	∧	∧	•
STX			x	x		x	x	x			•	•	∧	∧	•
SUB		x	x	x		x	x	x			•	•	∧	∧	∧
SWI	x										•	1	•	•	•
TAX	x										•	•	•	•	•
TST	x		x			x	x				•	•	∧	∧	•
TXA	x										•	•	•	•	•

Symbols for condition code:

H Half Carry (From Bit 3)
I Interrupt Mask
N Negative (Sign Bit)
Z Zero
C Carry/Borrow
∧ Test and Set if True, Cleared Otherwise
• Not Affected
? Load CC Register From Stack

Table 2-8 OP Code Map

	Bit Manipulation		Branch	Read/Modify/Write					Control		Register/Memory						
	Test & Branch	Set/ Clear	Rel	DIR	A	X	,X1	,X0	IMP	IMP	IMM	DIR	EXT	,X2	,X1	,X0	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	←HIGH
0	BRSET0	BSET0	BRA	NEG					RTI*	–	SUB						0
1	BRCLR0	BCLR0	BRN	–					RTS*	–	CMP						1
2	BRSET1	BSET1	BHI	–					–	–	SBC						2
3	BRCLR1	BCLR1	BLS	COM					SWI*	–	CPX						3 L
4	BRSET2	BSET2	BCC	LSR					–	–	AND						4 O
5	BRCLR2	BCLR2	BCS	–					–	–	BIT						5 W
6	BRSET3	BSET3	BNE	ROR					–	–	LDA						6
7	BRCLR3	BCLR3	BEQ	ASR					–	TAX	–	STA (+1)					7
8	BRSET4	BSET4	BHCC	LSL/ASL					–	CLC	EOR						8
9	BRCLR4	BCLR4	BHCS	ROL					–	SEC	ADC						9
A	BRSET5	BSET5	BPL	DEC					–	CLI	ORA						A
B	BRCLR5	BCLR5	BMI	–					–	SEI	ADD						B
C	BRSET6	BSET6	BMC	INC					–	RSP	–	JMP(–1)					C
D	BRCLR6	BCLR6	BMS	TST					–	NOP	BSR*	JSR(+1)		JSR		JSR(+1)	D
E	BRSET7	BSET7	BIL	–					–	–	LDX						E
F	BRCLR7	BCLR7	BIH	CLR					–	TXA	–	STX(+1)					F
	3/4 or 5	2/4	2/2 or 3	2/4	1/1	1/1	2/5	1/3	1/*	1/1	2/2	2/3	3/4	3/5	2/4	1/2	

(NOTES) 1. "–" is an undefined operation code.
2. The figure in the lowest row of each column gives the number of bytes and the cycles needed for the instruction.
The number of cycles for the asterisked (*) mnemonics is a follows:

RTI	7
RTS	4
SWI	9
BSR	4

3. The parenthesized figure must be added to the cycle count of the associated instruction.
4. If the instruction is branched, the cycle count is the larger figure.

3. Executable Instruction

3.1 Symbol and Abbreviation

Shown below are the meanings of symbols and abbreviations.

(1) Operation

( ): contents

←: movement direction

+: addition

-: subtraction

∧: AND

∨: OR

⊕: Exclusive OR

$\overline{x}$: NOT

(2) Register symbols in MPU

ACCA: accumulator A

CCR: condition codes register

IX: index register, 8 bits

PC: program counter, 12 bits

PCH: upper three bits of program counter

PCL: lower eight bits of program counter

SP: stack pointer, 5 bits

(3) Memory and addressing codes

M: stored address

MH: upper eight bits of stored address

ML: lower eight bits of stored address

M+1: stored address M plus 1

Msp: stored address indicated by stack pointer

Imm: immediate value

Disp: displacement value = M - (IX)

D: displacement value = M - (IX)

DH: displacement value = upper eight bits

DL: displacement value = lower eight bits

Rel: relative value

IMPLIED: implied addressing

RELATIVE: relative addressing

ACCUMULATOR: accumulator addressing

INDEX REG.: index register addressing

IMMEDIATE: immediate addressing

DIRECT: direct addressing

EXTENDED: extended addressing

INDEXED 0 BYTE OFFSET: indexed addressing 0 byte offset

INDEXED 1 BYTE OFFSET: indexed addressing 1 byte offset

INDEXED 2 BYTE OFFSET: indexed addressing 2 byte offset

EA: effective address

(4) Contents of bits 0 through 4 of condition codes register

C: carry - borrow bit 0

Z: zero bit 1

N: negative bit 2

I: interruption mask bit 3

H: half carry from bit 3 to bit 4 bit 4

(5) Status of each bit before execution of instruction

An: bit n of ACCA (n = 7, 6, 5, ..., 0)

Mn: bit n of M (n = 7, 6, 5, ..., 0)

Xn: bit n of IX (n= 7, 6, 5, ..., 0)

(6) Status of each bit on result after execution of instruction

Rn: bit n of result (n = 7, 6, 5, ..., 0)

(7) Symbols on instruction's format

P: each addressing mode on Immediate, Direct, Extended and index of 0, 1 and 2 byte offset

Q: each addressing mode on Direct and index of 0 and 1 byte offset

A: accumulator addressing mode

X: index register addressing mode

DR: direct addressing mode

dd: relative operand (8 bits)

n: bit n of memory (n = 7, 6, 5, ..., 0)

(8) Status of HD63L05'S interruption pin

INT: status of interruption pin (high, low)

Arithmetic Operation

ADC

ADC (ADd with Carry)

Format

ADC P

Operation

ACCA ← (ACCA) + (M) + (C)

Condition Codes

H: Set if there is a carry from bit 3, otherwise cleared.
I: Unaffected.
N: Set if the most significant bit of the result is set; otherwise cleared.
Z: Set if the result is 0; otherwise cleared.
C: Set if there was a carry from the most significant bit of the result; otherwise cleared.

Description

Adds the contents of the carry bit C to the sum of the contents of ACCA and M, and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	ADC	#Imm	A9	Imm		2	2
DIRECT	ADC	M	B9	M		2	3
EXTENDED	ADC	M	C9	MH	ML	3	4
INDEXED 0 BYTE OFFSET	ADC	0, X	F9			1	2
INDEXED 1 BYTE OFFSET	ADC	Disp,X	E9	D		2	4
INDEXED 2 BYTE OFFSET	ADC	Disp,X	D9	DH	DL	3	5

Example

```
LDA    VAL2     (EXVAL5, EXVAL6)+(VAL1, VAL2)
ADD    EXVAL6   *          =(EXVAL5, EXVAL6)
STA    EXVAL6   *
LDA    VAL1     *
ADC    EXVAL5   ***
STA    EXVAL5   *
```

Arithmetic Operation

ADD

ADD (ADD without carry)

Format

ADD P

Operation

ACCA ← (ACCA) + (M)

Condition Codes

H: Set if there is a carry from bit 3; otherwise cleared.
I: Unaffected.
N: Set if the significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0"; otherwise cleared.
C: Set if there is a carry from the most significant bit of the result; otherwise cleared.

Description

Adds the ACCA contents to the M contents and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	ADD	#Imm	AB	Imm		2	2
DIRECT	ADD	M	BB	M		2	3
EXTENDED	ADD	M	CB	MH	ML	3	4
INDEXED 0 BYTE OFFSET	ADD	0,X	FB			1	2
INDEXED 1 BYTE OFFSET	ADD	Disp,X	EB	D		2	4
INDEXED 2 BYTE OFFSET	ADD	Disp,X	DB	DH	DL	3	5

Example

```
LDA     VAL1      (VAL1)+(WORK)=(RESULT)
ADD     WORK      *
STA     RESULT    *
```

Logical Operation
AND

AND (logical AND)

Format

AND P

Operation

ACCA ← (ACCA) Λ (M)

Condition Codes

H: Unaffected
I: Unaffected
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0" otherwise cleared.
C: Unaffected

Description

Performs logical AND between the ACCA contents and M contents, and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	AND	#Imm	A4	Imm		2	2
DIRECT	AND	M	B4	M		2	3
EXTENDED	AND	M	C4	MH	ML	3	4
INDEXED 0 BYTE OFFSET	AND	0,X	F4			1	2
INDEXED 1 BYTE OFFSET	AND	Disp,X	E4	D		2	4
INDEXED 2 BYTE OFFSET	AND	Disp,X	D4	DH	DL	3	5

Example

```
LDA   0,X    ERASE UPPER 4 BITS
AND   #$0F   *
STA   0,X    (RESTORE)
INC   X      *
BRA   LOOP   *
```

ASL (Arithmetic Shift Left)

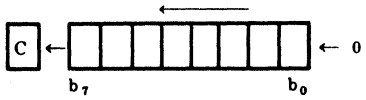
Format

```
ASL Q
ASL A
ASL X
```

Condition Codes

H: Unaffected
I: Unaffected
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0"; otherwise cleared.
C: Set if the most significant bit is "1", otherwise cleared.

Operation



Description

Shifts the contents of ACCA, IX or M one bit to the left. The bit 0 is loaded with "0". The carry bit C is loaded with the bit 7 of ACCA, IX or M.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	ASL	A	48			1	1
INDEX REG.	ASL	X	58			1	1
DIRECT	ASL	M	38	M		2	4
INDEXED 0 BYTE OFFSET	ASL	0,X	78			1	3
INDEXED 1 BYTE OFFSET	ASL	Disp,X	68	D		2	5

Example

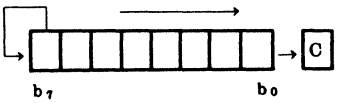
```
        LDA   WORK
CHECK   ASL   A       BRANCH FOLLOWING BIT
        BCS   BITON   *        7-6-5-4-3-2-1-0
BITOFF  EQU   *
        LDX   #100
```

ASR (Arithmetic Shift Right)

Format

ASR Q
ASR A
ASR X

Operation



Condition Codes

H: Unaffected
I: Unaffected
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0"; otherwise cleared.
C: Set if the least significant bit is "1" before a shift; otherwise cleared.

Description

Shifts the contents of ACCA, IX or M one bit to the right. The bit 7 is unaffected. The bit 0 is loaded into the carry bit C.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	ASR	A	47			1	1
INDEX REG.	ASR	X	57			1	1
DIRECT	ASR	M	37	M		2	4
INDEXED 0 BYTE OFFSET	ASR	0,X	77			1	3
INDEXED 1 BYTE OFFSET	ASR	Disp,X	67	D		2	5

Example

```
ASR WORK    BRANCH OPTION (KEEPING BIT7)
BCS OPT0
ASR WORK
BCS OPT1
ASR WORK
BCS OPT2
*
```

BCC (Branch if Carry Clear)

Format

BCC dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002+Rel if (C)=0

Description

Tests the state of the C bit and causes a branch if it is "0".
If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BCC	Rel	24	Rel		2	2 or 3

Example

```
        LDA     VAL2
        ADD     EXVAL6
        BCC     NORMAL      KETA AGARI NASHI
*
        INC     X           KETA AGARI
```

BCLR (Bit CLeaR bit n)

Format

BCLR n, DR

Condition Codes

Unaffected.

Operation

Mn ← 0

Description

Clears the bit n (n = 0 through 7) of M. The other bits are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	BCLR	0,M	11	M		2	4
DIRECT	BCLR	1,M	13	M		2	4
DIRECT	BCLR	2,M	15	M		2	4
DIRECT	BCLR	3,M	17	M		2	4
DIRECT	BCLR	4,M	19	M		2	4
DIRECT	BCLR	5,M	1B	M		2	4
DIRECT	BCLR	6,M	1D	M		2	4
DIRECT	BCLR	7,M	1F	M		2	4

Example

```
LDA     CNTRL     ** MAKE CONTROL CODE **
AND     #$FO      *
ORA     WORK      *
STA     CNTRL     *
BCLR    0,CNTRL   CLEAR BIT 0,6,7 ABSOLUTELY
BCLR    6,CNTRL
BCLR    7,CNTRL
```

Conditional Branch

BCS

BCS (Branch if Carry Set)

Format

BCS dd

Condition Codes

Unaffected

Operation

PC ← (PC)+0002+Rel if (C)=1

Description

Tests the state of the C bit and causes a branch if it is "1".
If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BCS	Rel	25	Rel		2	2 or 3

Example

```
        LDA     VAL1
        ADD     EXVAL6
        BCS     ABNML       KETA AGARI
*
        STA     EXVAL6      KETA AGARI  NASHI
```

BEQ (Branch of EQual)

Format

BEQ dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002+Rel if (Z)=1

Description

Tests the state of the Z bit and causes a branch if it is "1".
If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BEQ	Rel	27	Rel		2	2 or 3

Example

```
LDA     WORK
BEQ     AAAA        WORK = 0
CMP     RESULT
BEQ     BBBB        WORK = RESULT
```

Conditional Branch

BHCC

BHCC (Branch if Half Carry Clear)

Format

BHCC dd

Condition Codes

Unaffected

Operation

PC ← (PC)+0002+Rel if (H)=0

Description

Tests the state of the H bit and causes a branch if it is "0".
If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BHCC	Rel	28	Rel		2	2 or 3

Example

```
            CMP     #$9
            BLS     DAALOW      $99 ---> INPUT
DAAH6       LDX     #$60        HIGH NYBLE NEEDS CORRECTION
*
DAALOW      BHCC    DAAL9
            TXA
```

BHCS (Branch if Half Carry Set)

Format

BHCS dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002+Rel if (H)=1

Description

Tests the state of the H bit and causes a branch if it is 1.
If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BHCS	Rel	29	Rel		2	2 or 3

Example

```
        CMP     #$9
        BLS     DAALW1      $99 ---> INPUT
DAAH7   LDX     #$60        HIGH NYBLE NEEDS CORRECTION
*
DAALW1  BHCS    DAAL6
        NAD     #$F
```

Conditional Branch

BHI

BHI (Branch if HIgher)

Format

BHI dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002+Rel if (C V Z)=0
i.e. if (ACCA) > (M)
(unsigned binary numbers)

Description

Causes a branch if both bit C and bit Z are "0".
If the BHI instruction is executed immediately after execution of either of the instructions CMP or SUB, the branch will occur only if the unsigned binary number represented by the minuend (i.e. ACCA) was greater than the unsigned binary number represented by the subtrahend (i.e. M).
This instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BHI	Rel	22	Rel		2	2 or 3

Example

```
        LDA     VAL1
        CMP     VAL2
        BHI     ZIP25       VAL1 > VAL2     (IGNORE SIGN BIT)
*
        STA     WORK        VAL1 --> WORK   (LOWER OR SAME)
```

BHS (Branch if Higher or Same)

Format

BHS dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002+Rel if (C)=0

Description

Following to an unsigned compare or subtract, BHS will cause a branch if the register is higher than or the same as the location in M.
If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BHS	Rel	24	Rel		2	2 or 3

Example

```
        LDA     VAL1
        CMP     VAL2
        BHS     ZIP26       VAL1 >= VAL2    IGNORE SIGN BIT
*
        STA     WORK        VAL1 ---> WORK (LOWER)
```

Conditional Branch

BIH

BIH (Branch if Interrupt line is High)

Format

BIH dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002+Rel if INT=1 (high)

Description

Tests the state of the external interrupt pin ($\overline{INT}$) and causes a branch if it is high.

If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BIH	Rel	2F	Rel		2	2 or 3

Example

```
        BIH     INTH0   INT LINE CHECK
INTL1   LDA     #$28    OUTPUT DATA = $28
        BRA     NEXT2
INTH0   LDA     #$FF    OUTPUT DATA = $FF
NEXT2   STA     PIA     OUTPUT
```

BIL (Branch if Interrupt line is Low)

Format

BIL dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002+Rel if INT=0 (low)

Description

Tests the state of the external interrupt pin and causes a branch if it is low.

If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BIL	Rel	2E	Rel		2	2 or 3

Example

```
        BIL     INTL2     INT LINE CHECK
INTH3   LDA     #$45      OUTPUT DATA = $45
        BRA     NEXT4
INTL2   LDA     #$0       OUTPUT DATA = $00
NEXT4   STA     PIA       OUTPUT
```

Logical Operation
BIT

BIT (BIt Test)

Format

BIT P

Operation

(ACCA) Λ (M)

Condition Codes

H: Unaffected.
I: Unaffected
N: Set if the most significant bit of the result of the AND is "1"; otherwise cleared.
Z: Set if all the bits of the result of the AND are "0"; otherwise cleared.
C: Unaffected.

Description

Performs the logical AND operation of the ACCA contents and the M contents and modifies the respective condition codes. The contents of ACCA and M are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	BIT	#Imm	A5	Imm		2	2
DIRECT	BIT	M	B5	M		2	3
EXTENDED	BIT	M	C5	MH	ML	3	4
INDEXED 0 BYTE OFFSET	BIT	,X	F5			1	2
INDEXED 1 BYTE OFFSET	BIT	Disp,X	E5	D		2	4
INDEXED 2 BYTE OFFSET	BIT	Disp,X	D5	DH	DL	3	5

Example

```
EVBIT   LDA     VAL1
        BIT     #$F8
        BEQ     OK          0  = BIT ASSIGN (VAL1) <=7
*
NG      LDA     #227        SET ERROR NUMBER
        JMP     ERROR
```

BLO (Branch if LOwer)

Format

BLO dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002+Rel if (C)=1

Description

Following to a compare, BLO will cause a branch if the register is lower than the memory location.
Equivalent to the BCS executable instruction.
If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BLO	Rel	25	Rel		2	2 or 3

Example

```
        LDA     VAL1
        CMP     VAL2
        BLO     ZIP27       VAL1 < VAL2  (IGNORE SIGN BIT)
*
        STA     WORK        VAL1 --> WORK HIGHER OR SAME
```

Conditional Branch

BLS

BLS (Branch if Lower or Same)

Format

BLS dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002+Rel if (C V Z)=1
i.e. if (ACCA) ≦ (M)

Description

Causes a branch if either C or Z is "1".
If the BLS instruction is executed immediately after execution of either of the instructions CMP or SUB, the branch will occur only if the unsigned binary number represented by the minuend (i.e. ACCA) is less than or equal to the unsigned binary number represented by the subtrahend (i.e. M).

If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BLS	Rel	23	Rel		2	2 or 3

Example

```
        LDA     VAL1
        CMP     VAL2
        BLS     ZIP28       VAL1 <= VAL2 IGNORE SIGN BIT
*
        STA     WORK        VAL1 ---> WORK (HIGHER)
```

BMC (Branch if interrupt Mask is Clear)

Format

BMC dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002+Rel if (I)=0

Description

Tests the state of the I bit and causes a branch if I is "0".

If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BMC	Rel	2C	Rel		2	2 or 3

Example

```
        BMC     MSKOFF    INTMSK OFF?
        BIL     MSKOFF    INT LINE LOW?
        LDA     PIA       READ DATA
        STA     WORK
MSKOFF  RTS
```

Conditional Branch

BMI

BMI (Branch if MInus)

Format

BMI dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002+Rel if (N)=1

Description

Tests the state of the N bit, and causes a branch if N is "1".

If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BMI	Rel	2B	Rel		2	2 or 3

Example

```
        LDA     VAL1
        BMI     ZIP29       VAL1 < 0
*
        STA     WORK        VAL ---> WORK (PLUS)
```

BMS (Branch if interrupt Mask is Set)

Format

BMS dd

Condition Codes

Unaffected.

Operation

PC ← (PC+0002+Rel if (I)=1

Description

Tests the state of the I bit and causes a branch if I is "1".
If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BMS	Rel	2D	Rel		2	2 or 3

Example

```
        BMS     MSKON1      INTMSK ON?
MSKOF1  RTS                 NO
MSKON1  BIL     MSKOF1      INT LINE LOW?
        LDA     PIA         DATA
        STA     WORK
        RTS
```

BNE (Branch if Not Equal)

Format

BNE dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002+Rel if (Z)=0

Description

Tests the state of the Z bit and causes a branch if Z is "0".
Following to a compare or subtract instruction, BNE will cause a branch if the contents are different.

If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BNE	Rel	26	Rel		2	2 or 3

Example

```
        LDA     WORK
        BNE     CCCC        WORK NOT = 0
        CMP     RESULT
        BNE     DDDD        WORK NOT = RESULT
```

BPL (Branch if PLus)

Format

BPL dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002+Rel if (N)=0

Description

Tests the state of the N bit, and causes a branch if N is "0".
If branched, this instruction requires 3 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BPL	Rel	2A	Rel		2	2 or 3

Example

```
        LDA     VAL1
        BPL     ZIP31       VAL >= 0
*
        STA     WORK        VAL1 ---> WORK (MINUS)
```

Unconditional Branch

BRA

BRA (BRanch always)

Format

BRA dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002+Rel

Description

Causes an unconditional branch to the address given by the above expression.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BRA	Rel	20	Rel		2	3

Example

```
        LDA     EXVAL5
        STA     RESULT
        BRA     ENDO1       BRANCH TO ENDO1 ALWAYS
*
CHECK8  EQU     *
```

BRCLR (BRanch if bit n is CLeaR)

Format

BRCLR n, DR, dd

Operation

PC ← (PC)+0003+Rel if (Mn)=0

Condition Codes

H: Unaffected.
I: Unaffected.
N: Unaffected.
Z: Unaffected.
C: Set if (Mn)=1; otherwise cleared.

Description

Tests the bit n (n = 0 through 7) of M and causes a branch if the Mn contents are 0.
If branched, each instruction requires 5 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BRCLR	0,M,Rel	01	M	Rel	3	4 or 5
RELATIVE	BRCLR	1,M,Rel	03	M	Rel	3	4 or 5
RELATIVE	BRCLR	2,M,Rel	05	M	Rel	3	4 or 5
RELATIVE	BRCLR	3,M,Rel	07	M	Rel	3	4 or 5
RELATIVE	BRCLR	4,M,Rel	09	M	Rel	3	4 or 5
RELATIVE	BRCLR	5,M,Rel	0B	M	Rel	3	4 or 5
RELATIVE	BRCLR	6,M,Rel	0D	M	Rel	3	4 or 5
RELATIVE	BRCLR	7,M,Rel	0F	M	Rel	3	4 or 5

Example

```
        LDA     CNTRL       ** SET CONTROL CODE **
        AND     #$OF
        ORA     WORK
        STA     CNTRL
*               ** ACTION **
        BRCLR   4,CNTRL,ENGINE
        BRCLR   7,CNTRL,GASCHK
```

Unconditional Branch

BRN

BRN (BRanch Never)

Format

BRN dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002

Description

BRN is included here to demonstrate the nature of branches on the HD63L05 MCU. Each branch is matched with an inverse that varies only in the least significant bit of the opcode. BRN is the inverse of BRA. This instruction may have some use during program debugging.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BRN	Rel	21	Rel		2	2

Example

```
        STX     4,X
*
        BRN     *           ** DELAY **
        BRN     *
        BRN     *
        BRN     *
```

BRSET (BRanch if bit n is SET)

Format

BRSET n, DR, dd

Operation

PC ← (PC)+0003+Rel if (Mn)=1

Condition Codes

H: Unaffected.
I: Unaffected.
N: Unaffected.
Z: Unaffected.
C: Set if (Mn)=1; otherwise cleared.

Description

Tests the bit n (n = 0 through 7) of M, and causes a branch if the Mn contents are "1".

If branched, each instruction requires 5 cycles.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BRSET	0,M,Rel	00	M	Rel	3	4 or 5
RELATIVE	BRSET	1,M,Rel	02	M	Rel	3	4 or 5
RELATIVE	BRSET	2,M,Rel	04	M	Rel	3	4 or 5
RELATIVE	BRSET	3,M,Rel	06	M	Rel	3	4 or 5
RELATIVE	BRSET	4,M,Rel	08	M	Rel	3	4 or 5
RELATIVE	BRSET	5,M,Rel	0A	M	Rel	3	4 or 5
RELATIVE	BRSET	6,M,Rel	0C	M	Rel	3	4 or 5
RELATIVE	BRSET	7,M,Rel	0E	M	Rel	3	4 or 5

Example

```
        LDA     CNTRL     ** SET CONTROL CODE **
        AND     #$8E
        ORA     WORK
        STA     CNTRL
*                         ** ACTION **
PROC1   BRSET   0,CNTRL, OIL
PROC2   BRSET   7,CNTRL, GAS
```

BSET (Bit SET bit n)

Format

BSET n, DR

Condition Codes

Unaffected.

Operation

Mn ← 1

Description

Sets the bit n (n = 0 through 7) of M. All other bits are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	BSET	0,M	10	M		2	4
DIRECT	BSET	1,M	12	M		2	4
DIRECT	BSET	2,M	14	M		2	4
DIRECT	BSET	3,M	16	M		2	4
DIRECT	BSET	4,M	18	M		2	4
DIRECT	BSET	5,M	1A	M		2	4
DIRECT	BSET	6,M	1C	M		2	4
DIRECT	BSET	7,M	1E	M		2	4

Example

```
        LDA     RESULT
        BPL     PLUS
*                       (MINUS)
        BSET    2,CNTRL
        BSET    3,WORK
PLUS    EQU     *
        LDA     VAL2
```

BSR (Branch to SubRoutine)

Format

BSR dd

Condition Codes

Unaffected.

Operation

PC ← (PC)+0002
Msp ← (PCL), SP ← (SP)-0001
Msp ← (PCH), SP ← (SP)-0001
PC ← (PC)+Rel

Description

The program counter is increased by "2". The less significant byte of the contents of the program counter is pushed onto the stack. The stack pointer is then decreased by "1". The more significant byte of the contents of the program counter is then pushed onto the stack. Unused bits in the Program Counter high byte are stored as 1's on the stack. The stack pointer is again decreased by "1". A branch then occurs to the address specified by the program counter.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
RELATIVE	BSR	Rel	AD	Rel		2	4

Example

```
        LDA     #$3B        ACCA = INTERFACE(0011 1011)
        BSR     HAND
*
        LDA     #$1E        ACCA = INTERFACE(0001 1110)
        BSR     FING
```

CLC (CLear Carry)

Format

CLC

Operation

C ← 0

Condition Codes

H: Unaffected.
I: Unaffected.
N: Unaffected.
Z: Unaffected.
C: Cleared.

Description

Clears the carry bit C in the condition code register.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	CLC		98			1	1

Example

```
        BNE     CHK83
        STA     RESULT
        CLC                     RETURN CODE SET 'OK'
        RTS
*
*
```

CLI (CLear Interrupt mask)

Format

CLI

Condition Codes

H: Unaffected.
I: Cleared.
N: Unaffected.
Z: Unaffected.
C: Unaffected.

Operation

I ← 0

Description

Clears the interruption mask bit in the processor condition code register. This enables the MCU to service interruptions. Interruptions that were pending while the I bit was set will now begin to have effect.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	CLI		9A			1	1

Example

```
SEI             INTERRUPT DISABLE
RSP             RESET STACK POINTER
JSR     SYSINZ  SYSTEM INITIALIZE
CLI             INTERRUPT ENABLE
```

Arithmetic Operation

CLR

CLR (CLeaR)

Format

```
CLR Q
CLR A
CLR X
```

Condition Codes

H: Unaffected
I: Unaffected.
N: Cleared.
Z: Set.
C: Unaffected.

Operation

IX ← 0
or
ACCA ← 0
or
M ← 0

Description

The contents of IX, ACCA or M are replaced with "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	CLR A		4F			1	1
INDEX REG.	CLR X		5F			1	1
DIRECT	CLR	M	3F	M		2	4
INDEXED 0 BYTE OFFSET	CLR	,X	7F			1	3
INDEXED 1 BYTE OFFSET	CLR	Disp,X	6F	D		2	5

Example

```
*                    ** INITIALIZE **
        CLR     PNTR
        CLR     PNTR+1
        CLR     0,X
        CLR     A
```

CMP (CoMPare)

Format

CMP P

Operation

(ACCA)-(M)

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of the result of the subtraction is "1"; otherwise cleared.
Z: Set if the result of the subtraction is "0"; otherwise cleared.
C: Set if the absolute value of memory is greater than the absolute value of the accumulator; otherwise cleared.

Description

Compares the ACCA contents and the M contents, then sets the condition codes which may then be used for controlling the conditional branches. Both operands are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	CMP	#Imm	A1	Imm		2	2
DIRECT	CMP	M	B1	M		2	3
EXTENDED	CMP	M	C1	MH	ML	3	4
INDEXED 0 BYTE OFFSET	CMP	,X	F1			1	2
INDEXED 1 BYTE OFFSET	CMP	Disp,X	E1	D		2	4
INDEXED 2 BYTE OFFSET	CMP	Disp,X	D1	DH	DL	3	5

Example

```
        LDA     PNTR,X
*
        CMP     #'A
        BEQ     SECTA       ACCA = 'A'
        CMP     #'B
        BEQ     SECTB       ACCA = 'B'
        BRA     INPUT
```

Logical Operation

COM

COM (COMplement)

Format

```
COM Q
COM A
COM X
```

Operation

IX ← $(\overline{IX})$ = $FF-(IX)
or
ACCA ← $(\overline{ACCA})$ = $FF-(ACCA)
or
M ← $(\overline{M})$ = $FF-(M)

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0"; otherwise cleared.
C: Set.

Description

Replaces the contents of ACCA, IX or M with its 1's complement.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	COM	A	43			1	1
INDEX REG.	COM	X	53			1	1
DIRECT	COM	M	33	M		2	4
INDEXED 0 BYTE OFFSET	COM	,X	73			1	3
INDEXED 1 BYTE OFFSET	COM	Disp,X	63	D		2	5

Example

```
*
SUBIN   EQU     *
        INC     X
        LDA     PNTR, X
        COM     A           MODIFY DATA (REVERSE)
        RTS
*
```

CPX (ComPare indeX register)

Format

CPX P

Operation

(IX)-(M)

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0"; otherwise cleared.
C: Set if the absolute value of the contents of the memory is greater than the absolute value of the contents of IX; otherwise cleared.

Description

Compares the IX contents with M contents. The condition code can be collated by means of the next conditional branch instruction. Both operands are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	CPX	#Imm	A3	Imm		2	2
DIRECT	CPX	M	B3	M		2	3
EXTENDED	CPX	M	C3	MH	ML	3	4
INDEXED 0 BYTE OFFSET	CPX	,X	F3			1	2
INDEXED 1 BYTE OFFSET	CPX	Disp,X	E3	D		2	4
INDEXED 2 BYTE OFFSET	CPX	Disp,X	D3	DH	DL	3	5

Example

```
LDA     #$CC        ACCA = INTERFACE TO CR OR LF
LDX     PNTR
CPX     #$OD
BEQ     CR          CARRIAGE RETURN
CPX     #$OA
BEQ     LF          LINE FEED
```

DEC (DECrement)

Format

```
DEC Q
DEC A
DEC X
```

Operation

```
IX  ← (IX)-01
   or
ACCA ← (ACCA)-01
   or
M   ← (M)-01
```

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0"; otherwise cleared.
C: Unaffected

Description

Subtracts "1" from the contents of ACCA, IX or M.
N and Z bits are set and reset according to the result of this operation.
C bit is unaffected by this operation.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	DEC	A	4A			1	1
INDEX REG.	DEC	X	5A			1	1
DIRECT	DEC	M	3A	M		2	4
INDEXED 0 BYTE OFFSET	DEC	,X	7A			1	3
INDEXED 1 BYTE OFFSET	DEC	Disp,X	6A	D		2	5

Example

```
*           ** MOVE **
LOOP23 DEC   A
       BMI   NEXT
       LDX   0,X       *
       STX   $100,X    *
       INC   X         *
       BRA   LOOP23
*
NEXT   EQU   *
```

EOR (Exclusive OR)

Format

EOR P

Operation

ACCA ← (ACCA) ⊕ (M)

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0"; otherwise cleared.
C: Unaffected.

Description

Performs the logical EXCLUSIVE OR between the ACCA contents and M contents and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	EOR	#Imm	A8	Imm		2	2
DIRECT	EOR	M	B8	M		2	3
EXTENDED	EOR	M	C8	MH	ML	3	4
INDEXED 0 BYTE OFFSET	EOR	,X	F8			1	2
INDEXED 1 BYTE OFFSET	EOR	Disp,X	E8	D		2	4
INDEXED 2 BYTE OFFSET	EOR	Disp,X	D8	DH	DL	3	5

Example

```
*                          ** ARRANGE CONTROL CODE **
        LDA     CNTRL      XXXX XXXX
        EOR     #$99       1001 1001
        STA     CNTRL
        BRA     ACT01
```

Arithmetic Operation

INC

INC (INCrement)

Format

```
INC Q
INC A
INC X
```

Operation

```
IX ← (IX)+01
  or
ACCA ← (ACCA)+01
  or
M ← (M)+01
```

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0"; otherwise cleared.
C: Unaffected.

Description

Adds "1" to the contents of ACCA, IX or M.
N and Z bits are set or reset according to the result of this operation.
The C bit is not affected by this operation.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	INC	A	4C			1	1
INDEX REG.	INC	X	5C			1	1
DIRECT	INC	M	3C	M		2	4
INDEXED 0 BYTE OFFSET	INC	,X	7C			1	3
INDEXED 1 BYTE OFFSET	INC	Disp,X	6C	D		2	5

Example

```
LOOP3   INC     A           *
        CMP     #100
        BHI     EXIT        CHECK COUNTER (100 TIMES)
        LDX     0,X
        STX     $300,X      MOVE
        INC     X           *
        BRA     LOOP3
```

JMP (JuMP)

Format

JMP P

Condition Codes

Unaffected.

Operation

PC ← EA

Description

A jump occurs to the instruction stored at the effective address. The effective address is obtained according to the rules for EXTended, DIRect or INDexed addressing.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	JMP	M	BC	M		2	2
EXTENDED	JMP	M	CC	MH	ML	3	3
INDEXED 0 BYTE OFFSET	JMP	,X	FC			1	1
INDEXED 1 BYTE OFFSET	JMP	Disp,X	EC	D		2	3
INDEXED 2 BYTE OFFSET	JMP	Disp,X	DC	DH	DL	3	4

Example

```
        LDA     VAL1
        STA     EXVAL5
        LDA     VAL2
        STA     EXVAL6
        JMP     END90       DO TO END-ROUTINE
```

Subroutine Control

JSR

JSR (Jump to SubRoutine)

Format

JSR P

Condition Codes

Unaffected.

Operation

PC ← (PC)+n[Note)]
Msp ← (PCL), SP ← (SP)-0001
Msp ← (PCH), SP ← (SP)-0001
PC ← EA

Description

The program counter is increased by n[Note)] depending on the addressing mode, then pushed onto the 2-byte stack. And the stack point is updated. A jump occurs to the instruction stored at the effective address. The effective address is obtained according to the rules for EXTended, DIRect or INDexed addressing.

Note) n is equal to 1, 2 or 3, depending on the number of bytes in the instruction code. Refer to the addressing code and the number of MPU cycles shown below.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	JSR	M	BD	M		2	4
EXTENDED	JSR	M	CD	MH	ML	3	5
INDEXED 0 BYTE OFFSET	JSR	,X	FD			1	3
INDEXED 1 BYTE OFFSET	JSR	Disp,X	ED	D		2	4
INDEXED 2 BYTE OFFSET	JSR	Disp,X	DD	DH	DL	3	5

Example

```
*                    ** MAIN ROUTINE **
START    EQU    *
         JSR    INTRTN    INITIALIZE
         JSR    KBRTN     INPUT FROM KEY-BOARD
         JSR    ANARTN    ANALYSE
         JSR    PRCRTN    PROCESS
         JMP    ENDRTN    END
```

LDA (LoaD Accumulator)

Format

LDA P

Operation

ACCA ← (M)

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0"; otherwise cleared.
C: Unaffected.

Description

Loads the M contents into the accumulator.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	LDA	#Imm	A6	Imm		2	2
DIRECT	LDA	M	B6	M		2	3
EXTENDED	LDA	M	C6	MH	ML	3	4
INDEXED 0 BYTE OFFSET	LDA	,X	F6			1	2
INDEXED 1 BYTE OFFSET	LDA	Disp,X	E6	D		2	4
INDEXED 2 BYTE OFFSET	LDA	Disp,X	D6	DH	DL	3	5

Example

```
LDA     VAL1
STA     WORK
LDA     0,X
STA     RESULT
LDA     #$FF
```

LDX (LoaD indeX register)

Format

LDX P

Operation

IX ← (M)

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of IX is "1"; otherwise cleared.
Z: Set if all bits of IX of the result are "0"; otherwise cleared.
C: Unaffected.

Description

Loads the M contents into IX. The condition code is set according to data.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	LDX	#Imm	AE	Imm		2	2
DIRECT	LDX	M	BE	M		2	3
EXTENDED	LDX	M	CE	MH	ML	3	4
INDEXED 0 BYTE OFFSET	LDX	,X	FE			1	2
INDEXED 1 BYTE OFFSET	LDX	Disp,X	EE	D		2	4
INDEXED 2 BYTE OFFSET	LDX	Disp,X	DE	DH	DL	3	5

Example

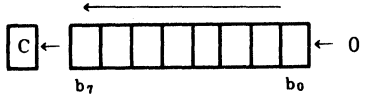
```
LDX     VAL1
STX     WORK
LDX     0,X
STX     RESULT
LDX     #$FF
```

LSL (Logical Shift Left)

Format

LSL Q
LSL A
LSL X

Operation



Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0"; otherwise cleared.
C: Set if the most significant bit of ACCA, IX or M is "1" before the execution of an instruction; otherwise cleared.

Description

Shifts the contents of ACCA, IX or M 1-bit to the left. The bit 0 is loaded with "0". The carry bit C is loaded with the most significant bit of ACCA, IX or M.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	LSL	A	48			1	1
INDEX REG.	LSL	X	58			1	1
DIRECT	LSL	M	38	M		2	4
INDEXED 0 BYTE OFFSET	LSL	,X	78			1	3
INDEXED 1 BYTE OFFSET	LSL	Disp,X	68	D		2	5

Example

```
LSL     WORK    ** MULTIPLY   X 8 **
LSL     WORK
LSL     WORK
```

LSR (Logical Shift Right)

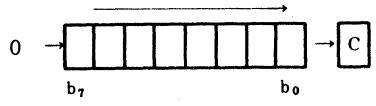
Format

LSR Q
LSR A
LSR X

Condition Codes

H: Unaffected.
I: Unaffected.
N: Cleared.
Z: Set if the result is "0"; otherwise cleared.
C: Set if, before execution of an instruction, the least significant bit of ACCA, IX or M is "1"; otherwise cleared.

Operation



Description

Shifts the contents of ACCA, IX or M 1-bit to the right. The bit 7 is loaded with "0". The carry bit C is loaded with the least significant bit of ACCA, IX or M.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	LSR	A	44			1	1
INDEX REG.	LSR	X	54			1	1
DIRECT	LSR	M	34	M		2	4
INDEXED 0 BYTE OFFSET	LSR	,X	74			1	3
INDEXED 1 BYTE OFFSET	LSR	Disp,X	64	D		2	5

Example

```
LSR     WORK    ** DIVIDE / 16 **
LSR     WORK
LSR     WORK
LSR     WORK
```

NEG (NEGate)

Format

```
NEG Q
NEG A
NEG X
```

Operation

IX ← (IX)=00-(IX)
or
ACCA ← (ACCA)=00-(ACCA)
or
M ← (M)=00-(M)

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0"; otherwise cleared.
C: Set if there is a borrow; otherwise cleared. Set if the contents of ACCA, IX or M are other than "0".

Description

Replaces the contents of ACCA, IX or M with its two's complement.
Note that $80 (-128) is unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	NEG	A	40			1	1
INDEX REG.	NEG	X	50			1	1
DIRECT	NEG	M	30	M		2	4
INDEXED 0 BYTE OFFSET	NEG	,X	70			1	3
INDEXED 1 BYTE OFFSET	NEG	Disp,X	60	D		2	5

Example

```
*          CHECK RANGE (RELATIVE ADDRESSING)
     CMP   #129     CHECK RANGE
     BCC   BERROR   *     BRANCH ERROR
     NEG   A        OFFSET
     BRA   SET
*
```

Unconditional Branch

NOP

NOP (No OPeration)

Format

NOP

Condition Codes

Unaffected.

Operation

Description

This is a single-byte instruction which causes only the program counter to be increased. No other registers are affected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	NOP		9D			1	1

Example

```
NOP       ** DELAY **
NOP
NOP
NOP
NOP
NOP
```

ORA (inclusive OR)

Format

ORA

Operation

ACCA ← (ACCA)V(M)

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if all bits of the result are "0"; otherwise cleared.
C: Unaffected.

Description

Performs logical OR between ACCA contents and M contents, and stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	ORA	#Imm	AA	Imm		2	2
DIRECT	ORA	M	BA	M		2	3
EXTENDED	ORA	M	CA	MH	ML	3	4
INDEXED 0 BYTE OFFSET	ORA	,X	FA			1	2
INDEXED 1 BYTE OFFSET	ORA	Disp,X	EA	D		2	4
INDEXED 2 BYTE OFFSET	ORA	Disp,X	DA	DH	DL	3	5

Example

```
        BCS     SKIP
*
ADCN    EQU     *           ** ADDITION CONTROL BIT **
        LDA     #$14        0001  0100
        ORA     CNTRL
        STA     CNTRL
SKIP    EQU     *
```

ROL (ROtate Left)

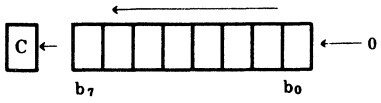
Format

ROL Q
ROL A
ROL X

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set of all the bits of the result are "0"; otherwise cleared.
C: Set if the most significant bit of ACCA, IX or M is "1" before the execution of an instruction; otherwise cleared.

Operation



Description

Shifts the contents of ACCA, IX or M 1-bit to the left. The bit 0 is loaded with the carry bit C, while the carry bit C is loaded with the most significant bit of ACCA, IX or M.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	ROL	A	49			1	1
INDEX REG.	ROL	X	59			1	1
DIRECT	ROL	M	39	M		2	4
INDEXED 0 BYTE OFFSET	ROL	,X	79			1	3
INDEXED 1 BYTE OFFSET	ROL	Disp,X	69	D		2	5

Example

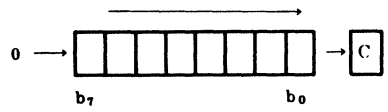
```
*               ** REPEAT ACTION FOLLOWING CNTRL **
        CLC
REPEAT  EQU     *
        ROL     CNTRL
        BCS     ACTION      ACTION & REPEAT OR ESCAPE
        NOP                 *
        NOP                 ** DELAY
        NOP                 *
        BRA     REPEAT
```

ROR (ROtate Right)

Format

ROR Q
ROR A
ROR X

Operation



Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if all the bits of the result are "0"; otherwise cleared.
C: Set if, before the execution of an instruction, the least significant bit of ACCA, IX or M was "1"; otherwise cleared.

Description

Shifts the contents of ACCA, IX or M 1-bit to the right. The bit 7 is loaded with the carry bit C, while the bit 0 is loaded with the carry bit C.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	ROR	A	46			1	1
INDEX REG.	ROR	X	56			1	1
DIRECT	ROR	M	36	M		2	4
INDEXED 0 BYTE OFFSET	ROR	,X	76			1	3
INDEXED 1 BYTE OFFSET	ROR	Disp,X	66	D		2	5

Example

```
*               ** REPEAT ACTION FOLLOWING CNTRL **
          CLC
REPT1     EQU   *
          ROR   CNTRL
          BCS   ACTN1       ACTION & REPEAT OR ESCAPE
          NOP               *
          NOP               ** DELAY **
          NOP               *
          BRA   REPT1
```

Stack Pointer Operation
RSP

RSP (Reset Stack Pointer)

Format

RSP

Operation

SP ← $7F

Condition Codes

Unaffected.

Description

Resets the stack pointer to the top ($7F) of the stack.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	RSP		9C			1	1

Example

```
        SEI             INTERRUPT DISABLE
        RSP             RESET STACK POINTER
        JSR     SYSINZ  SYSTEM INITIALIZE
        CLI             INTERRUPT ENABLE
```

RTI (ReTurn from Interrupt)

Format

RTI

Condition Codes

Set or cleared according to the first byte pulled from the stack.

Operation

SP ← (SP)+0001, CCR ← (SP)
SP ← (SP)+0001, ACCA ← (SP)
SP ← (SP)+0001, IX ← (SP)
SP ← (SP)+0001, PCH ← (SP)
SP ← (SP)+0001, PCL ← (SP)

Description

The Condition Codes, Accumulator, Index Register and the Program Counter are restored according to the state previously saved on the stack. Note that the interruption mask bit (I bit) will be reset only if the corresponding bit stored on the stack is "0".

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	RTI		80			1	7

Example

```
JSR     KEYSCN      KEY INPUT
STA     INKEY       STORE KEY CODE
JSR     EXSWIN      INPUT EXTERNAL SW
STA     INSW        STORE SW CONDITION
RTI                 RETURN TO INTERRUPT
```

Subroutine Control

RTS

RTS (Return from Subroutine)

Format

RTS

Condition Codes

Unaffected.

Operation

SP ← (SP)+0001, PCH ← (SP)
SP ← (SP)+0001, PCL ← (SP)

Description

The stack pointer is increased by "1". The contents of the byte of memory, pointed to by the stack pointer, are loaded into the high byte of the program counter. The stack pointer is again increased by "1". The byte pointed to by the stack pointer is loaded into the low byte of the program counter.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	RTS		81			1	4

Example

```
                STA     WORK
                LDA     EXVAL5
                STA     RESULT
                CLC                     RETURN CODE SET : OK
                RTS                     *
        *
```

SBC (SuBtract with Carry)

Format

SBC P

Operation

ACCA ← (ACCA)-(M)-(C)

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the result is "0"; otherwise cleared.
C: Set if the absolute value of the M contents plus the carry bit C is greater than the absolute value of the ACCA contents otherwise cleared.

Description

Subtracts the contents of the memory and the carry bit C from the ACCA contents then stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	SBC	#Imm	A2	Imm		2	2
DIRECT	SBC	M	B2	M		2	3
EXTENDED	SBC	M	C2	MH	ML	3	4
INDEXED 0 BYTE OFFSET	SBC	,X	F2			1	2
INDEXED 1 BYTE OFFSET	SBC	Disp,X	E2	D		2	4
INDEXED 2 BYTE OFFSET	SBC	Disp,X	D2	DH	DL	3	5

Example

```
*                (VAL1, VAL1+1)-(EXVAL5, EXVAL5+1)
*                              =(EXVAL5, EXVAL5+1)
*
        LDA     VAL1+1      *
        SUB     EXVAL5+1    *
        STA     EXVAL5+1    *
        LDA     VAL1        *
        SBC     EXVAL5      *
        STA     EXVAL5      *
```

SEC (SEt Carry)

Format

SEC

Condition Codes

H: Unaffected.
I: Unaffected.
N: Unaffected.
Z: Unaffected.
C: Set.

Operation

C bit ← 1

Description

Sets the carry bit C in the condition code register.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	SEC		99			1	1

Example

```
        BEQ     CHK84
        STA     RESULT
        SEC                     RETURN CODE SET : NG
        RTS
*
*
```

SEI (SEt Interrupt mask)

Format

SEI

Operation

I bit ← 1

Condition Codes

H: Unaffected.
I: Set.
N: Unaffected.
Z: Unaffected.
C: Unaffected.

Description

Sets the interruption mask bit in the processor condition code register. The MCU is inhibited from servicing interrupts, and will continue with execution of the instructions of the program until the interruption mask bit is cleared.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	SEI		9B			1	1

Example

```
SEI                 INTERRUPT DISABLE
RSP                 RESET STACK POINTER
JSR     SYSINZ      SYSTEM INITIALIZE
CLI                 INTERRUPT ENABLE
```

STA (STore Accumulator)

Format

STA P

Operation

M ← (ACCA)

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of ACCA is "1"; otherwise cleared.
Z: Set if the contents of ACCA is "0"; otherwise cleared.
C: Unaffected.

Description

Stores the ACCA contents into memory. The ACCA contents are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	STA	M	B7	M		2	4
EXTENDED	STA	M	C7	MH	ML	3	5
INDEXED 0 BYTE OFFSET	STA	,X	F7			1	3
INDEXED 1 BYTE OFFSET	STA	Disp,X	E7	D		2	5
INDEXED 2 BYTE OFFSET	STA	Disp,X	D7	DH	DL	3	6

Example

```
LDA     VAL1
STA     WORK
LDA     RESULT
STA     0,X
LDA     #$FF
STA     EXVAL5,X
```

STX (STore indeX register)

Format

STX P

Operation

M ← (IX)

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of IX is "1"; otherwise cleared.
Z: Set if the contents of IX is "0"; otherwise cleared.
C: Unaffected.

Description

Stores the IX contents into M. The IX contents are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
DIRECT	STX	M	BF	M		2	4
EXTENDED	STX	M	CF	MH	ML	3	5
INDEXED 0 BYTE OFFSET	STX	,X	FF			1	3
INDEXED 1 BYTE OFFSET	STX	Disp,X	EF	D		2	5
INDEXED 2 BYTE OFFSET	STX	Disp,X	DF	DH	DL	3	6

Example

```
LDX     VAL1
STX     WORK
LDX     RESULT
STX     0,X
LDX     #$FF
STX     EXVAL5,X
```

Arithmetic Operation

SUB

SUB (SUBtract)

Format

SUB P

Operation

ACCA ← (ACCA)-(M)

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of the result is "1"; otherwise cleared.
Z: Set if the contents of the result is 0; otherwise cleared.
C: Set if the absolute value of the M contents is greater than the absolute value of the contents of ACCA; otherwise cleared.

Description

Subtracts the M contents from ACCA contents, then stores the result into ACCA.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMMEDIATE	SUB	# Imm	A0	Imm		2	2
DIRECT	SUB	M	B0	M		2	3
EXTENDED	SUB	M	C0	MH	ML	3	4
INDEXED 0 BYTE OFFSET	SUB	,X	F0			1	2
INDEXED 1 BYTE OFFSET	SUB	Disp,X	E0	D		2	4
INDEXED 2 BYTE OFFSET	SUB	Disp,X	D0	DH	DL	3	5

Example

```
LDA     VAL1        (VAL1)-(WORK)=(RESULT)
SUB     WORK
STA     RESULT
```

SWI (SoftWare Interrupt)

Format

SWI

Condition Codes

H: Unaffected.
I: Set.
N: Unaffected.
Z: Unaffected.
C: Unaffected.

Operation

PC ← (PC)+0001
Msp ← (PCL), SP ← (SP)-0001
Msp ← (PCH), SP ← (SP)-0001
Msp ← (IX), SP ← (SP)-0001
Msp ← (ACCA), SP ← (SP)-0001
Msp ← (CCR), SP ← (SP)-0001
I bit ← 1
PC ← (SWI interrupt vector address)

Description

All the registers other than the stack pointer (SP) are pushed onto the stack. The interrupt mask bit is then set. Performs vectoring to the address indicated by the contents of the SWI interrupt vector address.Note)

Note) $7FC and $7FD for HD6805S.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	SWI		83			1	9

Example

```
LDA     #$FF        *
STA     TIMER+1     * TIMER COUNTER SET
LDA     #$3F        *
STA     TIMER       *
LDA     #3          TIMER CODE SET
SWI                 MONITOR SERVICE CALL
```

TAX (Transfer Accumulator to indeX register)

Format

TAX

Condition Codes

Unaffected.

Operation

IX ← (ACCA)

Description

Transfers the ACCA contents to IX. The ACCA contents are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	TAX		97			1	1

Example

```
TAX             SAVE ACCUMULATOR
LDA     #4      *
ADD     RESULT  ** ADD (RESULT+4)
STA     RESULT  *
TXA             REVIVE ACCUMULATOR
```

TST (TeST)

Format

```
TST Q
TST A
TST X
```

Operation

```
(IX) - 00
    or
(ACCA) - 00
    or
(M) - 00
```

Condition Codes

H: Unaffected.
I: Unaffected.
N: Set if the most significant bit of ACCA, IX or M is "1"; otherwise cleared.
Z: Set if the contents of ACCA, IX or M are "0"; otherwise cleared.
C: Unaffected.

Description

Sets N and Z bits of the condition code register according to the contents of ACCA, IX or M.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
ACCUMULATOR	TST	A	4D			1	1
INDEX REG.	TST	X	5D			1	1
DIRECT	TST	M	3D	M		2	4
INDEXED 0 BYTE OFFSET	TST	,X	7D			1	3
INDEXED 1 BYTE OFFSET	TST	Disp,X	6D	D		2	5

Example

```
        TST     CNTRL
        BEQ     INIT00      CNTRL=$00
*
        TST     WORK
        BMI     MINS00      WORK=(1XXX XXXX)
*
```

TXA (Transfer indeX register to Accumulator)

Format

TXA

Condition Codes

Unaffected.

Operation

ACCA ← (IX)

Description

Transfers the IX contents to ACCA. The IX contents are unaffected.

Addressing Mode and Number of CPU Cycles

Addressing Mode	Mnemonic	Operand type	Instruction code			Number of bytes	Number of CPU cycles
			Byte 1	Byte 2	Byte 3		
IMPLIED	TXA		9F			1	1

Example

```
TAX             SAVE ACCUMULATOR
LDA    #4       *
ADD    RESULT   ** ADD (RESULT+4)
STA    RESULT   *
TXA             REVIVE ACCUMULATOR
```

3.3 Limitation of SWI Instruction

Please pay attention to the following items when using the SWI instruction of the HD63L05.

1. Phenomenon

   When SWI (software interrupt) and other interrupt request occurs at the same time, the CPU executes usual interrupt sequence as follows.
   The CPU does not return from SWI interrupt service routine correctly, because an incorrect return address is saved onto the stack.
   Therefore, The CPU executes SWI instruction repeatedly, (Refer to Figure 1)

2. Countermeasure

   (1) There is no problem while all other interrupt is masked, or while you don't use SWI.

   (2) Please program your software as follows to avoid SWI and other interrupt occurring at the same time if you use SWI instruction.

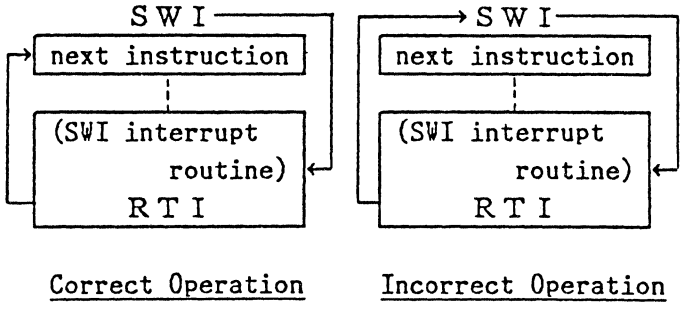


Figure 1 UNUSUAL PHENOMENON

(For Example)

```
        ↓    (User Program)
      SEI    (Other interrupts is masked)
      SWI
        ↓    (User Program)
```

SWI: Software Interrupt

3. Defect Description

As shown in Figure 2, when other interrupt request occurs in the cycle which the CPU fetches SWI instruction, the CPU pushes the content of incorrect Program Counter (PC) onto the stack.

The interrupt sequence is different from the SWI (Figure 3-②) and other interrupts (Figure 3-①) as shown in Figure 3.

Figure 2 shows the timing when the incorrect interrupt sequence occurs.

When interrupt request occurs at this timing, the CPU generates the SWI vector address and execute SWI interrupt service routine.

However, the "PC" which is incorrect return address. ("PC" is the address on which SWI instruction is programmed.) (Figure 3-③)

Therefore, the CPU exercise SWI instruction repeatedly after it returns from the SWI interrupt service routine.

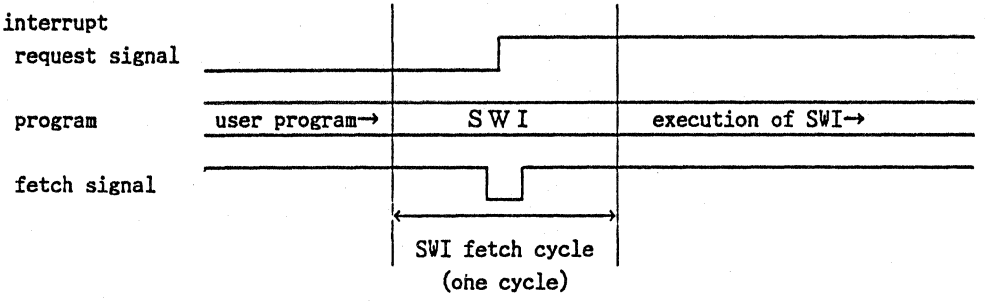


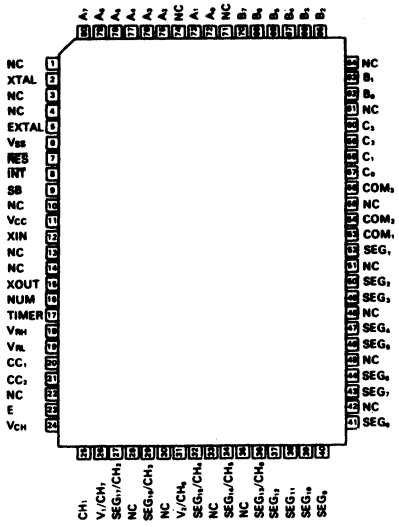
Figure 2 THE TIMING

Stack Address	①		②		③	
	other interrupts		SWI interrupt		incorrect operation	
SP	PCL	PC	PCL	PC+1	PCL	PC
SP−1	PCH	(return Address)	PCH	(return Address)	PCH	(return Address)
SP−2	Index Register		Index Register		Index Register	
SP−3	Accumulator		Accumulator		Accumulator	
SP−4	CCR		CCR		CCR	
	each interrupt routine		SWI interrupt routine		SWI interrupt routine	

CCR:Condition Code Register

Figure 3 INTERRUPT SEQUENCE

● HD63L05F1F (FP80)



● HD63L05F1P (DP64S)

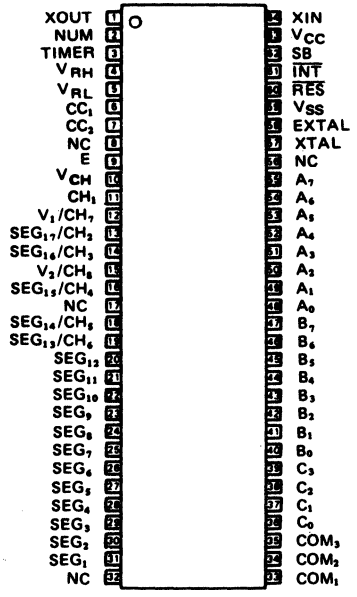
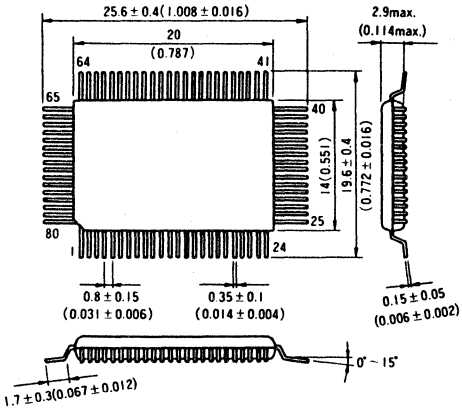
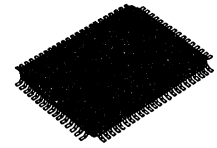


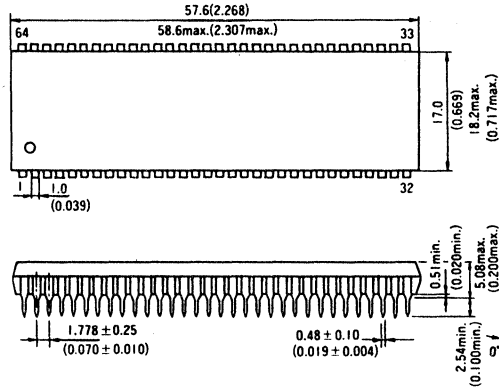
Figure 4-1 Pin Assignment (Top View)

● FP-80





● DP-64S



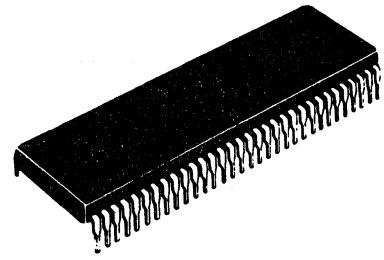


Figure 4-2 Package Information

5. Electrical Characteristics

■ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3~+5.5	V
Input Voltage	V_{in}	-0.3~V_{CC}+0.3	V
Output Voltage	V_{out}	-0.3~V_{CC}+0.3	V
Operating Temparature	T_{opr}	-20~+75	°C
Storage Temparature	T_{stg}	-55~+125	°C

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded.

● DC CHARACTERISTICS (V_{CC}=3.0±0.8V, V_{SS}=0V, Ta=-20∿+75°C, typ means typical value at V_{CC}=3.0V unless otherwise noted.)

Item			Symbol	Test Condition	min	typ	max	Unit
Input "High" Level Voltage	XTAL, XIN		V_{IH}	Connect C_L=0.5μF to V_{CH}	V_{CC}-0.3	-	V_{CC}	V
	$\overline{RES}$, $\overline{INT}$, SB				0.5V_{CC}+0.9	-	V_{CC}	V
	TIMER				0.8V_{CC}	-	V_{CC}	V
	NUM (Normal Mode)				V_{CC}-0.2	-	V_{CC}	V
Input "Low" Level Voltage	XTAL, XIN		V_{IL}	Connect C_L=0.5μF to V_{CH}	V_{CC}-2.1	-	V_{CC}-1.8	V
	$\overline{RES}$, $\overline{INT}$, SB				V_{SS}	-	0.2V_{CC}	V
	TIMER				V_{SS}	-	0.2V_{CC}	V
	NUM (Test Mode)				V_{SS}	-	0.2	V
Self Check Input Voltage	NUM (Self Check Mode)		V_{IM}		0.5V_{CC} -0.2	-	0.5V_{CC} +0.2	V
Input Pull-Up Current	$\overline{RES}$ ($\overline{INT}$:Mask Option) NUM		$-I_{R1}$	V_{CC}=3.0V, Vin=0V	3	15	30	μA
Input Leackage Current	TIMER, SB		$\|I_{IN}\|$	Vin = 0V∿V_{CC}	-	-	1.0	μA
Current Dissipation	* Crystal Oscillation	During System Operation	I_{CC1}	f =400kHz No load. Tested after setting up the internal status by self check.	-	100	200	μA
		At Halt			-	40	80	μA
		At Stand-By			-	2	5	μA
		At A/D Operation			-	200	600	μA
	* RC Oscillation	During System Operation	I_{CC2}	R =100kHz No load. Tested after setting up the internal status by self check.	-	120	200	μA
		At Halt			-	60**	100**	μA
		At Stand-By			-	2	5	μA
		At A/D Operation			-	220	600	μA
Output "Low" Level Voltage	E		V_{OL}	I_{OL} = 30μA	-	-	0.3	V

* By Mask option

** In the case that OSC1 is stopped by Halt; 60μA→30μA, 100μA→60μA

● AC CHARACTERISTICS (V_{CC}=3.0±0.8V, V_{SS}=0V, Ta=-20 ∿+75°C, typ means typical value at V_{CC}=3.0V unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	unit
Operating Clock Frequency		f_{cl}		100	400	500	kHz
Cycle Time		f_{cyc}		8	10	40	µs
Oscillation Frequency (Resistor Option)		f_{OSCR}	R =100kΩ ±1%	300	400	500	kHz
External Clock Duty		Duty		45	50	55	%
Oscillation Start Time (Crystal Option)		t_{OSCf}	C_D =10pF±20%, RS =1kΩ max	-	-	150	ms
Oscillation Start Time (Resistor Option)		t_{OSCR}	R=100kΩ±1%, Connect C_L=0.5µF to V_{CH}	-	-	2	ms
Oscillation Start Time (32kHz)		t_{OSC2}	C_G=10pF±20%. RS=20kΩ max	-	-	1	s
Internal Capacitance of Oscillator	EXTAL	C_D		-	10	-	pF
	XOUT			-	10	-	pF
Delay Time of Oscillation Delay Time		t_{DLY}	Selected by mask option	0	-	1	s
Reset Delay Time		t_{RLH}	External Capacitance =2.2µF	200	-	-	ms
$\overline{RES}$ Pulse Width		t_{RWL}	When OSC2 is used	48+1	-	-	µs
			When OSC2 is not used	1.5 t_{cyc}+1	-	-	µs
$\overline{INT}$ Pulse Width		t_{IWL}	When OSC2 is used	32	-	-	µs
			When OSC2 is not used	t_{cyc}+1	-	-	µs
TIMER Pulse Width		t_{TWL}	In the case of counter	t_{cyc}+1	-	-	µs

● PORT CHARACTERISTICS (V_{CC}=3.0±0.8V, V_{SS}=0V, Ta=-20 ∿+75°C, typ means typical value at V_{CC}=3.0V unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Output "High" Level Voltage	Port A,B,C	V_{OH}	CMOS Output, I_{OH}=-100µA	V_{CC}-0.3	-	-	V
	Port A,B,C		Key Load CMOS Output I_{OH}=-10µA	V_{CC}-0.3	-	-	V
Output "Low" Level Voltage	Port A,B,C	V_{OL}	I_{OL}=100µA	-	-	0.3	V
Input "High" Level Voltage	Port A,B,C	V_{IH}		0.8V_{CC}	-	V_{CC}	V
Input "Low" Level Voltage	Port A,B,C	V_{IL}		V_{SS}	-	0.2V_{CC}	V
Input Leackage Current	Port A,B,C	$\|I_{IN}\|$	V_{in}=0V ~ V_{CC}	-	-	1.0	µA
Input Pull-Up Current	Port A,B,C	$-I_{R2}$	V_{CC}=3.0V, V_{in}=0V	4	20	40	µA

5

●LCD DRIVER OUTPUT CHARACTERISTICS(V_{CC}=3.0V, V_{SS}=0V, Ta=-20∿ +75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Output "High" Level Voltage	Segment	V_{OH1}	V_1= 1.00V, V_2= 2.00V I_{OH}= -1μA	2.8	-	-	V
		V_{OH2}		1.8	-	-	V
		V_{OH3}		0.8	-	-	V
Output "Low" Level Voltage	Segment	V_{OL1}	V_1= 1.00V, V_2= 2.00S I_{OL}= 1μA	-	-	2.2	V
		V_{OL2}		-	-	1.2	V
		V_{OL3}		-	-	0.2	V
Output "High" Level Voltage	Common	V_{OH1}	V_1= 1.00V, V_2= 2.00V I_{OH}= -5μA	2.8	-	-	V
		V_{OH2}		1.8	-	-	V
		V_{OH3}		0.8	-	-	V
Output "Low" Level Voltage	Common	V_{OL1}	V_1= 1.00V, V_2= 2.00V I_{OL}= 5μA	-	-	2.2	V
		V_{OL2}		-	-	1.2	V
		V_{OL3}		-	-	0.2	V
Dividing Resistor		R_{LCD}	Tested between V_1 and V_2	45	90	180	kΩ
Output "High" Level Voltage	Segment	V_{OH}	In the case of Output Port, I_{OH}=-30μA	V_{CC}-0.3	-	-	V
Output "Low" Level Voltage	Segment	V_{OL}	In the case of Output Port, I_{OL}=30μA	-	-	0.3	V

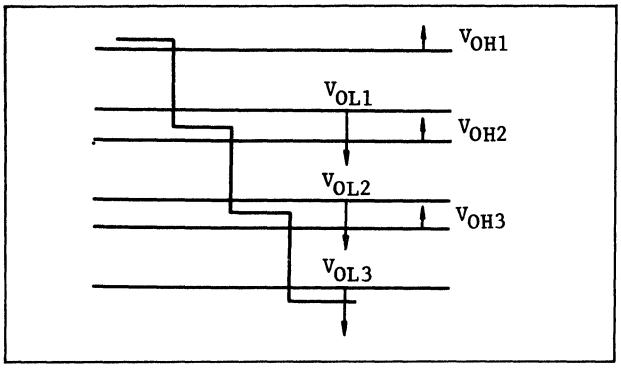


Figure 5-1 Output Level of SEG and COM

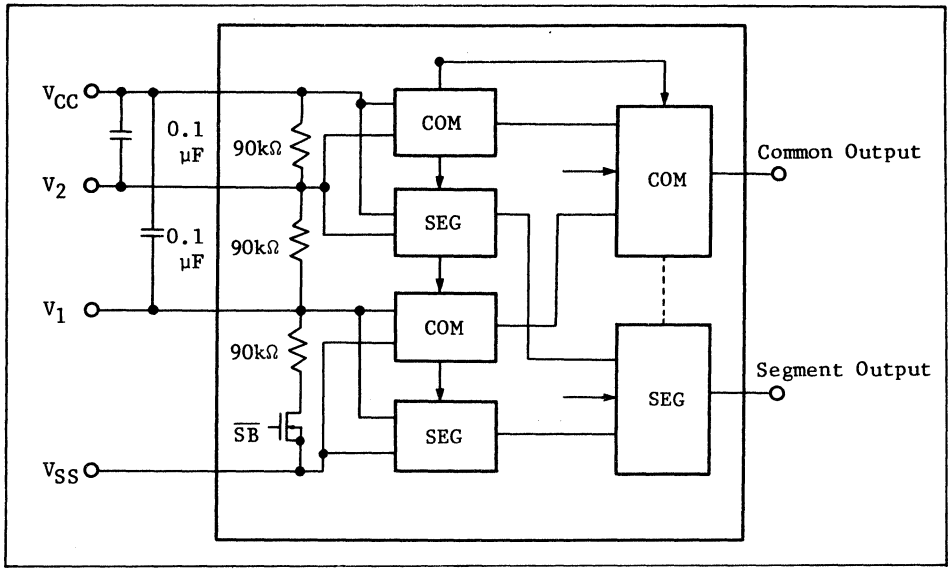


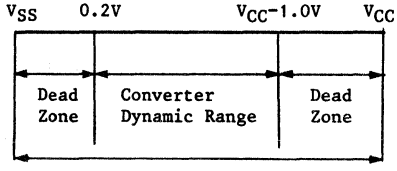
Figure 5-2 Power Supply Circuit for LCD Display

5

● A/D CONVERTER CHARACTERISTICS *(V_{CC}=3.0V, V_{SS}=0V, Ta=-20°C ∿ +75°C, C=300pF, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Conversion Accuracy	Resolution			-	-	8	bit
	Absolute Accuracy		V_{RL}=0.2V < V_{in}<V_{RH}=2.0V	-2	-	+2	LSB
Reference Voltage	"High" Side	V_{RH}		-	-	V_{CC}	V
	"Low" Side	V_{RL}		V_{SS}	-	-	V
	V_{RH} - V_{RL}	ΔV_{REF}		1.8	-	-	V
Input Voltage Range	Input Range	V_{IN}		V_{RL}	-	V_{RH}	V
	Input Dynamic Range	V_{DYN}		0.2	-	V_{CC}-1.0	V
Ladder Resistor (V_{RH}- V_{RL})		R_{HL}		40	80	160	kΩ
Conversion Time		t_{CNV}		2	-	4	ms
Programmable Voltage Comparison	Judge Error		V_{RL}=0.2V<V_{in} < V_{RH}=2.0V	-4	-	+4	LSB
	Judge Time	t_{CMP}		-	-	60	µs

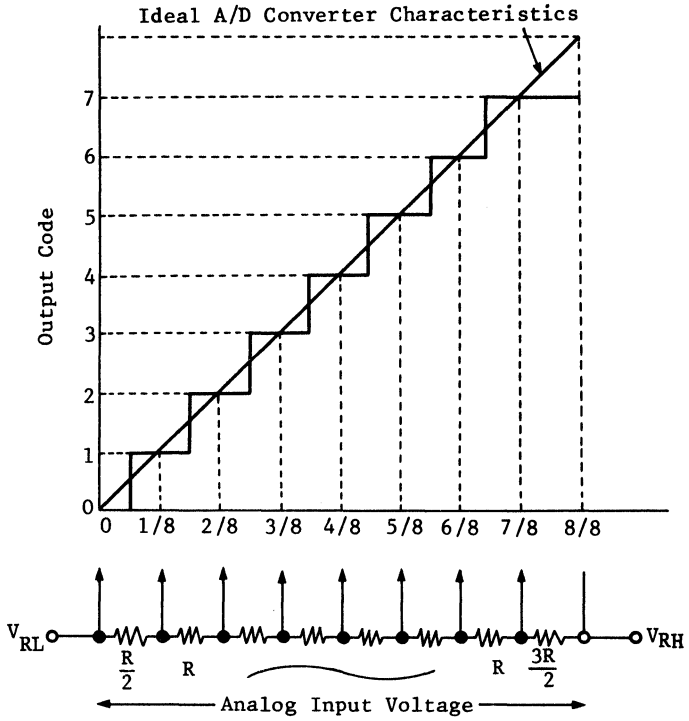
* These value can be changed without notice, because they are provisional.



Analog Input Voltage

(When the input voltage is in the dead zone, the result of the conversion is not guaranteed.)

Figure 5-3 Dynamic Range of the Comparator



(Ladder resistor indicates the compared voltage.)

Figure 5-4 Example of 3 bit Resolution

6. Application

6.1 How to Confirm Operation Frequency

When pin E of the HD63L05 MCU is connected to V_{CC} through the resistor, the system clock (E) divided by OSC1 is transmitted. The clock output from the OSC2 is available as outputs of the COM_1, COM_2, or COM_3.

6.2 Method of the DAA (Decimal Adjust Accumulator)

(1) Function

This subroutine is a simulation of the DAA instruction performed by the HD63L05 MCU. This is used immediately after the addition of two bytes (ADD and ADC) which consist of two-digit BCD (Binary Coded Decimal), respectively. This subroutine converts the result of the BCD addition into two-digit BCD to transmit it from the accumulator.

(2) Linkage

The digit to be converted is input to the accumulator and operation jumps to the routine.

(Example)

```
LDA   ARG1
ADD   ARG2
JSR   DAA---Jumps to the DAA subroutine
STA   ARG3
```

Two-digit BCDs are stored in the ARG1 and ARG2 respectively and the operation jumps to the DAA after addition. The result of the addition is converted into BCD and is stored in the ARG3.

(3) Result

The binary coded decimal digit is output to the accumulator.

(4) Register to be influenced

(i) IX guarantees the contents before jumping to the routine.

(ii) Of the CCR (Condition Code Register), the C bit is set when the result of the BCD addition or decimal conversion is rounded up. The previous contents in each bit of H, N and Z cannot be guaranteed.

(5) Program specification

Program specification is shown in Table 6-1.

Table 6-1 Program Specification

Number of words(B)	Work area(B)	Execution time(usec)	Reentrant	Relocation
31	2*	410	Not possible**	Possible

Intermediate interrupt	Indicator
Possible	—

*: It is necessary to provide this work area within the stored RAM ($020 $07F).

**: Depending on the situation, it may be necessary to mask the interrupt during the execution of this subroutine.

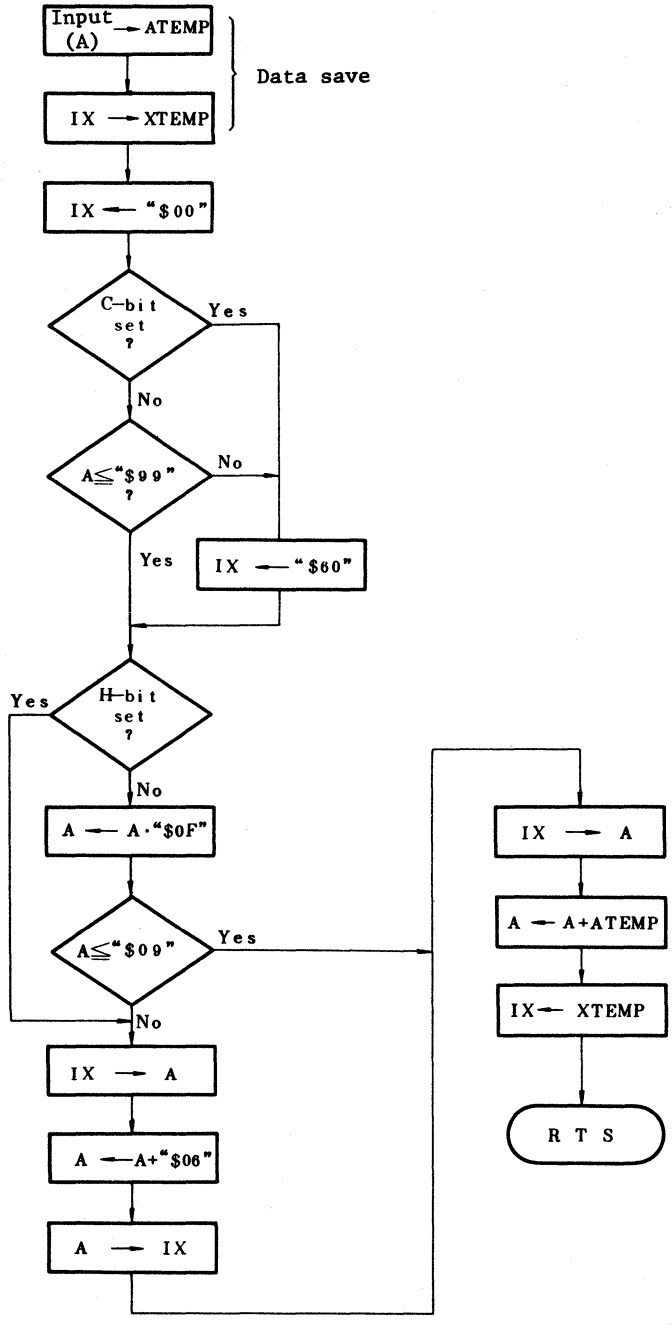


Figure 6-1 DAA Subroutine Flowchart

Table 6-2 DAA Subroutine Program List

ADDRESS	OP CODE					COMMENT
0044			ATEMP	RMB	1	Work area
0045			XTEMP	RMB	1	
0099	B7	44*	DAA	STA	ATEMP*	Saves the input value
009B	BF	45*		STX	XTEMP*	
009D	5F			CLR	X	Vacates the IX to store the converted value. Branches if the value is equal to or less than $99, otherwise the higher 4 bits are also converted.
009E	25	04		BCS	DAAH6	
00A0	A1	99		CMP	#$99	
00A2	23	02		BLS	DAALOW	
00A4	AE	60	DAAH6	LDX	#$60	
00A6	29	06	DAALOW	BHCS	DAAL6	
00A8	A4	0F		AND	#$0F	
00AA	A1	09		CMP	#$09	It is not necessary to convert if the lower 4 bits are less than $09.Therefore,branching is performed. The converted value of the lower 4 bits.
00AC	23	04		BLS	DAADNE	
00AE	9F		DAAL6	TXA		
00AF	AB	06		ADD	#$06	
00B1	97			TAX		
00B2	9F		DAADNE	TXA		Stores the converted value to A, adds it to the original(ATEMP) and prduces an output.
00B3	BB	44*		ADD	ATEMP*	
00B5	BE	45*		LDX	XTEMP*	
00B7	81			RTS		

* It is necessary to provide ATEMP and XTEMP within the stored RAM (Address $020~$07F) in the work area.

5

6.3 Cautions on the Programming the Write Only Register and Control Register

It is impossible to change the Write Only Register Contents (for example, the Data Direction Register (DDR) of the I/O port) of the HD63L05 MCU by applying the Read/Modify/Write instructions.

(1) The Write Only Register (for example the DDR of the I/O port) cannot read, the Read/Modify/Write instructions are executed in the following sequence.

(i) Reading the contents of the specified address
(ii) Changing the read-out data
(iii) Returning the changed data to the original address

Note that the Read/Modify/Write instructions cannot be applied to the Write Only Register such as the DDR.

(2) For the same reason, do not set the DDR of the I/O port or LCD register by using the BSET and BCLR instructions of the HD63L05 MCU.

(3) It is necessary to pay attention to the System Control Register, the Timer Control Register, and the A/D Control Register when BSET, BCLR, or Read/Modify/Write instructions are applied to them. If one's own interruption request occured onto the interruption request bit (bit 7) of the Control Register between read cycle and write cycle of these instructions, the bit 7 may be cleared in the write cycle and not acknowledged by CPU.

(4) Store instructions, such as STA or STX, are used to write correctly in the Write Only Register or to avoid missing an interruption of the Control Register.

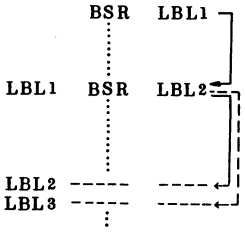
6.4 Cautions on Executing BSR (Branch SubRoutine) Instruction

Sometimes the HD63L05 MCU does not execute the address munipulation normally with the second BSR instruction, if there exists the BSR instruction in the address where branch has occurred. For this reason, do not execute the BSR instruction in the branched address.

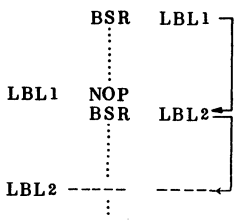
The following methods are available for substituting for the second BSR instruction.

(1) Insert the No OPeration (NOP) instruction before the second BSR instruction in order to cause a branch in the address in which the NOP instruction has been inserted.

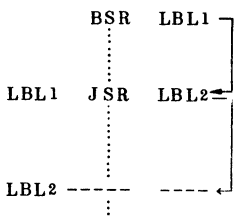
(2) Convert the BCR instruction in the branched address into Jump SubRoutine (JSR) instruction.



An example of a program which will not execute branch normally.



An example of a program in which NOP instruction is inserted.



An example of a program using JSR instruction.

6.5 Cautions

(1) To prevent the induced noise, the crystal oscillators should be located as close to the LSI as physically possible. Signal lines should not run pararell with, or across the clock oscillator inputs, such as XTAL, EXTAL, XIN, XOUT. In particular, these clock oscillator inputs should be separated as much as possible.

(2) Functions and Cautions

A. When STAND-BY goes "High", OSC1, system operation, and peripheral circuits stop. (OSC2 operates). On power up, or when reset, pull STAND-BY "Low".

B. OSC1, system operation, and peripheral circuit restart functioning by releasing them from stand-by mode after the delay time caused by the frequency divider.
If OSC2 is provided, clock is provided from OSC1 with the system after the delay time selected by $FFl.
If OSC2 is not provided, delay time means the period in which

OSC1 restarts functioning, after released from stand-by and the period in which judging output of frequency divider, changes by clock pulse transmitted to the delay circuit after restarting the functions.

Note that the restart time of the system operation cannot be prescribed precisely when using crystal oscillation, as crystal oscillation affects both the restart time of the oscillation and harmonic oscillation when restarting the oscillation.

C. In stand-by status, peripheral circuits (e.g. Time Base, Timer, A/D converter, and LCD driver) stop functioning. When Time Base, Timer, A/D converter go into stand-by mode during functioning, Timer and Time Base stop functioning and the result of A/D conversion cannot be assurred. Transmit the signal specifying if stand-by input is available in order to control the stand-by signal. Then input the stand-by signal to the port.

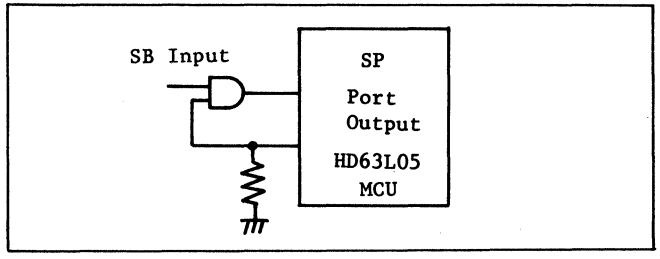


Figure 6-2 Example of SB Input Masking

7. Evaluation Chip (HD63L05E0)

The HD63L05E0 is a CMOS evaluation chip for the HD63L05 MCU. Connecting an external EPROM (HN482732) to the chip, it can be operated as a single chip microcomputer HD63L05 MCU. This chip is moulded in a 100 pins flat package. (Refer to Figure 7-1)

7.1 Block Diagram

The following describes input signals and output signals of the HD63L05E0 MCU. Basically, the same pin name means the same function as the HD63L05 MCU.

- V_{CC}, V_{SS}
  Power is supplied to the LSI by using these pins. V_{CC} has a voltage of 3.0V±0.8V and Vss is grounded.

- $\overline{INT}$
  This pin is used to interrupt the LSI processing externally.

- XTAL, EXTAL
  Input/Output Pins for the internal oscillator. A crystal (400 kHz typ.) is connected to these pins.

- TIMER
  An external input pin to count down the internal timer circuit.

- $\overline{RES}$
  This pin resets the LSI.

- STANDBY (SB)
  This external input pin stops the LSI and holds data.

- A/D Input Pins ($CH_1 \sim CH_8$)
  This is an input pins for analog voltages needed for A/D conversion. These may also be used as level check input under program control.

- VRH, VRL
  Reference voltages for A/D conversion are applied to these two pins.

- CC_1, CC_2
  These pins are not user application. They should be left open.

- XIN, XOUT

  A crystal (32.768kHz) is connected to these pins, for OSC2 oscillation. When OSC2 is not used, connect XIN to V_{CC}.

- VCH

  Decoupling pin from internal voltage regulator. Connect a capacitor (0.5μF) between VCH and V_{CC} to stabilize the voltage.

- $\overline{\text{MSET}}$

  This pin is not for user application. Connect it to V_{CC}.

- ADCLK

  This pin transmits a signal with 1/4 frequency of OSC1 (100kHz typ. synchonized with system clock E). NMOS open-drain output.

- U/M

  The HD63L05E0 MCU can take two operation modes based on the state of this pin. When the pin is connected to V_{CC}, the LSI operates as a single chip microcomputer with external EPROM. When the pin is grounded, however, the HD63L05E0 MCU operates in external extension mode.

- Input/Output Pins ($A_0 \sim A_7$, $B_0 \sim B_7$, $C_0 \sim C_3$)

  These 20 pins consist of two 8-bit ports and one 4-bit port. Each pin functions as an input or output under program control of the Data Direction Register. These are NMOS open-drain output.

- $D_0 \sim D_7$

  These pins are input pins for instruction or data from external data bus. For example, output from an external EPROM are applied to these pins.

- $E_0 \sim E_7$

  These pins are NMOS open-drain outputs. When the U/M is logical "1", the internal address bus ($A_0 \sim A_7$) is transmitted. When the U/M is logical "0", the port E functions either as address bus or as data bus. When system clock E is "Low", port E functions as address bus. When system clock E is "High", port E functions as the peripheral data bus.

- $F_0 \sim F_3$

  These pins are NMOS open-drain outputs. When the U/M is logical "1", the internal address bus ($A_8 \sim A_{11}$) is transmitted. When

the U/M is logical "0", the port F transmits internal address bus while system clock E is "Low".

- $\overline{CE}$/WR

  This pin is a NMOS open-drain output. When the U/M is "High", $\overline{CE}$ signal (means address bus is in from $080 to $FFF) is transmitted. When the U/M is "Low", Read/Write clock is transmitted.

- $\overline{LIR}$

  NMOS open-drain output. Fetch signal is available from this pin.

- $\overline{HALT}$

  NMOS open drain output. When stand-by signal is acknowledged, the output from this pin goes "Low" to control external clock source.

- V1, V2

  These are pins for LCD driver. When 1/3 bias - 1/3 duty drive, connect V1 and V2 to V_{CC} through condensor (0.1μF each).

- Liquid Crystal Driver Pins ($COM_1 \sim COM_3$, $SEG_1 \sim SEG_{17}$)

  $COM_1 \sim COM_3$ are for driving common electrodes, while $SEG_1 \sim SEG_{17}$ are output pins for driving segments. $SEG_{11} \sim SEG_{17}$ functions as either $SEG_{11} \sim SEG_{17}$ or as A/D inputs ($CH_2 \sim CH_8$) by specifying the function through selecting master-slice data. $SEG_{11} \sim SEG_{17}$ are multiplexed with A/D inputs ($CH_2 \sim CH_8$).

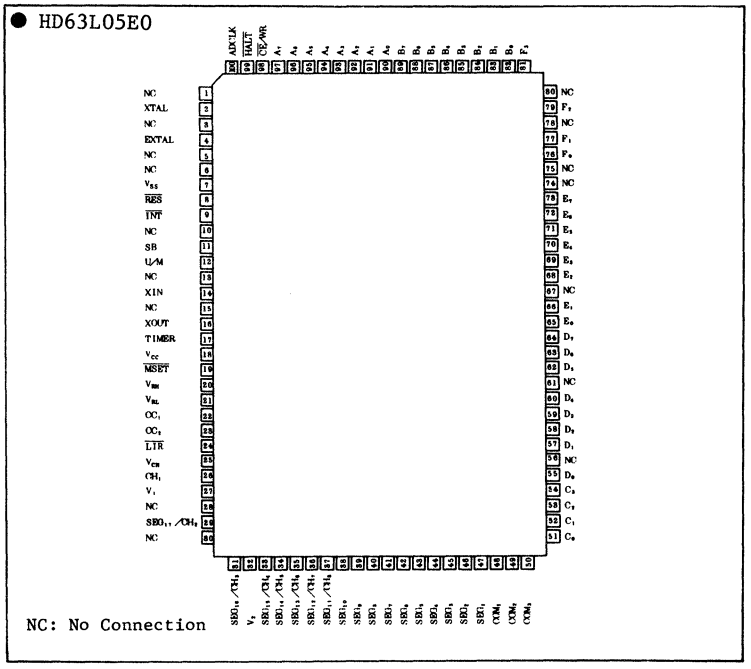


Figure 7-1 (a) Pin Assignment (Top View)

Unit: mm(inch)

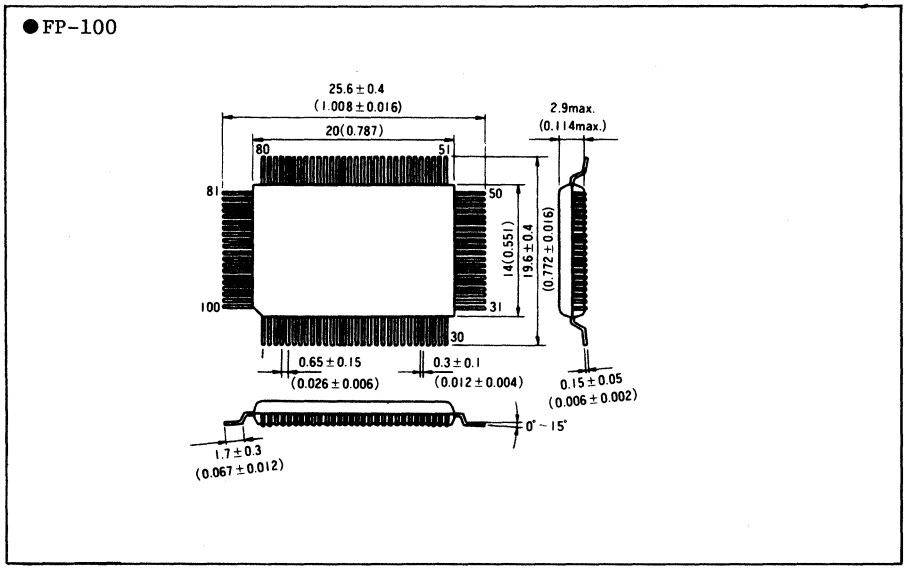
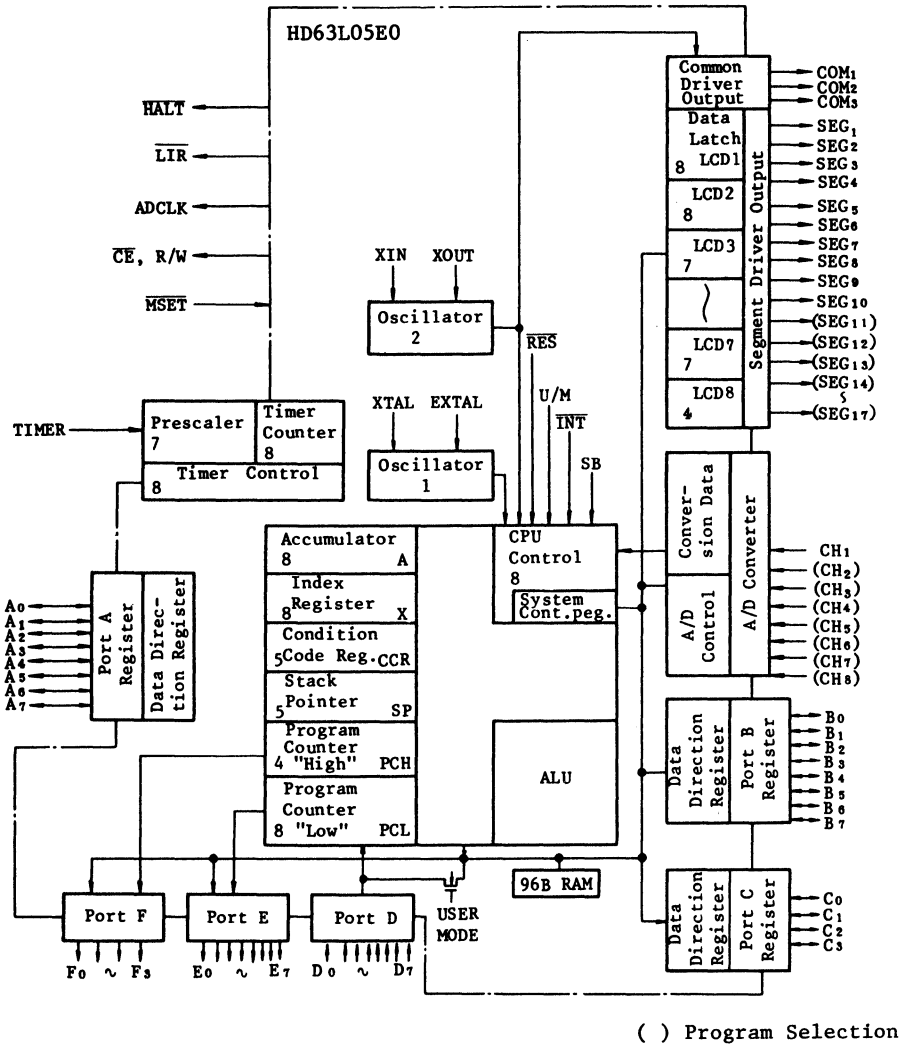


Figure 7-1 (b) Package Information



( ) Program Selection

Figure 7-2 HD63L05E0 Block Diagram

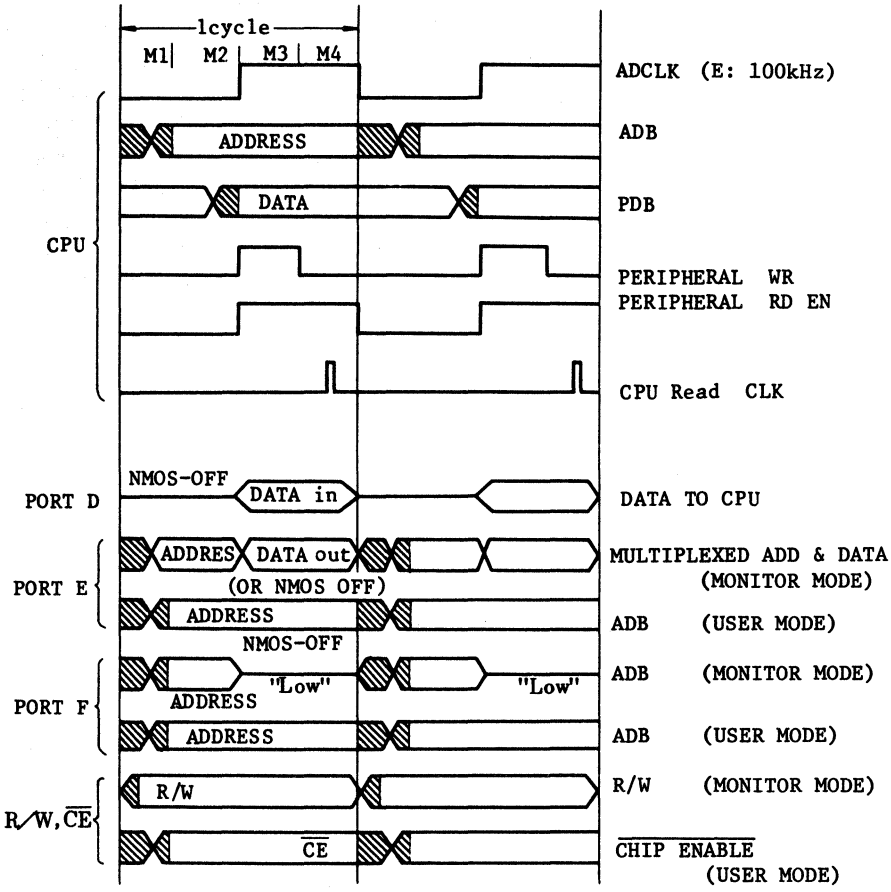


Figure 7-3 HD63L05E0 Timing Chart

7.2 Memory Map

Figure 7-4 shows a memory map of the HD63L05E0 MCU. $000∿$07F is contained in the LSI. $080∿$FFF is not contained, thus it is necessary to connect external EPROM (HN482732) to $080∿$FFF. $F30∿$FF3 is not for user application, for Hitachi will replace these addresses by the self check program.

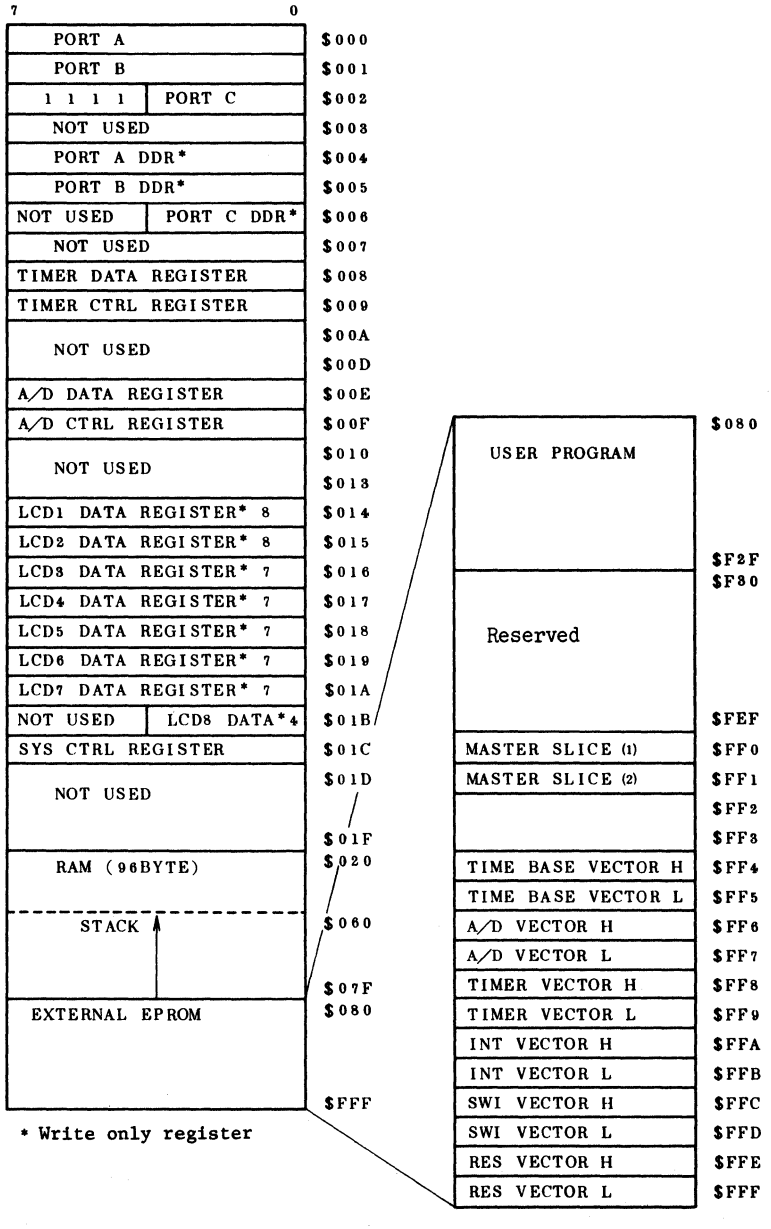


Figure 7-4 Memory Map

7.3 Pin Functions and Applications

(1) Power

3V is supplied to V_{CC}, then V_{DD} is grounded. V_{CC} is connected to V_{CH} through a condensor. ($V_{CH} \simeq +1V$ is transmitted) V_1 and V_2, which are connected to V_{CC} through a condensor, are voltage pins for LCD drive. ($V_1 \simeq +1V$, $V_2 \simeq +2V$ are transmitted.)

(2) Control Signals

Connect U/M and $\overline{MSET}$ to V_{CC}.

(3) Interfacing to the user system.

(a) I/O port ($A_0 \sim A_7$, $B_0 \sim B_7$, $C_0 \sim C_7$)

Each pin has NMOS open-drain output. Therefore, "High" level of the output is obtained by connecting a resistor to V_{CC}. When using as an input port, connect pull-up resistor to V_{CC}, if necessary.

(b) Control pins ($\overline{RES}$, $\overline{INT}$, SB, TIMER)

Only $\overline{RESET}$ is connected to V_{CC} through internal pull-up PMOS in the LSI. To avoid the floating input to $\overline{INT}$, S.B, or TIMER, connect pull-up resistors between these terminals and V_{CC}, if necessary.

(c) Others ($SEG_1 \sim SEG_{17}$, $COM_1 \sim COM_3$, $CH_1 \sim CH_8$, CC_1, CC_2, V_{RH}, V_{RL})

Bit correspondence between $SEG_1 \sim SEG_{17}$ and LCD register is fixed, which fixes the pin allocation and does not support the port function as an output only port.

Note that the pin allocation of the HD63L05E0 MCU is not equivalent to that of the HD63L05 MCU.

The selection between $SEG_{13} \sim SEG_{17}$ (and master slice analog input $CH_2 \sim CH_8$) can be specified by the external EPROM data. V_{RH} and V_{RL} of the HD63L05E0 MCU is equivalent to those of the HD63L05 MCU.

(4) Interfacing to External EPROM

(a) Data inputs ($D_0 \sim D_7$)

Connect the output pins from the EPROM to these pins.
Provide pull-down resistor to reduce input high level.

(b) Address Outputs ($E_0 \sim E_7$, $F_0 \sim F_3$)

The address signals ($A_0 \sim A_{11}$) is transmitted from these pins to EPROM. Connect them to V_{CC} of the EPROM with pull-up resistors.

(c) Chip Select Output ($\overline{CE}$)

Connect it to V_{CC} of the EPROM with pull-up resistor. When ROM address is selected (from $080 to $FFF), "Low" level is available.

(5) Others ($\overline{HALT}$, $\overline{LIR}$, $\overline{ADCLK}$)

Normally, these pins are not user application. Keep them open.

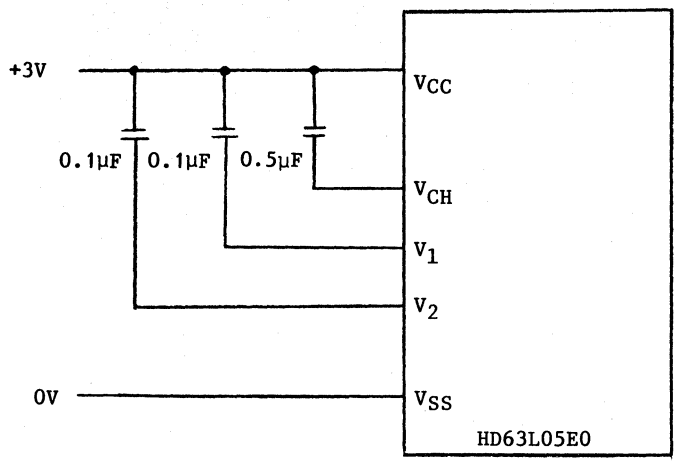
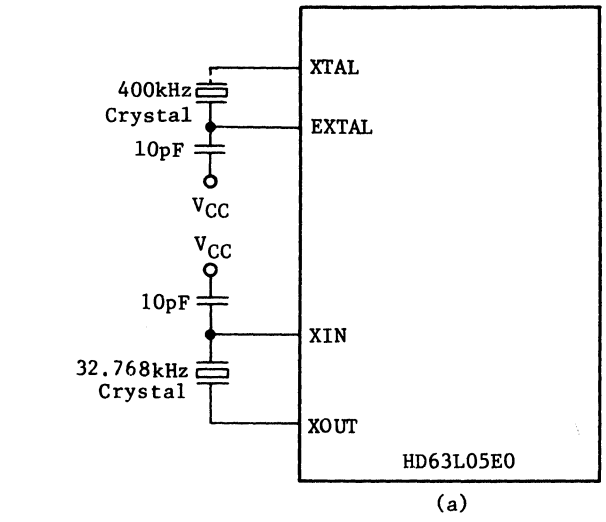
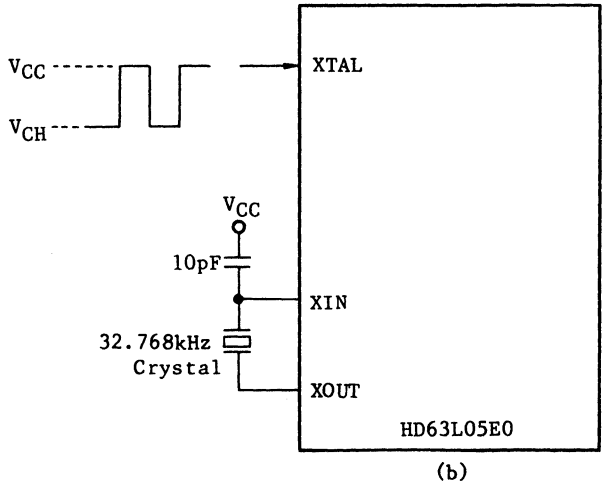


Figure 7-5 Connections for Power Supplying



(a)



(b)

Figure 7-6 Connections for the Oscillators

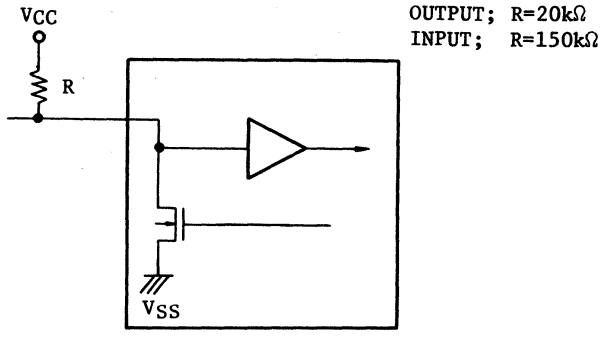


Figure 7-7 Configuration of NMOS open-drain Output

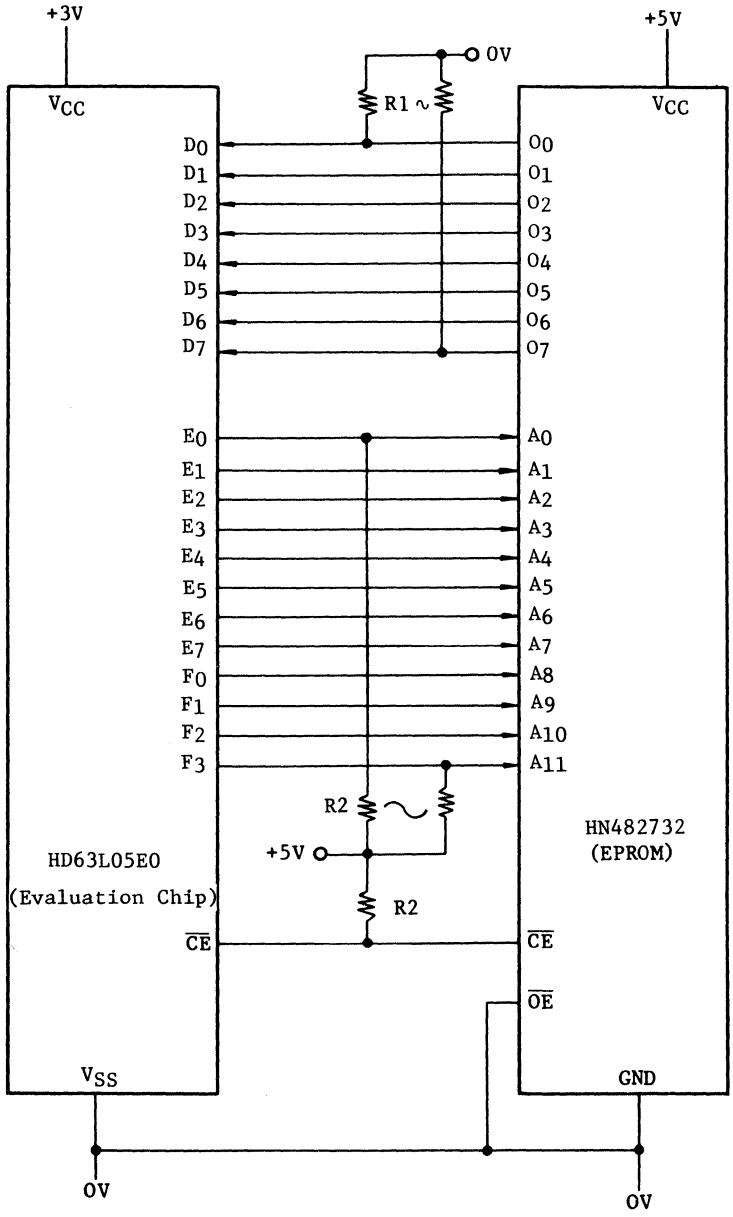


Figure 7-8 Interfacing between HD63L05E0 and EPROM

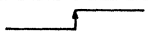
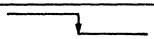
R1, R2 = 20kΩ

5

7.4 The Comparison between the HD63L05 MCU and the HD63L05E0

The HD63L05E0 MCU is an evaluation chip for the HD63L05 MCU, which supports the HD63L05 MCU function by connecting EPROM externally. However, these two devices have some differences in the architectures. Table 7-1 summarizes these.

Table 7-1 The Comparison between the HD63L05 MCU and the HD63L05E0 MCU

NO.	ITEM		HD63L05	HD63L05E0	
				HD63L05E0	NOTE
1	Operating Voltage		2.2V∿3.8V	←	Current cannot be supported.
2	OSC1		XTAL/CR Mask option	XTAL	CR oscillation is not supported.
3	OSC2		with OSC2/without mask option	←	Selectable
4	$\overline{RES}$			←	Master slice data is set while reset="L"
5	$\overline{INT}$			←	
6	TIMER		Timer input pin	←	
7	NUM	V_{CC}	Normal mode	User Mode	Equivarent to U/M
		V_{SS}	Test mode	Monitor Mode	Not supported
		$1/2V_{CC}$	Self check mode	—	Not supported
8	STANDBY		Standby input	←	Standby delay time is selectable
9	$\overline{MSET}$		—	Present	Not used
10	V_{RH}/V_{RL}		A/D Standard voltage	←	
11	CC_1/CC_2		Not used	←	
12	$CH_1 \sim CH_8$		A/D input pins	←	pin allocation of CH_7/CH_8 is different
13	Port input type		Pull-up R mask option	Without pull-up R	
14	Port output type		CMOS NMOS open drain	NMOS Open drain	
15	LCD pin		1/3 bias 1/3 duty, static,output pin, master slice	Only 1/3 bias 1/3 duty is supported	Fixed pin location
16	Pin location		Specified by mask option	Fixed pattern	
17	V_1, V_2		Liquid power source/ CH_7,CH_8,	Crystal source.	Pin allocation of CH_7/CH_8 is different
18	V_{CH}		LSI	←	
19	E		Goes to "Low" when Reset/Halt/S.B.	Goes to "Low" when S.B.	Equivalent to ADCLK
20	Port D		—	Data input port	
21	Port E		—	Address,data output port	
22	Port F		—	Address output port	
23	$\overline{HALT}$		—	Goes to "Low" when S.B.	
24	$\overline{LIR}$		—	Goes to "Low" when Fetch cycle	
25	$\overline{CE}$/WR		—	$\overline{CE}$="Low" in the address $080∿$FFF	
26	PACKAGE		FP-80/DP-64S	FP-100	

7.5 LCD Output

Note that LCD output of the HD63L05E0 MCU is for 1/3 bias-1/3 duty drive and its pin location block is fixed.

Table 7-2 Connections of the Pin location Block

LCD register	Timing	SEGMENT Pin
LCD1−0	COM_2	SEG_2
1	COM_3	SEG_2
2	COM_1	SEG_3
3	COM_1	SEG_2
4	COM_1	SEG_1
5	COM_2	SEG_1
6	COM_3	SEG_1
7	Not used	Not used
LCD2−0	COM_2	SEG_4
1	COM_3	SEG_5
2	COM_2	SEG_5
3	COM_1	SEG_4
4	COM_2	SEG_3
5	COM_3	SEG_3
6	COM_3	SEG_4
7	Not used	Not used
LCD3−0	COM_2	SEG_6
1	COM_2	SEG_7
2	COM_1	SEG_7
3	COM_1	SEG_6
4	COM_1	SEG_5
5	COM_3	SEG_6
6	COM_3	SEG_7
LCD4−0	COM_2	SEG_9
1	COM_3	SEG_9
2	COM_1	SEG_{10}
3	COM_1	SEG_9
4	COM_1	SEG_8
5	COM_2	SEG_8
6	COM_3	SEG_8
LCD5−0	COM_2	SEG_{11}
1	COM_3	SEG_{12}
2	COM_2	SEG_{12}
3	COM_1	SEG_{11}
4	COM_2	SEG_{10}
5	COM_3	SEG_{10}
6	COM_3	SEG_{11}
LCD6−0	COM_2	SEG_{13}
1	COM_2	SEG_{14}
2	COM_1	SEG_{14}
3	COM_1	SEG_{13}
4	COM_1	SEG_{12}
5	COM_3	SEG_{13}
6	COM_3	SEG_{14}
LCD7−0	COM_2	SEG_{16}
1	COM_3	SEG_{16}
2	COM_1	SEG_{17}
3	COM_1	SEG_{16}
4	COM_1	SEG_{15}
5	COM_2	SEG_{15}
6	COM_3	SEG_{15}
LCD8−0	COM_2	SEG_{17}
1	COM_3	SEG_{17}
2	Not used	Not used
3	Not used	Not used

7.6 Setting the mask-option data

The HD63L05E0 MCU contains two additional registers to specify the master-slice data for SEG_{11}∿SEG_{17} (CH_2∿CH_8) and internal oscillator mask options.

During the $\overline{RES}$ input is "Low", address bus (A_0∿A_{11}: from Port E, Port F) become alternately $FF0 and $FF1. Therefore, the output data from the external EPROM (Address are $FF0 and $FF1) can be written into these registers through Port D. 3 cycles are necessary for writing the master-slice data into the registers.

Table 7-3 Master-slice select register in EPROM

	Address	Data	Bit 7	6	5	4	3	2	1	0
Master Slice Register (1)	$FF0	0	*	CH_2	CH_3	CH_4	CH_5	CH_6	CH_7	CH_8
		1	*	SEG_{17}	SEG_{16}	SEG_{15}	SEG_{14}	SEG_{13}	SEG_{12}	SEG_{11}
Master Slice Register (2)	$FF1	0	*	*	*	OSC2 Not used	0 sec	1/16 sec	1/2 sec	1 sec
		1	*	*	*	OSC2 Used	Set 1-bit within 4 bit to "1". Set others to "0".			

* : These bits are not used, write "0".
Note that only one bit of OSC1 Delay Time select bits can be set to logical "0".

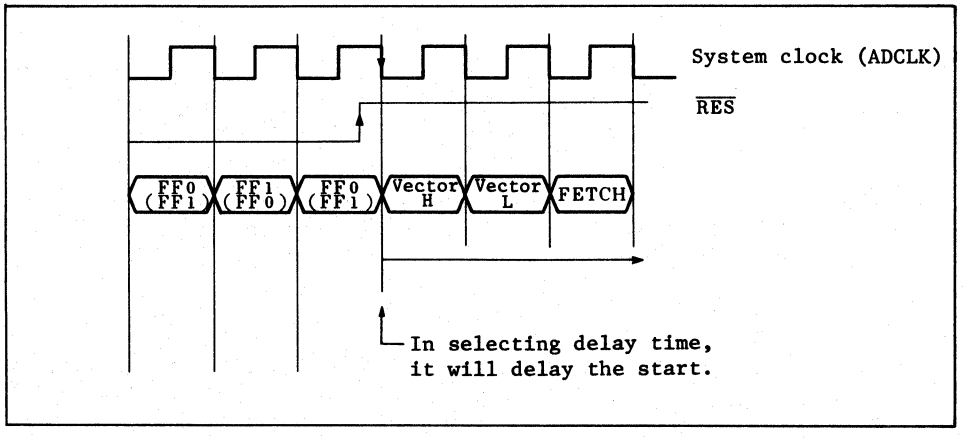


Fig. 7-9 Start from Reset

7.7 Electrical Characteristics

■ ABSOLUTE MAXIMUM RATINGS

Item	Symbol	Value	Unit
Supply Voltage	V_{CC}	-0.3 ∿ +5.5	V
Input Voltage	V_{in}	-0.3 ∿ V_{CC}+0.3	V
Output Voltage	V_{out}	-0.3 ∿ V_{CC}+0.3	V
Operating Temperature	T_{opr}	-20 ∿ +75	°C
Storage Temperature	T_{stg}	-55 ∿ +125	°C

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

● DC CHARACTERISTICS (V_{CC}=3.0±0.8V, V_{SS}=0V, Ta=-20 ∿ +75°C, typ means typical value at V_{CC}=3.0V unless otherwise noted.)

Item			Symbol	Test Condition	min	typ	max	Unit
Input "High" Level Voltage	XTAL, XIN		V_{IH}	Connect C_L=0.5μF to V_{CH}	V_{CC}-0.3	-	V_{CC}	V
	$\overline{RES}$, $\overline{INT}$, SB				0.5V_{CC}+0.9	-	V_{CC}	V
	TIMER				0.8V_{CC}	-	V_{CC}	V
	U/M (User Mode)				V_{CC}-0.2	-	V_{CC}	V
Input "Low" Level Voltage	XTAL, XIN		V_{IL}	Connect C_L=0.5μF to V_{CH}	V_{CC}-2.1	-	V_{CC}-1.8	V
	$\overline{RES}$, $\overline{INT}$, SB				Vss	-	0.2V_{CC}	V
	TIMER				Vss	-	0.2V_{CC}	V
	U/M (Monitor Mode)				Vss	-	0.2	V
Input Pull-Up Current	$\overline{RES}$		$-I_{R1}$	V_{CC}=3V, Vin=0V	3	15	30	μA
Input Leakage Current	TIMER, SB		$\|I_{IN}\|$	Vin=0V ∿ V_{CC}	-	-	1.0	μA
Current Dissipation	Crystal (400kHz)	During System Operation	I_{CC}	f=400kHz, No load. Tested after setting up the internal status.	-	100	200	μA
		At Halt			-	40	80	μA
		At Standby			-	2	5	μA
		At A/D Operation			-	200	600	μA

● AC CHARACTERISTICS (V_{CC}=3.0±0.8V, V_{SS}=0V, T_a=-20~+75°C, typ means typical value at V_{CC}=3.0V unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Operating Clock Frequency		fcl		100	400	500	kHz
Cycle Time		tcyc		8	10	40	µs
External Clock Duty		Duty		45	50	55	%
Oscillation Start Time (OSC1)		t_{OSCf}	C_D=10pF+20%, RS=1kΩ	-	-	150	ms
Oscillation Start Time (OSC2)		t_{OSC2}	C_D=10pF+20%, RS=20kΩ	-	-	1	s
Internal Capacitance of the Oscillator	OSC1	C_D	EXTAL	-	10	-	pF
	OSC2		XOUT	-	10	-	pF
Delay Time of Oscillation (Program)		t_{DLY}		0	-	1	s
Reset Delay Time		t_{RLH}		200	-	-	ms
$\overline{RES}$ Pulse Width		t_{RWL}		48	-	-	µs
$\overline{INT}$ Pulse Width		t_{IWL}		32	-	-	µs
TIMER Pulse Width		t_{TWL}		tcyc+1	-	-	µs

5

● PORT CHARACTERISTICS (V_{CC}=3.0±0.8V, V_{SS}=0V, T_a=-20∿+75°C, typ means typical value at V_{CC}=3.0V unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Output "Low" Level Voltage	Port A,B,C	V_{OL}	I_{OL}=100μA	—	—	0.3	V
Input "High" Level Voltage	Port A,B,C	V_{IH}		0.8V_{CC}	—	V_{CC}	V
Input "Low" Level Voltage	Port A,B,C	V_{IL}		Vss	—	0.2V_{CC}	V
Input Leackage Current	Port A,B,C	$\|I_{IN}\|$	Vin=0∿V_{CC}	—	—	1	μA
Output "Low" Level Voltage	ADCLK,$\overline{CE}$,$\overline{LIR}$ $\overline{HALT}$,Port E,F	V_{OL}	I_{OL}=200μA	—	—	0.3	V
Input "High" Level Voltage	Port D	V_{IH}		0.8V_{CC}	—	V_{CC}	V
Input "Low" Level Voltage	Port D	V_{IL}		Vss	—	0.2V_{CC}	V

● LCD DRIVER OUTPUT CHARACTERISTICS (V_{CC}=3.0V, V_{SS}=0V, T_a=20∿+75°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Output "High" Level Voltage	Segment	V_{OH1}	V_1= 1.00V, V_2= 2.00V I_{OH}= -1μA	2.8	-	-	V
		V_{OH2}		1.8	-	-	V
		V_{OH3}		0.8	-	-	V
Output "Low" Level Voltage	Segment	V_{OL1}	V_1= 1.00V, V_2= 2.00V I_{OL}= 1μA	-	-	2.2	V
		V_{OL2}		-	-	1.2	V
		V_{OL3}		-	-	0.2	V
Output "High" Level Voltage	Common	V_{OH1}	V_1= 1.00V, V_2= 2.00V I_{OH}= -5μA	2.8	-	-	V
		V_{OH2}		1.8	-	-	V
		V_{OH3}		0.8	-	-	V
Output "Low" Level Voltage	Common	V_{OL1}	V_1= 1.00V, V_2= 2.00V I_{OL}= 5μA	-	-	2.2	V
		V_{OL2}		-	-	1.2	V
		V_{OL3}		-	-	0.2	V
Dividing Resistor		R_{LCD}	Tested between V_1 and V_2	45	90	180	kΩ
Output "High" Level Voltage	Segment	V_{OH}	In the case of Output Port, I_{OH}=-30μA	V_{CC}-0.3	-	-	V
Output "Low" Level Voltage	Segment	V_{OL}	In the case of Output Port, I_{OL}= 30μA	-	-	0.3	V

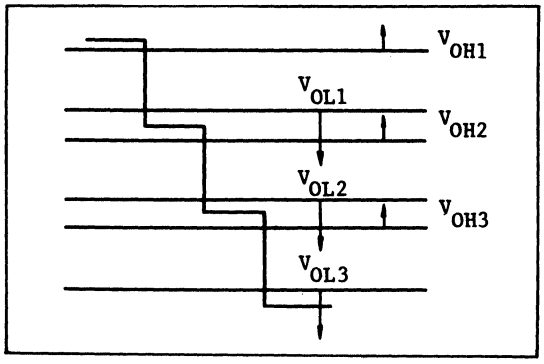


Figure 7-10 Output Level of SEG and COM

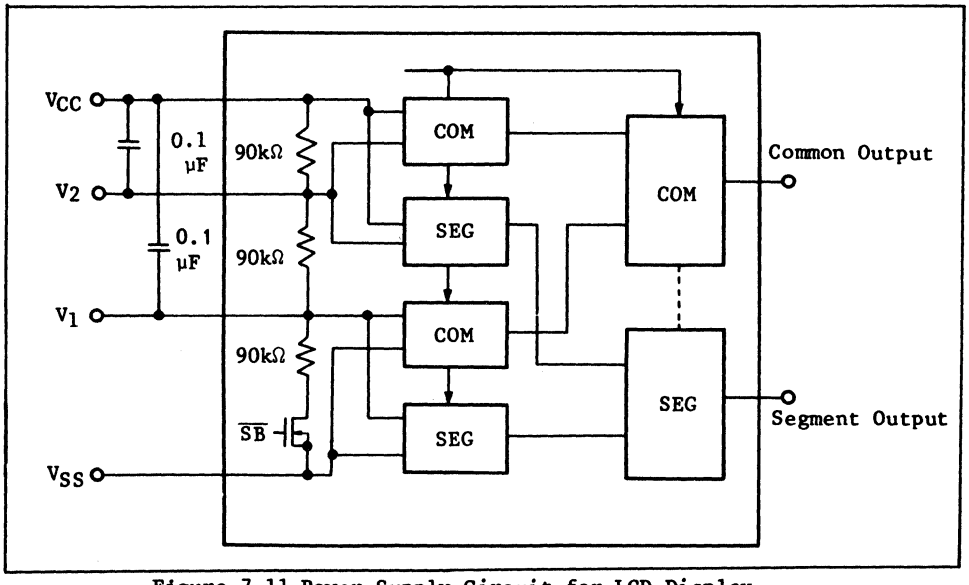
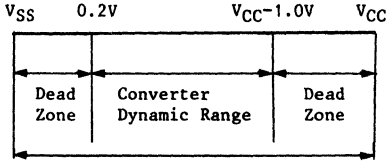


Figure 7-11 Power Supply Circuit for LCD Display

● A/D CONVERTER CHARACTERISTICS *(V_{CC}=3.0V, V_{SS}=0V, Ta=-20°C ∿+75°C unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Conversion Accuracy	Resolution			-	-	8	bit
	Absolute Accuracy		V_{RL}=0.2V < V_{in}<V_{RH}=2.0V	-2	-	+2	LSB
Reference Voltage	"High" Side	V_{RH}		-	-	V_{CC}	V
	"Low" Side	V_{RL}		V_{SS}	-	-	V
	V_{RH} - V_{RL}	ΔV_{REF}		1.8	-	-	V
Input Voltage Range	Input Range	V_{IN}		V_{RL}	-	V_{RH}	V
	Input Dynamic Range	V_{DYN}		0.2	-	V_{CC}-1.0	V
Ladder Resistor (V_{RH}- V_{RL})		R_{HL}		40	80	160	kΩ
Conversion Time		t_{CNV}		2	-	4	ms
Programmable Voltage Comparison	Judge Error		V_{RL}=0.2V<V_{in} < V_{RH}=2.0V	-4	-	+4	LSB
	Judge Time	t_{CMP}		-	-	60	µs

* These value can be changed without notice, because they are provisional.



(When the input voltage is in the dead zone, the result of the conversion is not guaranteed.)

Figure 7-12 Dynamic Range of the Comparator

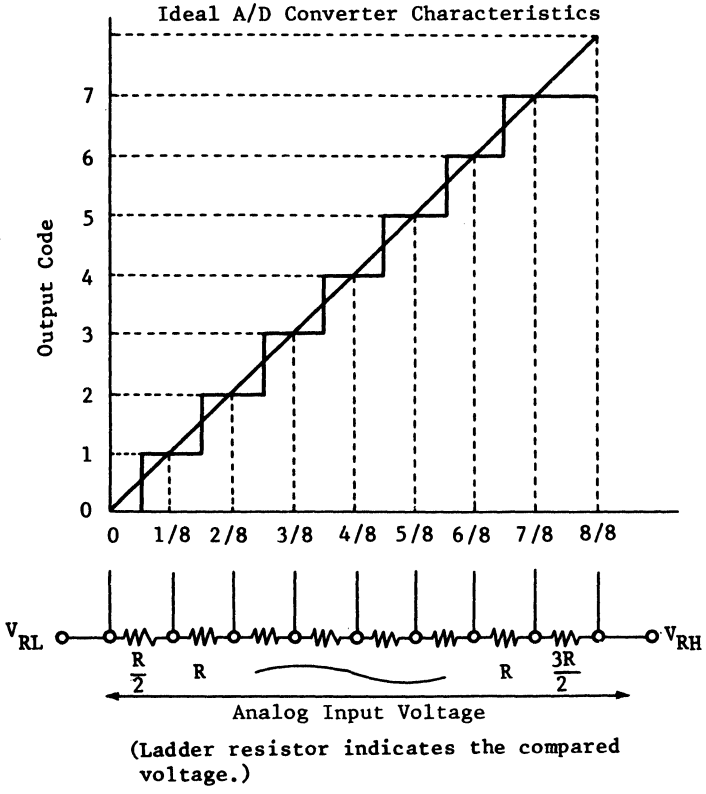


Figure 7-13 Example of 3 bit Resolution

8. ROM Code Order Method

User's programs are mask programmed into ROM by Hitachi to be shipped as LSI. Users are requested to hand in three EPROMs in which the same contents are written, order specifications, mask option list, and list of the ROM contents.

Relationship between the address of the mask ROM and that of the EPROM is shown in Table 8-1. Write $FF for the unused address data of the EPROM.

Table 8-1 Relationship between the Address of Mask ROM and that of EPROM

Type name	Address of Mask ROM	Address of EPROM	Remarks
HD63L05F1	$080 ≀ $F2F	$080 ≀ $F2F	User programs are written in this area.
	$F30 ≀ $FF3	$F30 ≀ $FF3	This area is used for the Self Check program by Hitachi.
	$FF4 ≀ $FFF	$FF4 ≀ $FFF	User supplied vectors are written into this area.

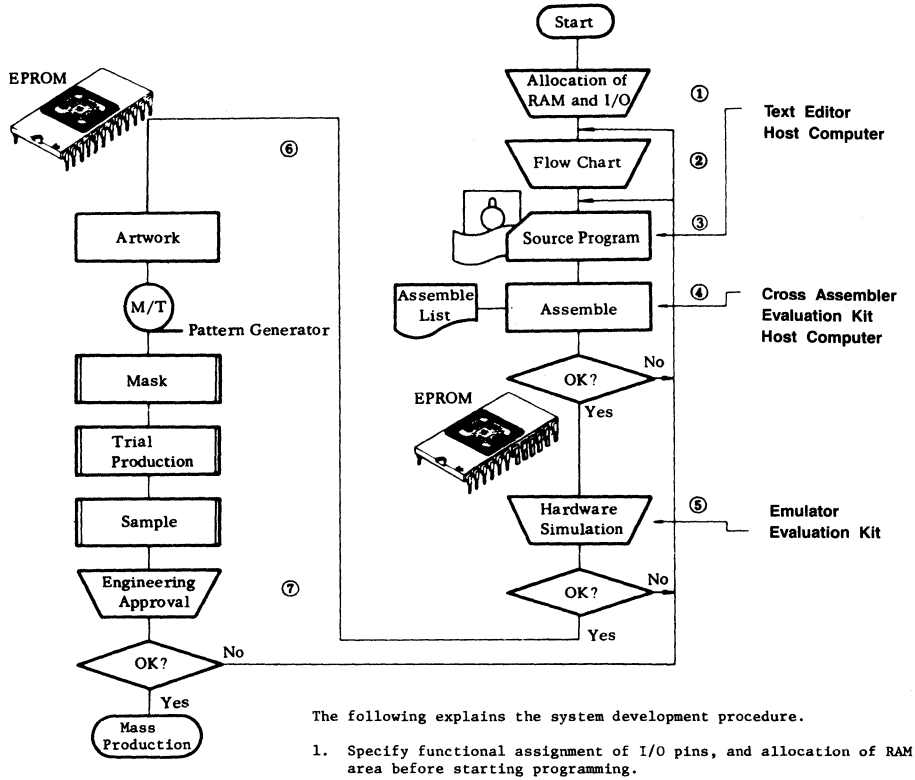
EPROM is a HN482732 or equivalent product.

APPENDIX

I. Design Procedure and Supporting Tool

Cross assembler and Hardware emulator, containing various kinds of computers, are available as supporting systems to develop users' programs. Hitachi will mask program users' programs into ROM to ship them as LSI.

Figure I-1 shows a typical program design procedure. Table I-1 summarizes a set of system development supporting tool for the HD63L05 MCU.



The following explains the system development procedure.

1. Specify functional assignment of I/O pins, and allocation of RAM area before starting programming.
2. Design flowchart to implement the functions and encode this flowchart with mnenic codes.
3. Write the coded format on cards or a floppy disk. This set of coded form is a source program.
4. Assemble the source program to form an object program with an assembler on either a resident system(evaluation kit) or cross system. Then check errors out.
5. Verify the program through hardware simulation by an aid of evaluation kit
6. Send the completed program in EPROM to Hitachi.
7. After Hitachi received users' specified ROM pattern and options, Hitachi will pilot product sample LSI for users' evaluation for the functions. If a user finds no problem in the sample LSI, Hitachi will start mass production of the LSI.

Figure I-1 Program Design Procedure

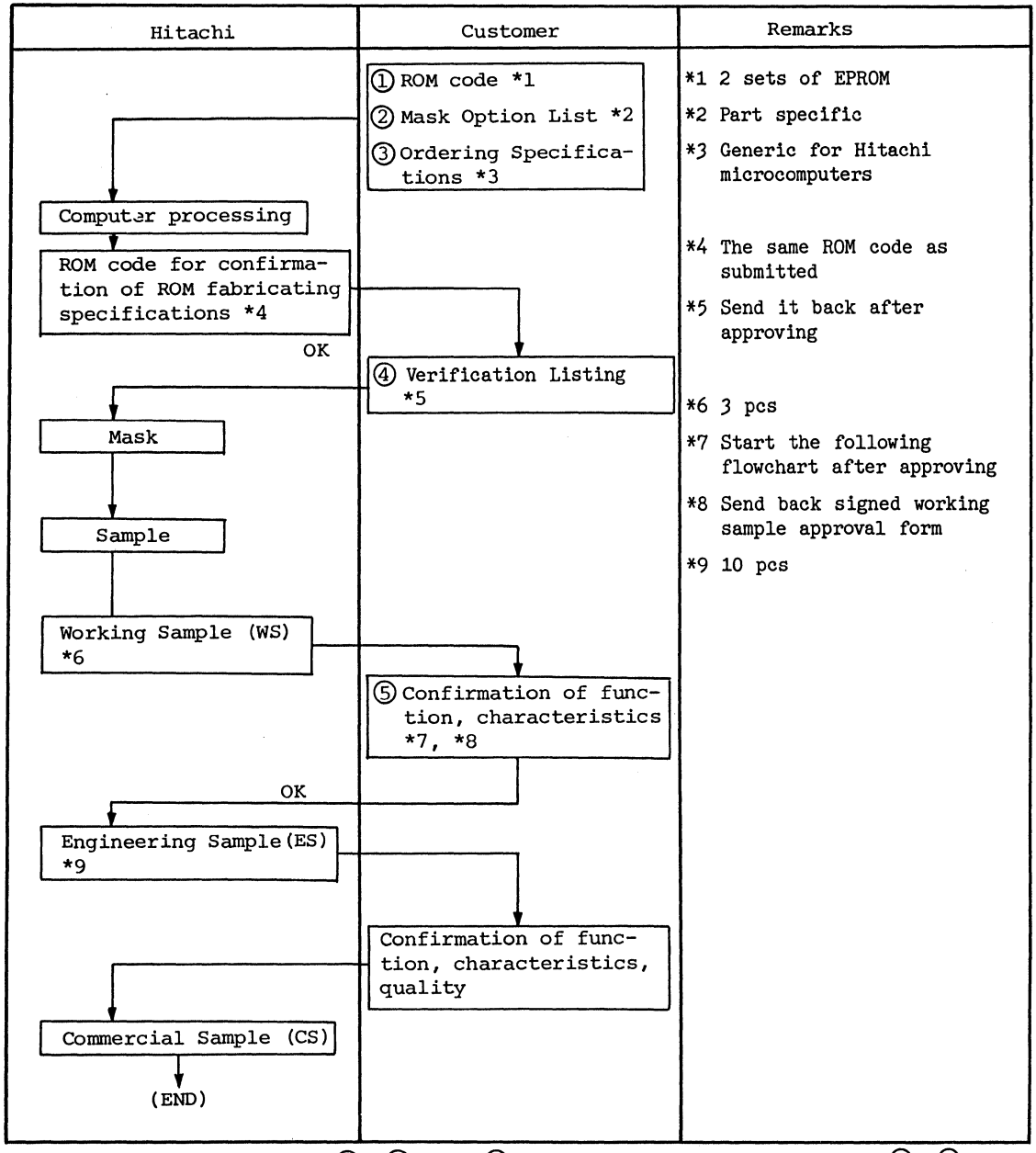
Table I-1 System Development Support Tool

Type No.	Evaluation Kit	IBM PC Cross Assembler
HD63L05F1	H3L5EVT1	S35IBMPC

Single Chip Microcomputer ROM Ordering Procedure

(1) Development Flowchart

Single chip microcomputer device is developed according to the following flowchart after program development.



(Note) Please send in ①, ②, and ③ at ROM ordering, and send back ④, ⑤ after approving.

Device Development Flowchart

(2) Data you send and precautions

(a) Ordering specifications ----- Common style for all Hitachi single chip microcomputer devices. Please enter as for the followings. The format is shown in the next page.

- Basic ITEM
- Environment Check List
- Check List of attached data
- Customer

(b) ROM code ----- Please send in the ordering ROM code by 2 sets of EPROM the same contents are written. Enter ROM code No. in them. It is desirable to send in program list for easy confirmation of the program contents.

(3) Change of ROM code

Note that if you change the ROM code once sended in or other specification, the ROM must be developed from the beginning. The cost of mask charge should be provided again in this case.

(4) Samples and Mass production

(Working Sample) ---------- Sample for confirmation of the contents of ROM code and that of mask option. Normally 3 samples are sent, but not guaranteed as for reliability. Please evaluate and approve immediately because the following sample making and mass production are set about after obtaining your evaluation.

(Engineering Sample) ------- Sample for evaluating also reliability. 10 pcs are included in mask charge.

(Commercial Sample) -------- Samples for pre-production which maybe purchased separately.

(Mass Product) ------------ Products for actual mass production. Please enter the plan of mass production in full.

HD63L05F1
ORDERING SPECIFICATIONS

(1) GENERAL CHARACTERISTICS (Fill in blank space or appropriate box☐).

Device Type		Package Outline (See Section) 3,4.1	☐ DP-40 ☐ CP-44 ☐ FP-54
Application (be specific)		Options/Remarks:	
Customer ROM Code ID			
ROM Code Media	☐ EPROM ☐ ZTAT™	Must Specify:	Customer Programmed Start Address _____ Customer Programmed Stop Address _____
Operating Temperature	☐ Standard	☐ J Specification (-40°C to +85°C), if offered	
Remask	☐ Yes ☐ No	Previous Hitachi P/N_____	

(2) OPERATING CHARACTERISTICS

LSI Ambient Temperature	Average	°C	Target Level Of Reliability	☐ 500 Fit ☐ (_____) ☐ 1000 Fit	
	Range	°C- °C			
LSI Ambient Humidity	Average	%	Acceptable Quality Level	☐ 1.0% ☐ 0.4% ☐ 0.65% ☐ (_____)	
	Range	%- %			
Power On Duration	Average	Hours/Day	Remarks:		
Maximum Applied Voltage To LSI	Power Supply	Max V			
	I/O	Max V			

(3) ELECTRICAL CHARACTERISTICS

☐ Purchasing Specifications _____	☐ Hitachi's Standard Specifications Refer To Data Sheet: _____

(4) CUSTOMER APPROVAL

Customer Name_____

PO#_____

Accepted By (print) _____

Accepted By (signature)_____

Date _____

For Hitachi Use Only

(5) ROM CODE VERIFICATION

LSI Type No.	
Shipping Date of ROM To Customer	
Approved Date of ROM From Customer	

HITACHI

HD63L05F MASK OPTION LIST

Date of Order	
Customer	
Dept.	
Accepted by	
ROM Code ID.	
LSI Type No.	HD63L05F

* Select one type for each item and check ●

(1) OSC Option

Select one type of OSC1 option.

Type of OSC1	Use of OSC2	Condition		Delay Time of Restart (Sec.)			
				0	1/16	1/2	1
*1 XTAL OSC.	Used	Stand-by Mode	Used	■	■		
			Not used				
	Not *2 used	Stand-by Mode	Used	■	■		
			Not used				
*3 CR OSC.	Used	Oscillation of OSC1 at halt	OSC1 OFF		■	■	■
			OSC1 ON				
	Not *2 used	Oscillation of OSC1 at halt	OSC1 OFF	■	■	■	■
			OSC1 ON				

*1 Crystal option of OSC1 is not allowed to stop at halt.
*2 If OSC2 is not used, the delay time is not accurate.
*3 All CR option of OSC1 allowed to use of stand-by mode.

(2) I/O Option

Specify an I/O option for each terminal.

Port	Mask Option			
	A	B	C	D
A_0				
A_1				
A_2				
A_3				
A_4				
A_5				
A_6				
A_7				
B_0				
B_1				
B_2				
B_3				
B_4				
B_5				
B_6				
B_7				
C_0				
C_1				
C_2				
C_3				

Pin	Mask Option	
	E	F
INT		

Pin	Mask Option		
	G	H	K
SEG_{13}/CH_6			■
SEG_{14}/CH_5			■
SEG_{15}/CH_4			■
SEG_{16}/CH_3			■
SEG_{17}/CH_2			■
$O18/CH_8/V_2$ *4			
$O19/CH_7/V_1$ *4			

*4 Specify K-type if L-type is selected at LCD driver.

A: Output without input pull-up PMOS.
B: Output with input pull-up PMOS.
C: Output for key scanning.
D: Open drain output. (Max. V_{CC})
E: Input without pull-up PMOS.
F: Input with pull-up PMOS.
G: A/D input.
H: Segment output.
K: Terminals for LCD display.

(3) LCD Driver

Specify a type of LCD driver.

	Mask Option		
	L	S	P
Segment			

L: 1/3 bias-1/3 duty LCD.
S: Static LCD.
P: Output Port.

... Mask options indicated as ■ are not available. ...

LCD Pin Location

LCD Register	BIT	Timing			Segment Output Terminal																			
		COM 1	COM 2	COM 3	SEG 1	SEG 2	SEG 3	SEG 4	SEG 5	SEG 6	SEG 7	SEG 8	SEG 9	SEG 10	SEG 11	SEG 12	SEG 13	SEG 14	SEG 15	SEG 16	SEG 17	O 18	O 19
LCD1	0																						
	1																						
	2																						
	3																						
	4																						
	5																						
	6																						
	7																						
LCD2	0																						
	1																						
	2																						
	3																						
	4																						
	5																						
	6																						
	7																						
LCD3	0																						
	1																						
	2																						
	3																						
	4																						
	5																						
	6																						
LCD4	0																						
	1																						
	2																						
	3																						
	4																						
	5																						
	6																						
LCD5	0																						
	1																						
	2																						
	3																						
	4																						
	5																						
	6																						
LCD6	0																						
	1																						
	2																						
	3																						
	4																						
	5																						
	6																						
LCD7	0																						
	1																						
	2																						
	3																						
	4																						
	5																						
	6																						
LCD8	0																						
	1																						
	2																						
	3																						
φWRITE		O																					
1/4-OSC1		O																					

* Specify the combination of timing and segment output terminal for each bit of LCD1 to LCD8.
* When static or output port is selected. The timing is fixed at COM1. So specify COM1.
* Don't specify the timing or segment output terminal for the terminal A/D input is selected at (2) I/O option.
* Specify the LCD pin-location even in such a case as unused segments are provided, and program to put out those segments. The indication of segments cannot be defined as Hitachi will connect unspecified segment to any registers to prevent circuit from malfunctioning.

Section Six

Software Application Notes

FOREWORD

The HD6305 is a family of 8-bit single chip CMOS microcomputers controlled by microprogramming. This family provides an easy to use instruction set by adding instructions for decimal adjustment and a low power consumption mode to those of the NMOS HD6805 FAMILY.

APPLICATION NOTES summarize typical programs for the HD6305 FAMILY to help users better understand the instruction set and to provide them with references for making more customized programs.

Programs described in APPLICATION NOTES have already been debugged. However, please be sure to check the operation in actual use.

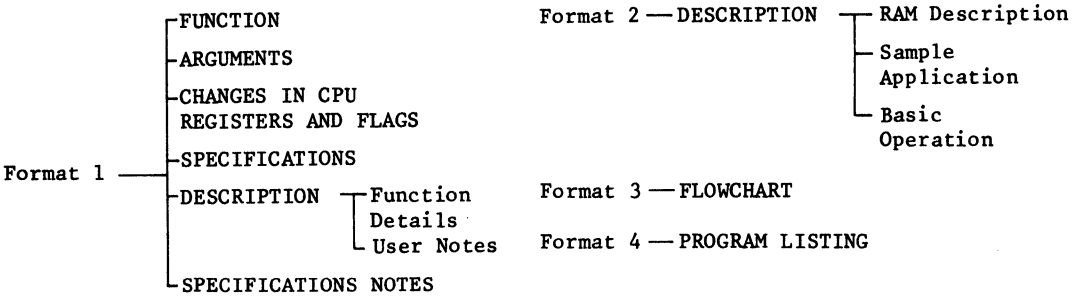
Section 6
Software Application Notes
Table of Contents

APPLICATION NOTES GUIDE

1. How to Use APPLICATION NOTES

1.1 Formats

APPLICATION NOTES consist of Formats 1 to 4, shown in Fig. 1.1.



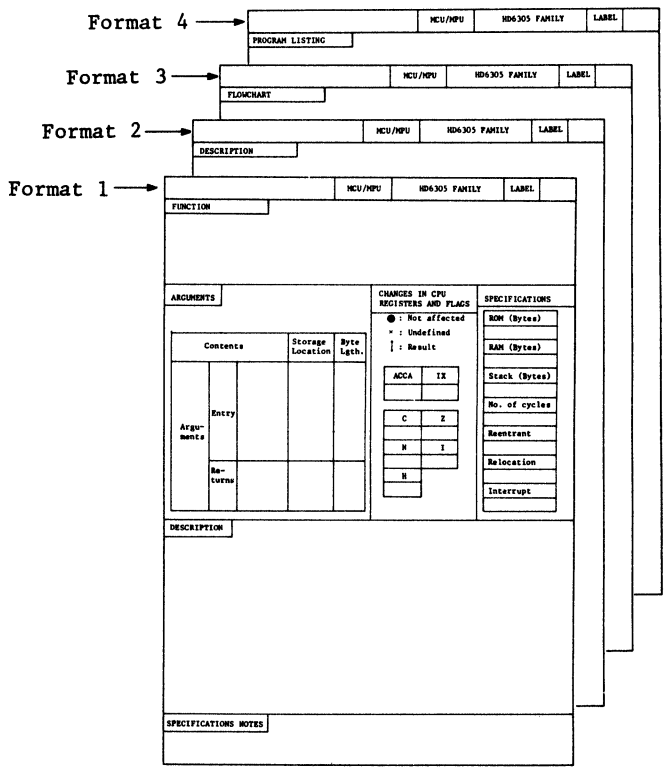


Fig. 1.1 APPLICATION NOTES Formats

Programs in APPLICATION NOTES can be implemented in two ways, i.e. (1) without change or (2) partially changed. Read the information that applies to the type of implementation to be carried out.

(1) Without change

(a) All of Format 1

(b) RAM Description and Sample Application in Format 2

(c) PROGRAM LISTING in Format 4

(2) Partially changed (user originals)

All of Formats 1 to 4 after reading these formats, change the FLOWCHART and PROGRAM LISTING according to user specification.

1.1.1 SPECIFICATION Format (Format 1)

SPECIFICATION Format is represented in Fig. 1.2. It gives program functions and specifications. Each item in the format is described using Fig. 1.2.

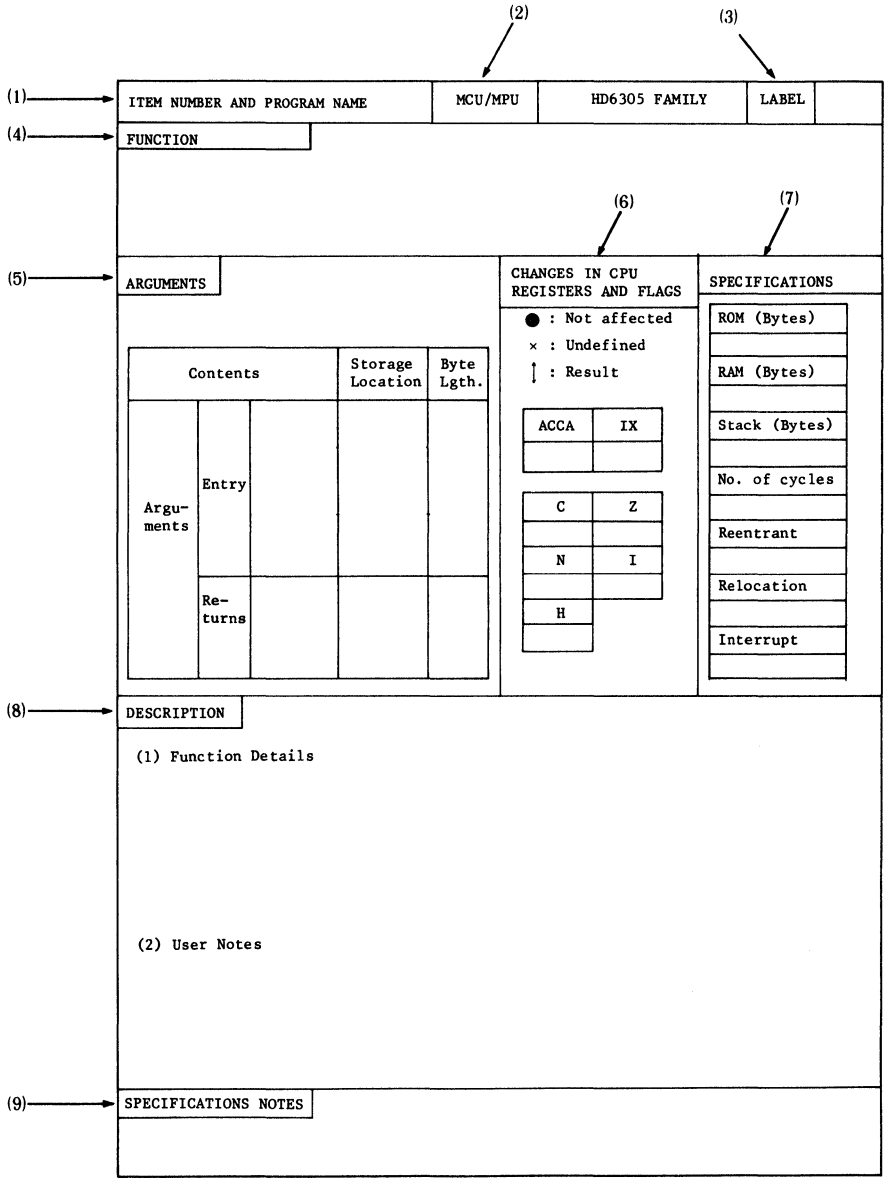
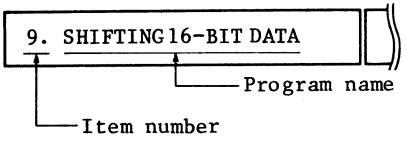


Fig. 1.2 SPECIFICATION Format

(1) ITEM NUMBER AND PROGRAM NAME:

Indicates item number and program name in APPLICATION NOTES.

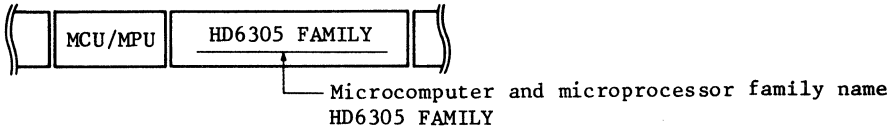
<Example>



(2) MCU/MPU:

Indicates names of microcomputer and microprocessor family applicable to a program.

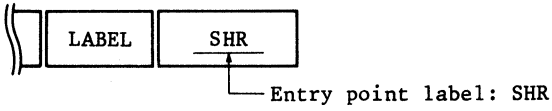
<Example>



(3) LABEL:

Indicates the name identifying program entry point.
When using a program as it is, call the label "SHR".

<Example>



(4) FUNCTION:

Explains program functions.

<Example>

FUNCTION
(a) Shifts 16-bit binary data in RAM to right. (b) Permits number of shifts to be freely determined. (c) Permits easy multiplication of 16-bit binary data by 2^{-n} (n : number of shifts).

(5) ARGUMENTS:

Explains entry arguments which must be set before execution of a program, and return arguments after execution.

(a) Contents:

Explains meanings of arguments.

(b) Storage Location:

Indicates registers and RAMs in which arguments are to be set. The RAM is presented as a label followed by "(RAM)".

(c) Byte Length:

Indicates byte length of the arguments.

<Example>

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	16-bit binary data to be shifted to right	SFT (RAM)	2
		Number of shifts	IX	1
	Returns	Shift results	SFT (RAM)	2

(6) CHANGES IN CPU REGISTERS AND FLAGS:

Explains changes in CPU registers after executing a program and flag changes of condition code register. Meanings of abbreviations and symbols in the table are given as follows:

(a) CPU register

ACCA: Accumulator A

IX: Index register

(b) Flags of condition code register

C: Carry/borrow flag (carry and borrow)

Z: Zero flag (Indication in case of 0)

N: Negative flag (Indication in case of negative)

I: Interrupt flag (Interrupt mask)

H: Half carry flag (Carry from bit 3 to bit 4)

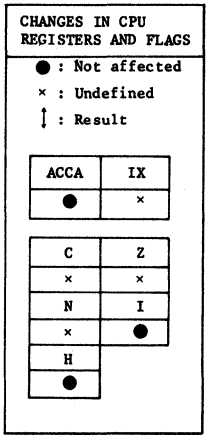
(c) State of CPU registers and condition code register flags

● : Not affected: Maintains previous values after executing a program.

× : Undefined : Does not maintain previous values after executing a program.

↕ : Result : Be set with the result of executing a program.

<Example>



(Notes)

In the example, after executing a program, contents of index register (IX), condition code register (CCR), bit C, bit N and bit Z will be destroyed. Thus, register contents which will be destroyed should be saved before executing a program.

(7) SPECIFICATIONS:

Explains a program specification.

(a) ROM (Bytes): Indicates ROM capacity used in a program.

(b) RAM (Bytes): Indicates RAM capacity used in a program.

(c) Stack (Bytes): Indicates stack size used in a program. The RAM capacity in this table does not include the stack size. When the program is executed, it is necessary to reserve the stack size in RAM.

(d) No. of cycles: Indicates maximum number of execution cycles when MCU executes a program. Calculate the execution time of the program as follows:

Execution time (sec) = Cycle number × cycle time

Cycle time (sec)= 4/(External oscillator (Hz))

(e) Reentrant : Indicates whether a program has a structure which can be called from two or more routines at the same time.

(f) Relocation : Indicates whether a program can be located in any memory space.

(g) Interrupt : Indicates whether MCU executes a program normally after serving an interrupt routine during program execution. If impossible, inhibit interrupt before the program is called.

<Example>

SPECIFICATIONS
ROM (Bytes)
8
RAM (Bytes)
2
Stack (Bytes)
0
No. of cycles
110
Reentrant
No
Relocation
No
Interrupt
Yes

(8) DESCRIPTION:

Explains detailed functions of a program and user notes.

(a) Function Details : Gives an execution example and detailed functions of a program.

(b) User Notes : Explains notes and limitations when executing a program.

* Be sure to read these items when using the programs without change.

<Example>

DESCRIPTION

(1) Function Details

(a) Argument details

SFT (RAM) : Holds 16-bit binary data to be shifted to right. After SHR execution, contains shift result.

IX : Holds number of 16-bit binary data to be shifted to right.

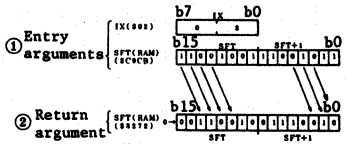


Fig. 1 Example of SHR execution

(b) Fig. 1 shows example of SHR execution. If entry arguments are held as shown in part ① of Fig. 1, 16-bit binary data is shifted to right as shown in part ② of Fig. 1.

(2) User Notes

Be sure to hold data into IX within range of $01 ≤ IX ≤ $0F.
When data outside this range is held, SFT (RAM) becomes "0".

(9) SPECIFICATIONS NOTES:

Explains notes on data process written in SPECIFICATIONS (7).

<Example>

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to shift 16-bit binary data to right by 7 bits.

1.1.2 DESCRIPTION Format (Format 2)

DESCRIPTION Format is represented in Fig. 1.3. It gives remaining Function Details, User Notes, RAM Description, Sample Application, and Basic Operation.

Each item in the format is described using Fig. 1.3.

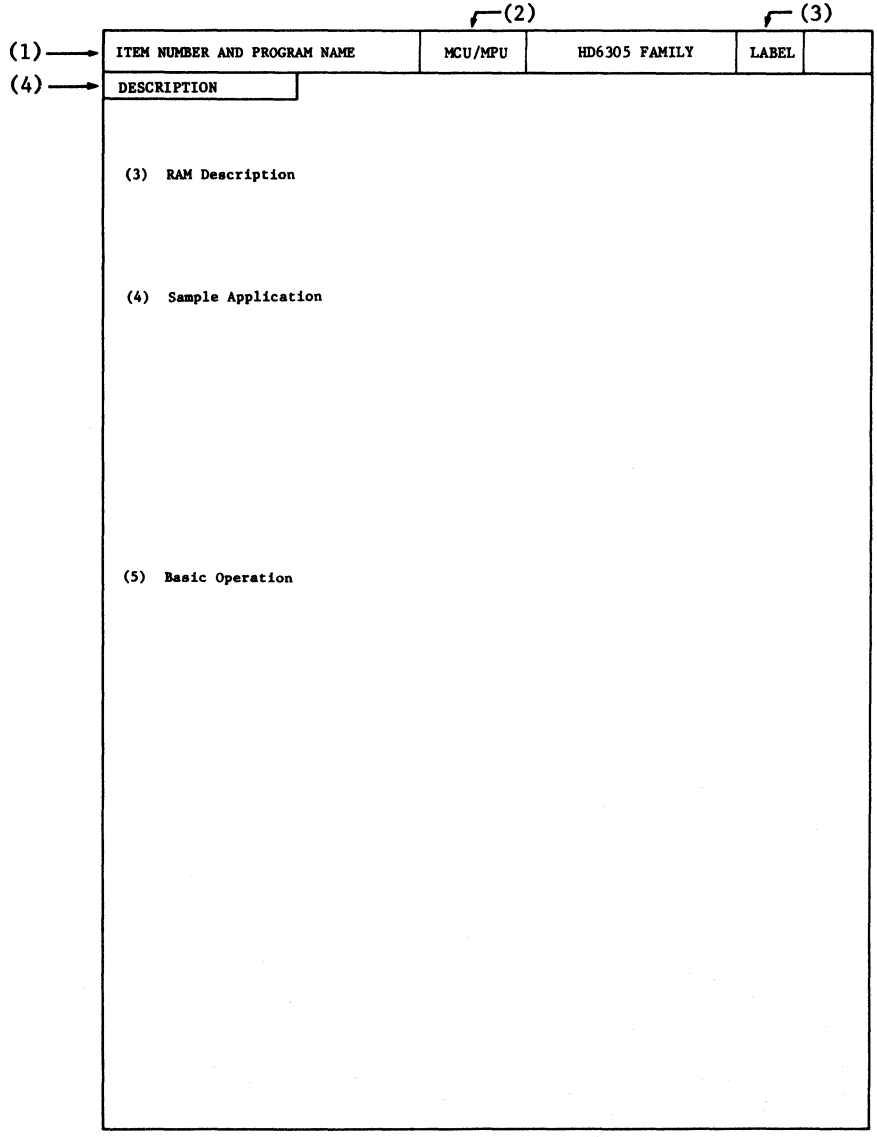


Fig. 1.3 DESCRIPTION Format

(1) ITEM NUMBER AND PROGRAM NAME
(2) MCU/MPU
(3) LABEL

} Same as SPECIFICATION Format

(4) DESCRIPTION:

Gives RAM Description, Sample Application, and Basic Operation

(a) RAM Description: Explains label and meaning of the RAM used in a program.

<Example>

9. SHIFTING 16-BIT DATA	MCU/MPU	HD6305 FAMILY	LABEL	SHR

DESCRIPTION

(3) RAM Description

Label	RAM		Description
	b7	b0	
SFT	Upper byte		16-bit binary data to be shifted to right is stored before execution.
	Lower byte		Shift result is stored after execution.

(b) Sample Application: Gives a sample application in actual use.

<Example>

(4) Sample Application

SHR subroutine is called after number of shifts and 16-bit binary data to be shifted to right are held.

```
WORK1   RMB   2        ------Reserves memory byte for 16-bit binary data.
WORK2   RMB   1        ----- Reserves memory byte for number of shifts.
WORK3   RMB   2        ----- Reserves memory byte for 16-bit binary
                             shift results.
WORK4   RMB   1        ----- Reserves memory byte for register contents
                             saving.
        :
        :
        STX   WORK4    ----- Saves register contents that will be
                             destroyed by executing SHR.
        LDA   WORK1
        STA   SFT      }----- Stores 16-bit binary data to be shifted
        LDA   WORK1+1  }      into right in entry argument (SFT).
        STA   SFT+1
        LDX   WORK2    ----- Loads number of shifts into entry argument
                             (IX).
        JSR   SHR      ----- Calls SHR subroutine.
        LDA   SFT
        STA   WORK3    }----- Stores shift results regarding 16-bit
        LDA   SFT+1    }      binary (return argument (SFT)) in RAM.
        STA   WORK3+1
        LDX   WORK4    ----- Restores register.
        :
        :
```

(c) Basic Operation: Indicates operating principles of a program.

<Example>

(5) Basic Operation

(a) Upper 8 bits in 16-bit binary are shifted to right. Here LSB is rotated to bit C. Lower 8 bits are then rotated to right. At this time, LSB in bit C is rotated to MSB of lower 8 bits.

(b) IX is used to keep track of number of shifts. IX is decremented each time (a) is executed. (a) is looped until IX is "0".

1.1.3 FLOWCHART Format (Format 3)

FLOWCHART format is represented in Fig. 1.4. It gives a program flowchart. Each item in the format is described using Fig. 1.4.

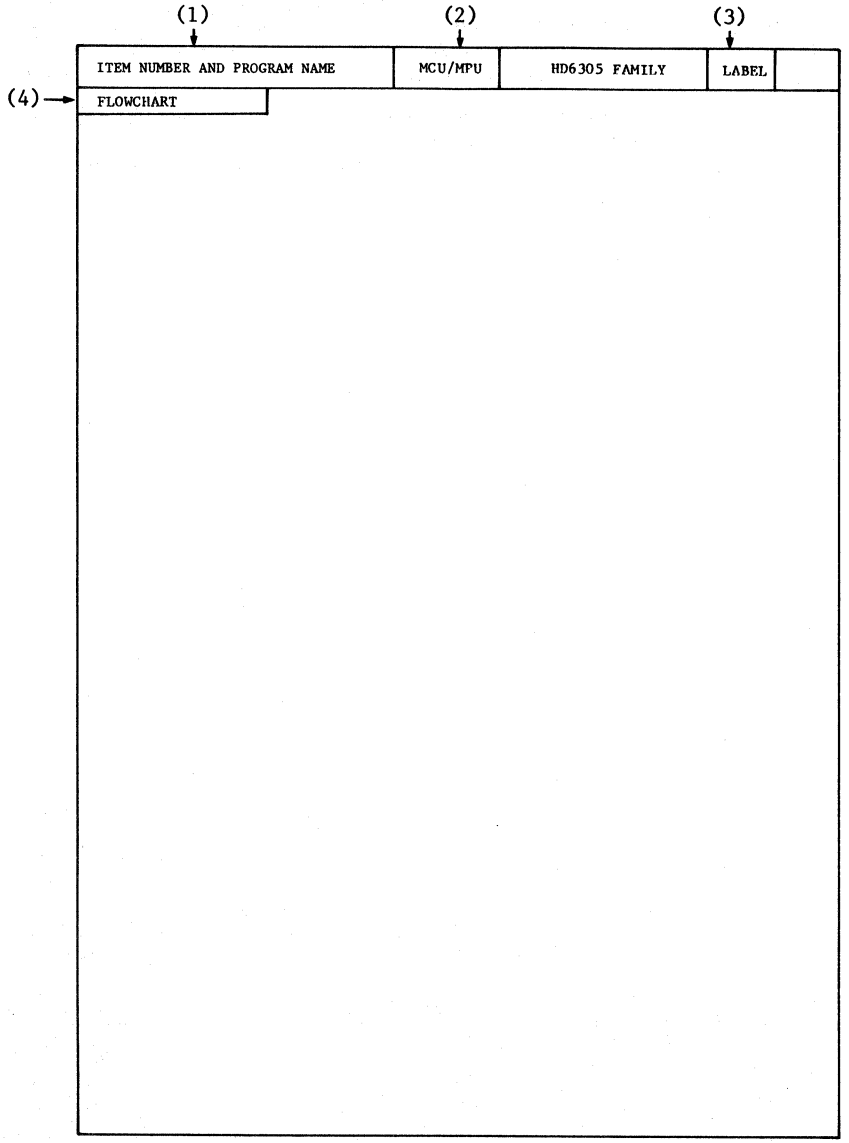


Fig. 1.4 FLOWCHART Format

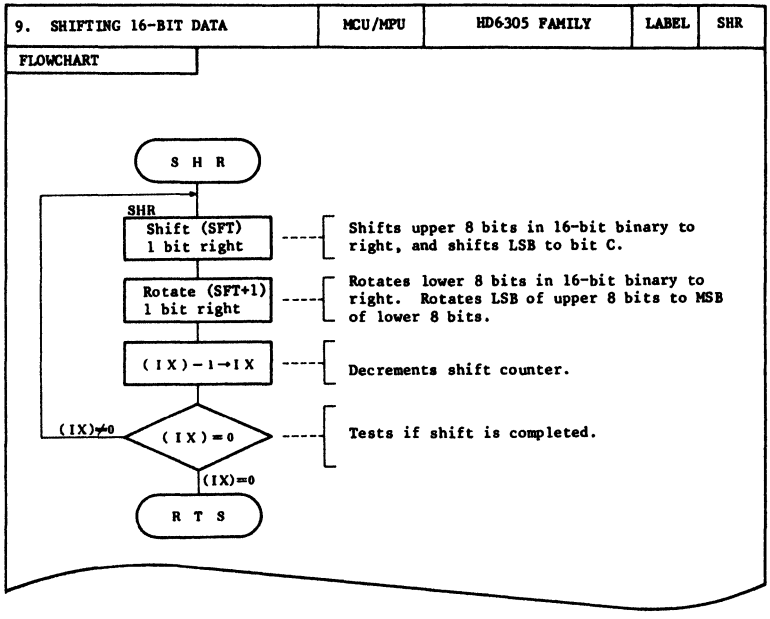
(1) ITEM NUMBER AND PROGRAM NAME
(2) MCU/MPU
(3) LABEL

} Same as SPECIFICATION Format

(4) FLOWCHART :

Comments on the flowchart are described in the column on the right.

<Example>



1.1.4 PROGRAM LISTING Format (Format 4)

PROGRAM LISTING Format is represented in Fig. 1.5. Each item in the format is described using Fig. 1.5

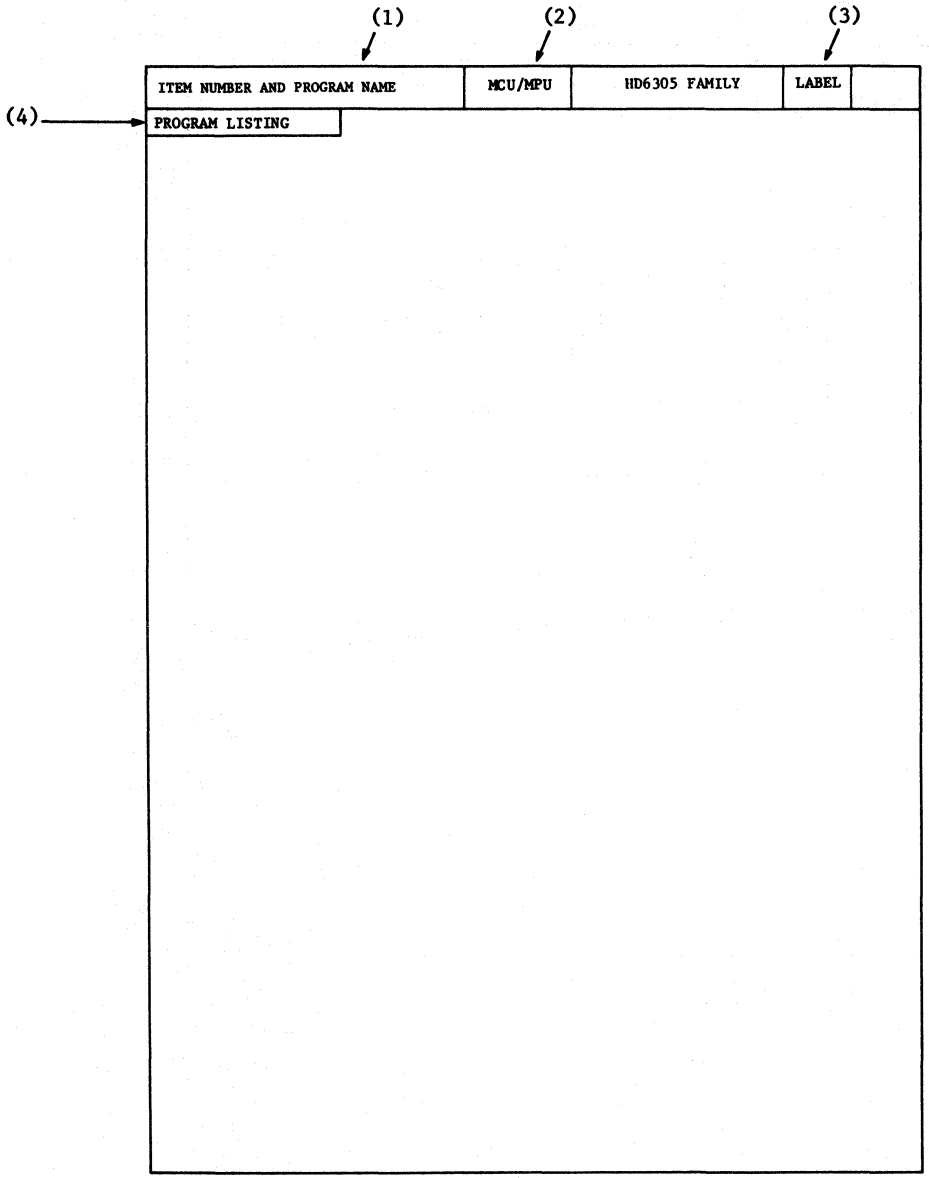


Fig. 1.5 PROGRAM LISTING Format

(1) ITEM NUMBER AND PROGRAM NAME
(2) MCU/MPU
(3) LABEL

} Same as SPECIFICATION Format

(4) PROGRAM LISTING:

<Example>

9. SHIFTING 16-BIT DATA	MCU/MPU	HD6305 FAMILY	LABEL	SHR

PROGRAM LISTING

```
00001          (a)  ********************************************
00002               *                                          *
00003               *     NAME : SHIFTING 16-BIT DATA  (SHR)   *
00004          (b)  *                                          *
00005               ********************************************
00006               *                                          *
00007          (c)  *       ENTRY : SFT  (16-BIT BINARY DATA)  *
00008               *               IX   (SHIFT COUNTER)       *
00009               *     RETURNS : SFT  (16-BIT BINARY DATA)  *
00010               *                                          *
00011               ********************************************
00012               *
00013 0080          SFT   ORG   $80
00014               *
00015 0080 0002     SFT   RMB   2          16-bit binary data
00016               *
00017 1000          (d)   ORG   $1000
00018               *
00019      1000     SHR   EQU   *          Entry point
00020 1000 34 80          LSR   SFT        Shift upper byte to right
00021 1002 36 81          ROR   SFT+1      Rotate lower byte to right
00022 1004 5A             DEC   X          Decrement shift counter
00023 1005 26 F9          BNE   SHR        Loop until shift counter = 0
00024 1007 81             RTS
```

(a) NAME: Name of a program. ( ) means entry point label.

(b) ENTRY: shows storage location and contents of entry arguments.

(c) RETURNS: shows storage location and contents of return arguments.

(d) SHR: shows entry point label.

1.2 How to Execute Programs

Relation between the programs in APPLICATION NOTES and user program is shown in Fig. 1.6. All programs in APPLICATION NOTES are formed as subroutine, they should be proceeded as shown in Fig. 1.6 and (1) to (5) on the next page.

An example of a user program in which a program in APPLICATION NOTES is accessed as a subroutine is shown in Fig. 1.7.

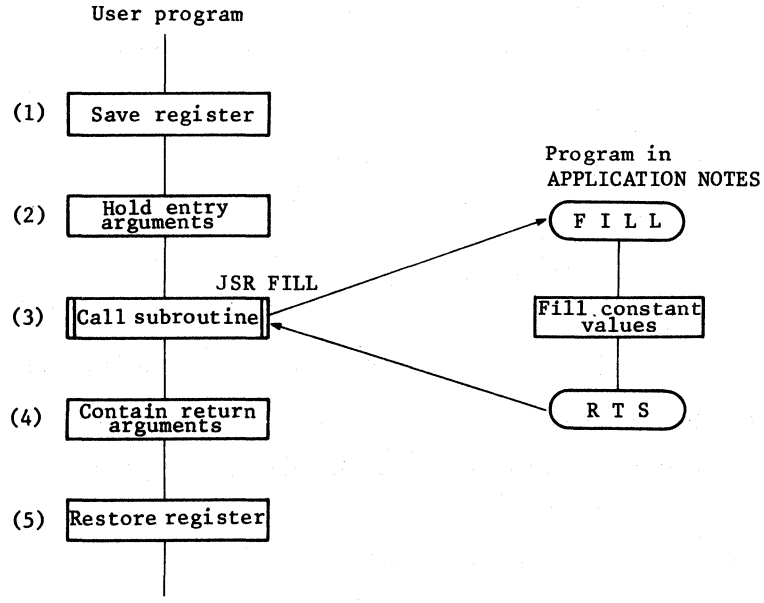


Fig. 1.6 Relation between User Program and Program in APPLICATION NOTES

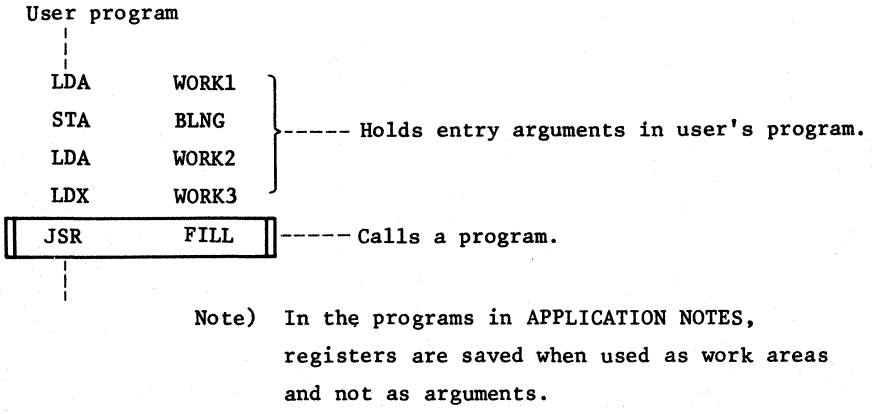


Fig. 1.7 Example Showing How to Execute a Program

(1) Save register contents that will be destroyed by program execution.

CPU registers used in the programs may return to the user program while destroying the contents of the registers. Thus, registers should be saved, if necessary. Refer to the "CHANGES IN CPU REGISTERS AND FLAGG" in SPECIFICATION (Format 1) for the register conditions after a program is executed.

(2) Hold entry arguments

Hold entry arguments to the CPU registers or a particular address in the memory before calling a program in the user program. Refer to "AGRUMENTS" in the SPECIFICATION (Format 1) for entry arguments to be held.

(3) Call subroutine

A program is called.

(4) Contain return arguments

After a program is executed, the result contained in the return arguments must be handled according to the user's purpose. Refer to "ARGUMENTS" in the SPECIFICATION (Format 1) for results.

(5) Restore register

Registers saved in (1) are restored here. When (1) is operated, (5) must also be operated.

Moreover, note that when a program is used as a subroutine, the stack area shown in the "SPECIFICATION" (Refer to (7) in Fig. 1.2) is necessary in addition to that for the subroutine call in the user program. When a subroutine is called, the above stack area must be assured.

1.3 Symbols

Symbols and abbreviations used in APPLICATION NOTES are defined as follows.

(a) Operation

( ) = Contents
(( )) = Index register addressing
→ = Data transfer direction
+ = Addition
− = Subtraction
× = Multiplication
/ = Division
∧ = AND
∨ = OR
⊕ = Exclusive OR
$\overline{\times}$ = NOT

(b) Register symbols in MCU/MPU

ACCA = Accumulator A
CCR = Condition code register
IX = Index register, 8 bits

(c) Contents of bits 0 through 4 of condition code register

C = Carry or borrow	bit 0
Z = Zero	bit 1
N = Negative	bit 2
I = Interrupt mask	bit 3
H = Carry from bit 3 to bit 4	bit 4

(d) Others

= = Equal sign
≠ = Not-equal sign
>, <, ≧, ≦ = Comparison signs
' ' = ASCII inside ' '
$ = Hexadecimal data
: = Labels of sequential addresses

PROGRAM APPLICATION EXAMPLES

PROGRAM APPLICATION TABLE

Item	Program	Label	Page
1	FILLING CONSTANT VALUES	FILL	24
2	MOVING MEMORY BLOCKS	MOVE	29
3	MOVING STRINGS	MOVES	36
4	BRANCHING FROM TABLE	CCASE	42
5	CONVERTING ASCII LOWERCASE INTO UPPERCASE	TPR	49
6	CONVERTING ASCII INTO 1-BYTE HEXADECIMAL	NIBBLE	54
7	CONVERTING 8-BIT BINARY DATA INTO ASCII	COBYTE	59
8	COUNTING NUMBER OF LOGICAL "1" BITS IN 8-BIT DATA	HCNT	64
9	SHIFTING 16-BIT DATA	SHR	69
10	4-DIGIT BCD COUNTER	DECNT	74
11	COMPARING 16-BIT BINARY DATA	CMP	79
12	ADDING 16-BIT BINARY DATA	ADD	85
13	SUBTRACTING 16-BIT BINARY DATA	SUB	91
14	MULTIPLYING 16-BIT BINARY DATA	MUL	97
15	DIVIDING 16-BIT BINARY DATA	DIV	103
16	ADDING 8-DIGIT BCD	ADDD	109
17	SUBTRACTING 8-DIGIT BCD	SUBD	115
18	16-BIT SQUARE ROOT	SQRT	121
19	CONVERTING 2-BYTE HEXADECIMALS INTO 5-DIGIT BCD	HEX	127
20	CONVERTING 5-DIGIT BCD INTO 2-BYTE HEXADECIMALS	BCD	132
21	SORTING	SORT	139

1. FILLING CONSTANT VALUES	MCU/MPU	HD6305 FAMILY	LABEL	FILL

FUNCTION

(a) Stores one-byte constant in RAM using direct page addressing.

(b) Permits RAM and byte length to be freely selected.

(c) Permits easy clearing of RAM.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Constant	ACCA	1
		Byte Length	BLNG (RAM)	1
		Start Address	IX	1
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
●	×

C	Z
●	×
N	I
×	●
H	
●	

SPECIFICATIONS

ROM (Bytes)
7
RAM (Bytes)
1
Stack (Bytes)
0
No. of cycles
229
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

ACCA: Holds one-byte constant in RAM using direct page addressing.

BLNG(RAM): Holds byte length of constant.

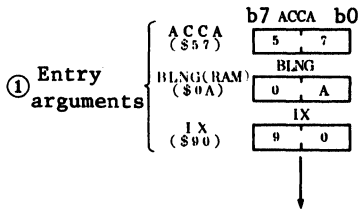
IX : Holds start address of RAM using direct page addressing.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to write constant in 16-byte RAM.

DESCRIPTION

(b) Fig. 1 shows example of FILL execution.
If entry arguments are as shown in part ① of Fig. 1, $57 in ACCA is stored in RAM as shown in part ② of Fig. 1.



(2) User Notes

(a) As BLNG (RAM) is only one byte in length, its data must be between $01 and $FF ($01≦BLNG≦$FF).

(b) Do not set BLNG (RAM) to "0", or FILL will not execute.

(c) BLNG (RAM) is located in internal RAM. Be careful not to overwrite BLNG (RAM) with FILL.

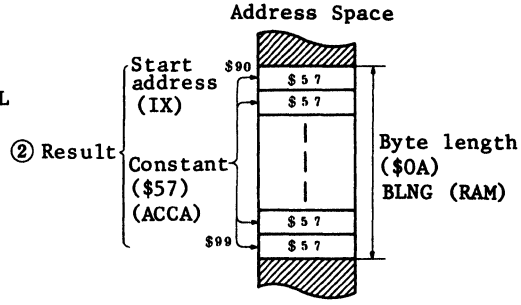


Fig. 1 Example of FILL execution

(3) RAM Description

Label	RAM	Description
BLNG	b7 b0	Byte length is stored in RAM with one-byte constant "BLNG".

1. FILLING CONSTANT VALUES	MCU/MPU	HD6305 FAMILY	LABEL	FILL

DESCRIPTION

(4) Sample Application

FILL subroutine is called after source start address, destination address and length of RAM to be filled are held.

```
WORK1    RMB    1        ----- Reserves memory byte for byte length.

WORK2    RMB    1        ----- Reserves memory byte for constant.

WORK3    RMB    1        ----- Reserves memory byte for start address.
          .
          .
          .
          .
         LDA    WORK1  }
         STA    BLNG   }  -----  Stores byte length into entry argument
                                  (BLNG).
         LDA    WORK2    ----- Loads constant into entry argument (ACCA).
         LDX    WORK3    ----- Loads start address into entry argument
                                  (IX).
         JSR    FILL     ----- Calls FILL subroutine.
```

(5) Basic Operation

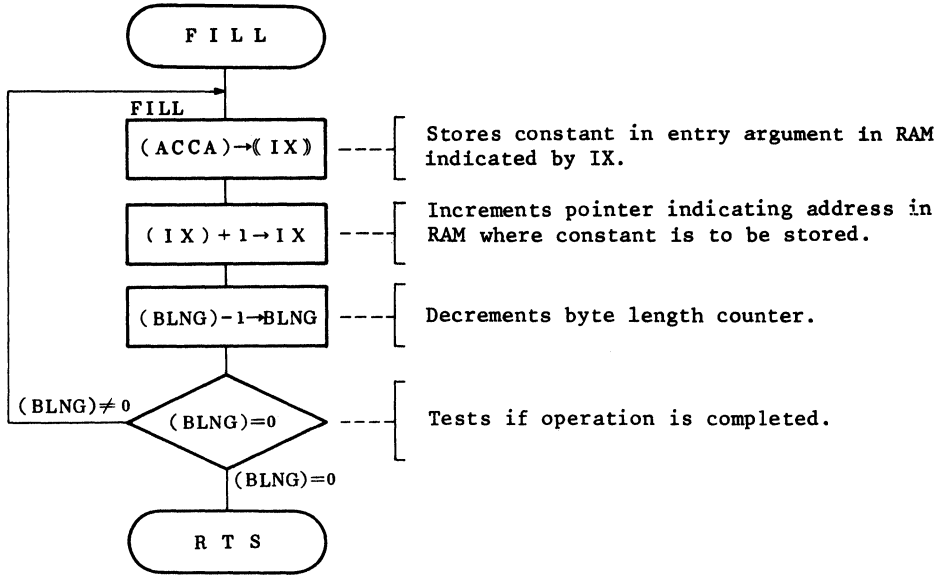
(a) IX is used to indicate address in RAM where constant is stored.

(b) Constant in ACCA in index addressing mode is stored in RAM in order.

(c) BLNG (RAM) is used to indicate byte length of constant. It is decremented each time constant is stored, until BLNG (RAM) is "0".

1. FILLING CONSTANT VALUES	MCU/MPU	HD6305 FAMILY	LABEL	FILL

FLOWCHART



1. FILLING CONSTANT VALUES	MCU/MPU	HD6305 FAMILY	LABEL	FILL

PROGRAM LISTING

```
00001                   ********************************************
00002                   *                                          *
00003                   *     NAME : FILLING CONSTANT VALUE (FILL)     *
00004                   *                                          *
00005                   ********************************************
00006                   *                                          *
00007                   *         ENTRY : ACCA    (CONSTANT)           *
00008                   *                 BLNG    (BYTE COUNTER)       *
00009                   *                 IX      (START ADDR)         *
00010                   *       RETURNS : NOTHING                  *
00011                   *                                          *
00012                   ********************************************
00013                   *
00014 0080                      ORG     $80
00015                   *
00016 0080 0001         BLNG    RMB     1         Byte counter
00017                   *
00018 1000                      ORG     $1000
00019                   *
00020       1000        FILL    EQU     *         Entry point
00021 1000 F7                   STA     0,X       Store constant
00022 1001 5C                   INC     X         Increment ADDR
00023 1002 3A 80                DEC     BLNG      Decrement byte counter
00024 1004 26 FA                BNE     FILL      Loop until byte counter = 0
00025 1006 81                   RTS
```

2. MOVING MEMORY BLOCKS	MCU/MPU	HD6305 FAMILY	LABEL	MOVE

FUNCTION

(a) Moves data block in memory to RAM using direct page addressing.

(b) Permits data block length and source and destination addresses to be freely selected in the memory.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Source start address	SOA (RAM)	2
		Destination start address	DEA (RAM)	1
		Byte length	MCNT (RAM)	1
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
×	×

C	Z
●	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
34
RAM (Bytes)
9
Stack (Bytes)
2
No. of cycles
720
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

SOA (RAM) : Holds source start address in 2-byte hexadecimals.

DEA (RAM) : Holds destination start address in 1-byte hexadecimal.

MCNT (RAM): Holds length of data block to be moved in 1-byte hexadecimal.

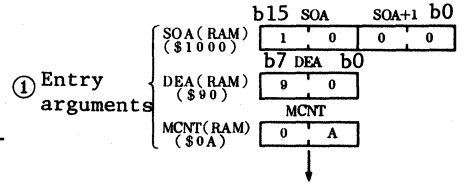
SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed for 16-byte data move.

2. MOVING MEMORY BLOCKS	MCU/MPU	HD6305 FAMILY	LABEL	MOVE

DESCRIPTION

(b) Fig. 1 shows example of MOVE execution.
If entry arguments are as shown in part ① of Fig. 1, data in source ($1000 - $1009) is moved to destination ($90 - $99) as shown in part ② of Fig. 1.



(2) User Notes

(a) As MCNT (RAM) is only one byte in length, its data must be between $01 and $FF ($01≦MCNT≦$FF).

(b) Do not hold MCNT (RAM) to "0", or MOVE will not execute.

(c) MOVE is located in internal RAM. Do not move data block to RAM where MOVE is stored, or execution can not be stopped.

(d) Hold entry arguments so that source area (Fig. 2 Ⓐ) and destination area (Fig. 2 Ⓒ) do not overlap. If they do, the source data in overlapping area (Fig. 2 Ⓑ) will be destroyed.

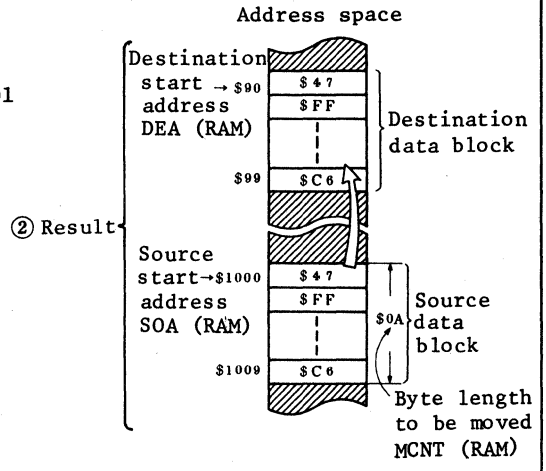


Fig. 1 Example of MOVE execution

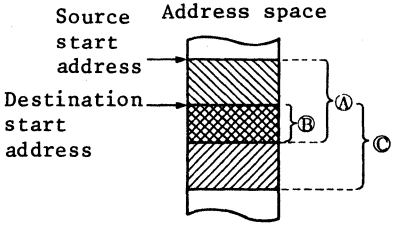
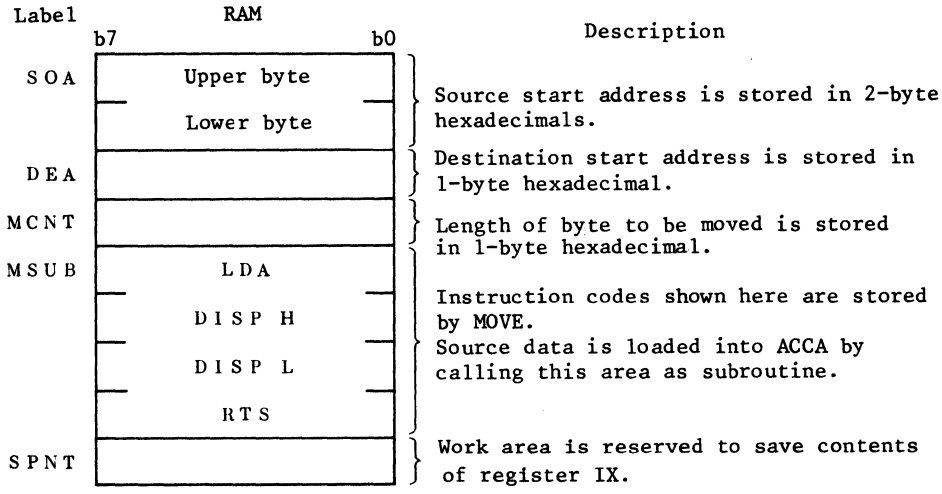


Fig. 2 Example of overlapping source area with destination area

DESCRIPTION

(3) RAM Description



2. MOVING MEMORY BLOCKS	MCU/MPU	HD6305 FAMILY	LABEL	MOVE

DESCRIPTION

(4) Sample Application

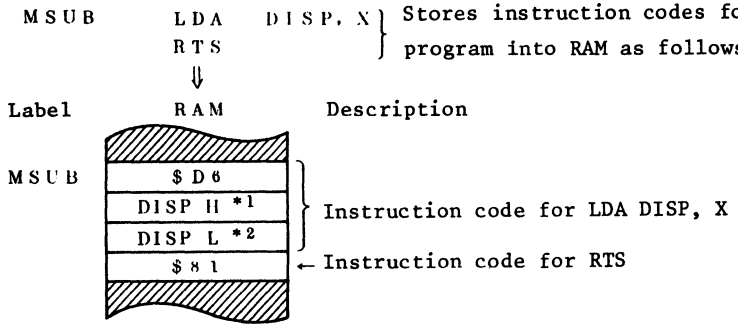
MOVE subroutine is called after source start address, destination start address and length of byte to be moved are held.

```
WORK1   RMB   2        ----- Reserves memory byte for source start
                             address.
WORK2   RMB   1        ----- Reserves memory byte for destination
                             start address.
WORK3   RMB   1        ----- Reserves memory byte for loading byte
                             length to be moved.
WORK4   RMB   2        ----- Reserves memory byte for register
                             contents saving.
        .
        .
        .
        .
        STA   WORK4    }
        STX   WORK4+1  }---- Saves register contents that will be
                             destroyed by executing MOVE.
        LDA   WORK1    }
        STA   SOA      }
        LDA   WORK1+1  }---- Stores source start address into entry
        STA   SOA+1    }     argument (SOA).
        LDA   WORK2    }
        STA   DEA      }---- Stores destination start address into
                             entry argument (DEA).
        LDA   WORK3    }
        STA   MCNT     }---- Stores length of byte to be moved into
                             entry argument (MCNT).
        JSR   MOVE     ----- Calls MOVE subroutine.
        LDA   WORK4    }
        LDX   WORK4+1  }---- Restores register.
        :
        :
        :
```

DESCRIPTION

(5) Basic Operation

(a) In the HD6305 family, IX is one byte in length. However, index addressing can be performed with source start address of 2 bytes by calling the subroutine shown in Fig. 3.



(Notes) *1: Store the upper byte of the start address in DISP H.
*2: Store the lower byte of the start address in DISP L.

Fig. 3 Details of instruction in RAM

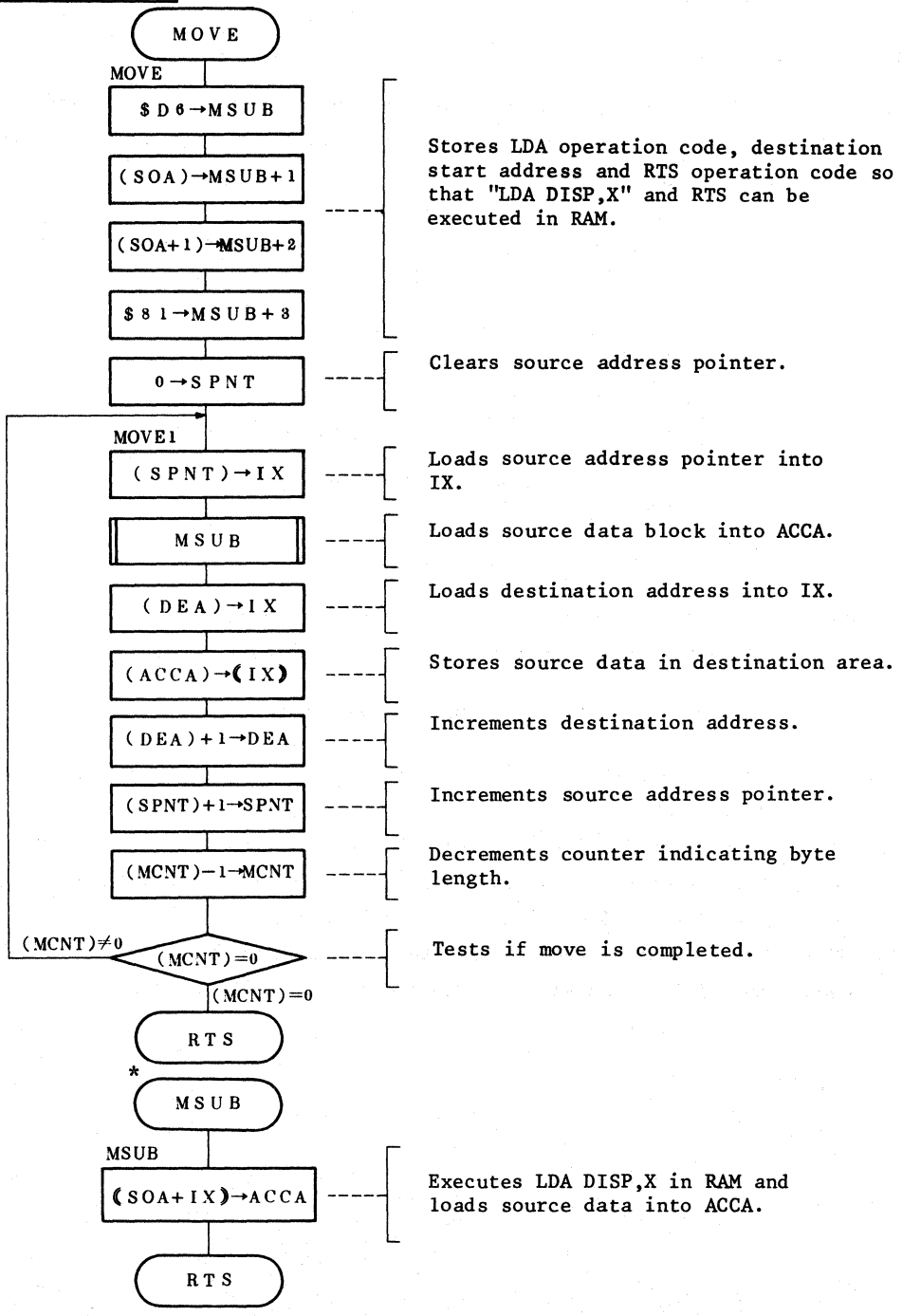
(b) IX is used to indicate source and destination addresses, which are alternately loaded into IX.

(c) Source data is loaded into ACCA by calling the subroutine shown in Fig. 3. Then, IX is saved, destination address is loaded into IX, and data in ACCA is moved to destination using index addressing mode.

(d) MCNT (RAM) is used to indicate length of byte to be moved. It is decrimented each time (c) is executed. (c) is looped until MCNT is "0".

2. MOVING MEMORY BLOCKS	MCU/MPU	HD6305 FAMILY	LABEL	MOVE

FLOWCHART



(Note) * MSUB does not appear in the PROGRAM LISTING because it is stored in RAM.

2. MOVING MEMORY BLOCKS	MCU/MPU	HD6305 FAMILY	LABEL	MOVE

PROGRAM LISTING

```
00001                   ************************************************
00002                   *                                              *
00003                   *   NAME : MOVING MEMORY BLOCKS (MOVE)         *
00004                   *                                              *
00005                   ************************************************
00006                   *                                              *
00007                   *        ENTRY : SOA   (SOURCE ADDR)           *
00008                   *                DEA   (DESTINATION ADDR)      *
00009                   *                MCNT  (TRANSFER COUNTER)      *
00010                   *      RETURNS : NOTHING                       *
00011                   *                                              *
00012                   ************************************************
00013                   *
00014 0080              ORG     $80
00015                   *
00016 0080 0002 SOA     RMB     2       Source ADDR
00017 0082 0001 DEA     RMB     1       Destination ADDR
00018 0083 0001 MCNT    RMB     1       Transfer counter
00019 0084 0004 MSUB    RMB     4       Work area for subroutine
00020 0088 0001 SPNT    RMB     1       Relative data of source ADDR
00021                   *
00022 1000              ORG     $1000
00023                   *
00024      1000 MOVE    EQU     *       Entry point
00025 1000 A6 D6        LDA     #$D6    Store instruction code (LDA Disp.X)
00026 1002 B7 84        STA     MSUB
00027 1004 B6 80        LDA     SOA     Store source ADDR (H)
00028 1006 B7 85        STA     MSUB+1
00029 1008 B6 81        LDA     SOA+1   Store source ADDR (L)
00030 100A B7 86        STA     MSUB+2
00031 100C A6 81        LDA     #$81    Store instruction code (RTS)
00032 100E B7 87        STA     MSUB+3
00033 1010 3F 88        CLR     SPNT    Clear relative data of source ADDR
00034 1012 BE 88 MOVE1  LDX     SPNT    Load relative data of source ADDR
00035 1014 BD 84        JSR     MSUB    Load transfer data
00036 1016 BE 82        LDX     DEA     Load destination ADDR
00037 1018 F7           STA     0,X     Store transfer data
00038 1019 3C 82        INC     DEA     Increment destination ADDR
00039 101B 3C 88        INC     SPNT    Increment relative data of source ADDR
00040 101D 3A 83        DEC     MCNT    Decrement transfer counter
00041 101F 26 F1        BNE     MOVE1   Loop until transfer counter = 0
00042 1021 81           RTS
```

6

3. MOVING STRINGS	MCU/MPU	HD6305 FAMILY	LABEL	MOVES

FUNCTION

(a) Moves data block in memory to RAM using direct page addressing.

(b) Terminates moving process when terminator $00 is found in data table.

(c) Permits source and destination addresses to be freely selected in the memory.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Source start address	SOAS (RAM)	2
		Destination start address	DEAS (RAM)	1
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
×	×

C	Z
●	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
34
RAM (Bytes)
8
Stack (Bytes)
2
No. of cycles
668
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

SOAS (RAM) : Holds source start address in 2-byte hexadecimals.

DEAS (RAM) : Holds destination start address in 1-byte hexadecimal.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to put terminator at the 16th byte.

DESCRIPTION

(b) Fig. 1 shows example of MOVES execution.

If entry arguments are as shown in part ① of Fig. 1, data in source ($1000) is moved to destination ($90) as shown in part ② of Fig. 1. When it loads terminator $00, MCU terminates moving process.

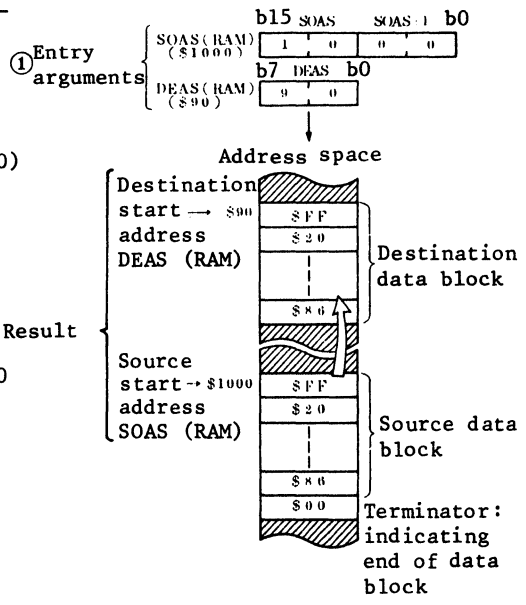


Fig. 1 Example of MOVES execution

(2) User Notes

(a) Source data must not contain any $00 function other than terminator.

(b) MOVES is located in internal RAM. Do not move data block to RAM where MOVES is held, or execution can not be stopped.

(c) Hold entry arguments so that source area (Fig. 2 Ⓐ) and destination area (Fig. 2 Ⓒ) do not overlap. If they do, the source data in overlapping area (Fig. 2 Ⓑ) will be destroyed.

(d) Amount of data block that can be moved is MCU address space using direct page addressing minus 8-byte RAM used by MOVES.

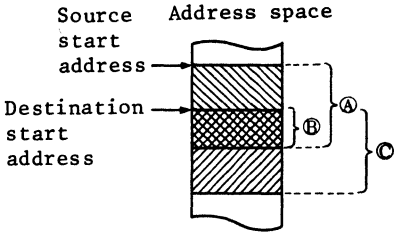


Fig. 2 Example of overlapping source area with destination area

6

DESCRIPTION

(3) RAM Description

Label	b7 RAM b0	Description
SOAS	Upper byte Lower byte	Source start address is stored in 2-byte hexadecimals.
DEAS		Destination start address is stored in 1-byte hexadecimal.
MSSUB	LDA DISP H DISP L RTS	Instruction codes shown here are stored by MOVES. Source data is loaded into ACCA by calling this area as subroutine.
SOPNT		Work area is reserved to save contents of register IX.

(4) Sample Application

MOVES subroutine is called after source start address and destination start address are held.

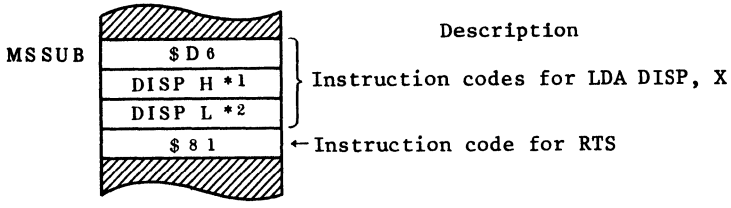
```
WORK1   RMB   2        ----- Reserves memory byte for source start
                             address.
WORK2   RMB   1        ----- Reserves memory byte for destination
                             start address.
WORK3   RMB   2        ----- Reserves memory byte for register contents
                             saving.
          .
          .
          .
          .
        STA   WORK3    }
        STX   WORK3+1  }----- Saves register contents that will be
                              destroyed by executing MOVES.
        LDA   WORK1    }
        STA   SOAS     }
        LDA   WORK1+1  }----- Stores source start address in entry
        STA   SOAS+1   }      argument (SOAS).
        LDA   WORK2    }
        STA   DEAS     }----- Stores destination start address in
                              entry argument (SOAS).
        JSR   MOVES    ----- Calls MOVES subroutine.
        LDA   WORK3    }
        LDX   WORK3+1  }----- Restores register.
          .
          .
          .
```

DESCRIPTION

(5) Basic Operation

(a) In the HD6305 family, IX is one byte in length. However, index addressing can be performed with source start address of 2 bytes by calling the subroutine shown in Fig. 3.

```
MSSUB   LDA   DISP,X  } Store instruction codes for
        RTS           } the program in RAM as follows.
          ⇓
Label     RAM
```



(Notes) *1: Store the upper byte of the start address in DISP H.
*2: Store the lower byte of the start address in DISP L.

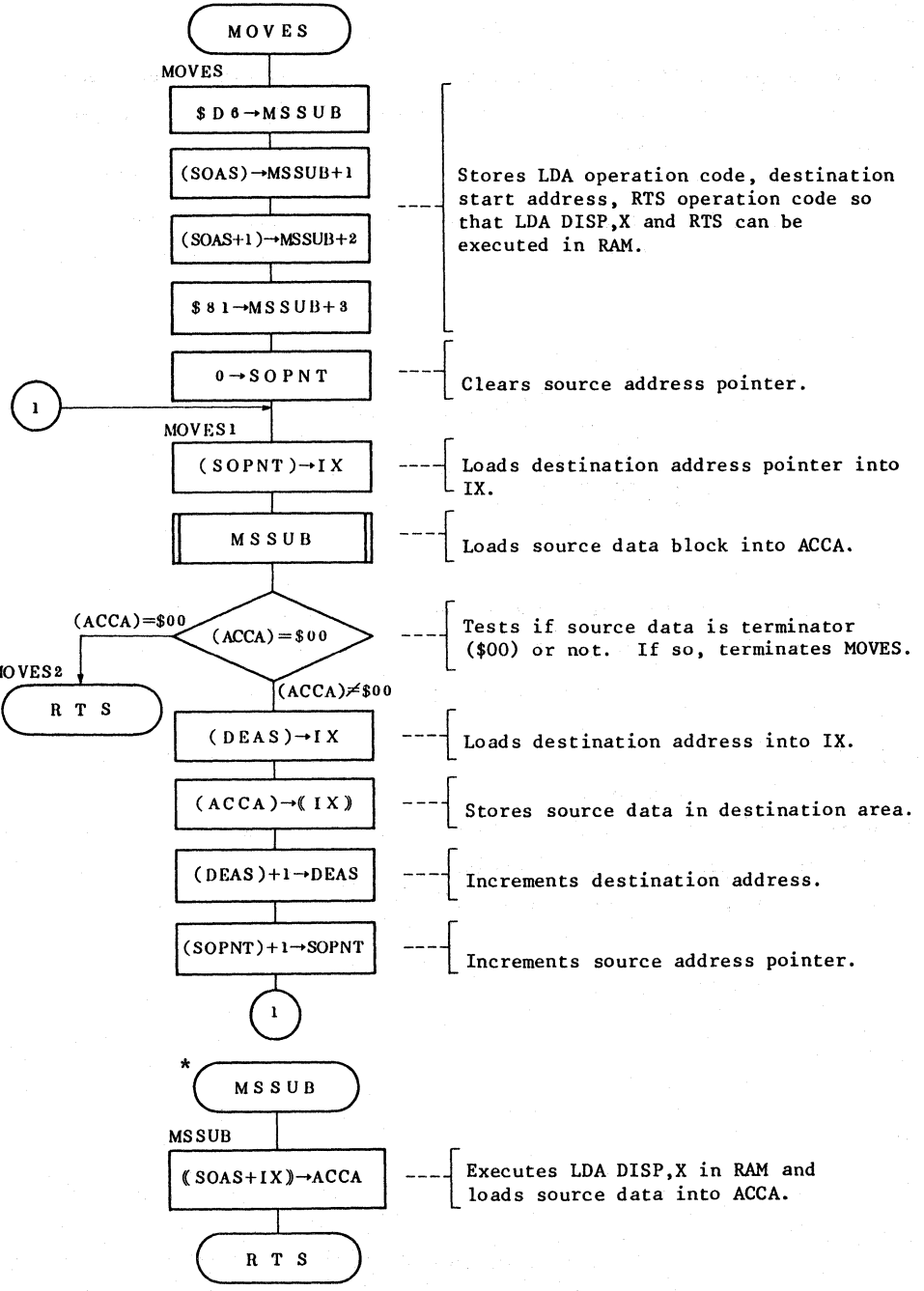
Fig. 3 Details of instruction in RAM

(b) IX is used to indicate source and destination addresses, which are alternately loaded into IX.

(c) Source data is loaded into ACCA by calling the subroutine shown in Fig. 3. ACCA data is tested if it is terminator. If so, MOVES is terminated. If not, moving process continues until the terminator is found.

3. MOVING STRINGS	MCU/MPU	HD6305 FAMILY	LABEL	MOVES

FLOWCHART



(Note) * MSSUB does not appear in the PROGRAM LISTING because it is stored in RAM.

3. MOVING STRINGS	MCU/MPU	HD6305 FAMILY	LABEL	MOVES

PROGRAM LISTING

```
00001                    ***********************************************
00002                    *                                             *
00003                    *      NAME : MOVING STRINGS (MOVES)          *
00004                    *                                             *
00005                    ***********************************************
00006                    *                                             *
00007                    *            ENTRY : SOAS (SOURCE ADDR)       *
00008                    *                    DEAS (DESTINATION ADDR)  *
00009                    *          RETURNS : NOTHING                  *
00010                    *                                             *
00011                    ***********************************************
00012                    *
00013 0080               *       ORG    $80
00014                    *
00015 0080 0002     SOAS    RMB    2         Source ADDR.
00016 0082 0001     DEAS    RMB    1         Destination ADDR.
00017 0083 0004     MSSUB   RMB    4         Work area for subroutine
00018 0087 0001     SOPNT   RMB    1         Relative data of source ADDR
00019                    *
00020 1000               *       ORG    $1000
00021                    *
00022      1000     MOVES   EQU    *         Entry point
00023 1000 A6 D6            LDA    #$D6      Store instruction code (LDA Disp.X)
00024 1002 B7 83            STA    MSSUB
00025 1004 B6 80            LDA    SOAS      Store source ADDR (H)
00026 1006 B7 84            STA    MSSUB+1
00027 1008 B6 81            LDA    SOAS+1    Store source ADDR (L)
00028 100A B7 85            STA    MSSUB+2
00029 100C A6 81            LDA    #$81      Store instruction code (RTS)
00030 100E B7 86            STA    MSSUB+3
00031 1010 3F 87            CLR    SOPNT     Clear relative data of source ADDR
00032 1012 BE 87    MOVES1  LDX    SOPNT     Load relative data of source ADDR
00033 1014 BD 83            JSR    MSSUB     Load transfer data
00034 1016 27 09            BEQ    MOVES2    Branch if transfer data = 0
00035 1018 BE 82            LDX    DEAS      Load destination ADDR
00036 101A F7               STA    0,X       Store transfer data
00037 101B 3C 82            INC    DEAS      Increment destination ADDR
00038 101D 3C 87            INC    SOPNT     Increment relative data of source ADDR
00039 101F 20 F1            BRA    MOVES1    Branch MOVES
00040 1021 81       MOVES2  RTS
```

6

4. BRANCHING FROM TABLE	MCU/MPU	HD6305 FAMILY	LABEL	CCASE

FUNCTION

(a) Stores service routine start address in RAM corresponding to the 1-byte command in RAM.

(b) Permits easy decoding and processing of keyboard and other data inputs.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Command	CMMD (RAM)	1
		Data table start address	TBTOP (RAM)	2
	Returns	Program start address	TBTOP (RAM)	2
		Command existance	Bit C (CCR)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
●	×

C	Z
↕	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
43
RAM (Bytes)
7
Stack (Bytes)
2
No. of cycles
169
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

CMMD (RAM): Holds command such as ASCII.

TBTOP(RAM): Holds data table address in which the command corresponding to that in CMMD (RAM) and command service routine start address have been stored in 3-byte units. After CCASE execution, it contains the service routine start address in 2-byte hexadecimals corresponding to the command in CMMD(RAM).

Bit C : Indicates CCASE termination.

Bit C=1 : Data in data table is the same as that in CMMD(RAM).

Bit C=0 : Data in data table differs from that in CMMD(RAM).

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to find data at the end of 3 data units.

DESCRIPTION

(b) Fig. 1 shows example of CCASE execution. If entry arguments are as shown in part ① of Fig. 1, CCASE locates start address of command service routine in data table (Fig. 2) and stores it in TBTOP (RAM) as shown in part ② of Fig. 1.

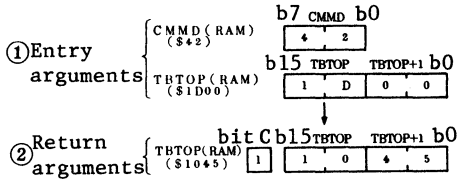


Fig. 1 Example of CCASE execution

(c) Data table shown in Fig. 2 must be set up before executing CCASE. It contains 3-byte data units beginning at $1D00 and terminator indicating the end of the table. The first byte of the 3-byte data units is command. The second and third bytes contain upper and lower bytes of command service routine start address respectively.

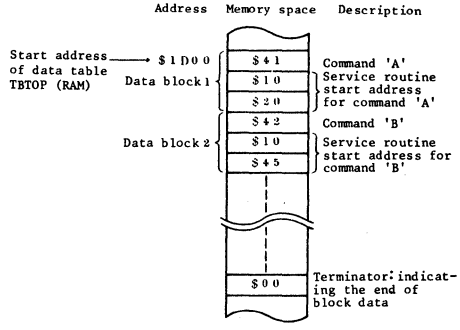


Fig. 2 Example of data table

(2) User Notes

Do not use $00 as argument (CMMD) or as command in data table.

It functions as terminator only.

(3) RAM Description

Label	RAM	Description
CMMD	b7 b0	1-byte command is stored.
TBTOP	Upper byte	Data table start address is stored in 2-byte hexadecimals before execution.
	Lower byte	Command service routine start address is stored in 2-byte hexadecimals after execution.
CSUB	LDA	Instruction codes shown here are stored by CCASE. Data in data table are loaded into ACCA by calling this area as subroutine.
	DISP H	
	DISP L	
	RTS	

6

4. BRANCHING FROM TABLE	MCU/MPU	HD6305 FAMILY	LABEL	CCASE

DESCRIPTION

(4) Sample Application

CCASE subroutine is called after command and start address of data table are held.

```
WORK1   RMB     1           ----- Reserves memory byte for command.

WORK2   RMB     1           ----- Reserves momory byte for register contents
          .                       saving.
          .
          .
          .
        STX     WORK2       ----- Saves register contents that will be
                                  destroyed by executing CCASE.
        LDA     #$1D
        STA     TBTOP       -----
        LDA     #$00                Stores data table start address into
        STA     TBTOP+1             entry argument (TBTOP).
        LDA     WORK1       -----
        STA     CMMD                Stores command into entry argument (CMMD).
        JSR     CCASE       ----- Calls CCASE subroutine.
        LDX     WORK2       ----- Restores register.
        BCC     ERROR       ----- Tests if there is data corresponding to
                                  inputted command in data table.
                *
    Program branching to
    command service
    routine
          :
          :
ERROR   ERROR program       ----- Executes error program because there is
                                  no data corresponding to inputted
                                  command in data table.
          .
          .
          .
        ORG     $1D00       ----- Start address of data table.
        FCC     'A'         ----- Command 'A'
        FDB     $1020       ----- Start address of command service routine
                                  in case of command 'A'.
        FCC     'B'         ----- Command 'B'
        FDB     $1045       ----- Start address of command service routine
          :                       in case of command 'B'.
          .
        FCB     $00         ----- Indicates the end of data block.
```

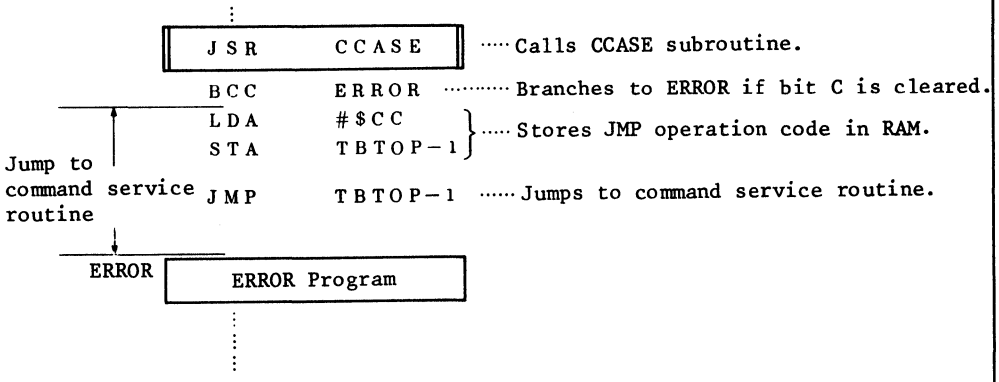
4. BRANCHING FROM TABLE	MCU/MPU	HD6305 FAMILY	LABEL	CCASE

DESCRIPTION

(Note)

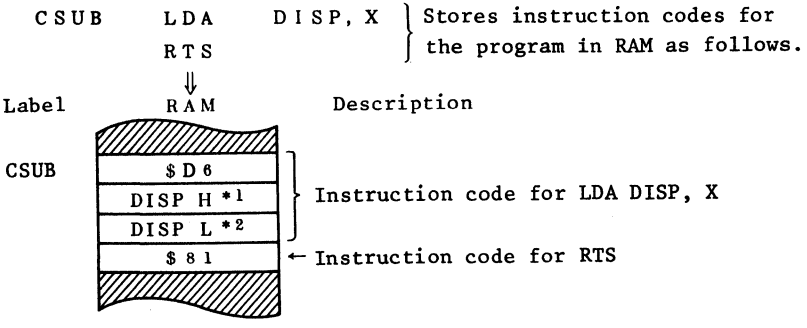
* Example of branching to command service routine after CCASE execution:

CCASE functions only to store start address of command service routine in TBTOP (RAM). Program as in the example below to branch to command service routine. Method used in this example is to store operation codes for JMP in TBTOP-1 (RAM) and jump to command service routine.



(5) Basic Operation

(a) In the HD6305 family, IX is one byte in length. However, index addressing can be performed with data table start address of 2 bytes, by calling the subroutine shown in Fig. 3.



(Notes) *1: Stores the upper byte of the start address in DISP H.

*2: Stores the lower byte of the start address in DISP L.

Fig. 3 Details of instruction in RAM

6

4. BRANCHING FROM TABLE	MCU/MPU	HD6305 FAMILY	LABEL	CCASE

DESCRIPTION

(b) IX is used to indicate data table start address.

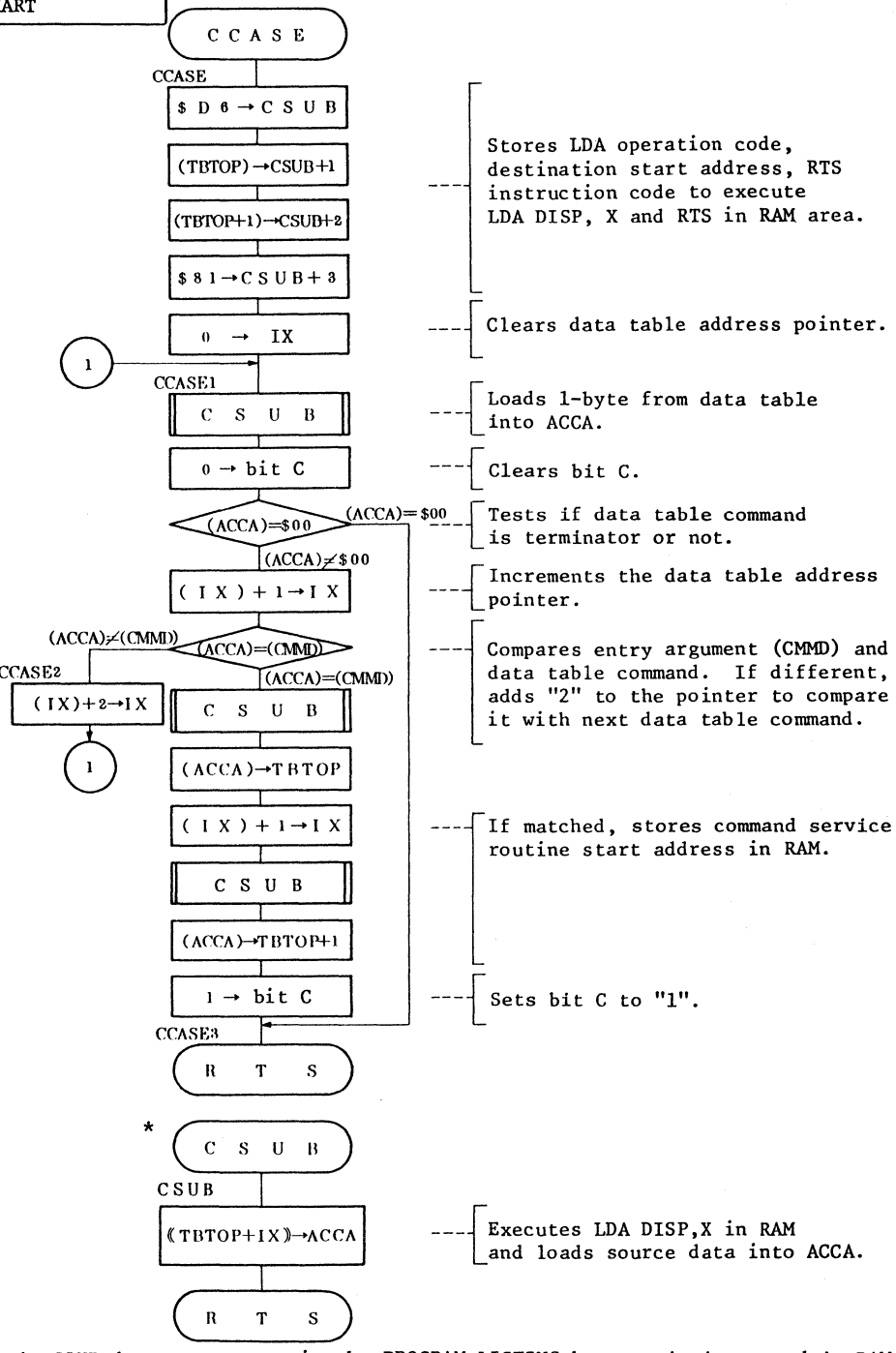
(c) By calling the subroutine shown in Fig. 3, data table commands are read in order from start address and compared with entry argument contents.

(d) If the data table commands match the entry argument contents (CMMD), bit C is stored and CCASE is terminated.

(e) If terminator $00 is found in data table, bit C is cleared and CCASE is terminated.

4. BRANCHING FROM TABLE	MCU/MPU	HD6305 FAMILY	LABEL	CCASE

FLOWCHART



(Note) * CSUB does not appear in the PROGRAM LISTING because it is stored in RAM.

4, BRANCHING FROM TABLE	MCU/MPU	HD6305 FAMILY	LABEL	CCASE

PROGRAM LISTING

```
00001                   ************************************************
00002                   *                                              *
00003                   *         NAME : BRANCHING FROM TABLE (CCASE)  *
00004                   *                                              *
00005                   ************************************************
00006                   *                                              *
00007                   *            ENTRY : CMMD   (COMMAND )         *
00008                   *                    TBTOP  (TABLE ADDR)       *
00009                   *            RETURNS : TBTOP  (MODULE ADDR)    *
00010                   *                      CARRY  (C=1:TRUE,C=0:FALSE) *
00011                   *                                              *
00012                   ************************************************
00013                   *
00014 0080                      ORG     $80
00015                   *
00016 0080 0001         CMMD    RMB     1       Command
00017 0081 0002         TBTOP   RMB     2       Table ADDR -> Module ADDR
00018 0083 0004         CSUB    RMB     4       Work area for subroutine
00019                   *
00020 1000                      ORG     $1000
00021                   *
00022       1000        CCASE   EQU     *       Entry point
00023 1000 A6 D6                LDA     #$D6    Store instruction code (LDA Disp,X)
00024 1002 B7 83                STA     CSUB
00025 1004 B6 81                LDA     TBTOP   Store table ADDR (H)
00026 1006 B7 84                STA     CSUB+1
00027 1008 B6 82                LDA     TBTOP+1 Store table ADDR (L)
00028 100A B7 85                STA     CSUB+2
00029 100C A6 81                LDA     #$81    Store instruction code (RTS)
00030 100E B7 86                STA     CSUB+3
00031 1010 5F                   CLR     X       Clear pointer of table ADDR
00032 1011 BD 83        CCASE1  JSR     CSUB    Load command of table
00033 1013 4D                   TST     A       Command of table = 0 ?
00034 1014 98                   CLC             Clear carry
00035 1015 27 13                BEQ     CCASE3  Branch if ZERO
00036 1017 5C                   INC     X       Increment pointer of table ADDR
00037 1018 B1 80                CMP     CMMD    Command of table = comand ?
00038 101A 27 04                BEQ     CCASE2  Branch if equal
00039 101C 5C                   INC     X       Increment pointer of table ADDR
00040 101D 5C                   INC     X
00041 101E 20 F1                BRA     CCASE1  Branch CCASE1
00042 1020 BD 83        CCASE2  JSR     CSUB    Store ADDR(H) of module ADDR
00043 1022 B7 81                STA     TBTOP
00044 1024 5C                   INC     X       Increment pointer of table ADDR
00045 1025 BD 83                JSR     CSUB    Store ADDR(L) of module ADDR
00046 1027 B7 82                STA     TBTOP+1 Set carry to "1"
00047 1029 99                   SEC
00048 102A 81           CCASE3  RTS
```

5. CONVERTING ASCII LOWERCASE INTO UPPERCASE	MCU/MPU	HD6305 FAMILY	LABEL	TPR

FUNCTION

(a) Converts ASCII lowercase data in ACCA into uppercase and loads result into ACCA.

(b) Utilizes 7-bit ASCII in arguments.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Lowercase (ASCII)	ACCA	1
	Returns	Uppercase (ASCII)	ACCA	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
↕	●

C	Z
×	×
N	I
×	●
H	
●	

SPECIFICATIONS

ROM (Bytes)
11
RAM (Bytes)
0
Stack (Bytes)
0
No. of cycles
17
Reentrant
Yes
Relocation
Yes
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

ACCA: Holds ASCII lowercase data.
After executing TPR, contains the corresponding uppercase data.

(b) Fig. 1 shows example of TPR execution. If entry argument lowercase 'a' ($61) is held in ACCA as shown in part ① of Fig. 1, it is converted into uppercase 'A' ($41), and the result is contained in ACCA as shown in part ② of Fig. 1.

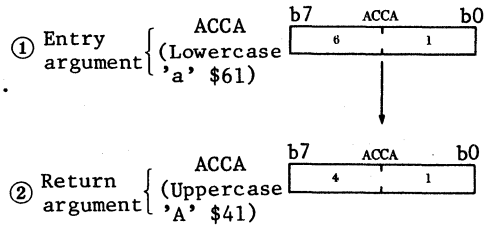


Fig. 1 Example of TPR execution

SPECIFICATIONS NOTES

5. CONVERTING ASCII LOWERCASE INTO UPPERCASE	MCU/MPU	HD6305 FAMILY	LABEL	TPR

DESCRIPTION

(2) User Notes

Lowercase data should be held into ACCA. If ASCII other than lowercase are held into ACCA, these are saved and not converted into uppercase.

(3) RAM Description

RAM is not used in TPR subroutine.

(4) Sample Application

TPR subroutine is called after lowercase data is held into ACCA.

```
WORK1   RMB   1        ----- Reserves memory byte for lowercase.

WORK2   RMB   1        ----- Reserves memory byte for uppercase.

        :

        LDA   WORK1    ----- Loads lowercase data into entry argument
                             (ACCA).

        JSR   TPR      ----- Calls TPR subroutine.

        STA   WORK2    ----- Stores uppercase data (return argument
                             (ACCA)) in RAM.
        :
```

(5) Basic Operation

(a) A compare command (CMP) is used to determine whether entry argument in ACCA is lowercase or not.

(b) Entry argument and $DF are ANDed by AND command, and lowercase is converted into uppercase by clearing the fifth bit of lowercase as shown in Fig. 2.

(c) If entry argument is other than in lowercase, TPR does not execute any operation, and entry argument is saved.

5. CONVERTING ASCII LOWERCASE INTO UPPERCASE	MCU/MPU	HD6305 FAMILY	LABEL	TPR

DESCRIPTION

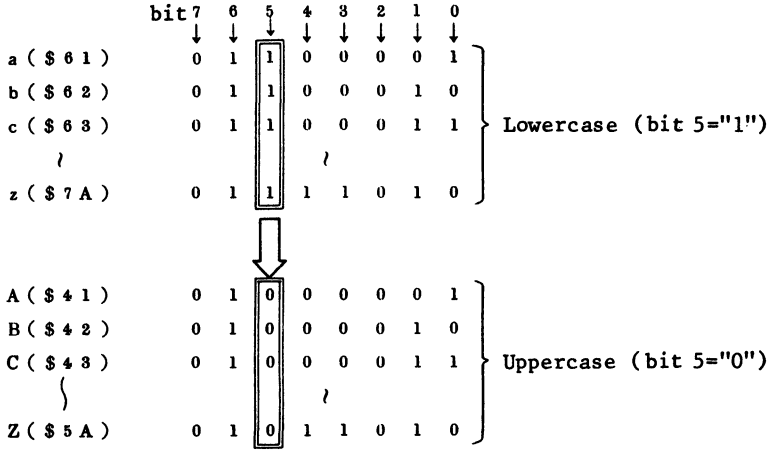
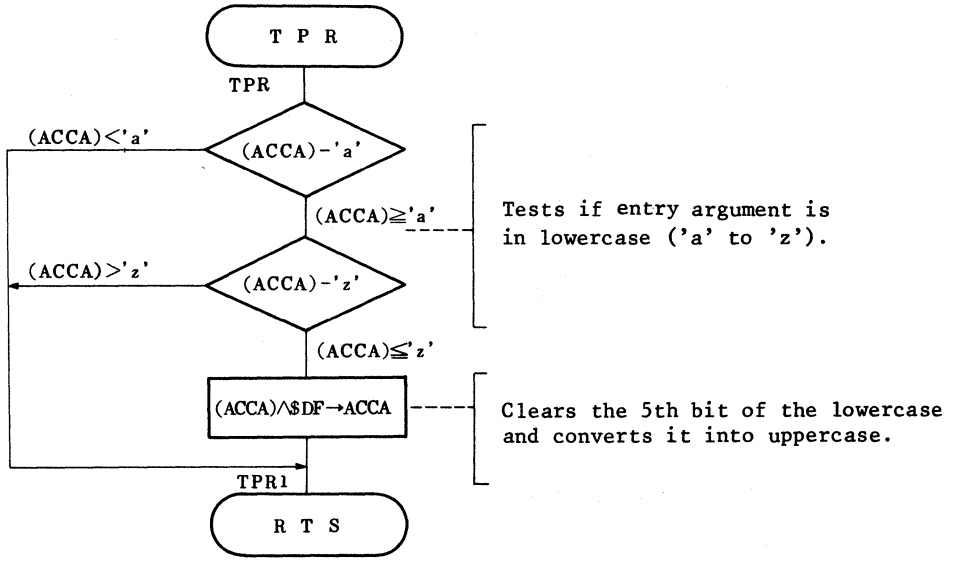


Fig. 2 Lowercase and uppercase of 7-bit ASCII

5. CONVERTING ASCII LOWERCASE INTO UPPERCASE	MCU/MPU	HD6305 FAMILY	LABEL	TPR

FLOWCHART



5. CONVERTING ASCII LOWERCASE INTO UPPERCASE	MCU/MPU	HD6305 FAMILY	LABEL	TPR

PROGRAM LISTING

```
00001                   ***********************************************
00002                   *                                             *
00003                   *   NAME : CONVERTING ASCII LOWERCASE INTO    *
00004                   *            ASCII                  (TPR)     *
00005                   *                                             *
00006                   ***********************************************
00007                   *                                             *
00008                   *            ENTRY : ACCA (ASCII LOWERCASE)   *
00009                   *          RETURNS : ACCA (ASCII UPPERCASE)   *
00010                   *                                             *
00011                   ***********************************************
00012                   *
00013 1000                      ORG     $1000
00014                   *
00015       1000        TPR     EQU     *       Entry point
00016 1000 A1 61                CMP     #'a     ACCA - 'a' ?
00017 1002 25 06                BCS     TPR1    Branch if ACCA < 'a'
00018 1004 A1 7A                CMP     #'z     ACCA - 'z' ?
00019 1006 22 02                BHI     TPR1    Branch if ACCA > 'z'
00020 1008 A4 DF                AND     #$DF    Convert lowercase into Uppercase
00021 100A 81           TPR1    RTS
```

6. CONVERTING ASCII INTO 1-BYTE HEXADECIMAL	MCU/MPU	HD6305 FAMILY	LABEL	NIBBLE

FUNCTION

(a) Converts ASCII '0' to '9' and 'A' to 'F' in ACCA into 1-byte hexadecimal and loads result into ACCA.

(b) Utilizes 7-bit ASCII in arguments.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	ASCII	ACCA	1
	Returns	1-byte hexadecimal	ACCA	1
		Conversion/not conversion	bit C (CCR)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
↕	●

C	Z
↕	×
N	I
×	●
H	
×	

SPECIFICATIONS

ROM (Bytes)
20
RAM (Bytes)
0
Stack (Bytes)
0
No. of cycles
28
Reentrant
Yes
Relocation
Yes
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

ACCA : Holds ASCII. After NIBBLE execution, contains 1-byte hexadecimal.

Bit C (CCR): Shows state when NIBBLE is executed.

Bit C=1 : Shows ASCII other than '0' to '9' or 'A' to 'F'.

Bit C=0 : Shows ASCII within range of '0' to '9' and 'A' to 'F'.

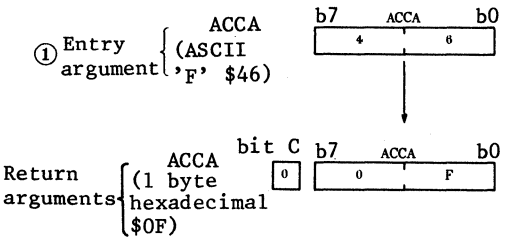


Fig. 1 Example of NIBBLE execution

SPECIFICATIONS NOTES

6. CONVERTING ASCII INTO 1-BYTE HEXADECIMAL	MCU/MPU	HD6305 FAMILY	LABEL	NIBBLE

DESCRIPTION

(b) Fig. 1 shows example of NIBBLE execution. If entry argument is as shown in part ① of Fig. 1, $0F, data converted from ASCII into 1-byte hexadecimal, is held in ACCA as shown in part ② of Fig. 1.

(2) User Notes

If data, other than ASCII '0' to '9' or 'A' to 'F', is stored in ACCA after NIBBLE execution, ACCA data is destroyed.

(3) RAM Description

RAM is not used in NIBBLE subroutine.

(4) Sample Application

NIBBLE subroutine is called after ASCII is held.

```
WORK1   RMB   1        ----- Reserves memory byte for 1-digit ASCII.

WORK2   RMB   1        ----- Reserves memory byte for 1-byte
        :                    hexadecimal.
        :
        :
        LDA   WORK1    ----- Loads ASCII into entry argument (ACCA).

        JSR   NIBBLE   ----- Calls NIBBLE subroutine.

        BCS   SKIP     ----- If ASCII is other than '0' to '9' or 'A'
                             to 'F', branches to service routine for
                             other ASCII.
        STA   WORK2    ----- Stores 1-byte hexadecimal (return
                             argument (ACCA)) in RAM.
        :
        :
        :
        :
SKIP    Service routine for ASCII
        other than
        '0' to '9' or 'A' to 'F'
        :
        :
        :
        :
```

DESCRIPTION

(5) Basic Operation

(a) Bit C resulting from comparison and subtraction of data in ACCA is used to test if the data is within range of '0' to 'F' in the ASCII table (note ▭ blocked area in table).

(b) Addition continues between '0' and '@', ':' to '@' (note ▩ cross hatched area in table) is deleted.

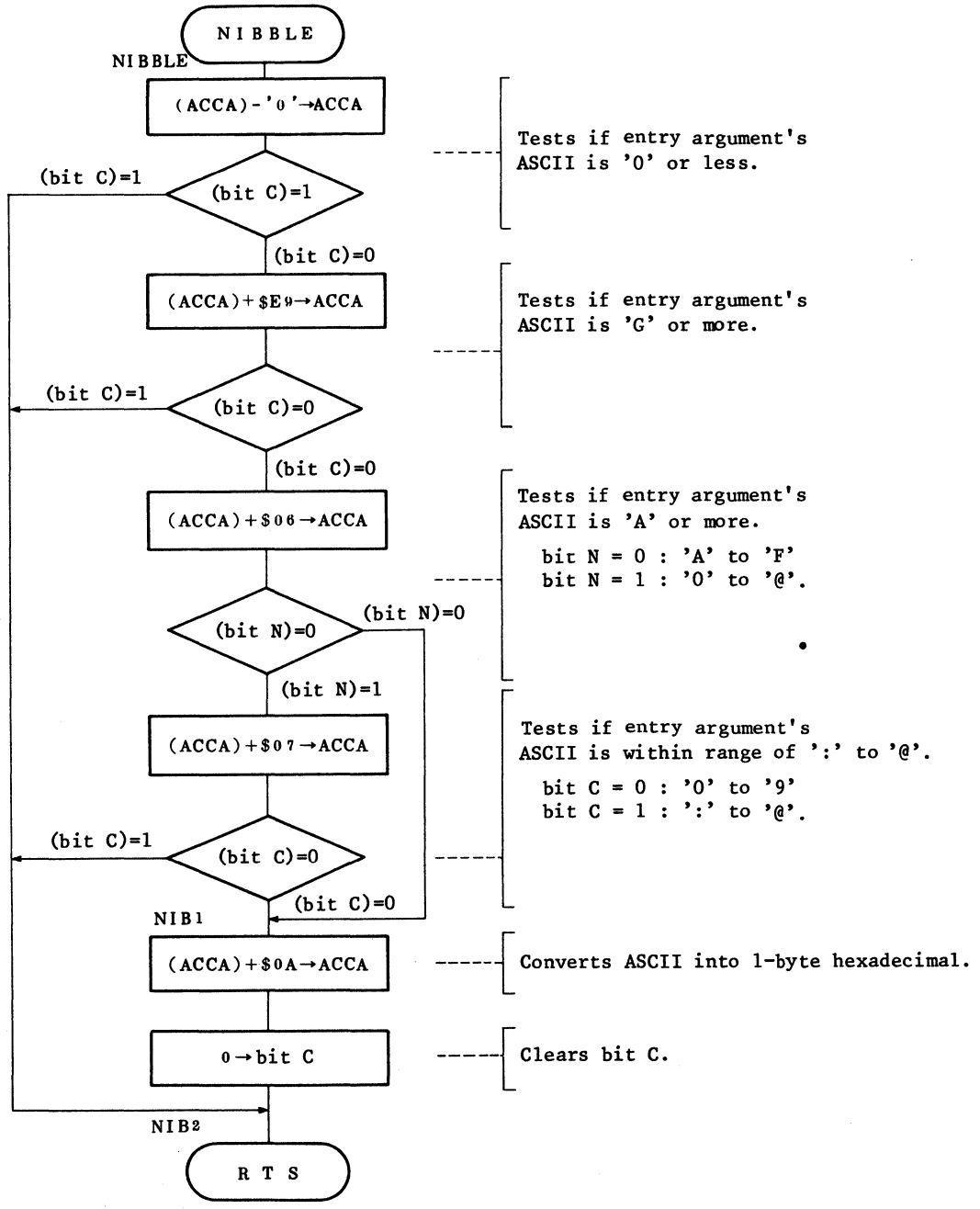
(c) In cases other than between '0' and '9' or 'A' and 'F', bit C is set during (a) or (b) above.

Table 1 ASCII Table

LSD \ MSD	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 0000	NUL	DLE	SP	0	@	P	`	p
1 0001	SOH	DC1	!	1	A	Q	a	q
2 0010	STX	DC2	"	2	B	R	b	r
3 0011	ETX	DC3	#	3	C	S	c	s
4 0100	EOT	DC4	$	4	D	T	d	t
5 0101	ENG	NAK	%	5	E	U	e	u
6 0110	ACK	SYN	&	6	F	V	f	v
7 0111	BEL	ETB	'	7	G	W	g	w
8 1000	BS	CAN	(	8	H	X	h	x
9 1001	HT	EM	)	9	I	Y	i	y
A 1010	LF	SUB	*	:	J	Z	j	z
B 1011	VT	ESC	+	;	K	[	k	{
C 1100	FF	FS	,	<	L	\	l	|
D 1101	CR	GS	-	=	M	]	m	}
E 1110	SO	RS	.	>	N	↑	n	~
F 1111	SI	VS	/	?	O	←	o	DEL

6. CONVERTING ASCII INTO 1-BYTE HEXADECIMAL	MCU/MPU	HD6305 FAMILY	LABEL	NIBBLE

FLOWCHART



6

6. CONVERTING ASCII INTO 1-BYTE HEXADECIMAL	MCU/MPU	HD6305 FAMILY	LABEL	NIBBLE

PROGRAM LISTING

```
00001                   ************************************************
00002                   *                                              *
00003                   *       NAME : CONVERTING ASCII                *
00004                   *              INTO 1-BYTE HEXADECIMAL (NIBBLE)  *
00005                   *                                              *
00006                   ************************************************
00007                   *                                              *
00008                   *            ENTRY : ACCA  (ASCII)             *
00009                   *          RETURNS : ACCA  (BINARY DATA)       *
00010                   *                    CARRY (C=0:TRUE,C=1:FALSE) *
00011                   *                                              *
00012                   ************************************************
00013                   *
00014 1000                        ORG     $1000
00015                   *
00016       1000        NIBBLE EQU     *        Entry point
00017 1000 A0 30                   SUB     #'0      ACCA (ASCII code) - '0' ?
00018 1002 25 0F                   BCS     NIB2     Branch if ACCA<'0'
00019 1004 AB E9                   ADD     #$E9     ACCA - 'G' -> ACCA
00020 1006 25 0B                   BCS     NIB2     Branch if ACCA>='G'
00021 1008 AB 06                   ADD     #6       Test '0'-'@' or 'A'-'F'
00022 100A 2A 04                   BPL     NIB1     Branch if ACCA = 'A'-'F'
00023 100C AB 07                   ADD     #7       Test '0'-'9' or ':'-'@'
00024 100E 25 03                   BCS     NIB2     Branch if ACCA = ':'-'@'
00025 1010 AB 0A        NIB1       ADD     #$A      Convert ASCII into binary dat
00026 1012 98                      CLC              Clear carry
00027 1013 81           NIB2       RTS
```

7. CONVERTING 8-BIT BINARY DATA INTO ASCII	MCU/MPU	HD6305 FAMILY	LABEL	COBYTE

FUNCTION

(a) Converts 8-bit binary data in ACCA into two ASCII characters and stores result in RAM.

(b) Utilizes 7-bit ASCII in arguments.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	8-bit binary data	ACCA	1
	Returns	2-digit ASCII	COB (RAM)	2

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
×	×

C	Z
×	×
N	I
×	●
H	
×	

SPECIFICATIONS

ROM (Bytes)
26
RAM (Bytes)
2
Stack (Bytes)
2
No. of cycles
63
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

ACCA : Holds 8-bit binary data to be converted into ASCII.

COB (RAM) : Holds data converted, from upper and lower 4 bits of 8-bit binary data into 2-digit ASCII, is set.

(b) Fig. 1 shows example of COBYTE execution. If entry argument is as shown in part ① of Fig. 1, data converted from 8-bit binary data into ASCII is contained to COB (RAM) as shown in part ② of Fig. 1.

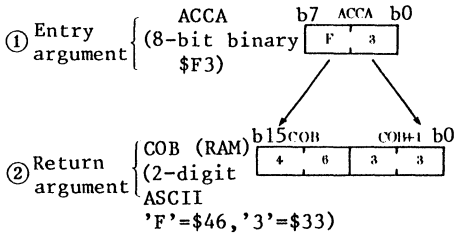


Fig. 1 Example of COBYTE execution

SPECIFICATIONS NOTES

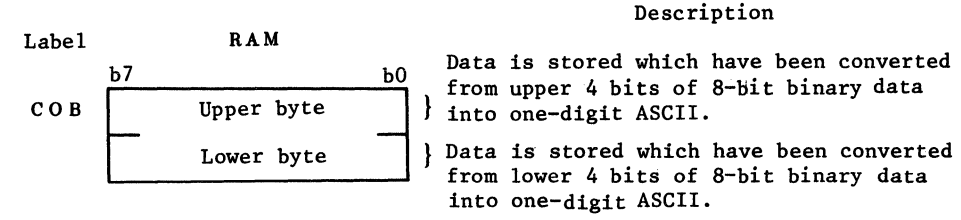
"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to convert 8-bit binary data into ASCII.

DESCRIPTION

(2) User Notes

If 8-bit binary data in ACCA needs to be retained after COBYTE execution, it should be saved in memory before execution.

(3) RAM Description



(4) Sample Application

COBYTE subroutine is called after 8-bit binary data is held.

```
WORK1   RMB   1          ----- Reserves memory byte for 8-bit binary
                                data.
WORK2   RMB   2          ----- Reserves memory byte for 2-digit ASCII.

WORK3   RMB   1          ----- Reserves memory byte for register contents
        :                      saving.
        STX   WORK3      ----- Saves register contents that will be
                                destroyed by COBYTE execution.
        LDA   WORK1      ----- Loads 8-bit binary data into entry
                                argument.
        JSR   COBYTE     ----- Calls COBYTE subroutine.
        LDA   COB
        STA   WORK2      -----
        LDA   COB+1             Stores 2-digit ASCII (return argument
        STA   WORK2+1           (COB)) in RAM.
        LDX   WORK3      ----- Restores register.
        .
        .
        .
```

DESCRIPTION

(5) Basic Operation

(a) 8-bit binary data in ACCA is divided into 4 upper and 4 lower bits.

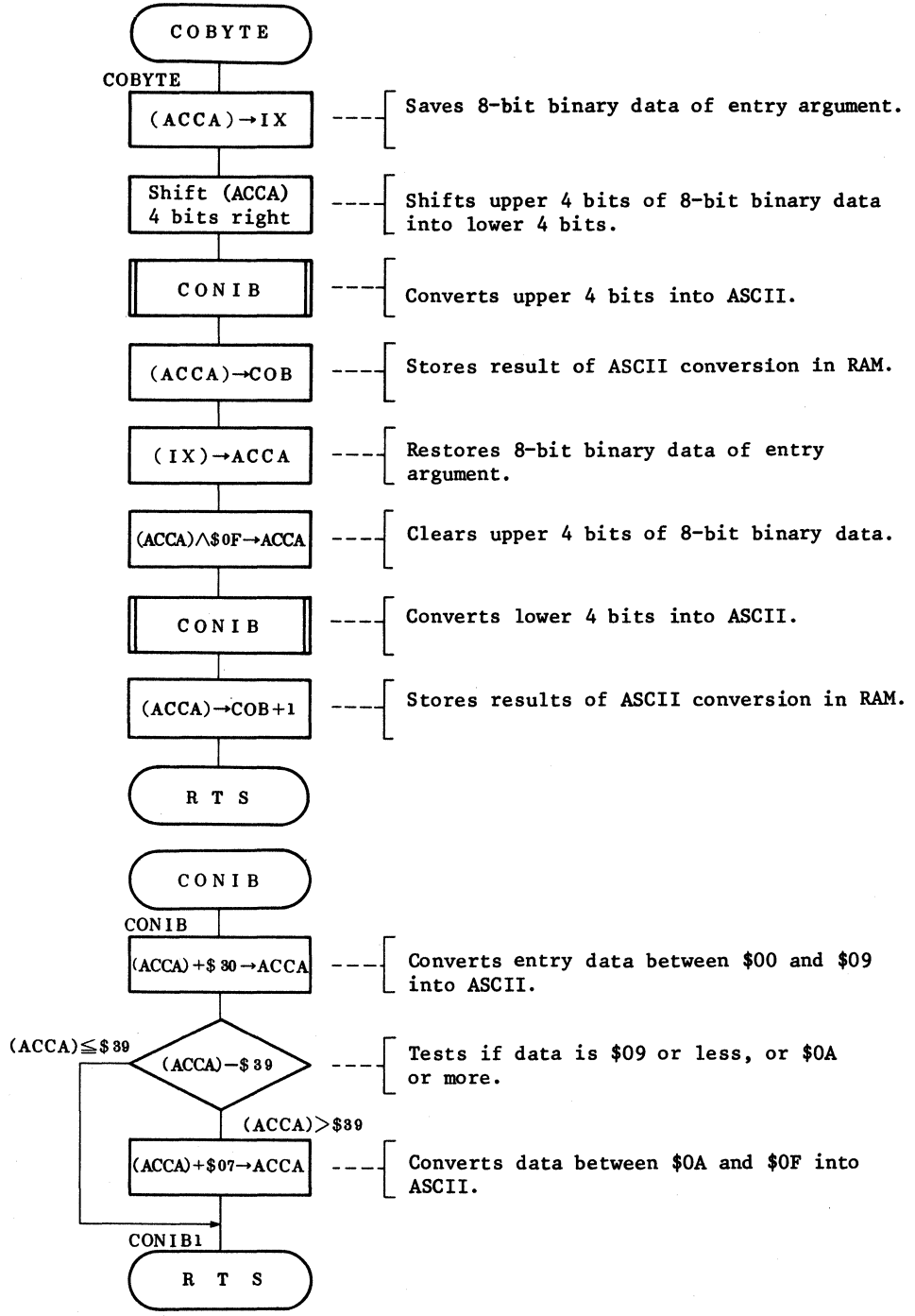
(b) Divided data is then checked by a comparison (instruction CMP). If data is between $00 and $09 (: blocked area in ASCII table shown as Table 1), $30 is added. If data is between $0A and $0F (in Table 1), $37 is added. Result is converted into ASCII.

Table 1 ASCII Table

MSD / LSD	0 000	1 001	2 010	3 011	4 100	5 101	6 110	7 111
0 0000	NUL	DLE	SP	0	@	P	`	p
1 0001	SOH	DC1	!	1	A	Q	a	q
2 0010	STX	DC2	"	2	B	R	b	r
3 0011	ETX	DC3	#	3	C	S	c	s
4 0100	EOT	DC4	$	4	D	T	d	t
5 0101	ENQ	NAK	%	5	E	U	e	u
6 0110	ACK	SYN	&	6	F	V	f	v
7 0111	BEL	ETB	'	7	G	W	g	w
8 1000	BS	CAN	(	8	H	X	h	x
9 1001	HT	EM	)	9	I	Y	i	y
A 1010	LF	SUB	*	:	J	Z	j	z
B 1011	VT	ESC	+	;	K	[	k	{
C 1100	FF	FS	,	<	L	\	l	|
D 1101	CR	GS	-	=	M	]	m	}
E 1110	SO	RS	.	>	N	↑	n	~
F 1111	SI	US	/	?	O	←	o	DEL

7. CONVERTING 8-BIT BINARY DATA INTO ASCII	MCU/MPU	HD6305 FAMILY	LABEL	COBYTE

FLOWCHART



7. CONVERTING 8-BIT BINARY DATA INTO ASCII	MCU/MPU	HD6305 FAMILY	LABEL	COBYTE

PROGRAM LISTING

```
00001                   ***********************************************
00002                   *                                             *
00003                   *     NAME : CONVERTING 8-BIT BINARY DATA     *
00004                   *            INTO ASCII        (COBYTE)       *
00005                   *                                             *
00006                   ***********************************************
00007                   *                                             *
00008                   *        ENTRY : ACCA (8-BIT BINARY DATA)     *
00009                   *      RETURNS : COB  (2-BYTE ASCII)          *
00010                   *                                             *
00011                   ***********************************************
00012                   *
00013 0080                      ORG     $80
00014                   *
00015 0080 0002         COB     RMB     2           2-byte ASCII
00016                   *
00017 1000                      ORG     $1000
00018                   *
00019       1000        COBYTE  EQU     *           Entry point
00020 1000 97                   TAX                 Transfer 8-bit binary data
00021 1001 44                   LSR     A           Shift upper 4 bits to lower 4 bits
00022 1002 44                   LSR     A
00023 1003 44                   LSR     A
00024 1004 44                   LSR     A
00025 1005 AD 0A                BSR     CONIB       Convert upper 4 bits into ASCII
00026 1007 B7 80                STA     COB         Store ASCII
00027 1009 9F                   TXA                 Transfer 8 bit binary data
00028 100A A4 0F                AND     #$0F        Mask upper 4 bits
00029 100C AD 03                BSR     CONIB       Convert lower 4 bits into ASCII
00030 100E B7 81                STA     COB+1       Store ASCII
00031 1010 81                   RTS
00032 1011 AB 30        CONIB   ADD     #'0         Convert into ASCII ('0'-'9')
00033 1013 A1 39                CMP     #'9         ASCII = '0'-'9' or 'A'-'F' ?
00034 1015 23 02                BLS     CONIB1      Branch if ACCA = '0'-'9'
00035 1017 AB 07                ADD     #$07        Convert into ASCII ('A'-'F')
00036 1019 81           CONIB1  RTS
```

6

8. COUNTING NUMBER OF LOGICAL "1" BITS IN 8-BIT DATA	MCU/MPU	HD6305 FAMILY	LABEL	HCNT

FUNCTION

(a) Counts number of logical "1" bits in 8-bit data string in ACCA, and stores result in RAM.

(b) Permits easy parity checking.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	8-bit data	ACCA	1
	Returns	Number of logical "1" bits	HBIT (RAM)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
●	×

C	Z
●	×
N	I
×	●
H	
●	

SPECIFICATIONS

ROM (Bytes)
14
RAM (Bytes)
1
Stack (Bytes)
0
No. of cycles
134
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

ACCA : Holds 8-bit data in which number of logical "1" bits is counted.

HBIT (RAM) : Holds number of logical "1" bit in 8-bit data.

(b) Fig. 1 shows example of HCNT execution. If entry argument is as shown in part ① of Fig. 1, number of logical "1" bits in 8-bit data string is contained in HBIT (RAM) as shown in part ② of Fig. 1.

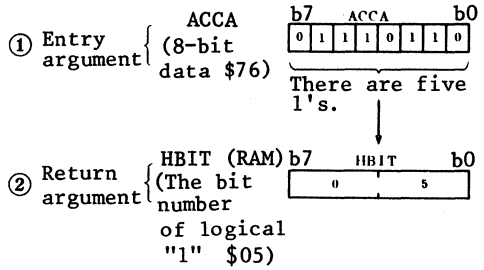


Fig. 1 Example of HCNT execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to count number of logical "1" bits in 8-bit data string.

8. COUNTING NUMBER OF LOGICAL "1" BITS IN 8-BIT DATA	MCU/MPU	HD6305 FAMILY	LABEL	HCNT

DESCRIPTION

(c) Contents of ACCA are saved after HCNT execution.

(2) User Notes

When counting number of logical "0" bits, take 1's complement of ACCA before HCNT execution.

(3) RAM Description

Label	RAM	Description
HBIT	b7 … b0	Number of logical "1" bits is stored in 1-byte hexadecimal.

(4) Sample Application

Subroutine HCNT is called after 8-bit data is held.

```
WORK1   RMB   1          ----- Reserves memory byte for 8-bit data.

WORK2   RMB   1          ----- Reserves memory byte for number of
                               logical "1" bits in 8-bit data string.

WORK3   RMB   1          ----- Reserves memory byte for register contents
        :                      saving.
        STX   WORK3      ----- Saves register contents that will be
                               destroyed by HCNT execution.
        LDA   WORK1      ----- Loads 8-bit data into entry argument
                               (ACCA).
        JSR   HCNT       ----- Calls HCNT subroutine.
        LDA   HBIT       }
        STA   WORK2      }----- Stores number of logical "1" bits (return
                               argument (HBIT)) in RAM.
        LDX   WORK3      ----- Restores register.
        :
```

6

8. COUNTING NUMBER OF LOGICAL "1" BITS IN 8-BIT DATA	MCU/MPU	HD6305 FAMILY	LABEL	HCNT

DESCRIPTION

(5) Basic Operation

(a) IX is used to indicate number of 8-bit data rotations.

(b) Using rotate (instruction ROL), data in ACCA is loaded into bit C one by one.

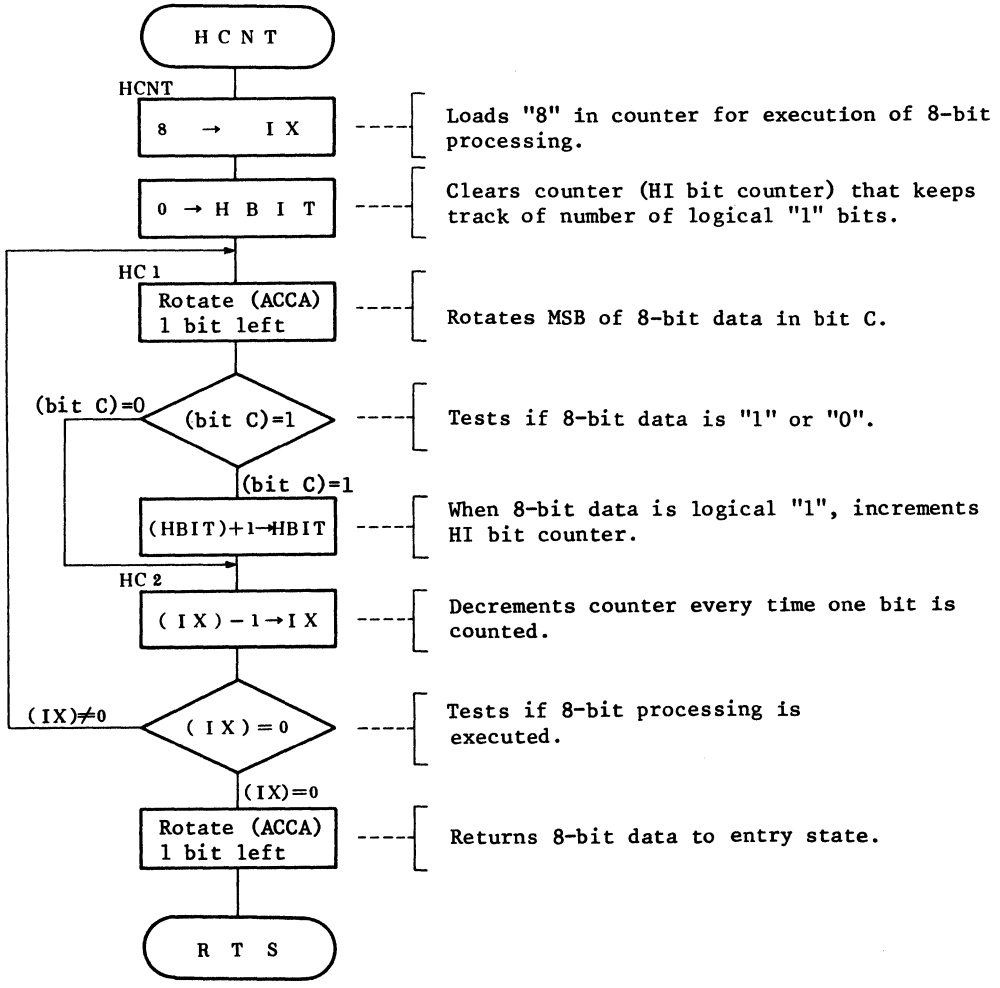
(c) Bit C is checked. If "1", HBIT (RAM) is incremented; if "0", no operation applied.

(d) IX is decremented each time (b) and (c) are executed. (b) and (c) are looped until IX is "0".

Number of logical "1" bits in 8-bit data is then determined.

8. COUNTING NUMBER OF LOGICAL "1" BITS IN 8-BIT DATA	MCU/MPU	HD6305 FAMILY	LABEL	HCNT

FLOWCHART



8. COUNTING NUMBER OF LOGICAL "1" BITS IN 8-BIT DATA	MCU/MPU	HD6305 FAMILY	LABEL	HCNT

PROGRAM LISTING

```
00001                   ***********************************************
00002                   *                                             *
00003                   *       NAME : COUNTING NUMBER OF LOGICAL "1"  *
00004                   *              BITS IN 8-BIT DATA   (HCNT)     *
00005                   *                                             *
00006                   ***********************************************
00007                   *                                             *
00008                   *            ENTRY : ACCA (8-BIT DATA)        *
00009                   *          RETURNS : HBIT (HIGH BIT COUNTER)  *
00010                   *                                             *
00011                   ***********************************************
00012                   *
00013 0080                      ORG     $80
00014                   *
00015 0080 0001         HBIT    RMB     1         High bit counter
00016                   *
00017 1000                      ORG     $1000
00018                   *
00019       1000        HCNT    EQU     *         Entry point
00020 1000 AE 08                LDX     #8        Set rotate counter
00021 1002 3F 80                CLR     HBIT      Clear high bit counter
00022 1004 49           HCNT1   ROL     A         Rotate 8-bit data left
00023 1005 24 02                BCC     HCNT2     Branch if carry=0
00024 1007 3C 80                INC     HBIT      Increment high bit counter
00025 1009 5A           HCNT2   DEC     X         Decrement rotate counter
00026 100A 26 F8                BNE     HCNT1     Loop until rotate counter=0
00027 100C 49                   ROL     A         Replace 8-bit data
00028 100D 81                   RTS
```

9. SHIFTING 16-BIT DATA	MCU/MPU	HD6305 FAMILY	LABEL	SHR

FUNCTION

(a) Shifts 16-bit binary data in RAM to right.

(b) Permits number of shifts to be freely determined.

(c) Permits easy multiplication of 16-bit binary data by 2^{-n} (n : number of shifts).

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	16-bit binary data to be shifted to right	SFT (RAM)	2
		Number of shifts	IX	1
	Returns	Shift results	SFT (RAM)	2

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
●	×

C	Z
×	×
N	I
×	●
H	
●	

SPECIFICATIONS

ROM (Bytes)
8
RAM (Bytes)
2
Stack (Bytes)
0
No. of cycles
110
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

SFT (RAM) : Holds 16-bit binary data to be shifted to right. After SHR execution, contains shift result.

IX : Holds number of 16-bit binary data to be shifted to right.

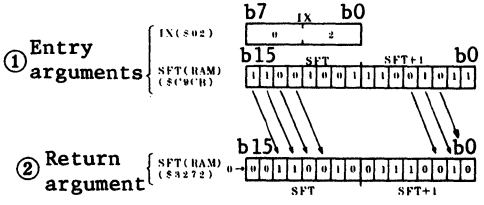


Fig. 1 Example of SHR execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to shift 16-bit binary data to right by 7 bits.

9. SHIFTING 16-BIT DATA	MCU/MPU	HD6305 FAMILY	LABEL	SHR

DESCRIPTION

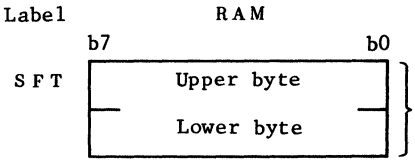
(b) Fig. 1 shows example of SHR execution. If entry arguments are held as shown in part ① of Fig. 1, 16-bit binary data is shifted to right as shown in part ② of Fig. 1.

(2) User Notes

Be sure to hold data into IX within range of $01 ≦ IX ≦ $0F.

When data outside this range is held, SFT (RAM) becomes "0".

(3) RAM Description



(4) Sample Application

SHR subroutine is called after number of shifts and 16-bit binary data to be shifted to right are held.

```
WORK1   RMB   2        ------Reserves memory byte for 16-bit binary data.
WORK2   RMB   1        ----- Reserves memory byte for number of shifts.
WORK3   RMB   2        ----- Reserves memory byte for 16-bit binary
                             shift results.
WORK4   RMB   1        ----- Reserves memory byte for register contents
                             saving.
          :
          :
        STX   WORK4    ----- Saves register contents that will be
                             destroyed by executing SHR.
        LDA   WORK1
        STA   SFT      -----
        LDA   WORK1+1        Stores 16-bit binary data to be shifted
        STA   SFT+1          into right in entry argument (SFT).
        LDX   WORK2    ----- Loads number of shifts into entry argument
                             (IX).
        JSR   SHR      ----- Calls SHR subroutine.
        LDA   SFT
        STA   WORK3
        LDA   SFT+1    -----
        STA   WORK3+1        Stores shift results regarding 16-bit
                             binary (return argument (SFT)) in RAM.
        LDX   WORK4    ----- Restores register.
          :
          :
```

9. SHIFTING 16-BIT DATA	MCU/MPU	HD6305 FAMILY	LABEL	SHR

DESCRIPTION

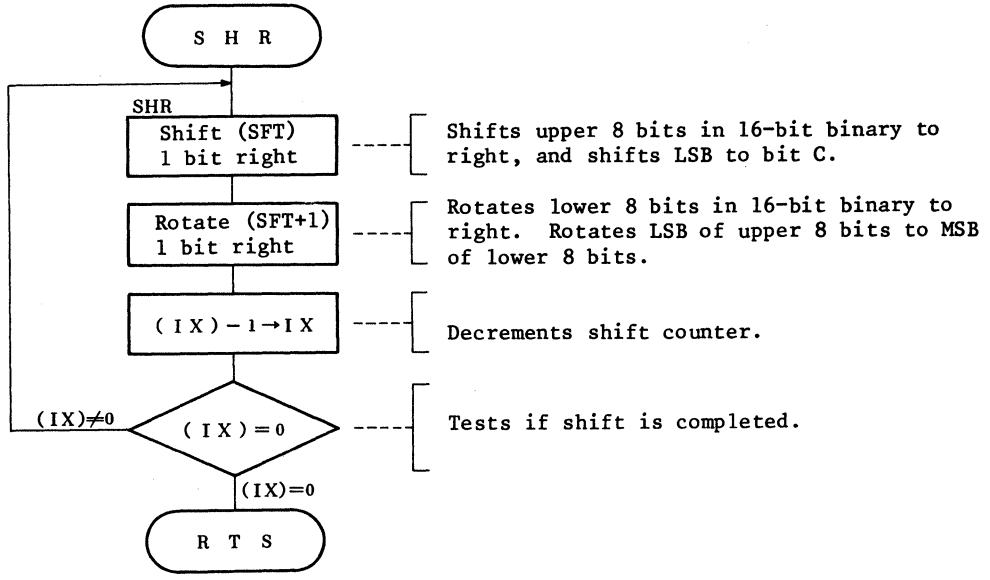
(5) Basic Operation

(a) Upper 8 bits in 16-bit binary are shifted to right. Here LSB is rotated to bit C. Lower 8 bits are then rotated to right. At this time, LSB in bit C is rotated to MSB of lower 8 bits.

(b) IX is used to keep track of number of shifts. IX is decremented each time (a) is executed. (a) is looped until IX is "0".

9. SHIFTING 16-BIT DATA	MCU/MPU	HD6305 FAMILY	LABEL	SHR

FLOWCHART



9. SHIFTING 16-BIT DATA	MCU/MPU	HD6305 FAMILY	LABEL	SHR

PROGRAM LISTING

```
00001                   ***********************************************
00002                   *                                             *
00003                   *       NAME : SHIFTING 16-BIT DATA  (SHR)    *
00004                   *                                             *
00005                   ***********************************************
00006                   *                                             *
00007                   *           ENTRY : SFT  (16-BIT BINARY DATA) *
00008                   *                   IX   (SHIFT COUNTER)      *
00009                   *         RETURNS : SFT  (16-BIT BINARY DATA) *
00010                   *                                             *
00011                   ***********************************************
00012                   *
00013 0080                      ORG    $80
00014                   *
00015 0080 0002         SFT     RMB    2          16-bit binary data
00016                   *
00017 1000                      ORG    $1000
00018                   *
00019       1000        SHR     EQU    *          Entry point
00020 1000 34 80                LSR    SFT        Shift upper byte to right
00021 1002 36 81                ROR    SFT+1      Rotate lower byte to right
00022 1004 5A                   DEC    X          Decrement shift counter
00023 1005 26 F9                BNE    SHR        Loop until shift counter = 0
00024 1007 81                   RTS
```

10. 4-DIGIT BCD COUNTER	MCU/MPU	HD6305 FAMILY	LABEL	DECNT

FUNCTION

(a) Increments 4-digit BCD in RAM.

(b) Permits easy counting of interrupts (external, timer, etc).

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	—	—	—
	Returns	4-digit BCD counter	DCNTR (RAM)	2
		Counter over-flow or not?	Bit C (CCR)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
×	×

C	Z
↕	×
N	I
×	●
H	
●	

SPECIFICATIONS

ROM (Bytes)
13
RAM (Bytes)
2
Stack (Bytes)
0
No. of cycles
42
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

DCNT (RAM) : Used as counter of 4-digit BCD. Counts at every DECNT execution.

Bit C (CCR) : Indicates counter status after DECNT execution.

Bit C=1 : Counter overflows (see Fig. 2).

Bit C=0 : Counter has returned normal.

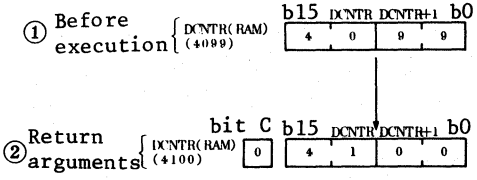


Fig. 1 Example of DECNT execution

SPECIFICATIONS NOTES

10. 4-DIGIT BCD COUNTER	MCU/MPU	HD6305 FAMILY	LABEL	DECNT

DESCRIPTION

(b) Fig. 1 shows example of DECNT execution. When DECNT is executed, 4-digit BCD is counted as shown in part ② of Fig. 1.

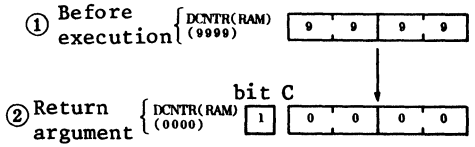


Fig. 2 Example of counter overflow

(2) User Notes

If counter overflows as shown in Fig. 2, counter is reset to "0".

(3) RAM Description

Label	RAM	Description
DCNTR	b7 Upper byte / Lower byte b0	Counter value is stored by 4-digit BCD.

(4) Sample Application

```
WORK1   RMB   2          ----- Reserves memory byte for results of
                               counting by 4-digit BCD.
WORK2   RMB   2          ----- Reserves memory byte for register
                               contents.
        :
        :
        STA   WORK2      }
        STX   WORK2+1    } ----- Saves register contents that will be
                               destroyed by DECNT execution.
        JSR   DECNT      ----- Calls DECNT subroutine.
        BCS   OVER       ----- When BCD counter overflows, branching is
                               effected to service routine for counter
                               overflow.
        LDA   DCNTR+1    }
        STA   WORK1+1    } ----- Stores result of counting by 4-digit BCD
        LDA   DCNTR      }       (return argument (DCNTR)) in RAM.
        STA   WORK1      }
        LDA   WORK2      }
        LDX   WORK2+1    } ----- Restores register.
        :
        :
OVER    Service routine in case
        of counter overflow.
        :
```

10. 4-DIGIT BCD COUNTER	MCU/MPU	HD6305 FAMILY	LABEL	DECNT

DESCRIPTION

(5) Basic Operation

(a) IX is used to indicate address of BCD counter and is also used to keep track of number of additions.

(b) Sets bit C for counting "1"s.

(c) Executes (Formula 1) using index addressing mode.
(Bit C is set at first addition, and a carry is performed after (Formula 1) execution.

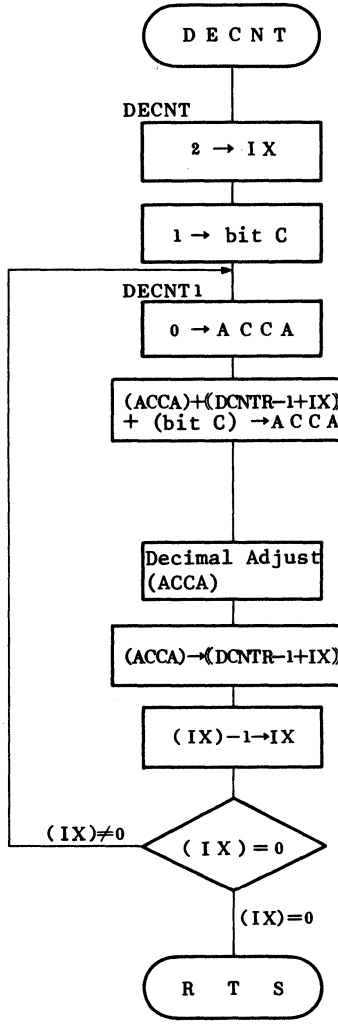
$$0 + ((DCNTR - 1 + IX)) + (\text{bit C}) \rightarrow ACCA \quad \text{-------- (Formula 1)}$$

(d) Decrements IX.

(e) Loops addition of upper digit and bit C until IX is "0".

10. 4-DIGIT BCD COUNTER	MCU/MPU	HD6305 FAMILY	LABEL	DECNT

FLOWCHART



Loads "2" into pointer indicating address of 4-digit BCD counter, and into counter indicating number of additions.

Bit C is set and ACCA is cleared in order to count up 4-digit BCD counter.

In the first loop, counts-up lower 2 digits in 4-digit BCD counter.
In the second loop, adds carry to upper 2 digits.

Adjusts result of addition into decimal, and stores it in 4-digit BCD counter's RAM.

Decrements pointer indicating address of 4-digit BCD counter and counter indicating number of additions.

Tests if counts-up is completed.

6

10. 4-DIGIT BCD COUNTER	MCU/MPU	HD6305 FAMILY	LABEL	DECNT

PROGRAM LISTING

```
00001                   ************************************************
00002                   *                                              *
00003                   *     NAME : 4-DIGIT BCD COUNTER (DECNT)       *
00004                   *                                              *
00005                   ************************************************
00006                   *                                              *
00007                   *        ENTRY : NOTHING                       *
00008                   *      RETURNS : DCNTR  (BCD COUNTER)          *
00009                   *                CARRY  (C=0:TRUE,C=1:OVER FLOW) *
00010                   *                                              *
00011                   ************************************************
00012                   *
00013 0080                      ORG    $80
00014                   *
00015 0080 0002         DCNTR   RMB    2         BCD counter
00016                   *
00017 1000                      ORG    $1000
00018                   *
00019      1000         DECNT   EQU    *         Entry point
00020 1000 AE 02                LDX    #2        Load ADDR pointer ( addition counter)
00021 1002 99                   SEC              Set carry
00022 1003 4F           DECNT1  CLR    A         Clear ACCA
00023 1004 E9 7F                ADC    DCNTR-1,X  Increment BCD counter
00024 1006 8D                   DAA              Convert into BCD
00025 1007 E7 7F                STA    DCNTR-1,X  Store BCD counter
00026 1009 5A                   DEC    X         Decrement ADDR pointer
00027 100A 26 F7                BNE    DECNT1    Loop until ADDR pointer=0
00028 100C 81                   RTS
```

11. COMPARING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	CMP

FUNCTION

(a) Determines larger than / smaller than relationship (> ,=, <) of 16-bit binary data of 2 groups, and loads result into bit C and bit Z of CCR.

(b) Utilizes unsigned integers in arguments.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	First value	CMD (RAM)	2
		Second value	CMT (RAM)	2
	Returns	Comparison results	bit C bit Z (CCR)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
×	●

C	Z
↕	↕
N	I
×	●
H	
●	

SPECIFICATIONS

ROM (Bytes)
11
RAM (Bytes)
4
Stack (Bytes)
0
No. of cycles
20
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

CMD (RAM): Holds the first 16-bit binary value.

CMT (RAM): Holds the second 16-bit binary value.

Bit C, Bit Z (CCR): Bit C and bit Z of CCR are held according to comparison results.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to make equal comparand and comparative number.

DESCRIPTION

(b) Table 1 shows example of CMP execution. If entry arguments are as shown in Table 1, bit C and bit Z of CCR are set accordingly.

Table 1. Example of CMP execution

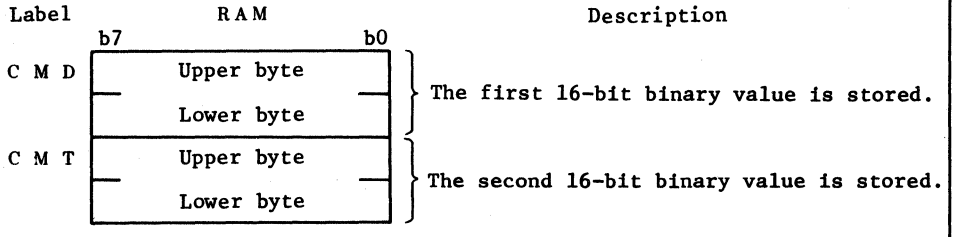
Entry arguments			Return arguments	
First value	Large/small relationship	Second value	C C R	
CMD (2 bytes RAM)		CMT (2 bytes RAM)	bit C	bit Z
$F67D	>	$2001	0	0
$2200	=	$2200	0	1
$4001	<	$F000	1	0

(c) After CMP execution, entry arguments are contained.

(2) User Notes

When not using upper byte, set "0" in the upper byte. If "0" is not set, comparison is performed with undefined data set in the upper byte, and correct data cannot be obtained.

(3) RAM Description



11. COMPARING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	CMP

DESCRIPTION

(4) Sample Application

CMP subroutine is called after the first value and the second value are held.

```
WORK1   RMB    2        ----- Reserves memory byte for the first
                               16-bit binary value.
WORK2   RMB    2        ----- Reserves memory byte for the second
                               16-bit binary value.
          :
          :
        TAX             ----- Saves register contents that will be
                               destroyed by CMP execution.
        LDA    WORK1
        STA    CMD
        LDA    WORK1+1  -----  Stores the first 16-bit binary value
        STA    CMD+1            into entry argument (CMD).
        LDA    WORK2
        STA    CMT      -----  Stores the second 16-bit binary value
        LDA    WORK2+1          into entry argument (CMT).
        STA    CMT+1
        JSR    CMP      ----- Calls CMP subroutine.
        TXA             ----- Restores register.
        BEQ    SKIP2    ----- Branches to service routine when
                               CMD=CMT.
        BCC    SKIP1    ----- Branches to service routine when
                               CMD>CMT.
      Service routine in case
      of CMD<CMT
        BRA    SKIP3
SKIP2 Service routine in case
      of CMD=CMT
        BRA    SKIP3
SKIP1 Service routine in case
      of CMD>CMT
SKIP3 User program
          :
          :
```

6

11. COMPARING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	CMP

DESCRIPTION

(5) Basic Operation

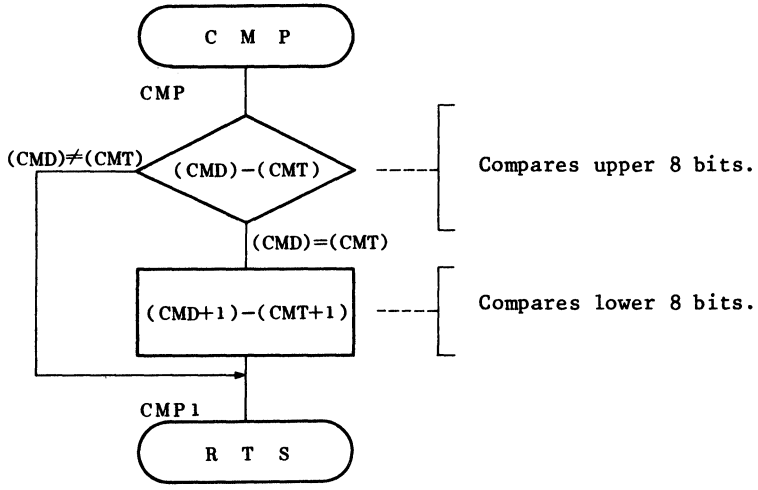
(a) When more than 2 bytes are compared, perform comparison for each byte.

(b) Bit C and bit Z of CCR are determined as return argument, after comparison (instruction CMP) are executed.

(c) Upper bytes are compared using one-byte (comparison instruction CMP).
When equal, lower bytes are compared.
When not equal, CMP is terminated.

11. COMPARING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	CMP

FLOWCHART



11. COMPARING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	CMP

PROGRAM LISTING

```
00001                   ********************************************
00002                   *                                          *
00003                   *   NAME : COMPARING 16-BIT BINARY DATA (CMP)  *
00004                   *                                          *
00005                   ********************************************
00006                   *                                          *
00007                   *   ENTRY : CMD           (COMPARAND)          *
00008                   *           CMT           (COMPARATIVE NUMBER) *
00009                   * RETURNS : CARRY & BIT Z (COMPARISON RESULT)  *
00010                   *                                          *
00011                   ********************************************
00012                   *
00013 0080                      ORG     $80
00014                   *
00015 0080 0002         CMD     RMB     2       Comparand
00016 0082 0002         CMT     RMB     2       Comparative number
00017                   *
00018 1000                      ORG     $1000
00019                   *
00020      1000         CMP     EQU     *       Entry point
00021 1000 B6 80                LDA     CMD     Compare with upper byte
00022 1002 B1 82                CMP     CMT     CMD - CMT ?
00023 1004 26 04                BNE     CMP1    Branch if not equal
00024 1006 B6 81                LDA     CMD+1   Compare with lower byte
00025 1008 B1 83                CMP     CMT+1   CMD+1 - CMT+1 ?
00026 100A 81           CMP1    RTS
```

12. ADDING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	ADD

FUNCTION

(a) Performs addition of 16-bit binary data in RAM, and stores result in RAM.

(b) Utilizes unsigned integers in arguments.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Augend	ADED (RAM)	2
		Addend	ADER (RAM)	2
	Returns	Addition results	ADED (RAM)	2
		Carry or no carry	Bit C (CCR)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
×	●

C	Z
↕	×
N	I
×	●
H	
×	

SPECIFICATIONS

ROM (Bytes)
13
RAM (Bytes)
4
Stack (Bytes)
0
No. of cycles
23
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

ADED (RAM) : Holds 16-bit binary augend. After ADD execution, contains addition result.

ADER (RAM) : Holds 16-bit binary addend.

Bit C (CCR): Indicates whether carry is generated or not after ADD execution.

Bit C = 1 : Carry is generated in addition result. (see Fig. 2)

Bit C = 0 : No carry is generated in addition result.

SPECIFICATIONS NOTES

12. ADDING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	ADD

DESCRIPTION

(b) Fig. 1 shows example of ADD execution. If entry arguments are as shown in part ① of Fig. 1, addition result is held in ADED (RAM) as shown in part ② of Fig. 1.

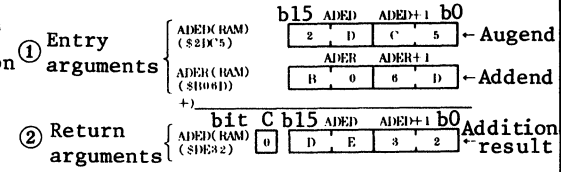


Fig. 1 Example of ADD execution

(2) User Notes

(a) As shown in Fig. 3, when not using upper byte, hold "0" in the upper byte. If "0" is not held, addition is performed with undefined data held in the upper byte, and correct result cannot be obtained.

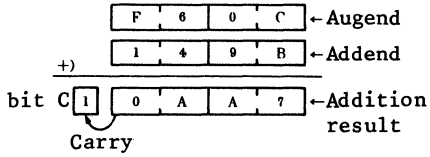


Fig. 2 Example of addition when carry is generated

(b) After ADD execution, augend is destroyed because addition result is contained in ADED (RAM). If augend needs to be retained after ADD execution, it should be saved in memory before execution.

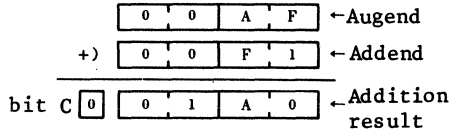
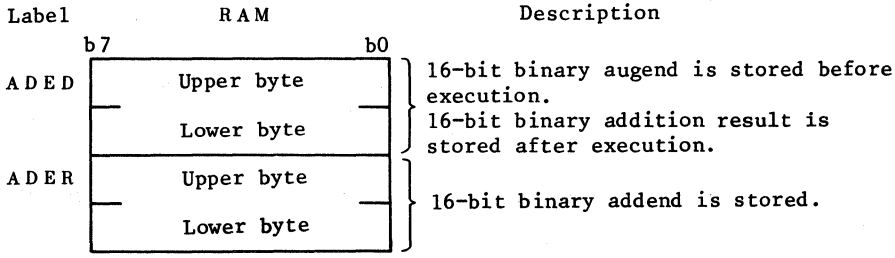


Fig. 3 Example of addition when upper byte is not used

(3) RAM Description



12. ADDING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	ADD

DESCRIPTION

(4) Sample Application

ADD subroutine is called after augend and addend are held.

```
WORK1   RMB   2       ----- Reserves memory byte for 16-bit binary
                            augend.
WORK2   RMB   2       ----- Reserves memory byte for 16-bit binary
                            addend.
WORK3   RMB   2       ----- Reserves memory byte for 16-bit binary
                            addition results.
WORK4   RMB   1       ----- Reserves memory byte for register contents.
          :
          :
        STA   WORK4   ----- Saves register contents that will be
                            destroyed by ADD execution.
        LDA   WORK1
        STA   ADED    -----
        LDA   WORK1+1       Loads 16-bit binary augend into entry
        STA   ADED+1        argument (ADED).
        LDA   WORK2
        STA   ADER    -----
        LDA   WORK2+1       Loads 16-bit binary addend into entry
        STA   ADER+1        argument (ADER).
        JSR   ADD     ----- Calls ADD subroutine.
        BCS   OVER    ----- If carry is generated in addition result,
                            branches to service carry routine.
        LDA   ADED
        STA   WORK3
        LDA   ADED+1  -----   Stores addition result (return argument
        STA   WORK3+1         (ADED)) in RAM.
        LDA   WORK4   ----- Restores register.
          :
          :
OVER    Service routine in case
        of carry
          :
          :
```

6

12. ADDING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	ADD

DESCRIPTION

(5) Basic Operation

(a) When more than 2 bytes are added, perform addition for each byte.

(b) Lower bytes (ADED+1, ADER+1) shown in (Formula 1) are added using 1-byte addition (instruction ADD) ignoring bit C.
When carry is generated after performing (Formula 1), bit C is set.

(ADED + 1) + (ADER + 1) → ADED + 1 ------- (Formula 1)

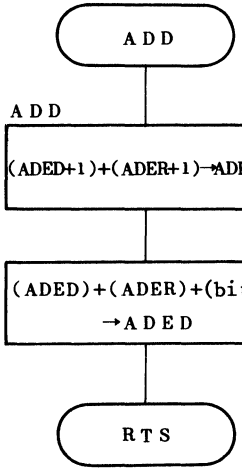
(c) Upper bytes (ADED, ADER) shown in (Formula 2) are added using 1-byte addition (instruction ADC) based on consideration of bit C.

(ADED) + (ADER) + (bit C) → ADED ----------(Formula 2)

Bit C is taken into consideration because there is carry involved with the addition result regarding lower bytes executed in (b).

12. ADDING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	ADD

FLOWCHART



Adds lower 8 bits and stores result in augend (ADED) RAM.

Adds upper 8-bits considering carry (bit C) which occurred in lower 8 bits addition, and stores result in augend RAM.

6

12. ADDING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	ADD

PROGRAM LISTING

```
00001                   ************************************************
00002                   *                                              *
00003                   *       NAME : ADDING 16-BIT BINARY DATA  (ADD)  *
00004                   *                                              *
00005                   ************************************************
00006                   *                                              *
00007                   *         ENTRY : ADED  (AUGEND)               *
00008                   *                 ADER  (ADDEND)               *
00009                   *       RETURNS : ADED  (SUM)                  *
00010                   *                 CARRY (C=0;TRUE,C=1;OVER FLOW) *
00011                   *                                              *
00012                   ************************************************
00013                   *
00014 0080                        ORG     $80
00015                   *
00016 0080 0002         ADED      RMB     2         Augend -> sum
00017 0082 0002         ADER      RMB     2         Addend
00018                   *
00019 1000                        ORG     $1000
00020                   *
00021       1000        ADD       EQU     *         Entry point
00022 1000 B6 81                  LDA     ADED+1    Load lower augend
00023 1002 BB 83                  ADD     ADER+1    Add lower addend
00024 1004 B7 81                  STA     ADED+1    Store lower augend area
00025 1006 B6 80                  LDA     ADED      Load upper augend
00026 1008 B9 82                  ADC     ADER      Add upper addend with carry
00027 100A B7 80                  STA     ADED      Store upper augend area
00028 100C 81                     RTS
```

13. SUBTRACTING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	SUB

FUNCTION

(a) Performs subtraction of 16-bit binary data in RAM, and stores result in RAM.

(b) Utilizes unsigned integers in arguments.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Minuend	SBED (RAM)	2
		Subtrahend	SBER (RAM)	2
	Returns	Subtraction results	SBED (RAM)	2
		Borrow or no borrow	bit C (CCR)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
×	●

C	Z
↕	×
N	I
×	●
H	
●	

SPECIFICATIONS

ROM (Bytes)
13
RAM (Bytes)
4
Stack (Bytes)
0
No. of cycles
23
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

SBED (RAM) : Holds 16-bit binary minuend. After SUB execution, contains subtraction result.

SBER (RAM) : Holds 16-bit binary subtrahend.

Bit C (CCR): Indicates whether borrow is generated or not after SUB execution.

Bit C=1 : Borrow is generated in subtraction result. (see Fig. 2)

Bit C=0 : No borrow is generated in subtraction result.

SPECIFICATIONS NOTES

13. SUBTRACTING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	SUB

DESCRIPTION

(b) Fig. 1 shows example of SUB execution. If entry arguments are as shown in part ① of Fig. 1, subtraction result is stored in SBED (RAM) as shown in part ② of Fig. 1.

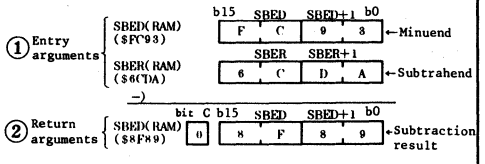


Fig. 1 Example of SUB execution

(2) User Notes

(a) When subtraction result is negative, (Minuend < Subtrahend), the result is 2's complement.

(b) As shown in Fig. 3, when not using upper byte, hold "0" in the upper byte. If "0" is not held, subtraction is performed with undefined data held in the upper byte, and correct result cannot be obtained.

(c) After SUB execution, minuend is destroyed because subtraction result is contained in SBED (RAM). If minuend needs to be retained after SUB execution, it should be saved in memory before execution.

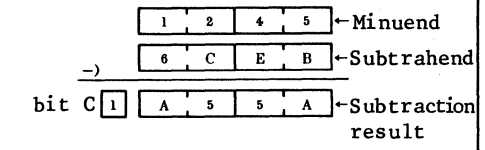


Fig. 2 Example of subtraction when borrow is generated

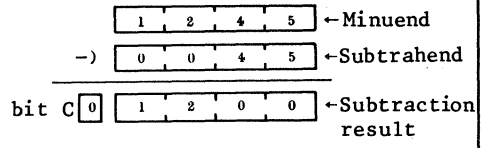
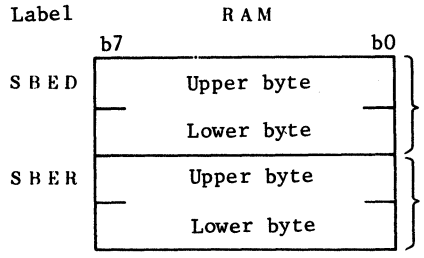


Fig. 3 Example of subtraction when upper byte is not used

(3) RAM Description



Description

16-bit binary minuend is stored before execution.

16-bit binary subtraction result is stored after execution.

16-bit binary subtrahend is stored.

13. SUBTRACTING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	SUB

DESCRIPTION

(4) Sample Application

SUB subroutine is called after minuend and subtrahend are held.

```
WORK1   RMB     2       ----- Reserves memory byte for 16-bit binary
                              minuend.
WORK2   RMB     2       ----- Reserves memory byte for 16-bit binary
                              subtrahend.
WORK3   RMB     2       ----- Reserves memory byte for 16-bit binary
                              subtraction results.
WORK4   RMB     1       ----- Reserves memory byte for register contents.
          .
          .
          .
        STA     WORK4   ----- Saves register contents that will be
                              destroyed by SUB execution.
        LDA     WORK1
        STA     SBED    -----
        LDA     WORK1+1       Stores 16-bit binary minuend into entry
        STA     SBED+1        argument(SBED).

        LDA     WORK2
        STA     SBER    -----
        LDA     WORK2+1       Stores 16-bit binary subtrahend into entry
        STA     SBER+1        argument (SBER).

        JSR     SUB     ----- Calls SUB subroutine.

        BCS     OVER    ----- If borrow is generated in subtraction
                              result, branches to service borrow routine.

        LDA     SBED
        STA     WORK3   -----
        LDA     SBED+1        Stores subtraction result (return
        STA     WORK3+1       argument (SBED)) in RAM.

        LDA     WORK4   ----- Restores register.
          .
          .
          .
OVER    Service routine in case
        of borrow.
          .
          .
          .
```

13. SUBTRACTING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	SUB

DESCRIPTION

(5) Basic Operation

(a) When more than 2 bytes are subtracted, perform subtraction for each byte.

(b) Lower bytes (SBED+1, SBER+1) shown in (Formula 1) are subtracted using 1-byte subtraction (instruction SUB) ignoring bit C. When borrow is generated after performing (Formula 1), bit C is set.

(SBED + 1) - (SBER + 1) → SBED + 1 ----- (Formula 1)

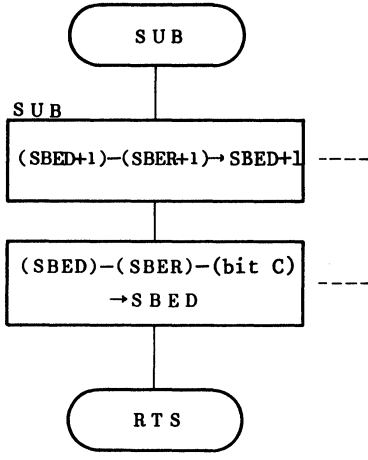
(c) Upper bytes (SBED, SBER) shown in (Formula 2) are subtracted using 1-byte subtraction (instruction SBC) based on consideration of bit C.

(SBED) - (SBER) - (bit C) → SBED ----- (Formula 2)

Bit C is taken into consideration because there is borrow involved with the subtraction result regarding lower bytes executed in (b).

13. SUBTRACTING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	SUB

FLOWCHART



Subtracts lower 8 bits, and stores result in minuend RAM (SBED).

Subtracts upper 8 bits considering borrow (bit C) which occurred in lower 8 bits, and stores result in minuend RAM (SBED).

13. SUBTRACTING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	SUB

PROGRAM LISTING

```
00001                   ************************************************
00002                   *                                              *
00003                   *    NAME : SUBTRACTING 16-BIT BINARY DATA (SUB)  *
00004                   *                                              *
00005                   ************************************************
00006                   *                                              *
00007                   *             ENTRY : SBED  (MINUEND)          *
00008                   *                     SBER  (SUBTRAHEND)       *
00009                   *           RETURNS : SBED  (RESIDUAL)         *
00010                   *                     CARRY (C=0:TRUE,C=1:BORROW)  *
00011                   *                                              *
00012                   ************************************************
00013                   *
00014 0080                      ORG     $80
00015                   *
00016 0080 0002         SBED    RMB     2       Minuend ->Residual
00017 0082 0002         SBER    RMB     2       Subtrahend
00018                   *
00019 1000                      ORG     $1000
00020                   *
00021      1000         SUB     EQU     *       Entry point
00022 1000 B6 81                LDA     SBED+1  Load lower minuend
00023 1002 B0 83                SUB     SBER+1  Subtract lower subtrahend
00024 1004 B7 81                STA     SBED+1  Store lower minuend area
00025 1006 B6 80                LDA     SBED    Load upper minuend
00026 1008 B2 82                SBC     SBER    Subtract upper subtrahend
00027 100A B7 80                STA     SBED    Store upper minuend area
00028 100C 81                   RTS
```

14. MULTIPLYING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	MUL

FUNCTION

(a) Performs multiplication of 16-bit binary data in RAM and stores result in 32-bit binary in RAM.

(b) Utilizes 16-bit unsigned integers in arguments.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Multiplicand	MCAND (RAM)	2
		Multiplier	MER (RAM)	2
	Returns	Product (Upper 2 bytes)	PRDCT (RAM)	2
		Product (Lower 2 bytes)	MER (RAM)	2

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
×	×

C	Z
×	×
N	I
×	●
H	
×	

SPECIFICATIONS

ROM (Bytes)
33
RAM (Bytes)
6
Stack (Bytes)
0
No. of cycles
785
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTIONS

(1) Function Details

(a) Argument details

MCAND (RAM): Holds 16-bit binary multiplicand.

MER (RAM): Holds 16-bit binary multiplier.
After MUL execution, the lower 16 bits of multiplication result is contained.

PRDCT (RAM): Contains upper 16 bits of product.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to hold $FFFF as entry argument.

14. MULTIPLYING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	MUL

DESCRIPTION

(b) Fig. 1 shows example of MUL execution. If entry arguments are as shown in part ① of Fig. 1, multiplication result is stored in PRDCT (RAM) and MER (RAM) as shown in part ② of Fig. 1.

(c) Table 1 shows result when "0" is held as entry argument.

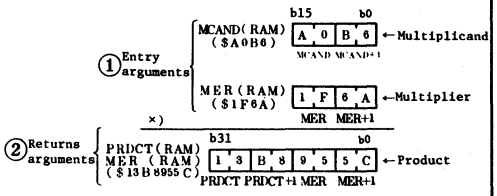


Fig. 1 Example of MUL execution

(2) User Notes

(a) As shown in Fig. 2, when not using upper byte, hold "0" in the upper byte. If "0" is not held, multiplication is performed with undefined data, and correct product cannot be obtained.

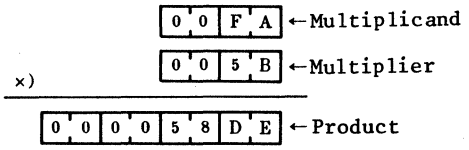


Fig. 2 Multiplication example when upper byte is not used

Table 1 Product when not holding "0" as entry arguments

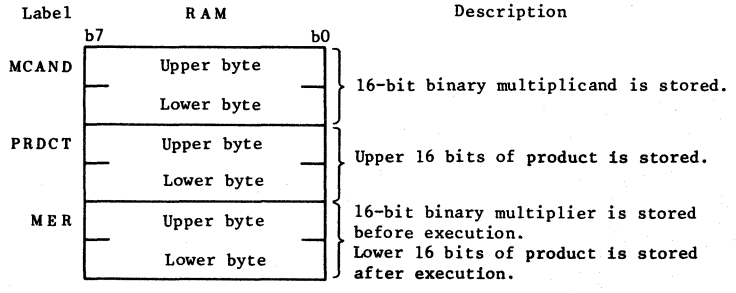
Entry argument		Return argument
Multiplicand (MCAND:MCAND+1)	Multiplier (MER:MER+1)	Product (PRDCT:PRDCT+1: MER:MER+1)
$**** *	$0000	$00000000
$0000	$**** *	$00000000
$0000	$0000	$00000000

(Note) * $**** indicates hexadecimals.

(b) After MUL execution, multiplier is destroyed because lower 2 bytes of Product is held in MER (RAM).

If minuend needs to be retained after MUL execution, it should be saved in memory before execution.

(3) RAM Description



14. MULTIPLYING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	MUL

DESCRIPTION

(4) Sample Application

MUL subroutine is called after multiplicand and multiplier are held.

```
WORK1   RMB     2       ----- Reserves memory byte for 16-bit binary
                              multiplicand.
WORK2   RMB     2       ----- Reserves memory byte for 16-bit binary
                              multiplier.
WORK3   RMB     4       ----- Reserves memory byte for product.

WORK4   RMB     2       ----- Reserves memory byte for register contents.
        :
        :
        :
        STA     WORK4   }-----
        STX     WORK4+1 }       Saves register contents that will be
                                destroyed by MUL execution.
        LDA     WORK1   }
        STA     MCAND   }-----
        LDA     WORK1+1 }       Stores 16-bit binary multiplicand into
        STA     MCAND+1 }       entry argument (MER).
        LDA     WORK2   }
        STA     MER     }-----
        LDA     WORK2+1 }       Stores 16-bit binary multiplier into entry
        STA     MER+1   }       argument (MCAND).
        JSR     MUL     ----- Calls MUL subroutine.
        LDA     PRDCT   }
        STA     WORK3   }
        LDA     PRDCT+1 }
        STA     WORK3+1 }-----  Stores 32-bit binary product
        LDA     MER     }       (return argument (PRDCT, MER)) in RAM.
        STA     WORK3+2 }
        LDA     MER+1   }
        STA     WORK3+3 }
        LDA     WORK4   }-----  Restores register.
        LDX     WORK4+1 }
        :
        :
        :
```

14. MULTIPLYING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	MUL

DESCRIPTION

(5) Basic Operation

(a) Fig. 3 shows example of 4-bit binary multiplication.

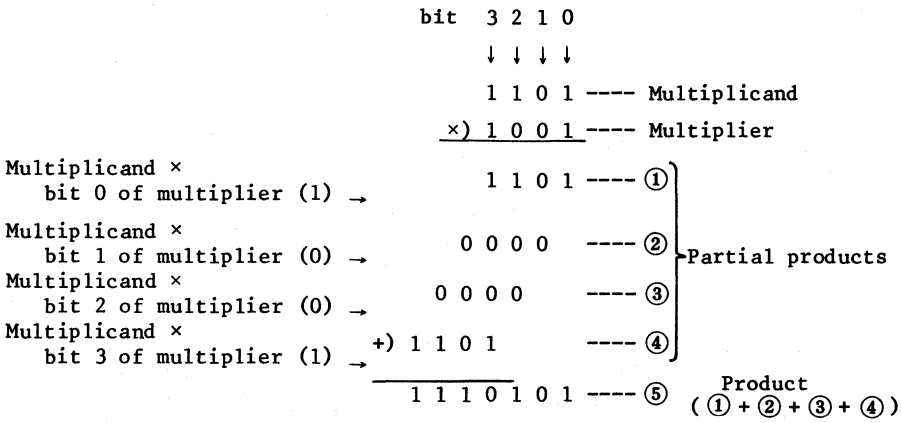


Fig. 3 A multiplication example ($0D × $09 = $75)

Multiplication of 4-bit binary data requires obtaining partial products, as shown in ① , ② , ③ and ④ , and adding them. (⑤ in Fig. 3)
Each bit of binary data is either "0" or "1". If multiplier bit is 1, its partial product is multiplicand (① and ④ in Fig. 3), while if multiplier bit is "0", its partial product is "0". (② and ③ in Fig. 3)

(b) Using Fig. 3, the program is explained as follows:

(i) Clears RAM (PRDCT).

(ii) IX is used to indicate number of partial product.

(iii) To determine partial product for each 1 bit, tests whether LSB of multiplier (MER), is "1" or "0".

(iv) If "1", Adds multiplicand to RAM for product, and stores sum in RAM for product, because if LSB of RAM for multiplier is "1", partial product is multiplicand. If LSB of RAM for multiplier is "0", partial product is "0", and there is no execution.

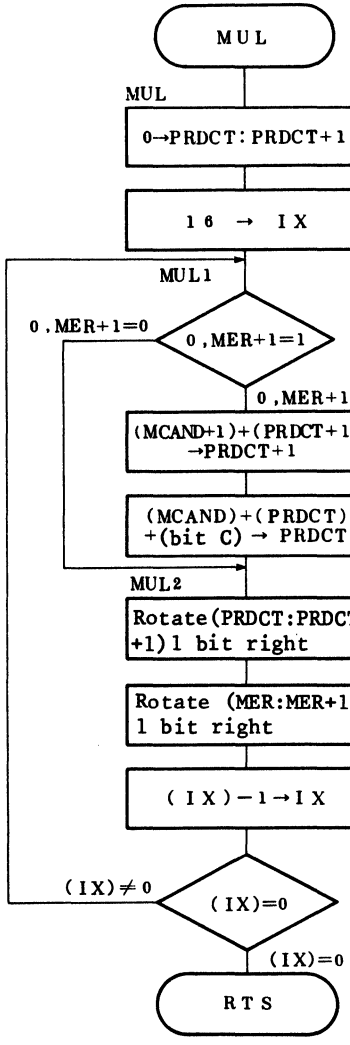
(v) To acquire the partial product for the next bit in RAM for multiplier, rotates RAM for product and RAM for multiplier 1 bit right.

(vi) Decrements IX.

(vii) Loops (iii) to (vi) until IX is "0".

14. MULTIPLYING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	MUL

FLOWCHART



Clears RAM where product is to be stored.

Stores counter indicating number of partial products.

Tests if LSB of multiplier is "1" or "0".

Adds multiplicand to product because partial product is multiplicand when LSB of multiplier is "1".

Shifts LSB of product to upper bit of multiplier, and shifts LSB of multiplier to bit C by rotating RAM for the product and multiplier 1 bit right.

Decrements counter indicating number of partial products.

Tests if all the partial products have been obtained or not.

6

14. MULTIPLYING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	MUL

PROGRAM LISTING

```
00001                    **********************************************
00002                    *                                            *
00003                    *       NAME : 16-BIT MULTIPLATION (MUL)     *
00004                    *                                            *
00005                    **********************************************
00006                    *                                            *
00007                    *         ENTRY : MACAND (MULTIPLICAND)      *
00008                    *                 MER    (MULTIPLIER)        *
00009                    *       RETURNS : PRDCT  (UPPER PRODUCT)     *
00010                    *                 MER    (LOWER PRODUCT)     *
00011                    *                                            *
00012                    **********************************************
00013                    *
00014 0080               *       ORG     $80
00015                    *
00016 0080 0002          MCAND   RMB     2       Multiplicand
00017 0082 0002          PRDCT   RMB     2       Product (upper bytes)
00018 0084 0002          MER     RMB     2       Multiplier->Product (lower bytes)
00019                    *
00020 1000               *       ORG     $1000
00021                    *
00022 1000 3F 82         MUL     CLR     PRDCT   Clear product area
00023 1002 3F 83                 CLR     PRDCT+1
00024 1004 AE 10                 LDX     #16     Set bit counter
00025 1006 01 85 0C      MUL1    BRCLR   0,MER+1,MUL2  Branch if MER(0)=0
00026 1009 B6 81                 LDA     MCAND+1 MCAND+PRDCT->PRDCT
00027 100B BB 83                 ADD     PRDCT+1
00028 100D B7 83                 STA     PRDCT+1
00029 100F B6 80                 LDA     MCAND
00030 1011 B9 82                 ADC     PRDCT
00031 1013 B7 82                 STA     PRDCT
00032 1015 36 82         MUL2    ROR     PRDCT   Rotate product area
00033 1017 36 83                 ROR     PRDCT+1
00034 1019 36 84                 ROR     MER     Rotate multiplier area-
00035 101B 36 85                 ROR     MER+1   -and set LSB of MER to carry
00036 101D 5A                    DEC     X       Decrement bit counter
00037 101E 26 E6                 BNE     MUL1    Loop until bit counter=0
00038 1020 81                    RTS
```

15. DIVIDING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	DIV

FUNCTION

(a) Performs divisions of 16-bit binary data in RAM and stores result (quotient and residual) in 16-bit binary in RAM.

(b) Utilizes unsigned integers in arguments.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Dividend	DVD (RAM)	2
		Divisor	DVS (RAM)	2
	Returns	Quotient	DVD (RAM)	2
		Residual	RSD (RAM)	2

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
×	×

C	Z
×	×
N	I
×	●
H	
×	

SPECIFICATIONS

ROM (Bytes)
47
RAM (Bytes)
6
Stack (Bytes)
0
No. of cycles
1137
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

DVD (RAM): Holds 16-bit binary dividend. Contains quotient after DIV execution.

DVS (RAM): Holds 16-bit binary divisor.

RSD (RAM): Holds 16-bit binary residual.

SPECIFICATIONS NOTES

DESCRIPTION

(b) Fig. 1 shows example of DIV execution. If entry arguments are as shown in part ① of Fig. 1, division result is stored in DVD (RAM) and RSD (RAM) as shown part ② of Fig. 1.

(c) Table 1 shows result of holding "0" as argument.

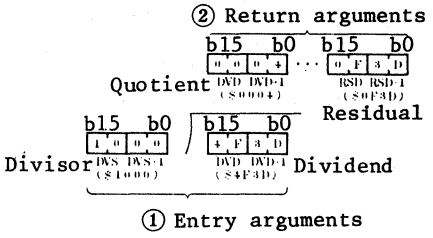


Fig. 1 Example of DIV execution

(2) User Notes

(a) When not using upper byte as shown in Fig. 2, hold "0" in the upper byte. If "0" is not held, division is performed with undefined data in the upper byte, and correct result cannot be obtained.

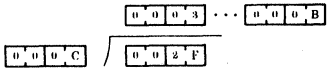


Fig. 2 DIV example when upper bytes are not used

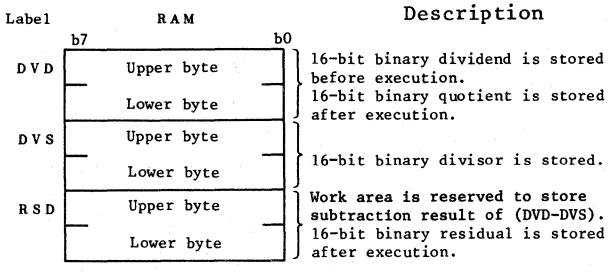
(b) After executing DIV, dividend is destroyed because quotient is contained in DVD (RAM). If dividend needs to be retained after DIV execution, it should be saved in memory before execution.

TABLE 1 Results when holding "0" as entry arguments

Entry Arguments		Return Arguments	
Dividend (DVD)	Divisor (DVS)	Quotient (DVD)	Residual (RSD)
$ * * * * *	$ 0 0 0 0	$ F F F F	$ F F F F
$ 0 0 0 0	$ * * * * *	$ 0 0 0 0	$ 0 0 0 0
$ 0 0 0 0	$ 0 0 0 0	$ F F F F	$ 0 0 0 0

(Note) * $**** indicates hexadecimals.

(3) RAM Description



15. DIVIDING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	DIV

DESCRIPTION

(4) Sample Application

DIV subroutine is called after dividend and divisor are held.

```
WORK1   RMB     2       ----- Reserves memory byte for 16-bit binary
                              divisor.
WORK2   RMB     2       ----- Reserves memory byte for 16-bit binary
                              dividend.
WORK3   RMB     2       ----- Reserves memory byte for 16-bit binary
                              quotient.
WORK4   RMB     2       ----- Reserves memory byte for 16-bit binary
                              residual.
WORK5   RMB     2       ----- Reserves memory byte in stack register.
          .
          .
          .
        STA     WORK5   }-----
        STX     WORK5+1 }     Saves register contents that will be
                              destroyed by executing DIV.
        LDA     WORK1   }
        STA     DVS     }-----
        LDA     WORK+1  }     Stores 16-bit binary dividend into entry
        STA     DVS+1   }     argument (DVS).
        LDA     WORK2   }
        STA     DVD     }-----
        LDA     WORK2+1 }     Stores 16-bit binary dividend into entry
        STA     DVD+1   }     argument (DVD).

        JSR     DIV     ----- Calls DIV subroutine.

        LDA     DVD     }
        STA     WORK3   }
        LDA     DVD+1   }
        STA     WORK3+1 }-----
        LDA     RSD     }     Stores division result (return arguments
        STA     WORK4   }     (DVD, RSD)) in RAM.
        LDA     RSD+1   }
        STA     WORK4+1 }
        LDA     WORK5   }-----
        LDX     WORK5+1 }     Restores register.
          .
          .
          .
```

6

DESCRIPTION

(5) Basic Operation

(a) In binary code division, quotient and residual are obtained by repeated subtraction. Fig. 3 shows example of division ($0D ÷ $03).

```
                    ③ ⑥
                     :  :
                     1  0  0   ← Quotient
Divisor → 1 1 ) 1  1  0  1   ← Dividend
          -)  1  1           ---- ①
              0  0           ---- ②
          -)     1  1        ---- ④
              -  0  1        ---- ⑤
          +)     1  1
              0  0  1
          -)        1  1
                 -1  0
          +)        1  1
              0  0  1   ← Residual
```

Fig. 3 Division example ($0D ÷ $03)

(b) Using Fig. 3, the flowchart is explained as follows;

(i) IX is used to indicate number of shifts.

(ii) Clears work area (RSD: RSD+1) where residual will be stored.

(iii) Sets bit C in advance to store "1" in quotient RAM.

(iv) Rotates DVD:DVD+1 (dividend → quotient and RSD:RSD+1 left 1 bit, sets "1" in quotient RAM and simultaneously shifts MSB of DVD: DVD+1 to LSB of RSD:RSD+1. This is because when performing subtraction, the upper bits are fetched one by one from dividend.

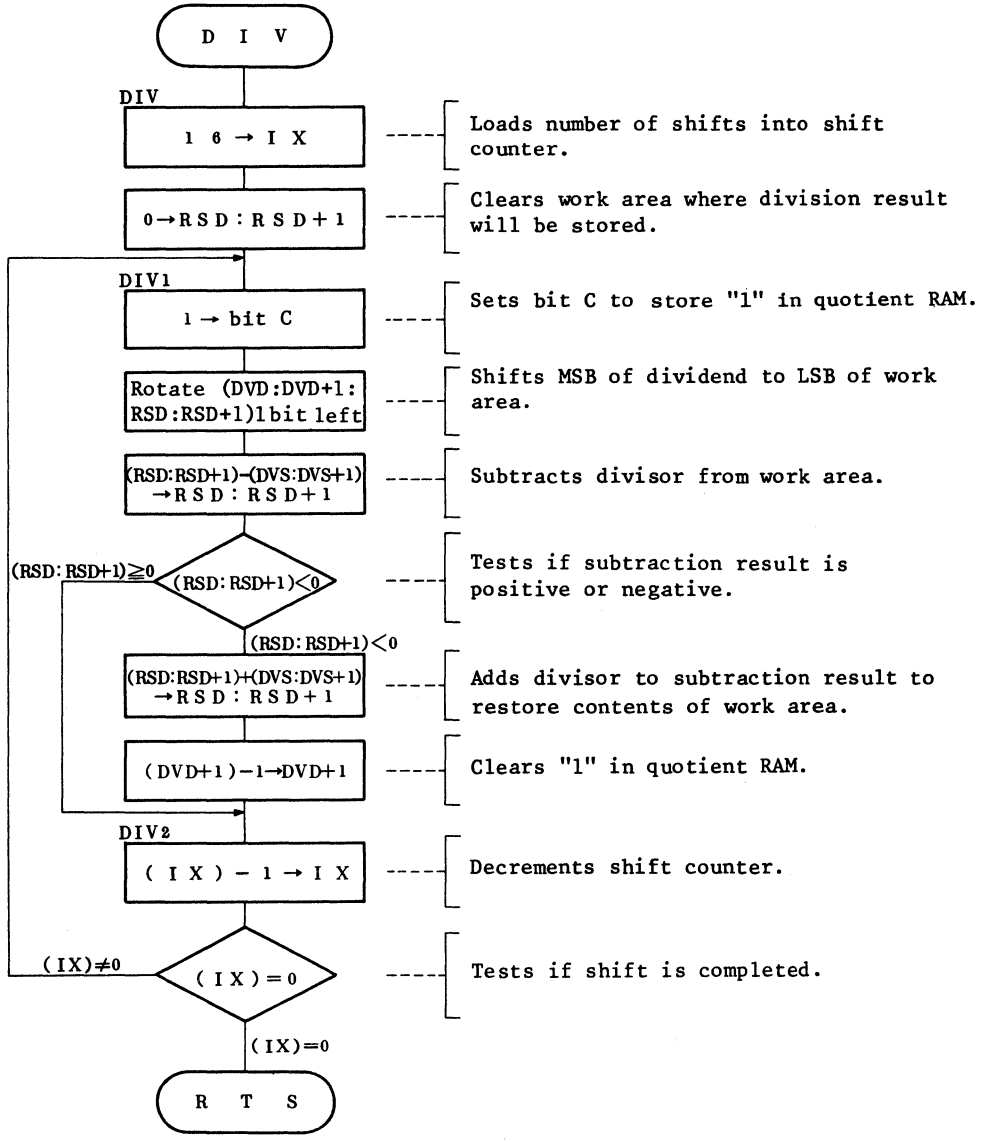
(v) Subtracts divisor, DVS:DVS+1 from RSD:RSD+1. If subtraction result is positive, sets LSB of DVD:DVD+1 to "1".
(Fig. 3 ①→②→③)
If subtraction result is negative, adds divisor to subtraction result and clears "1" set in quotient RAM.
(Fig. 3 ④→⑤→⑥)

(vi) Decrements shift counter.

(vii) Loops (iii) to (vi) until shift counter is "0".

15. DIVIDING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	DIV

FLOWCHART



15. DIVIDING 16-BIT BINARY DATA	MCU/MPU	HD6305 FAMILY	LABEL	DIV

PROGRAM LISTING

```
00001                   ************************************************
00002                   *                                              *
00003                   *     NAME : DIVING 16-BIT BINARY DATA  (DIV)  *
00004                   *                                              *
00005                   ************************************************
00006                   *                                              *
00007                   *           ENTRY : DVD   (DIVIDEND)           *
00008                   *                   DVS   (DIVISOR)            *
00009                   *         RETURNS : DVD   (QUOTIENT)           *
00010                   *                   RSD   (RESIDUAL)           *
00011                   *                                              *
00012                   ************************************************
00013                   *
00014 0080                        ORG     $80
00015                   *
00016 0080 0002         DVD       RMB     2         Dividend -> Quotient
00017 0082 0002         DVS       RMB     2         Divisor
00018 0084 0002         RSD       RMB     2         Residual
00019                   *
00020 1000                        ORG     $1000
00021                   *
00022       1000        DIV       EQU     *         Entry point
00023 1000 AE 10                  LDX     #16       Set shift counter
00024 1002 3F 84                  CLR     RSD       Clear work (set quotient afterward)
00025 1004 3F 85                  CLR     RSD+1
00026 1006 99           DIV1      SEC               Set LSB of residual area to high
00027 1007 39 81                  ROL     DVD+1     Shift dividend and set MSB -
00028 1009 39 80                  ROL     DVD       -of dividend to LSB of work
00029 100B 39 85                  ROL     RSD+1
00030 100D 39 84                  ROL     RSD
00031 100F B6 85                  LDA     RSD+1     Work - divisor -> Work
00032 1011 B0 83                  SUB     DVS+1
00033 1013 B7 85                  STA     RSD+1
00034 1015 B6 84                  LDA     RSD
00035 1017 B2 82                  SBC     DVS
00036 1019 B7 84                  STA     RSD
00037 101B 24 0E                  BCC     DIV2      Branch if work > divisor
00038 101D B6 83                  LDA     DVS+1     Work + Divisor -> Work
00039 101F BB 85                  ADD     RSD+1
00040 1021 B7 85                  STA     RSD+1
00041 1023 B6 82                  LDA     DVS
00042 1025 B9 84                  ADC     RSD
00043 1027 B7 84                  STA     RSD
00044 1029 3A 81                  DEC     DVD+1     Clear LSB of residual area
00045 102B 5A           DIV2      DEC     X         Decrement shift counter
00046 102C 26 D8                  BNE     DIV1      Loop until shift counter = 0
00047 102E 81                     RTS
```

16. ADDING 8-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	ADDD

FUNCTION

(a) Performs addition of 8-digit BCD in RAM, and stores result in 8-digit BCD in RAM.

(b) Utilizes unsigned integers in arguments.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Augend	ABD (RAM)	4
		Addend	ACD (RAM)	4
	Returns	Addition results	ABD (RAM)	4
		Carry or no carry	bit C (CCR)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
×	×

C	Z
↕	×

N	I
×	●

H
×

SPECIFICATIONS

ROM (Bytes)
14
RAM (Bytes)
8
Stack (Bytes)
0
No. of cycles
84
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTIONS

(1) Function Details

(a) Argument details

ABD (RAM): Holds 8-digit BCD augend. After ADDD execution, Contains addition result.

ACD (RAM): Holds 8-digit BCD added.

Bit C (CCR): Indicates whether a carry is generated or not after ADDD execution.

Bit C=1: Carry is generated in addition result.

Bit C=0: No carry is generated in addition result.

(See Fig. 2)

SPECIFICATIONS NOTES

DESCRIPTION

(b) Fig. 1 shows example of ADDD execution. If entry arguments are as shown in part ① of Fig. 1, addition results are contained in ABD (RAM) as shown in part ② of Fig. 1.

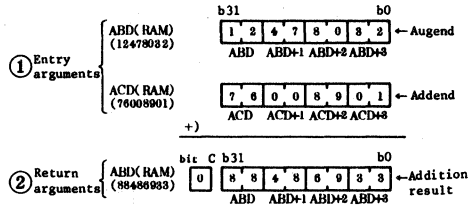


Fig. 1 Example of ADDD execution

(2) User Notes

(a) As shown in Fig. 3, when not using upper digits, hold "0" in the upper digits. If "0" is not held, addition is performed with undefined data in the upper digits, and correct result cannot be obtained.

(b) After ADDD execution, augend is destroyed because addition result are contained in ABD (RAM). If augend needs to be ratained after ADDD execution, it should be saved in memory before execution.

(c) Hold BCD data as augend and addend. If data other than in BCD from is held, correct addition result cannot be obtained.

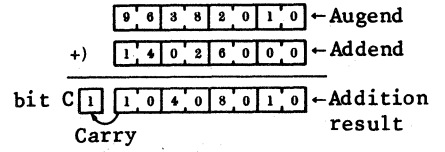


Fig. 2 Addition example when carry is generated

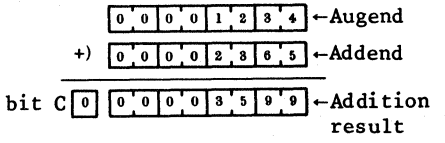
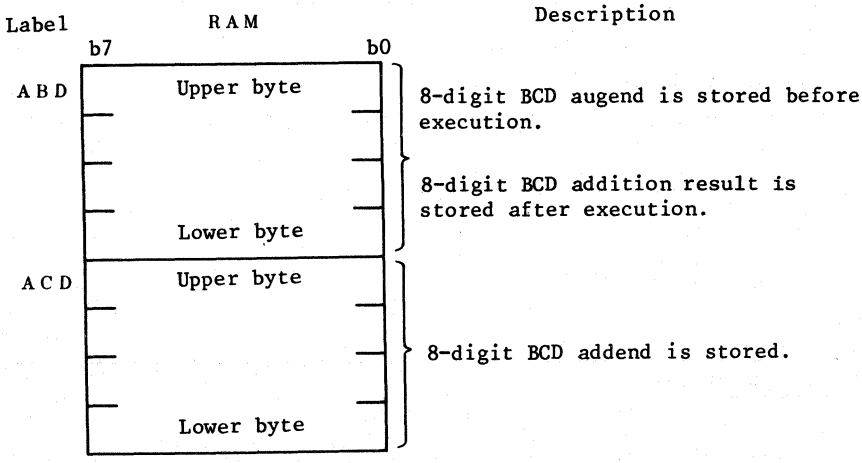


Fig. 3 Addition example when upper digit is not used

(3) RAM Description



16. ADDING 8-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	ADDD

DESCRIPTION

(4) Sample Application

ADDD subroutine is called after augend and addend are held.

```
WORK1   RMB     4       ----- Reserves memory byte for 8-digit BCD
                              augend.
WORK2   RMB     4       ----- Reserves memory byte for 8-digit BCD
                              addend.
WORK3   RMB     4       ----- Reserves memory byte for addition result.

WORK4   RMB     2       ----- Reserves memory byte for register contents
                              saving.
        :                     :
        :                     :
        :                     :
        STA     WORK4   }-----
        STX     WORK4+1 }     Saves register contents that will be
                              destroyed by executing ADDD.
        LDA     WORK1   ]
        STA     ABD     ]
        LDA     WORK+1  ]
        STA     ABD+1   }-----
        LDA     WORK1+2 ]     Stores 8-digit BCD added into entry
        STA     ABD+2   ]     argument (ABD).
        LDA     WORK1+3 ]
        STA     ABD+3   ]
        LDA     WORK2   ]
        STA     ACD     ]
        LDA     WORK2+1 ]
        STA     ACD+1   }-----
        LDA     WORK2+2 ]     stores 8-digit BCD addend into entry
        STA     ACD+2   ]     argument (ACD).
        LDA     WORK2+3 ]
        STA     ACD+3   ]
        JSR     ADDD    ----- Calls ADDD subroutine.
        BCS     OVER    ----- If there is carry in addition result,
                              branches to service carry routine.
        LDA     ABD     ]
        STA     WORK3   ]
        LDA     ABD+1   ]
        STA     WORK3+1 }-----
        LDA     ABD+2   ]     Stores addition result (return argument
        STA     WORK3+2 ]     (ABD)) in RAM.
        LDA     ABD+3   ]
        STA     WORK3+3 ]
        LDA     WORK4   }-----
        LDX     WORK4+1 }     Restores register.
        :                     :
        :                     :
        :                     :
OVER    Service routine in case
        of carry
        :
        :
        :
```

6

16. ADDING 8-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	ADDD

DESCRIPTION

(5) Basic Operation

(a) When more than 2 bytes are added, perform addition for each byte.

(b) IX is used to indicate augend and addend addresses and is also used to count number of addition.

(c) Clears bit C at first.

(d) Performs (Formula 1) on each byte of augend and addend using index addressing mode.

Augend + Addend + (bit C) → ACCA ------- (Formula 1)

Bit C is added in (Formula 1) because addition result of lower bytes generate carry.

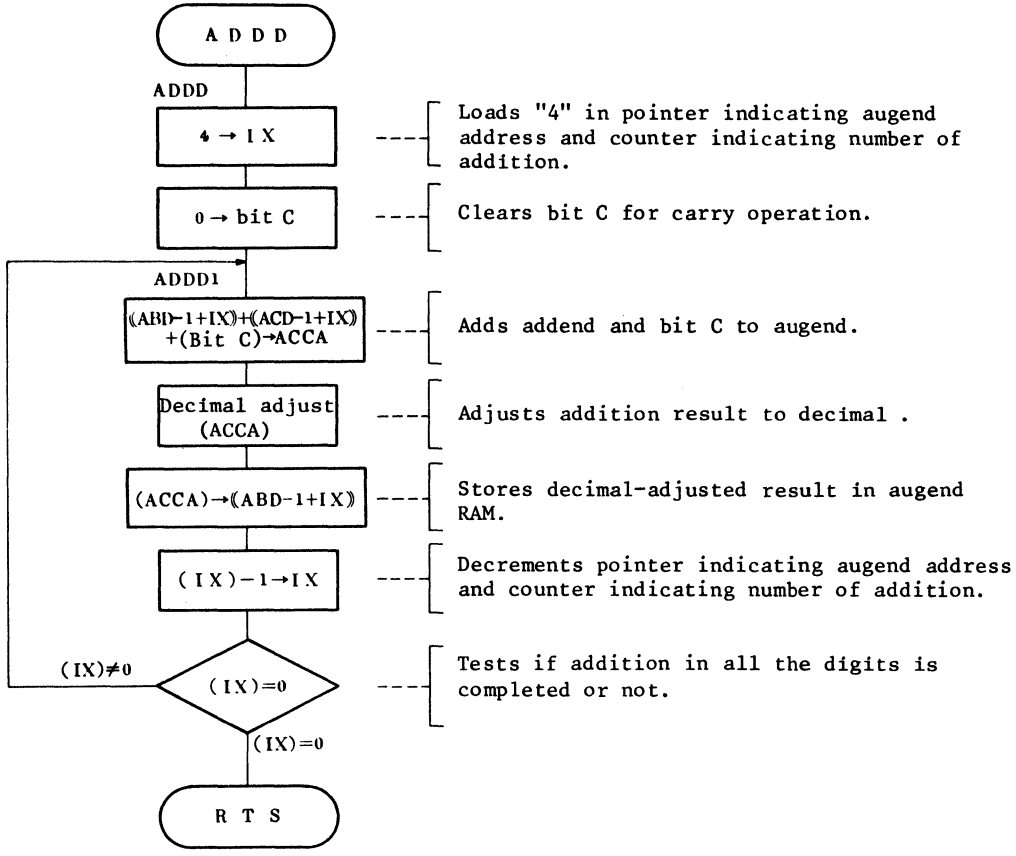
(e) Adjusts addition result of (d) to decimal value using decimal adjust (instruction DAA), and stores it in augend RAM (ABD).

(f) Decrements IX each time (d) and (e) are executed.

(g) Loops (c) to (f) until IX is "0".

16. ADDING 8-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	ADDD

FLOWCHART



6

16. ADDING 8-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	ADDD

PROGRAM LISTING

```
00001                   ************************************************
00002                   *                                              *
00003                   *         NAME : ADDING 8-DIGIT BCD  (ADDD)    *
00004                   *                                              *
00005                   ************************************************
00006                   *                                              *
00007                   *        ENTRY : ABD   (AUGEND)                *
00008                   *                ACD   (ADDEND)                *
00009                   *      RETURNS : ABD   (SUM)                   *
00010                   *                CARRY (C=0:TRUE,C=1:OVER FLOW) *
00011                   *                                              *
00012                   ************************************************
00013                   *
00014 0080                        ORG     $80
00015                   *
00016 0080 0004         ABD       RMB     4           Augend -> Sum
00017 0084 0004         ACD       RMB     4           Addend
00018                   *
00019 1000                        ORG     $1000
00020                   *
00021      1000         ADDD      EQU     *           Entry point
00022 1000 AE 04                  LDX     #4          Set ADDR pointer (addition counter)
00023 1002 98                     CLC                 Clear carry
00024 1003 E6 7F        ADDD1     LDA     ABD-1,X     Augend + addend
00025 1005 E9 83                  ADC     ACD-1,X
00026 1007 8D                     DAA                 Convert into BCD
00027 1008 E7 7F                  STA     ABD-1,X     Store in augend area
00028 100A 5A                     DEC     X           Decrement ADDR pointer
00029 100B 26 F6                  BNE     ADDD1       Loop until ADDR pointer = 0
00030 100D 81                     RTS
```

17. SUBTRACTING 8-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	SUBD

FUNCTION

(a) Performs subtraction of 8-digit BCD in RAM and stores result in 8-digit BCD in RAM.

(b) Utilizes unsigned integers in arguments.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Minuend	SUBEDS (RAM)	4
		Subtrahend	SUBERS (RAM)	4
	Returns	Subtraction result	SUBEDS (RAM)	4
		Borrow or no borrow	bit C (CCR)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
×	×

C	Z
↕	×

N	I
×	●

H
×

SPECIFICATIONS

ROM (Bytes)
25
RAM (Bytes)
8
Stack (Bytes)
0
No. of cycles
146
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

SUBEDS (RAM): Holds 8-digit BCD minuend. After SUBD execution, Contains subtraction result.

SUBERS (RAM): Holds 8-digit BCD Subtrahend.

Bit C (CCR) : Indicates whether borrow is generated or not after SUBD execution.

Bit C=0: Borrow is generated in subtraction result.

Bit C=1: No borrow is generated in subtraction result.

(See Fig. 2)

SPECIFICATIONS NOTES

17. SUBTRACTING 8-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	SUBD

DESCRIPTION

(b) Fig. 1 shows example of executing SUBD. If entry arguments are as shown in part ① of Fig. 1, subtraction result is stored in SUBEDS (RAM) as shown in part ② of Fig. 1.

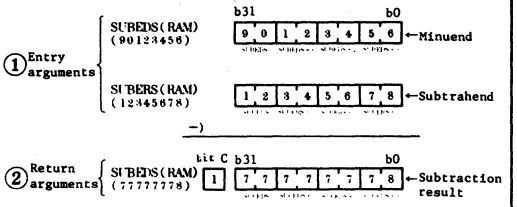


Fig. 1 Example of SUBD execution

(2) User Notes

(a) As shown in Fig. 3, when not using upper digits, hold "0" in the upper digits. If "0" is not held, subtraction is performed with undefined data held in the upper digits, and correct result cannot be obtained.

(b) After SUBD execution, minuend is destroyed because subtraction result is contained in SUBEDS (RAM). If minuend needs to be retained after SUBD execution, it should be saved in memory before execution.

(c) Hold BCD data as minuend and subtrahend. If data other than BCD is held, correct subtraction result cannot be obtained.

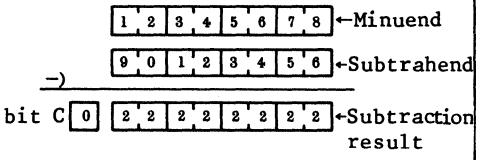


Fig. 2 Subtraction example when borrow is generated

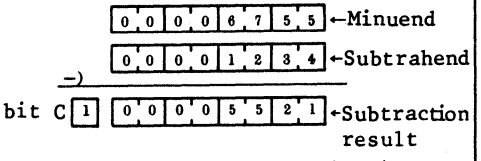
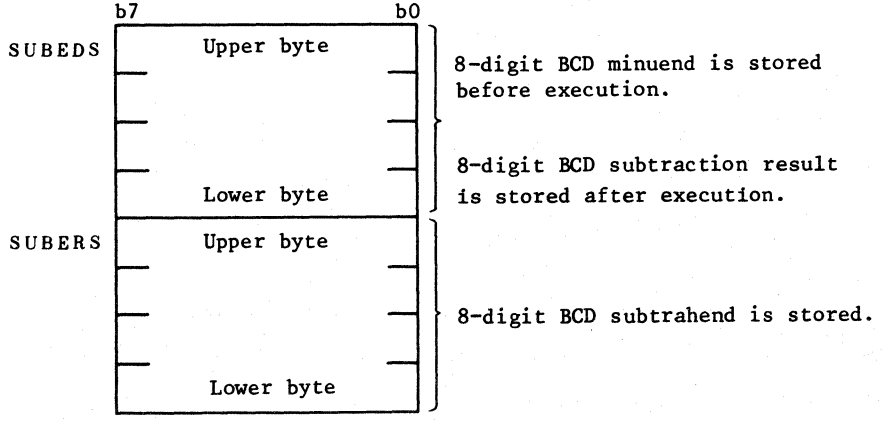


Fig. 3 Subtraction example when upper digit is not used

(3) RAM Description



17. SUBTRACTING 8-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	SUBD

DESCRIPTION

(4) Sample Application

SUBD subroutine is called after minuend and subtrahend are held.

```
WORK1   RMB     4       ----- Reserves memory byte for 8-digit BCD
                              minuend.
WORK2   RMB     4       ----- Reserves memory byte for 8-digit BCD
                              subtrahend.
WORK3   RMB     4       ----- Reserves memory byte for 8-digit BCD
                              subtraction result.
WORK4   RMB     2       ----- Reserves memory byte for register
                              contents saving.
        :
        :
        STA     WORK4   }
        STX     WORK4+1 }-----  Saves register contents that will be
        LDA     WORK1   }       destroyed by executing SHR.
        STA     SUBEDS  }
        LDA     WORK1+1 }
        STA     SUBEDS+1}-----  Stores 8-digit BCD minuend into entry
        LDA     WORK1+2 }       argument (SUBEDS).
        STA     SUBEDS+2}
        LDA     WORK1+3 }
        STA     SUBEDS+3}
        LDA     WORK2   }
        STA     SUBERS  }
        LDA     WORK2+1 }
        STA     SUBERS+1}-----  Stores 8-digit BCD subtrahend into entry
        LDA     WORK2+2 }       argument (SUBERS).
        STA     SUBERS+2}
        LDA     WORK2+3 }
        STA     SUBERS+3}
        JSR     SUBD    ------  Calls SUBD subroutine.
        BCC     OVER    ------  If borrow is generated in subtraction
                                result, branches to service borrow routine.
        LDA     SUBEDS  }
        STA     WORK3   }
        LDA     SUBEDS+1}
        STA     WORK3+1 }-----  Stores subtraction result (return argument
        LDA     SUBEDS+2}       (SUBEDS)) in RAM.
        STA     WORK3+2 }
        LDA     SUBEDS+3}
        STA     WORK3+3 }
        LDA     WORK4   }
        LDX     WORK4+1 }-----  Restores register.
        :
        :
OVER    Service routine
        in case of borrow
        :
        :
        :
```

17. SUBTRACTING 8-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	SUBD

DESCRIPTION

(5) Basic Operation

(a) Subtraction of BCD can be performed by (Formula 1) and (Formula 2).

Minuend-Subtrahend=Minuend+10's complement of subtrahend .....(Formula 1)

10's complement of minuend=$99-Subtrahend+1 .....(Formula 2)

(b) (Formula 1) and (Formula 2) are described as follows:

(i) Takes 10's complements of 8-digit subtrahend by (Formula 2) and stores them in SUBERS (RAM).

(ii) IX is used to indicate minuend and subtrahend, and is also used to count number of subtraction.

(iii) Performs (Formula 3) on every byte from LSB of 10's complement of minuend and subtrahend using index addressing mode.

Minuend+10's complement of subtrahend+(bit C) → ACCA ... (Formula 3)

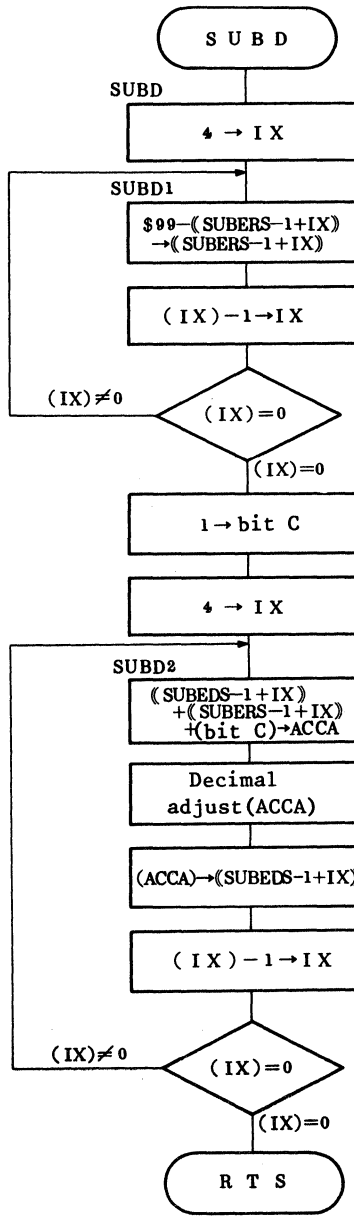
(iv) Adjusts subtraction result of (iii) to decimal value using decimal adjust (instruction DAA) and stores it in minuend RAM, (SUBERS).

(v) Decrements IX.

(vi) Loops (iii) to (iv) until IX is "0".

17. SUBTRACTING 8-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	SUBD

FLOWCHART



Takes 10's complement of subtrahend, and stores it in subtrahend RAM. However, only ($99-subtrahend) is calculated in this part. Adds "1" in the next step by setting bit C. (See Formula 2 in "(5) Basic Operation")

Loads "4" in pointer indicating minuend address and counter indicating number of subtraction.

Adds 10's complement of subtrahend and bit C to minuend.

Adjusts addition result to decimal.

Stores decimal adjusted result in minuend RAM.

Decrements pointer indicating minuend address and counter indicating number of subtraction.

Tests if subtraction in all the digits is completed or not.

6

17. SUBTRACTING 8-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	SUBD

PROGRAM LISTING

```
00001                   ************************************************
00002                   *                                              *
00003                   *     NAME : SUBTRACTING 8-DIGIT BCD  (SUBD)   *
00004                   *                                              *
00005                   ************************************************
00006                   *                                              *
00007                   *        ENTRY : SUBEDS (MINUEND)              *
00008                   *                SUBERS (SUBTRAHEND)           *
00009                   *      RETURNS : SUBEDS (RESIDUAL)             *
00010                   *                CARRY  (C=1:TRUE,C=0:BORROW)  *
00011                   *                                              *
00012                   ************************************************
00013                   *
00014 0080                        ORG    $80
00015                   *
00016 0080 0004         SUBEDS RMB    4          Minuend -> Residual
00017 0084 0004         SUBERS RMB    4          Subtrahend
00018                   *
00019 1000                        ORG    $1000
00020                   *
00021      1000         SUBD   EQU    *          Entry point
00022 1000 AE 04                  LDX    #4         Set subtraction counter
00023 1002 A6 99        SUBD1  LDA    #$99       99999999-Subtraend
00024 1004 E0 83                  SUB    SUBERS-1,X  -> SUBERS
00025 1006 E7 83                  STA    SUBERS-1,X
00026 1008 5A                     DEC    X
00027 1009 26 F7                  BNE    SUBD1
00028 100B 99                     SEC                Set carry bit
00029 100C AE 04                  LDX    #4         Set ADDR pointer(subtraction counter)
00030 100E E6 7F        SUBD2  LDA    SUBEDS-1,X  Minuend + negative of subtrahend
00031 1010 E9 83                  ADC    SUBERS-1,X
00032 1012 8D                     DAA                Convert into BCD
00033 1013 E7 7F                  STA    SUBEDS-1,X  Store in minuend area
00034 1015 5A                     DEC    X          Decrement ADDR pointer
00035 1016 26 F6                  BNE    SUBD2      Loop until ADDR pointer = 0
00036 1018 81                     RTS
```

18. 16-BIT SQUARE ROOT	MCU/MPU	HD6305 FAMILY	LABEL	SQRT

FUNCTION

(a) Obtains square root of 16-bit binary data in RAM and stores result in RAM.
(b) Utilizes unsigned integers in argument.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Data to be squared	SQRD (RAM)	2
	Returns	Square root	SANS+1 (RAM)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	×

C	Z
×	×
N	I
×	●
H	
×	

SPECIFICATIONS

ROM (Bytes)
71
RAM (Bytes)
6
Stack (Bytes)
0
No. of cycles
861
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

SQRD (RAM) : Holds 16-bit binary data to be squared.

SANS+1 (RAM): Contains 16-bit binary square root.

(b) Fig. 1 shows example of SQRT execution. If entry argument is as shown in part ① of Fig. 1, square contained in SANS+1 (RAM) as shown in part ② of Fig. 1.

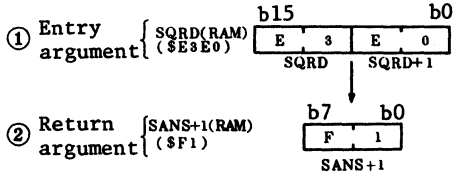


Fig. 1 Example of SQRT execution

SPECIFICATIONS NOTES

"No. of cycles" represents the number of cycles needed to get square root of $FFFF.

DESCRIPTION

(2) User Notes

(a) When not using upper byte as shown in Fig. 2, holds "0" in the upper byte. If "0" is not held, square root is obtained with undefined data Held in the upper byte, and correct result cannot be obtained.

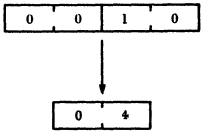


Fig. 2 Example when not upper bytes are not used

(b) Values to the right of the binary point are truncated.

(3) RAM Description

Label	RAM	Description
	b7 b0	
SQRD	Upper byte Lower byte	16-bit binary data is stored.
SANS	Upper byte	Work area is reserved to store square root before execution. 8-bit binary square root is stored in SANS+1, and "0" is stored in SANS after execution.
(SANS+1)	Lower byte	
SWORK	Upper byte Lower byte	Work area is reserved to operate on upper 2 bits of data to be squared.

18. 16-BIT SQUARE ROOT	MCU/MPU	HD6305 FAMILY	LABEL	SQRT

DESCRIPTION

(4) Sample Application

SQRT subroutine is called after data to be squared is held.

```
WORK1   RMB     2       ----- Reserves memory byte for binary
                              data to be squared.

WORK2   RMB     1       ----- Reserves memory byte for 16-bit binary
                              square root.
WORK3   RMB     2       ----- Reserves memory byte for register contents
                              saving.
        :
        :
        :
        STA     WORK3   }_____ Saves register contents that will be
        STX     WORK3+1 }      destroyed by SQRT execution.
        LDA     WORK1   }
        STA     SQRD    }_____ Stores data to be squared into entry
        LDA     WORK1+1 }      argument (SQRD).
        STA     SQRD+1  }
        JSR     SQRT    ----- Calls SQRT subroutine.
        LDA     SANS+1  }      Stores 16-bit binary square root (return
        STA     WORK2   }_____ argument (SANS+1)) in RAM.
        LDA     WORK3   }
        LDX     WORK3+1 }_____ Restores register.
        :
        :
        :
```

DESCRIPTION

(5) Basic Operation

(a) Fig. 3 shows calculation to obtain square root of 16-bit binary data, $05 of hexadecimal $22 data to be squared.

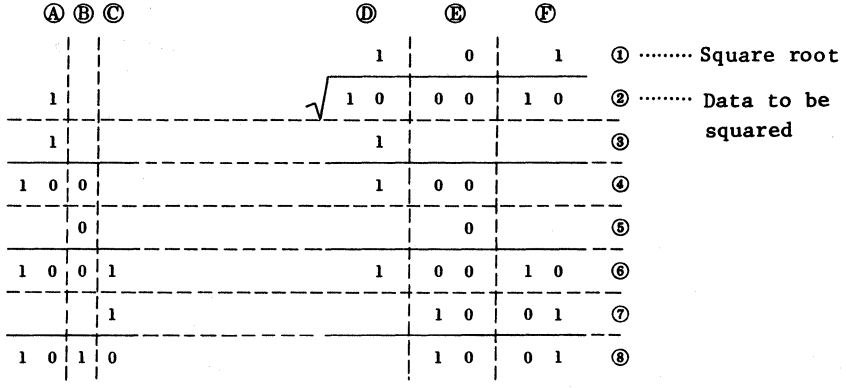


Fig. 3 Calculating a square root

(b) The calculation in Fig. 3, is explained as follows:

(i) Clears square root area, SANS:SANS+1 and work area SWORK:SWORK+1.

(ii) Rotates SQRD:SQRD+1 and SWORK:SWORK+1 2 bits left to fetch upper 2 bits of data to be squared, and sets upper 2 bits of data to be squared in SWORK:SWORK+1. (Fig. 3 Ⓓ - ②)

(iii) Sets "1" in 2-byte area, SANS:SANS+1 from RAM address shown in SANS. (Fig. 3 Ⓐ - ②)

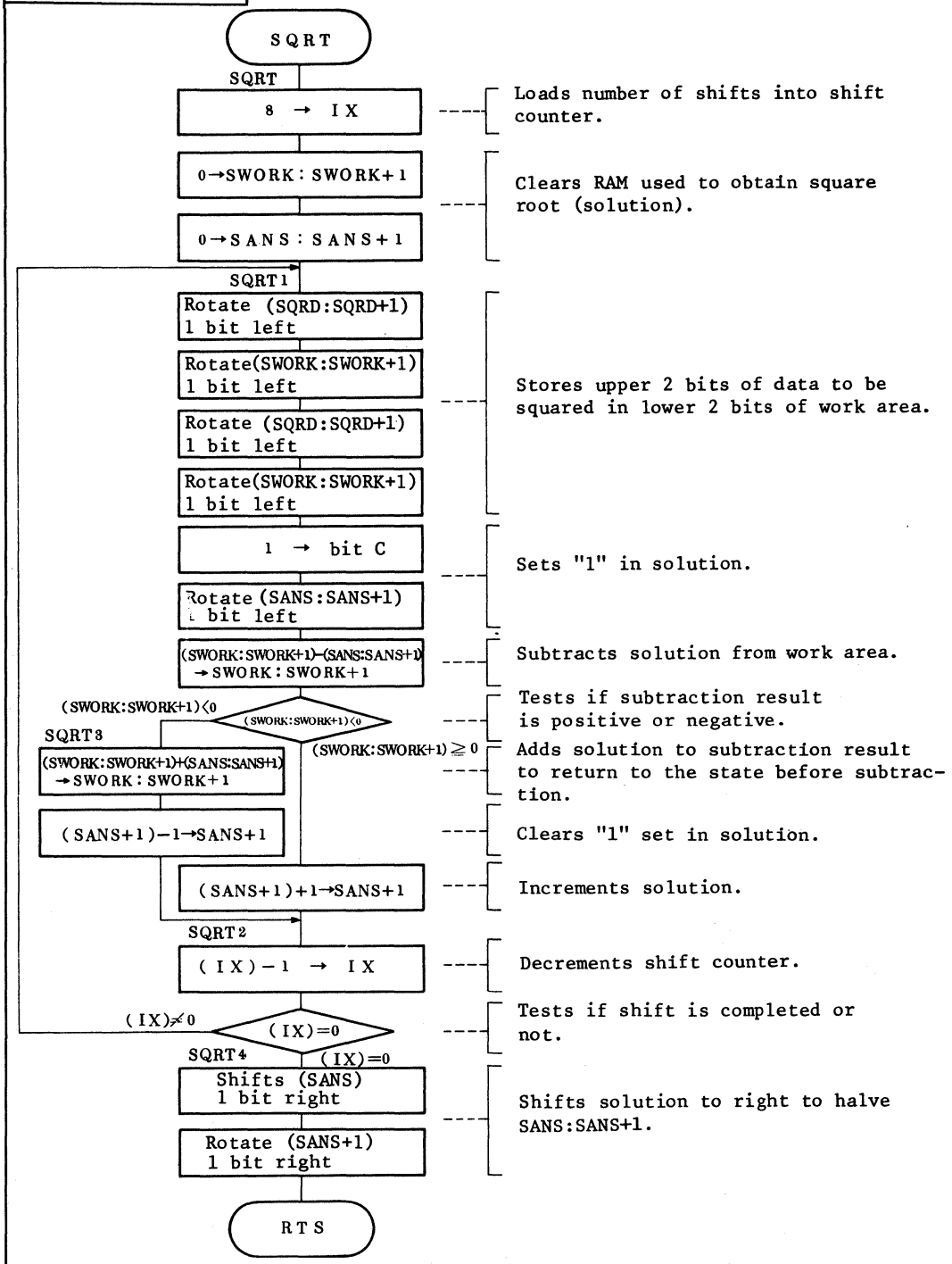
(iv) Subtracts SANS:SANS+1 from SWORK:SWORK+1, and stores result in SWORK:SWORK+1. (Fig. 3 Ⓓ - ② , ③ , ④)

(v) When result is positive, increments SANS+1. (Fig. 3 Ⓐ - ④)
When result is negative, decrements SANS+1, and adds SANS: SANS+1 to SWORK:SWORK+1. (Fig. 3 Ⓓ , Ⓔ - ⑥)

(c) In SQRT, loops (ii) to (v) 8 times and then shifts SANS:SANS+1 1 bit. right to halve SANS: SANS+1. (Fig. 3 Ⓐ , Ⓑ - ⑧ is square root).

18. 16-BIT SQUARE ROOT	MCU/MPU	HD6305 FAMILY	LABEL	SQRT

FLOWCHART



18. 16-BIT SQUARE ROOT	MCU/MPU	HD6305 FAMILY	LABEL	SQRT

PROGRAM LISTING

```
00001                    ***********************************************
00002                    *                                             *
00003                    *       NAME : 16-BIT  SQUARE ROOT (SQRT)     *
00004                    *                                             *
00005                    ***********************************************
00006                    *                                             *
00007                    *           ENTRY : SQRD (16-BIT  BINARY DATA) *
00008                    *         RETURNS : SANS (SQUARE ROOT)        *
00009                    *                                             *
00010                    ***********************************************
00011                    *
00012 0080               *       ORG     $80
00013                    *
00014 0080 0002          SQRD    RMB     2         Input 16-bit binary data
00015 0082 0002          SANS    RMB     2         Square root
00016 0084 0002          SWORK   RMB     2         Work area
00017                    *
00018 1000               *       ORG     $1000
00019                    *
00020      1000          SQRT    EQU     *         Entry point
00021 1000 AE 08                 LDX     #8        Set shift counter
00022 1002 3F 84                 CLR     SWORK     Clear Work area
00023 1004 3F 85                 CLR     SWORK+1
00024 1006 3F 82                 CLR     SANS      Clear square root area
00025 1008 3F 83                 CLR     SANS+1
00026 100A 39 81         SQRT1   ROL     SQRD+1    Shift upper 2 bits of SQRD area
00027 100C 39 80                 ROL     SQRD      -and set lower 2 bits of swork
00028 100E 39 85                 ROL     SWORK+1
00029 1010 39 84                 ROL     SWORK
00030 1012 39 81                 ROL     SQRD+1
00031 1014 39 80                 ROL     SQRD
00032 1016 39 85                 ROL     SWORK+1
00033 1018 39 84                 ROL     SWORK
00034 101A 99                    SEC               Set LSB of SANS area
00035 101B 39 83                 ROL     SANS+1
00036 101D 39 82                 ROL     SANS
00037 101F B6 85                 LDA     SWORK+1   Swork - SANS -> Swork
00038 1021 B0 83                 SUB     SANS+1
00039 1023 B7 85                 STA     SWORK+1
00040 1025 B6 84                 LDA     SWORK
00041 1027 B2 82                 SBC     SANS
00042 1029 B7 84                 STA     SWORK
00043 102B 25 0A                 BCS     SQRT3     Branch if minus
00044 102D 3C 83                 INC     SANS+1    SANS + 1 -> SANS
00045 102F 5A            SQRT2   DEC     X         Decrement shift counter
00046 1030 26 D8                 BNE     SQRT1     Loop until shift counter = 0
00047 1032 37 82                 ASR     SANS      Halve SANS:SANS+1
00048 1034 36 83                 ROR     SANS+1
00049 1036 81                    RTS
00050 1037 B6 85         SQRT3   LDA     SWORK+1   Add again
00051 1039 BB 83                 ADD     SANS+1
00052 103B B7 85                 STA     SWORK+1
00053 103D B6 84                 LDA     SWORK
00054 103F B9 82                 ADC     SANS
00055 1041 B7 84                 STA     SWORK
00056 1043 3A 83                 DEC     SANS+1    SANS -1 -> SANS
00057 1045 20 E8                 BRA     SQRT2     Branch SQRT2
```

19. CONVERTING 2-BYTE HEXA-DECIMALS INTO 5-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	HEX

FUNCTION

(a) Converts 2-byte hexadecimals data in RAM into 5-digit BCD data and stores result in RAM.

(b) Utilizes unsigned integers in argument.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	2-byte hexadecimal	HEXD (RAM)	2
	Returns	5-figure BCD	DECD (RAM)	3

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
×	×

C	Z
×	×
N	I
×	●
H	
×	

SPECIFICATIONS

ROM (Bytes)
31
RAM (Bytes)
6
Stack (Bytes)
0
No. of cycles
1257
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

HEXD (RAM) : Holds 2-byte hexadecimals to be converted into BCD.

DECD (RAM) : Contains 5-gidit BCD.

(b) Fig. 1 shows example of HEX execution. If argument is as shown in part ① of Fig. 1, 5-digit BCD, in this case "52734" is held in DECD (RAM) (see part ② of Fig. 1).

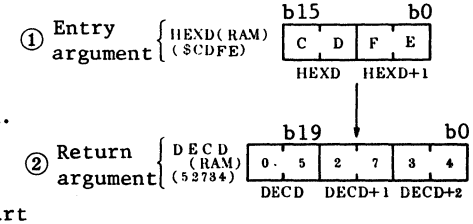


Fig. 1 Example of HEX EXECUTION

(2) User Notes

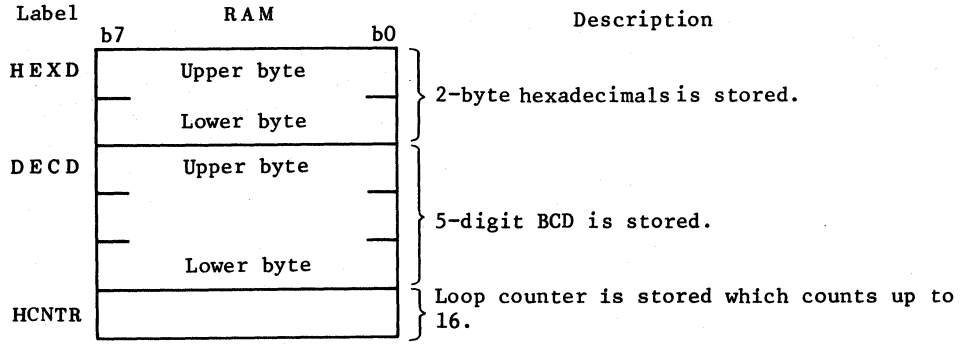
"0" is held as MSD (10^5) of return argument.

SPECIFICATIONS NOTES

19. CONVERTING 2-BYTE HEXA-DECIMALS INTO 5-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	HEX

DESCRIPTION

(3) RAM Description



(4) Sample Application

HEX subroutine is called after 2-byte hexadecimals is held.

```
WORK1   RMB     2       ----- Reserves memory byte for 2-byte
                              hexadecimals.
WORK2   RMB     3       ----- Reserves memory byte for 5-digit BCD.

WORK3   RMB     2       ----- Reserves memory byte for register contents
        :                     saving.
        :
        :
        STA     WORK3   }-----
        STX     WORK3+1 }     Saves register contents that will be
                              destroyed by HEX execution.
        LDA     WORK1   }
        STA     HEXD    }-----
        LDA     WORK1+1 }     Stores 2-byte hexadecimals into entry
        STA     HEXD+1  }     argument (HEX).
        JSR     HEX     ----- Calls HEX subroutine.
        LDA     DECD    }
        STA     WORK2   }
        LDA     DECD+1  }
        STA     WORK2+1 }-----
        LDA     DECD+2  }     Stores 5-digit BCD (return argument (DECD))
        STA     WORK2+2 }     in RAM.
        LDA     WORK3   }-----
        LDX     WORK3+1 }     Restores register.
        :
        :
```

DESCRIPTION

(5) Basic Operation

(a) 4-bit binary (ABCD) construction is shown in Fig. 2 (Formula 1, Formula 2).

$$ABCD = A \times 2^3 + B \times 2^2 + C \times 2^1 + D \times 2^0 \quad \cdots\cdots \text{(Formula 1)}$$

$$= \{\{(A \times 2) + B\} \times 2 + C\} \times 2 + D \quad \cdots\cdots \text{(Formula 2)}$$

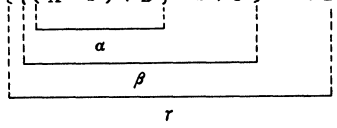


Fig. 2 4-bit binary (ABCD)

(b) 2-byte hexadecimals can be converted into 5-digit BCD by calculating (Formula 2). First, calculate $\alpha = (A \times 2) + B$, and adjust result into decimal. Next, the same calculation is done for

$\beta = (\alpha \times 2) + C$, and

$\gamma = (\beta \times 2) + D$, both of which are adjusted into decimals.

(c) HEX uses HEXD and DECD (RAM) for $\alpha = (A \times 2) + B$

(i) Shifts 2-byte hexadecimals (HEXD) 1 bit left and rotates MSB to bit C.

(ii) Loads 5-digit BCD (DECD) into ACCA and calculates (ACCA) + (DECD) + (bit C) → (ACCA), where $\alpha = (A \times 2) + B$ is executed.

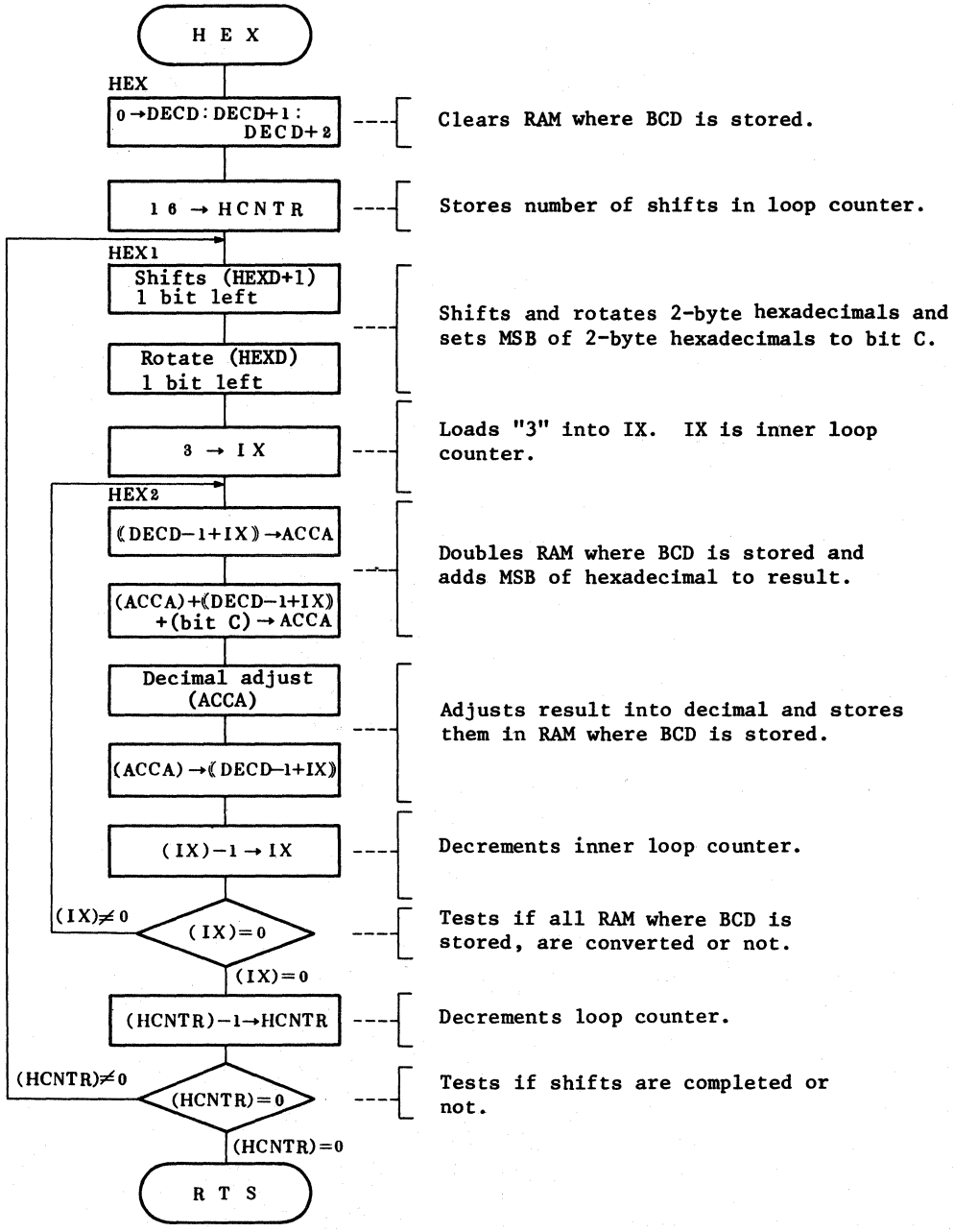
(iii) Adjusts result into decimal and stores them in DECD.

(iv) Loops (i) to (iii) sixteen times to convert 2-byte hexadecimals into 5-digit BCD.

6

19. CONVERTING 2-BYTE HEXA-DECIMALS INTO 5-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	HEX

FLOWCHART



19. CONVERTING 2-BYTE HEXA-DECIMALS INTO 5-DIGIT BCD	MCU/MPU	HD6305 FAMILY	LABEL	HEX

PROGRAM LISTING

```
00001                   ************************************************
00002                   *                                              *
00003                   *    NAME : CONVERTING 2-BYTE HEXADECIMALS     *
00004                   *           INTO 5-DIGIT BCD        (HEX)     *
00005                   *                                              *
00006                   ************************************************
00007                   *                                              *
00008                   *         ENTRY : HEXD (2-BYTE HEXADECIMALS)   *
00009                   *       RETURNS : DECD (5-DIGIT BCD)           *
00010                   *                                              *
00011                   ************************************************
00012                   *
00013 0080              *       ORG     $80
00014                   *
00015 0080 0002         HEXD    RMB     2       2-byte hexadecimals
00016 0082 0003         DECD    RMB     3       5-digit BCD
00017 0085 0001         HCNTR   RMB     1       Subtraction counter
00018                   *
00019 1000              *       ORG     $1000
00020                   *
00021      1000         HEX     EQU     *       Entry point
00022 1000 3F 82                CLR     DECD    Clear 5-digit BCD area
00023 1002 3F 83                CLR     DECD+1
00024 1004 3F 84                CLR     DECD+2
00025 1006 A6 10                LDA     #16     Set shift counter
00026 1008 B7 85                STA     HCNTR
00027 100A 38 81        HEX1    ASL     HEXD+1  Shift MSB of HEXD to carry
00028 100C 39 80                ROL     HEXD
00029 100E AE 03                LDX     #3      Set ADDR pointer (addition counter)
00030 1010 E6 81        HEX2    LDA     DECD-1,X DECD * 2 + C -> ACCA
00031 1012 E9 81                ADC     DECD-1,X
00032 1014 8D                   DAA             Convert into BCD
00033 1015 E7 81                STA     DECD-1,X Store 5-digit BCD area
00034 1017 5A                   DEC     X       Decrement ADDR pointer
00035 1018 26 F6                BNE     HEX2    Loop until ADDR pointer = 0
00036 101A 3A 85                DEC     HCNTR   Decrement shift counter
00037 101C 26 EC                BNE     HEX1    Loop until shift counter = 0
00038 101E 81                   RTS
```

6

20. CONVERTING 5-DIGIT BCD INTO 2-BYTE HEXADECIMALS	MCU/MPU	HD6305 FAMILY	LABEL	BCD

FUNCTION

(a) Converts 5-digit BCD data in RAM into 2-byte hexadecimals and stores result in RAM.

(b) Utilizes unsigned integers in argument.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	5-digit decimals	DEC (RAM)	3
	Returns	2-byte hexadecimals	HDATA (RAM)	2

CHANGES IN CPU REGISTERS AND FLAGS

●: Not affected

×: Undefined

↕: Result

ACCA	IX
×	×

C	Z
×	×
N	I
×	●
H	
×	

SPECIFICATIONS

ROM (Bytes)
70
RAM (Bytes)
8
Stack (Bytes)
0
No. of cycles
440
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

DEC (RAM) : Holds 5-digit BCD to be converted into hexadecimal.

HDATA (RAM): Contains 2-byte hexadecimals.

(b) Fig. 1 shows example of BCD execution. If entry argument is as shown in part ① of Fig. 1, 2-byte hexadecimals is contained in HDATA (RAM) as shown in part ② of Fig. 1.

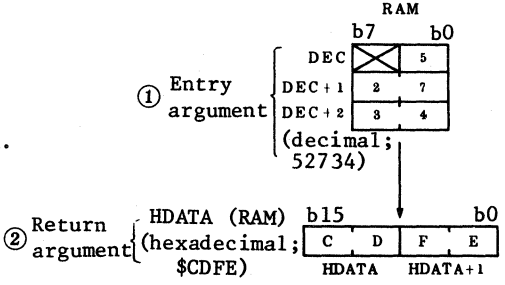


Fig. 1 Example of BCD execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to convert 59999 into hexadecimal.

20. CONVERTING 5-DIGIT BCD INTO 2-BYTE HEXADECIMALS	MCU/MPU	HD6305 FAMILY	LABEL	BCD

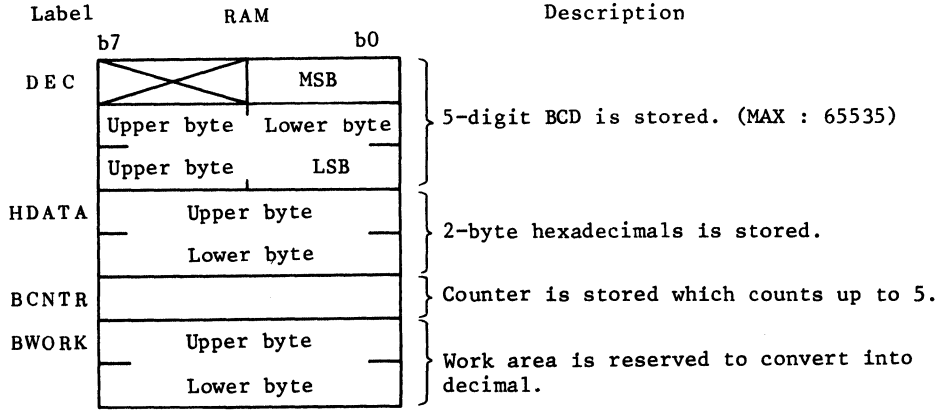
DESCRIPTION

(2) User Notes

(a) If 65536 or more is held as 5-digit BCD, correct result cannot be obtained.

(b) Holds BCD in entry argument. If other data is held, correct result cannot be obtained.

(3) RAM Description



(4) Sample Application

BCD subroutine is called after 5-digit decimals is held.

```
WORK1   RMB     3       ----- Reserves memory byte for 5-digit BCD.

WORK2   RMB     2       ----- Reserves memory byte for 2-byte
                              hexadecimals.
WORK3   RMB     2       ----- Reserves memory byte for register contents
                              saving.
          :
          :
          :
        STA     WORK3   }_____ Saves register contents that will be
        STX     WORK3+1 }      destroyed by BCD execution.
        LDA     WORK1   }
        STA     DEC     }
        LDA     WORK1+1 }----- Stores 5-digit BCD into entry argument
        STA     DEC+1   }      (DEC).
        LDA     WORK1+2 }
        STA     DEC+2   }
        JSR     BCD     ----- Calls BCD subroutine.
```

6

20. CONVERTING 5-DIGIT BCD INTO 2-BYTE HEXADECIMALS	MCU/MPU	HD6305 FAMILY	LABEL	BCD

DESCRIPTION

```
LDA     HDATA
STA     WORK2       }------ Stores 2-byte hexadecimals.
LDA     HDATA+1
STA     WORK2+1
LDA     WORK3
LDX     WORK3+1     }------ Restores register.
  .
  .
  .
```

(5) Basic Operation

(a) BCD consists of 2 operations; one is to fetch 5-digit BCD, digit by digit as shown in Fig. 2, the other is to convert fetched data into hexadecimal by 4 bits units.

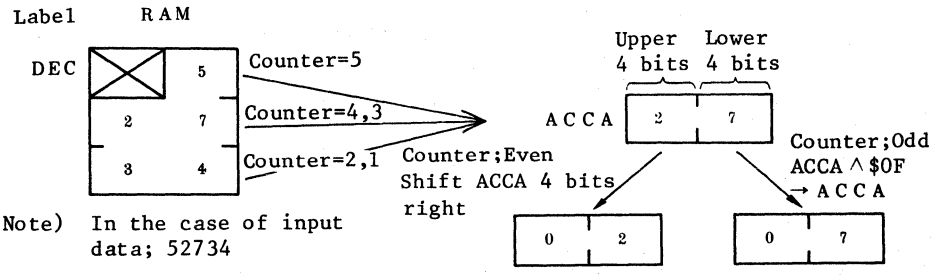


Fig. 2 Dividing 1-byte of RAM data into two parts

(b) Fetching (see Fig. 2)

(i) IX is used to indicate memory address of input 5-digit BCD.stores "5" in counter to convert 5 digits.

(ii) Loads input data into ACCA in order from MSB using index addressing mode and selects upper or lower 4 bits.

(iii) Decrements counter every time 1 digit is loaded into ACCA.

(iv) Loops (ii),(iii) untill counter is "0".

(v) During (ii),(iii) CPU checks whether counter is an even or an odd number. If odd, AND ACCA to $0F and fetch lower 4 bits. If even, shift ACCA 4 bits right and fetch upper 4 bits.

DESCRIPTION

(c) Converting BCD into hexadecimal

(i) 4-digit BCD construction is shown in Fig. 3 (Formula 1, Formula 2).

$$ABCD = A \times 10^3 + B \times 10^2 + C \times 10^1 + D \times 10^0 \quad \cdots\cdots \text{(Formula 1)}$$

$$= [\{(A \times 10) + B\} \times 10 + C] \times 10 + D \quad \cdots\cdots \text{(Formula 2)}$$

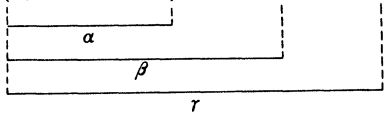


Fig. 3 4-digit BCD (ABCD)

(ii) 5-digit BCD can be converted into hexadecimal as follows. First, calculate $\alpha = (A \times 10) + B$ to determine in Fig. 3 (Formula 2). Then calculate $\beta = (\alpha \times 10) + C$ and $\gamma = (\beta \times 10) + D$ to determine.

(iii) Calculation of A × 10 is shown in (Formula 3, Formula 4);

$$A \times 10 = A \times (2 + 8) \text{ ---------- (Formula 3)}$$

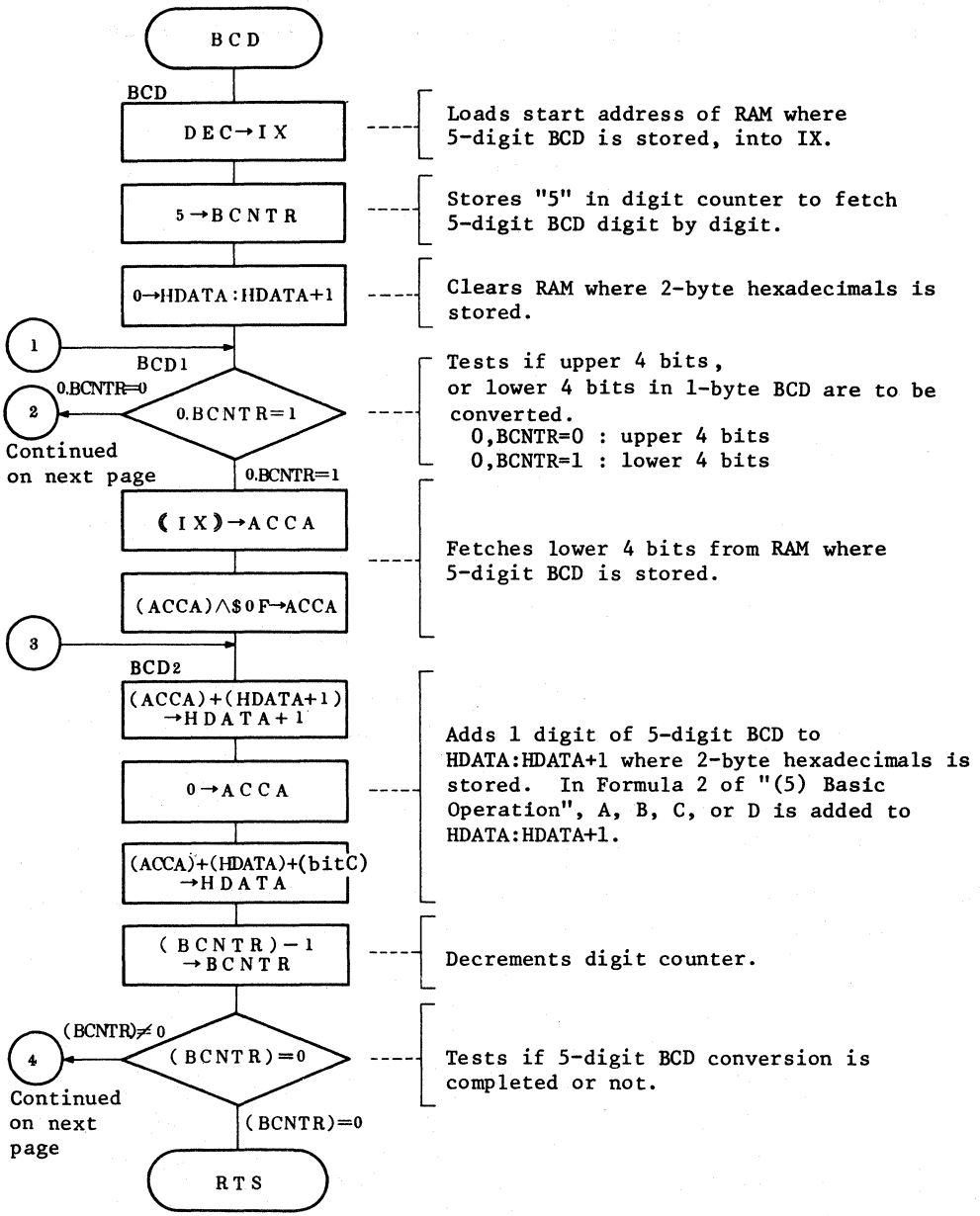
$$= A \times 2\,(1 + 2^2) \text{ ---------- (Formula 4)}$$

(iv) When calculating (Formula 4), BCD uses RAM of HDATA:HDATA+1 and BWORK:BWORK+1. That is, store A in HDATA:HDATA+1 (Formula 4), shift HDATA:HDATA+1 left 1 bit and store result in BWORK:BWORK+1. Next, shift HDATA:HDATA+1 left 2 bits and add HDATA:HDATA+1 to BWORK:BWORK+1 to determine A × 10.

(d) Loop (b) and (c) five times to complete conversion of 5-digit BCD into 16-bit binary number.

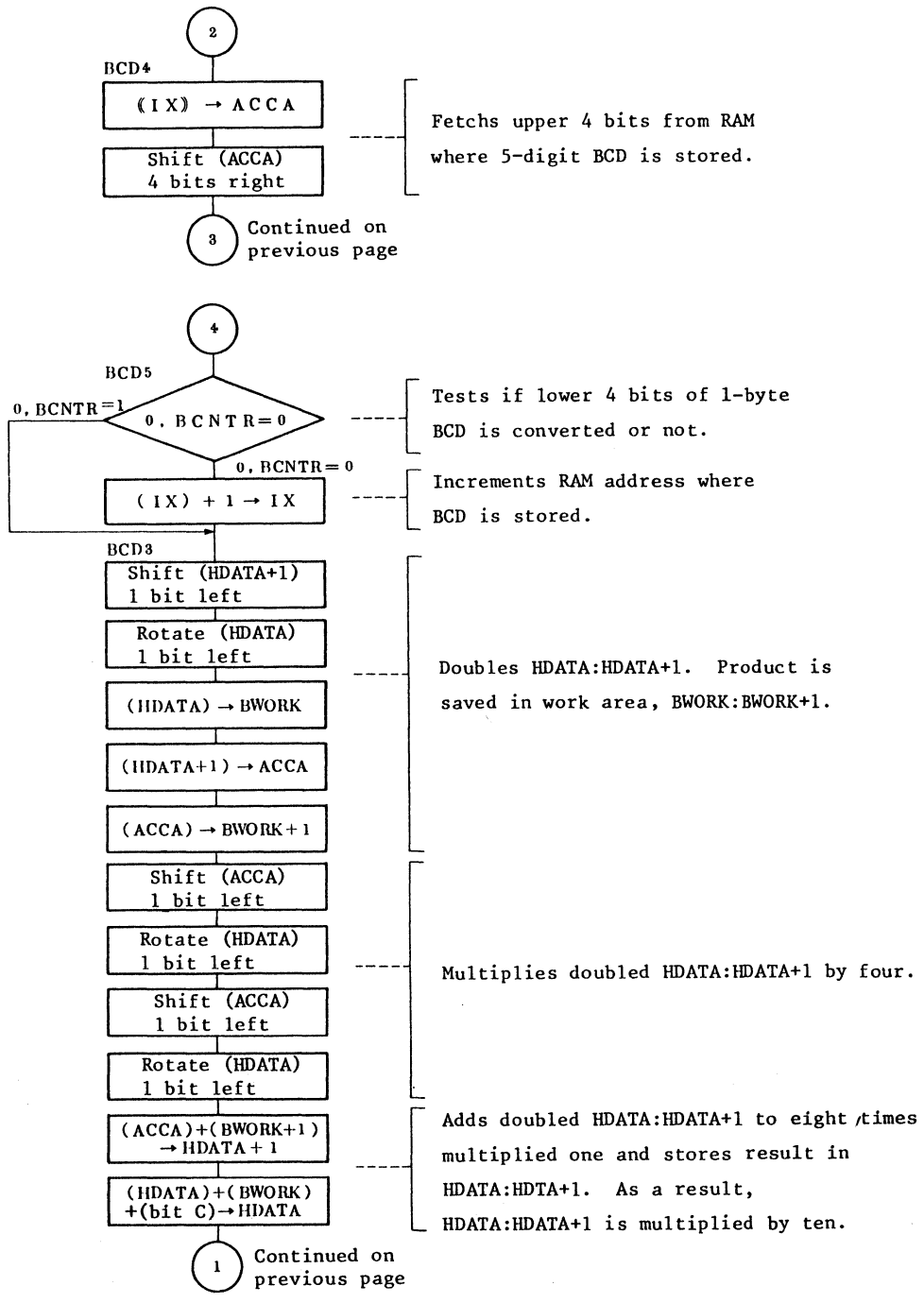
20. CONVERTING 5-DIGIT BCD INTO 2-BYTE HEXADECIMALS	MCU/MPU	HD6305 FAMILY	LABEL	BCD

FLOWCHART



20. CONVERTING 5-DIGIT BCD INTO 2-BYTE HEXADECIMALS	MCU/MPU	HD6305 FAMILY	LABEL	BCD

FLOWCHART



20. CONVERTING 5-DIGIT BCD INTO 2-BYTE HEXADECIMALS	MCU/MPU	HD6305 FAMILY	LABEL	BCD

PROGRAM LISTING

```
00001                    ***********************************************
00002                    *                                             *
00003                    *     NAME : CONVERTING 5-DIGIT BCD           *
00004                    *            INTO 2-BYTE HEXADECIMALS  (BCD)  *
00005                    *                                             *
00006                    ***********************************************
00007                    *                                             *
00008                    *       ENTRY : DEC    (5-DIGIT BCD)          *
00009                    *     RETURNS : HDATA  (2-BYTE HEXADECIMALS)  *
00010                    *                                             *
00011                    ***********************************************
00012                    *
00013 0080                        ORG    $80
00014                    *
00015 0080 0003          DEC      RMB    3          5-digit BCD
00016 0083 0002          HDATA    RMB    2          2-byte hexadecimals
00017 0085 0001          BCNTR    RMB    1          Digit counter
00018 0086 0002          BWORK    RMB    2          Work area
00019                    *
00020 1000                        ORG    $1000
00021                    *
00022      1000          BCD      EQU    *          Entry point
00023 1000 AE 80                  LDX    #DEC       Load BCD DATA ADDR
00024 1002 A6 05                  LDA    #5         Set figure counter
00025 1004 B7 85                  STA    BCNTR
00026 1006 3F 83                  CLR    HDATA      Clear Hex area
00027 1008 3F 84                  CLR    HDATA+1
00028 100A 01 85 11 BCD1          BRCLR  0,BCNTR,BCD4  Branch if bit0 = 0
00029 100D F6                     LDA    0,X
00030 100E A4 0F                  AND    #$F        Set lower 4 bits of BCD data
00031 1010 BB 84          BCD2    ADD    HDATA+1    ACCA + HDATA+1 -> HDATA+1
00032 1012 B7 84                  STA    HDATA+1
00033 1014 4F                     CLR    A          Clear ACCA
00034 1015 B9 83                  ADC    HDATA      ACCA(0) + HDATA + C->HDATA
00035 1017 B7 83                  STA    HDATA
00036 1019 3A 85                  DEC    BCNTR      Decrement digit counter
00037 101B 26 08                  BNE    BCD5       Loop until figure counter = 0
00038 101D 81                     RTS
00039 101E F6             BCD4    LDA    0,X        Set upper 4 bits of BCD data
00040 101F 44                     LSR    A
00041 1020 44                     LSR    A
00042 1021 44                     LSR    A
00043 1022 44                     LSR    A
00044 1023 20 EB                  BRA    BCD2       Branch BCD2
00045 1025 00 85 01 BCD5          BRSET  0,BCNTR,BCD3  Branch if bit0 = 1
00046 1028 5C                     INC    X          Increment BCD data ADDR
00047 1029 38 84          BCD3    ASL    HDATA+1    HDATA:HDATA+1*2->BWORK:BWORK+1
00048 102B 39 83                  ROL    HDATA
00049 102D B6 83                  LDA    HDATA
00050 102F B7 86                  STA    BWORK
00051 1031 B6 84                  LDA    HDATA+1
00052 1033 B7 87                  STA    BWORK+1
00053 1035 48                     ASL    A          HDATA:HDATA+1*4->HDATA:HDATA+1
00054 1036 39 83                  ROL    HDATA
00055 1038 48                     ASL    A
00056 1039 39 83                  ROL    HDATA
00057 103B BB 87                  ADD    BWORK+1    HDATA+1 + BWORK+1->HDATA+1
00058 103D B7 84                  STA    HDATA+1
00059 103F B6 86                  LDA    BWORK
00060 1041 B9 83                  ADC    HDATA      HDATA + BWORK + C -> HDATA
00061 1043 B7 83                  STA    HDATA
00062 1045 20 C3                  BRA    BCD1       Branch BCD1
```

21. SORTING	MCU/MPU	HD6305 FAMILY	LABEL	SORT

FUNCTIONS

(a) Sorts unsigned byte oriented data in RAM in descending order.

(b) Permits number of bytes to be sorted to be freely selected.

(c) Utilizes unsigned integers in arguments.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	No. of bytes to be sorted	ACCA	1
		Starting address of data to be sorted	IX	1
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected

× : Undefined

↕ : Result

ACCA	IX
×	×

C	Z
×	×
N	I
×	●
H	
×	

SPECIFICATIONS

ROM (Bytes)
36
RAM (Bytes)
10
Stack (Bytes)
0
No. of cycles
474
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

ACCA: Holds number of bytes to be sorted; (No. of bytes to be stored - 1) 1-byte hexadecimal.

IX : Holds starting address of data in RAM to be sorted in 1-byte hexadecimal.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to sort 5-byte ascending data to descending.

DESCRIPTION

(b) Fig. 1 shows example of SORT execution.
If entry arguments are as shown in part ① of Fig. 1, sorted data is stored from address $90 in descending order (see part ② of Fig. 1).
As data to be sorted is 5-byte, $04 (Number of bytes to be sorted; ($05)-1) is held in ACCA.

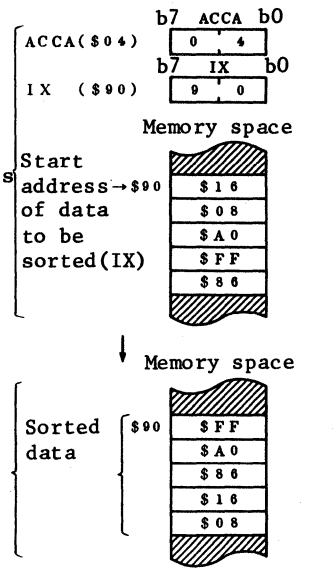


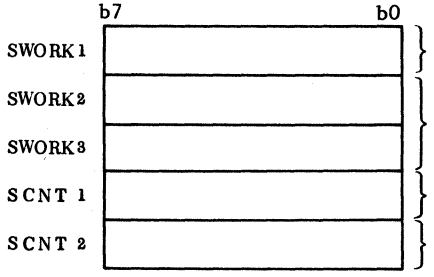
Fig. 1 Example of SORT execution

(2) User Notes

(a) When loading number of bytes to be sorted, holds (No. of bytes to be sorted - 1) to ACCA for effective loop processing.

(b) Stores data to be sorted in RAM using direct addressing.

(3) RAM Description



Description

IX (address pointer) is saved.

Work area is reserved to exchange data during sorting operation.

Counter is stored which shows how many bytes remain to be sorted.

Counter is stored which shows how many bytes remain to be compared.

DESCRIPTION

(4) Sample Application

SORT subroutine is called after start address and number of bytes to be sorted are held.

```
WORK1    RMB    1        ----- Reserves memory byte for number of
                                 bytes to be sorted.
WORK2    RMB    1        ----- Reserves memory byte for start address
          .                      to be sorted.
          .
          .
          .
         LDA    WORK1    ----- Loads number of bytes to be sorted into
                                 entry argument (ACCA).
         LDX    WORK2    ----- Loads start address of data to be sorted
                                 into entry argument (IX).
         JSR    SORT     ----- Calls SORT subroutine.
          .
          .
          .
          .
          .
```

DESCRIPTION

(5) Basic Operation

(a) Fig. 2 shows how three byte values are sorted in descending order.

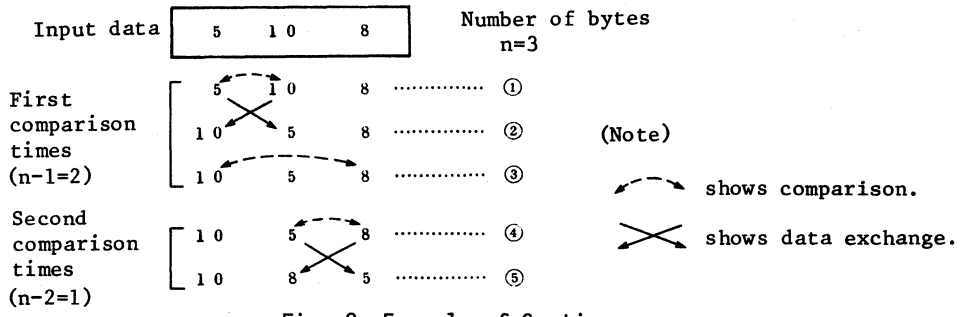


Fig. 2 Example of Sorting

(i) Finds the largest value among three and puts it into left position. (See Fig. 2 ①, ② and ③)

(ii) Compares middle and right values and puts larger one in middle. (See Fig. 2 ①, ④ and ⑤)

(b) Program processing

(i) Uses IX as two pointers.

(ii) First, use IX as pointer showing memory address where data is stored.

(iii) Loads this data into ACCA to be compared.

(iv) Increments address where data is stored and compares new value with value (indicated with using index addressing mode) in ACCA.

(v) If value is larger than compared value in ACCA (memory > ACCA), exchange them.

(vi) Loop (iv) to (v) until counter SCNT2, showing number of remaining bytes, reaches "0".

(vii) When SCNT2 reaches "0", the largest data is stored in ACCA.

(viii) Then use IX as pointer when storing the largest data.

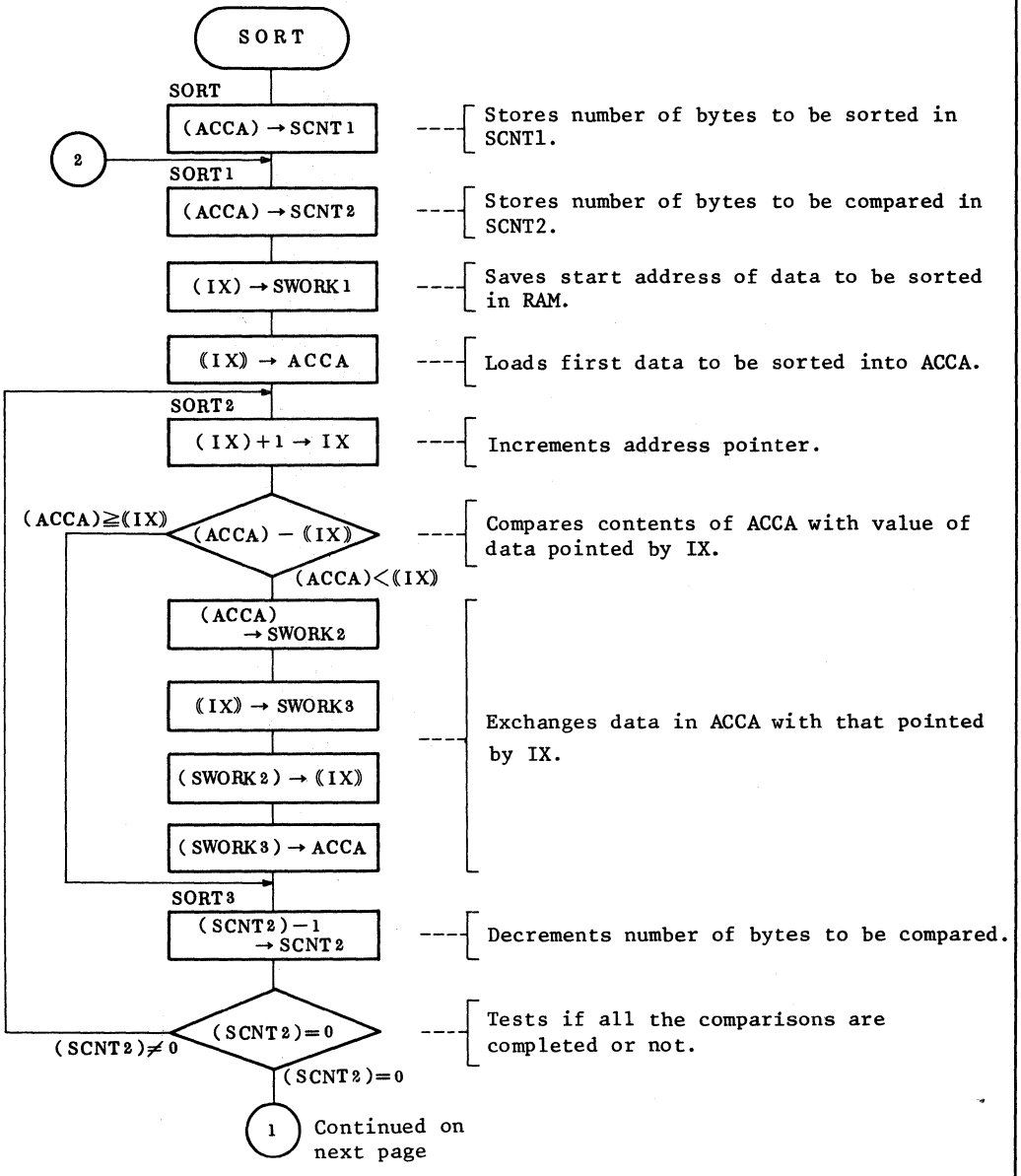
(ix) Stores contents of ACCA in address IX points, and loads next address, at which the next largest data is to be stored, into IX.

(x) Decrements counter SCNT1 showing how many bytes remain to be sorted.

(xi) Loops (iv) to (x) until SCNT1 is "0".

21. SORTING	MCU/MPU	HD6305 FAMILY	LABEL	SORT

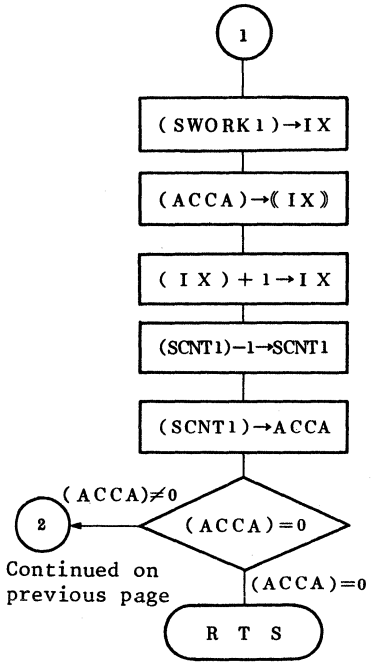
FLOWCHART



6

21. SORTING	MCU/MPU	HD6305 FAMILY	LABEL	SORT

FLOWCHART



Stores maximum data in RAM in descending order.

Increments address whose data is loaded to ACCA.

Decrements number of bytes remaining to be sorted.

Loads the number of bytes remaining to be compared.

Tests if all the data is completed or not.

21. SORTING	MCU/MPU	HD6305 FAMILY	LABEL	SORT

PROGRAM LISTING

```
00001                   ************************************************
00002                   *                                              *
00003                   *            NAME : SORTING (SORT)             *
00004                   *                                              *
00005                   ************************************************
00006                   *                                              *
00007                   *       ENTRY : ACCA (VOLUME OF SORTING DATA)  *
00008                   *                 IX   (TOP ADDR OF SORTING DATAS)*
00009                   *     RETURNS : NOTING                         *
00010                   *                                              *
00011                   ************************************************
00012                   *
00013 0080              ORG     $80
00014                   *
00015 0080 0001    SWORK1 RMB   1          Work for ADDR pointer
00016 0081 0001    SWORK2 RMB   1          Work for data exchange
00017 0082 0001    SWORK3 RMB   1          Work for data exchange
00018 0083 0001    SCNT1  RMB   1          Counter for sorting data
00019 0084 0001    SCNT2  RMB   1          Counter for comparing data
00020                   *
00021 1000              ORG     $1000
00022                   *
00023      1000    SORT   EQU   *          Entry point
00024 1000 B7 83        STA     SCNT1      Store counter for sorting data
00025 1002 B7 84   SORT1  STA     SCNT2     Store counter for comparing data
00026 1004 BF 80        STX     SWORK1     Store sorting data ADDR
00027 1006 F6           LDA     0,X        Load sorting data
00028 1007 5C      SORT2  INC     X          Set next sorting DATA ADDR
00029 1008 F1           CMP     0,X        Compare comparing DATA with sorting data
00030 1009 24 0A        BCC     SORT3      Branch if comparing DATA > sorting data
00031 100B B7 81        STA     SWORK2     Exchange each data
00032 100D F6           LDA     0,X
00033 100E B7 82        STA     SWORK3
00034 1010 B6 81        LDA     SWORK2
00035 1012 F7           STA     0,X
00036 1013 B6 82        LDA     SWORK3     Load exchanged data
00037 1015 3A 84   SORT3  DEC     SCNT2     Decrement compare data counter
00038 1017 26 EE        BNE     SORT2      Loop until compare data counter=0
00039 1019 BE 80        LDX     SWORK1     Load sorting data ADDR
00040 101B F7           STA     0,X        Store max data
00041 101C 5C           INC     X          Increment sorting data ADDR
00042 101D 3A 83        DEC     SCNT1      Decrement sorting data counter
00043 101F B6 83        LDA     SCNT1      Load sorting data counter
00044 1021 26 DF        BNE     SORT1      Loop until sorting data counter=0
00045 1023 81           RTS
```

6

Section Seven

Hardware Application Notes

The HD6305 is a series of CMOS 8-bit single chip microcomputers controlled by microprogramming. The CPU, clock generator, ROM, RAM, I/O, timer and serial communication interface are all resident on the chip. This series can be applied to a wide variety of systems, both small and large.

APPLICATION NOTES are written to help users design hardware systems using examples of typical application functions with specific circuit diagrams, timing charts and program examples.

Application examples in APPLICATION NOTES used in actual systems should be tested for proper operation.

NOTE

The following hardware application notes were prepared for HD6305X and HD6305Y devices. The applications, however, are generic in nature and also apply to HD6305U, HD6305V, and HD63L05 devices.

Section 7
Hardware Application Notes
Table of Contents

SCI APPLICATIONS

EXTERNAL EXPANSION APPLICATION

LOW POWER DISSIPATION/FAIL SAFE APPLICATION

APPLICATION NOTES GUIDE

1. HOW TO USE APPLICATION NOTES

This chapter describes the configuration for each system application example following this chapter.

1.1 APPLICATION EXAMPLE CONFIGURATION

Each application example in APPLICATION NOTES is divided into 5 sections, as shown in Fig. 1.1.

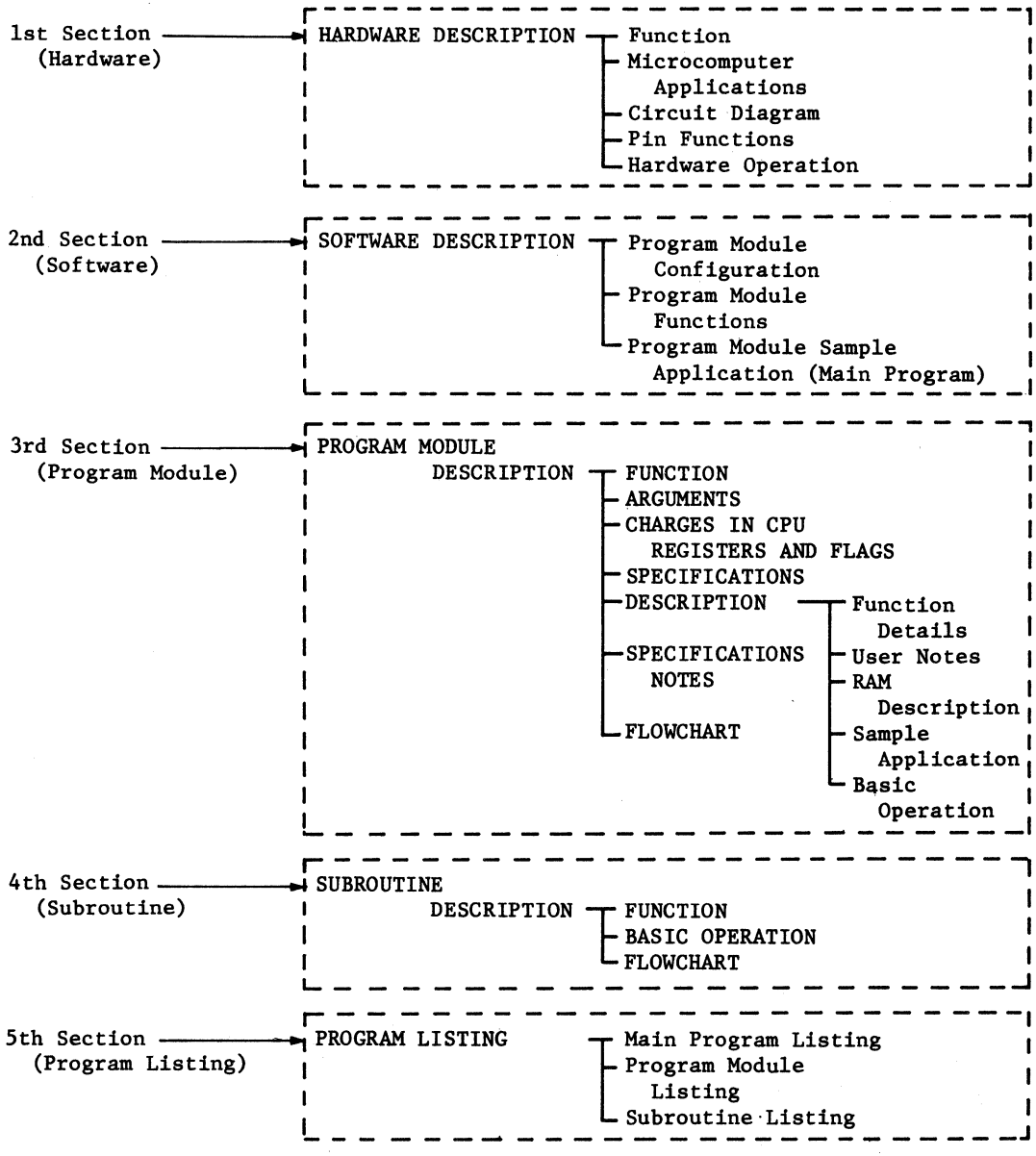


Fig. 1.1 Application Example Configuration

(1) 1st Section (Hardware)

Describes the function, circuit diagram or hardware operation of the HD6305 hardware example.

(2) 2nd Section (Software)

Describes the program module to control the hardware example in the 1st section and shows the main program when using all program modules.

(3) 3rd Section (Program Module)

Describes the program modules presented in the 2nd section in detail, using a modular format for more efficient system use.

(4) 4th Section (Subroutine)

Describes the subroutines used in the above program modules. Refer to it necessary when you use program module.

(5) 5th Section (Program Listing)

Presents sample application program listing for the above modules.

A detailed explanation of all five sections follows.

1.2 1ST SECTION (HARDWARE)

(1) Function

Describes system specifications for the hardware used in the particular application.

Example:

(1) Function

(a) Controls dot matrix liquid crystal controller driver HD44780 (hereinafter, LCD-II) using the HD6305X0 and displays in character mode on liquid crystal module H2570.

(b) H2570 displays 5×7 dot characters in 1-column × 16-row.

(c) Transfers ASCII from the HD6305X0 as display data to LCD-II. LCD-II automatically controls liquid crystal driver HD44100 and LCD by controlling LCD-II with the HD6305X0.

(2) Microcomputer Applications

Describes the typical functions of the microcomputer utilized for the particular application.

Example:

(2) Microcomputer Applications

Displays characters on H2570 by controlling LCD-II data bus ($DB_0 \sim DB_7$) and control signals (E, RS and R/W) through port G and port F.

(3) Circuit Diagram

Shows the circuit diagram for the hardware specified above.

Note) All the microcomputers described in APPLICATION NOTES are used Plastic DIP.

Example:

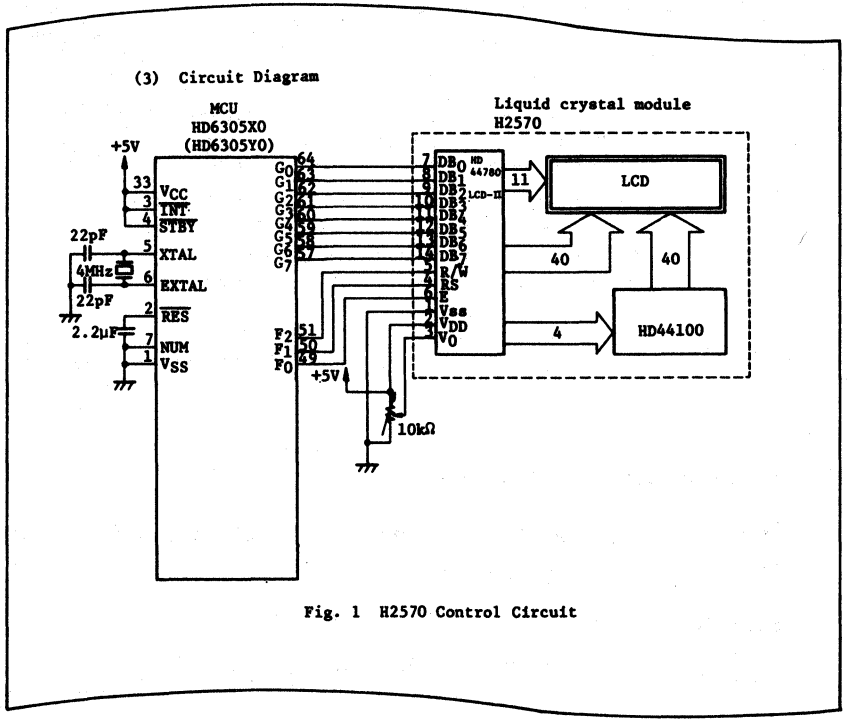


Fig. 1 H2570 Control Circuit

(4) Pin Functions

Describes pin functions for interfacing external circuits using a table.

Example:

(4) Pin Functions

Pin functions at the interface between the HD6305X0 and LCD-II are shown in Table 1.

Table 1 Pin Functions

Pin Name (HD6305X0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (LCD-II)	Program Label
F_0	Output	High	Enable signal	E	PFDTR
F_1	Output	Low	Selects instruction register	RS	
		High	Selects data register		
F_2	Output	Low	Writes data (Microcomputer → LCD-II)	R/W	
		High	Reads data (Microcomputer ← LCD-II)		
G_0	Input/ Output	—	Data lines	DB_0	PGDTR
G_1	Input/ Output	—		DB_1	
G_2	Input/ Output	—		DB_2	
G_3	Input/ Output	—		DB_3	
G_4	Input/ Output	—		DB_4	
G_5	Input/ Output	—		DB_5	
G_6	Input/ Output	—		DB_6	
G_7	Input/ Output	—		DB_7	

(a) "Active Level" in Table 1 indicates as follow;

High : Logical 1

Low : Logical 0

— : Logical 1 or Logical 0

(b) ⊘ in "Program Label" in Table 1 means there is no program label.

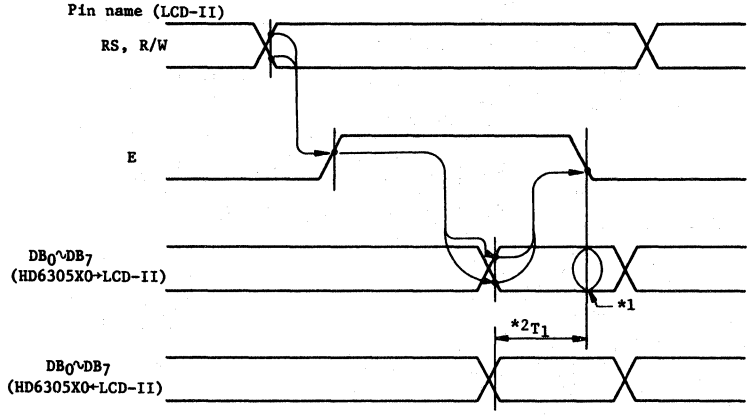
(5) Hardware Operation

Describes hardware operation required to control external circuits showing timing charts.

Example:

(5) Hardware Operation

LCD-II control signals (RS, R/W, E) are controlled by the program. Each LCD-II control signal timing chart is shown in Fig. 2.
The HD6305 family utilizes I/O port rather than buses to control LCD-II to eliminate any timing restrictions on LCD-II.



*1 Data is written to LCTC at the falling edge of E.
*2 Data from LCD-II can be read during period T_1.

Fig. 2 HD6305X0 ↔ LCD-II Interface

1.3 2ND SECTION (SOFTWARE)

(1) Program Module Configuration

Describes the necessary program modules to control the hardware specified in the 1ST SECTION. Each module in the Program Module Configuration figure has module No. (number of modules: 1 ∿ N) at the upper right. The module No. of the main program is '0'.

Example:

(1) Program Module Configuration

The program module configuration for character display on the liquid crystal module is shown in Fig. 3.

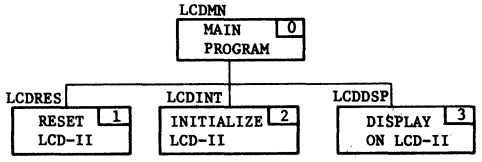


Fig. 3 Program Module Configuration

(2) Program Module Functions

Explains functions of each program module presented in Program Module Configuration. "No." in the table matches module No. in Program Module Configuration.

Example:

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	LCDMN	Demonstrates character display on H2570.
1	RESET LCD-II	LCDRES	Resets LCD-II using instruction.
2	INITIALIZE LCD-II	LCDINT	Initializes LCD-II to display characters on LCD.
3	DISPLAY ON LCD	LCDDSP	Transfers ASCII code to LCD-II and displays on H2570.

(3) Program Module Sample Application (Main Program)

Explains a sample program (Main Program) in flowchart format using program modules described in Program Module Configuration.

Example:

(3) Program Module Sample Application (Main Program)

The flowchart in Fig. 4 is an example of character display on H2570 performed by program module in Fig. 3.

The program in Fig. 4 demonstrates the display on liquid crystal shown in Fig. 5.

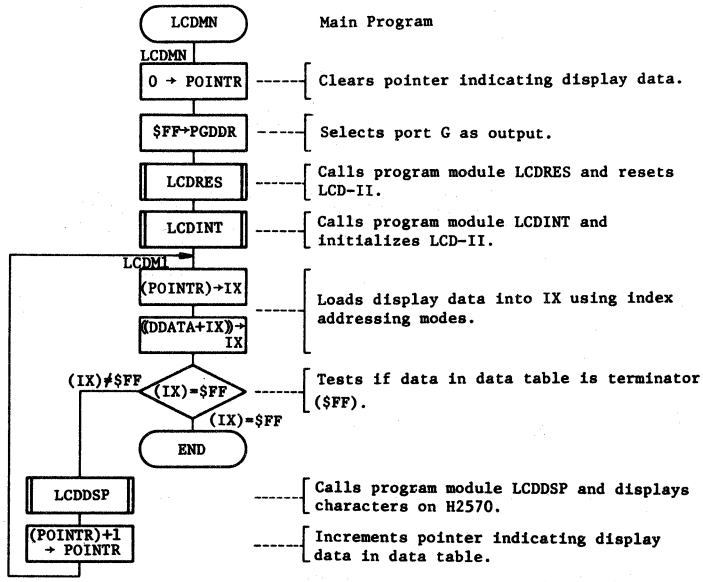


Fig. 4 Program Module Flowchart

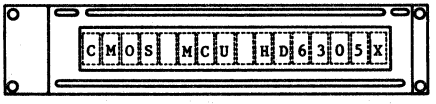


Fig. 5 Example of displaying on LCD

1.4 3RD SECTION (PROGRAM MODULE)

Program module detailed description consists of Format 1 to 3 as shown in Fig. 1.2.

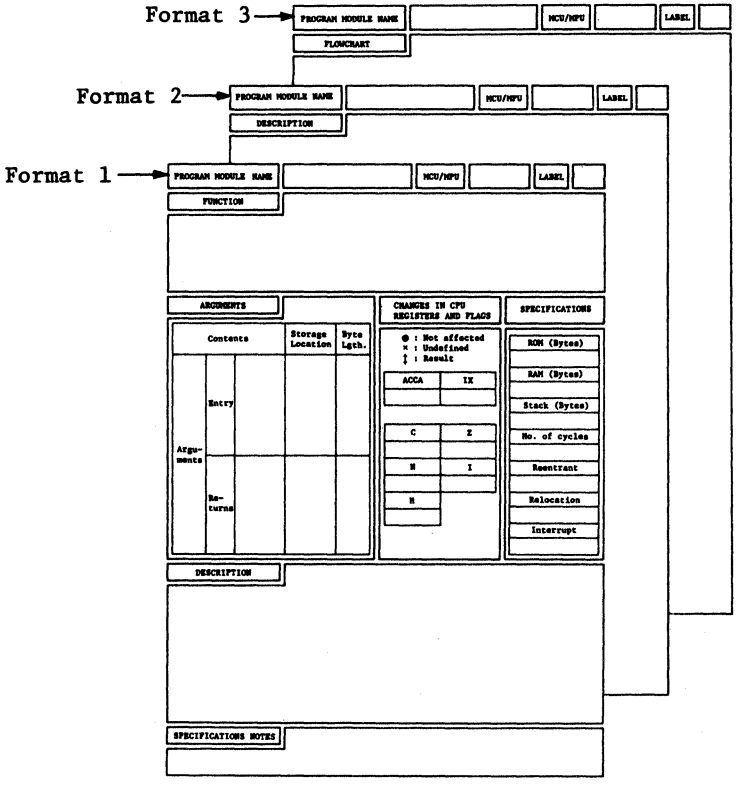


Fig. 1.2 Program Module Format

1.4.1 SPECIFICATION Format (Format 1)

The SPECIFICATION Format represented in Fig. 1.3. It describes functions and specifications for the program module used in the hardware sample application. Each numbered item is described referring to Fig. 1.3.

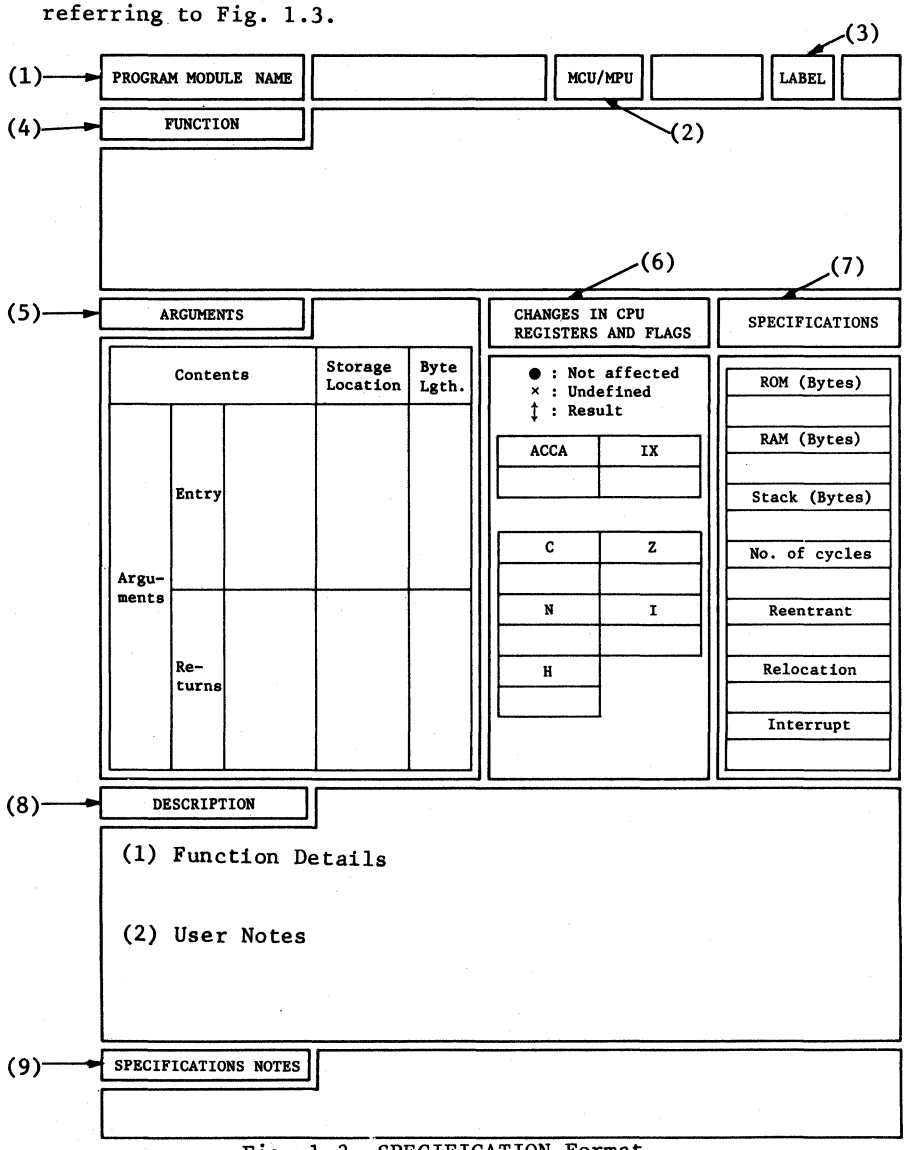


Fig. 1.3 SPECIFICATION Format

(1) PROGRAM MODULE NAME:

Example:

PROGRAM MODULE NAME	DISPLAY ON LCD

(2) MCU/MPU:

Indicates names of microcomputer and microprocessor the program module is used on.

Example:

MCU/MPU	HD6305X0/Y0

(3) LABEL:

Indicates the name identifying program entry point. When using the program module as a subroutine, call the label.

Example:

LABEL	LCDDSP

(4) FUNCTION:

Describes program module functions.

Example:

FUNCTION
Stores ASCII code in DDRAM of LCD-II and displays characters on LCD.

(5) ARGUMENTS:

Describes both entry and return arguments for the program module.

(a) Contents:

Describes the contents of arguments.

(b) Storage Location:

Indicates registers or RAM in which arguments must be located. When an argument is stored in RAM, the storage location is denoted by a label followed by "(RAM)".

(c) Byte Length:

Indicates byte length of the arguments.

Example:

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Display data	IX	1
	Returns	—	—	—

(6) CHANGES IN CPU REGISTERS AND FLAGS:

Describes changes in CPU registers after executing a program module as well as flag changes in the condition code register. Explanation of abbreviations and symbols in the table is given as follows:

(a) CPU register

ACCA : Accumulator A

IX : Index register

(b) Flags of condition code register

C : Carry/Borrow flag (Operation resulted in a carry or borrow)

Z : Zero flag (Zero result)

N : Negative flag (Negative result)

I : Interrupt flag (Interrupt mask)

H : Half carry flag (Carry from bit 3 to bit 4)

(c) Status of CPU registers and condition code register flags

● : Not affected : Previous values maintained after executing a program module.

× : Undefined : Previous values destroyed after executing a program module.

↕ : Results : Contains results from program module execution.

Example:

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
×	×
N	I
×	●
H	
●	

Note: In the example shown, contents of Accumulator A (ACCA) and Condition Code Register (CCR) (bit C, bit N and bit Z) are destroyed after executing the program module. Register contents thus destroyed could be saved before executing program module.

7

(7) SPECIFICATIONS:

Describes program module specifications as follows.

(a) ROM (Bytes):

Amount of ROM used in the program module.

(b) RAM (Bytes):

Amount of RAM used in the program module. RAM used for stack is not included.

(c) Stack (Bytes):

Stack size used in the program module. Stack size used by a subroutine called from a user program is not included. When a program module is executed, memory for the stack must be reserved in RAM.

(d) Number of cycles:

Maximum number of execution cycles required by the program module calculated as follows;

Execution time (sec) = Number of cycles × Cycle time

Cycle time (sec) = 4 / (External oscillator (Hz))

(e) Reentrant:

Indicates whether a program module has a structure which can be called from two or more routines at the same time.

(f) Relocation:

Indicates whether a program module can be located in any memory space.

(g) Interrupt:

Indicates whether CPU will continue with normal execution after serving an interrupt routine. If not, inhibit interrupt before and after the program module is called.

Example:

SPECIFICATIONS
ROM (Bytes)
42
RAM (Bytes)
0
Stack (Bytes)
2
No. of cycles
67
Reentrant
No
Relocation
No
Interrupt
Yes

(8) DESCRIPTION:

Describes the function of the program module in detail and precautions to follow.

(1) Function Details:

Gives an execution example and detailed functions of a program module.

(2) User Notes:

Explains notes and limitations when executing a program module.

* Be sure to read these items when using program modules without change.

Example:

DESCRIPTION

(1) Function Details

(a) Argument details
IX: Holds 1-byte ASCII.

(b) Fig. 7 shows an example of program module LCDDSP execution. If entry argument is held in IX as shown in part ① of Fig. 7, ASCII is stored in current address of DDRAM, and characters are displayed on LCD.

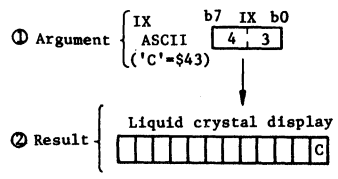


Fig. 7 Example of LCDDSP Execution

(c) Program module LCDDSP calls subroutines shown in Table 5.

Table 5 Subroutine Called in LCDDSP

Subroutine Name	Label	Function
CHECK BUSY FLAG	LCDBSY	Checks busy flag.

(2) User Notes

(a) Selects DDR of port G as output.

(b) Calls program modules LCDRES and LCDINT and initializes LCD-II to make H2570 display characters using LCD-II.

(9) SPECIFICATIONS NOTES:

Explains notes on data process written in (7) SPECIFICATIONS.

Example:

SPECIFICATIONS NOTES

(1) "SPECIFICATIONS" includes subroutine LCDBSY.

(2) "No. of cycles" in "SPECIFICATIONS" represents the number of cycles required when subroutine LCDBSY is executed by the minimum cycles.

1.4.2 DESCRIPTION Format (Format 2)

The DESCRIPTION Format is represented in Fig. 1.4. It describes RAM description, Sample Application and Basic Operation. Each numbered is described referring to Fig. 1.4.

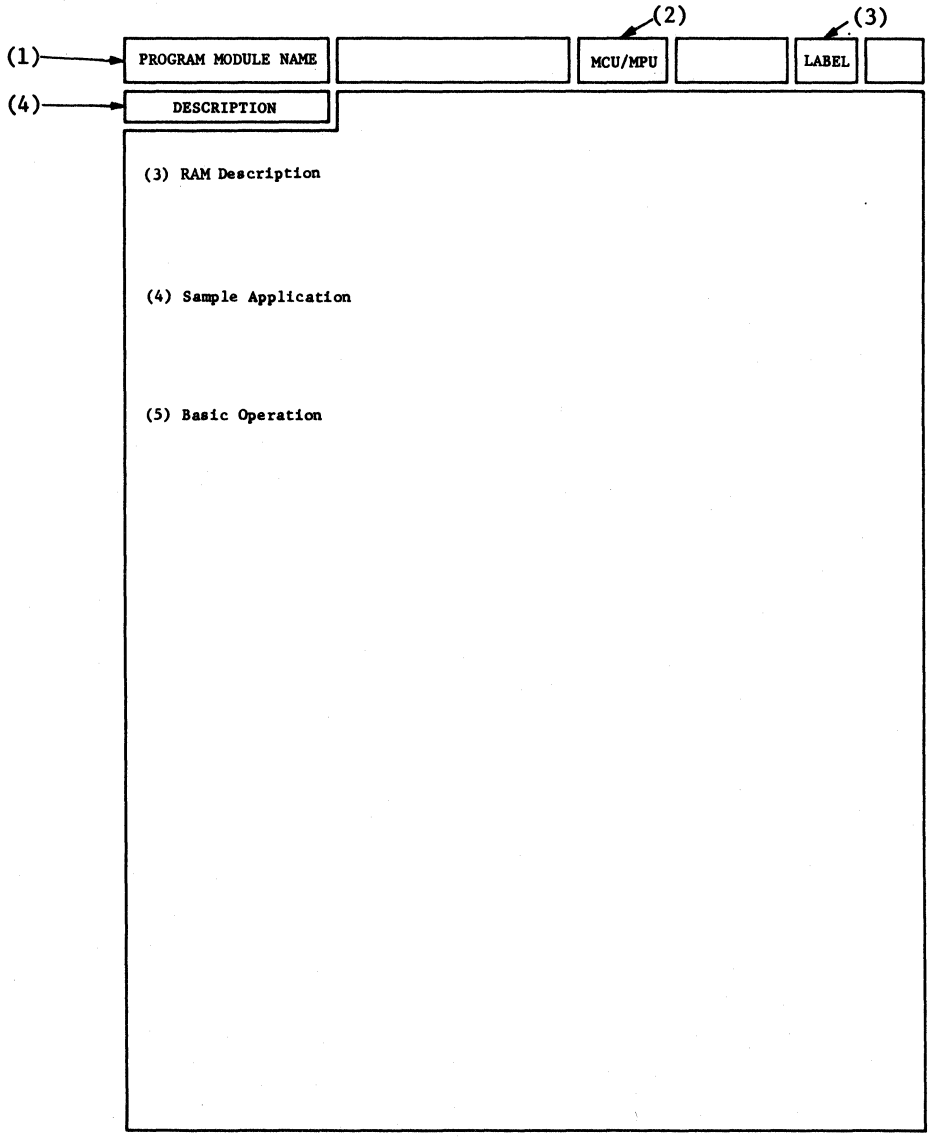


Fig. 1.4 DESCRIPTION Format

(1) PROGRAM MODULE NAME
(2) MCU/MPU
(3) LABEL

} Same as SPECIFICATION Format.

(4) DESCRIPTION:

Gives RAM Description, Sample Application and Basic Operation.

(a) RAM Description:

Explains label and function of the RAM used in the program module.

Example:

PROGRAM MODULE NAME	DISPLAY ON LCD	MCU/MPU	HD6305X0/Y0	LABEL	LCDDSP

DESCRIPTION

(3) RAM Description

RAM is not used in program module LCDDSP.

(b) Sample Application:

Gives a sample application in actual use.

Example:

(4) Sample Application

Program module LCDDSP is called after selecting I/O port, resetting LCD-II, initializing LCD-II and storing display data.

```
    :
    LDA     #$FF      }
    STA     PGDDR     }--Selects port G as output.
    JSR     LCDRES ----Calls program module LCDRES and resets LCD-II.
    JSR     LCDINT ----Calls program module LCDINT and initializes LCD-II.
    LDX     #$41   ----Stores display data in entry argument.
    JSR     LCDDSP --Calls program module LCDDSP.
    :
```

(c) Basic Operation:

Explains basic operation of the program module.

Example:

(5) Basic Operation

(a) Microcomputer cannot write data into LCD-II when LCD-II is in operation. Whether LCD-II is in operation or not can be checked by busy flag. Therefore, program module LCDDSP calls subroutine LCDBSY and checks LCD-II busy flag to test if LCD-II is in operation.
Busy flag = 1 : LCD-II is in operation. Data cannot be written.
Busy flag = 0 : Data can be written into LCD-II.

(b) Controls LCD-II control signal and writes data into LCD-II with timing illustrated in "2.1 HARDWARE DESCRIPTION, (5) Hardware Operation".

1.4.3 FLOWCHART Format (Format 3)

The FLOWCHART Format is represented in Fig. 1.5. Each numbered item is described referring to Fig. 1.5.

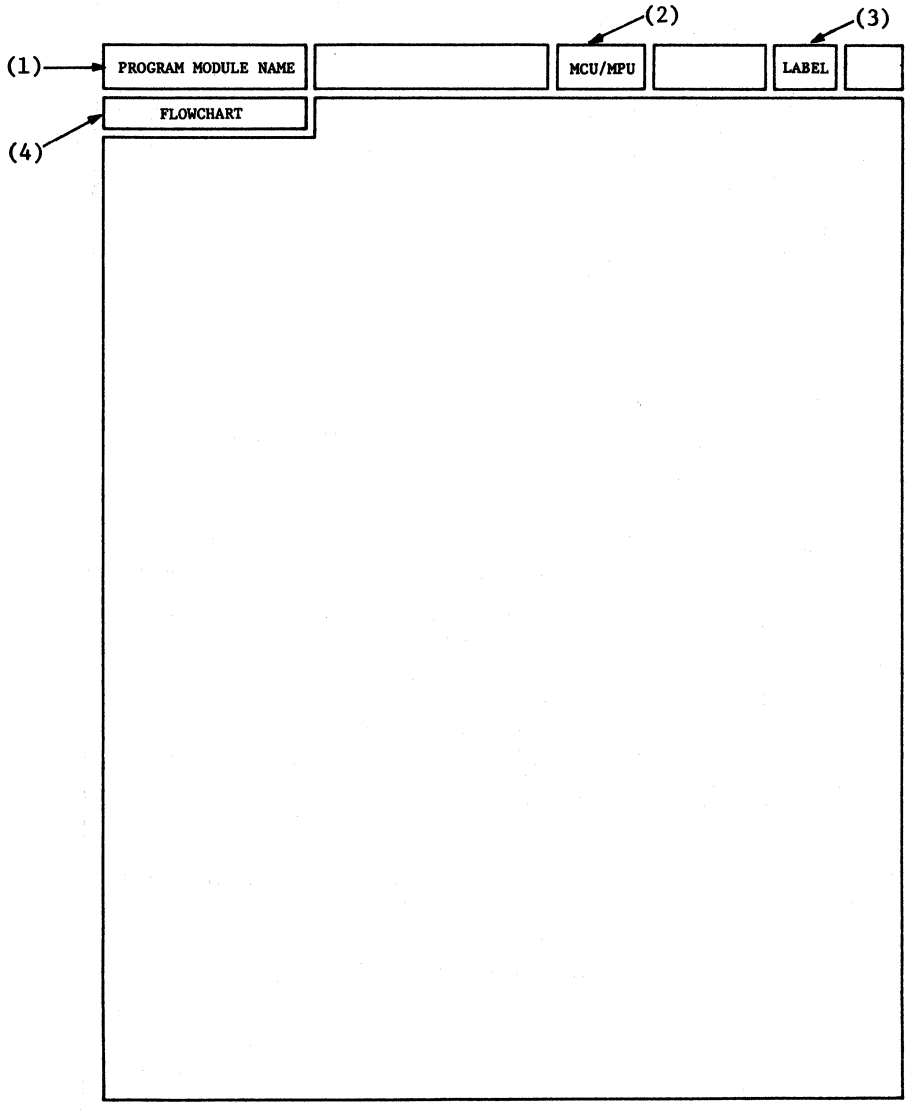


Fig. 1.5 FLOWCHART Format

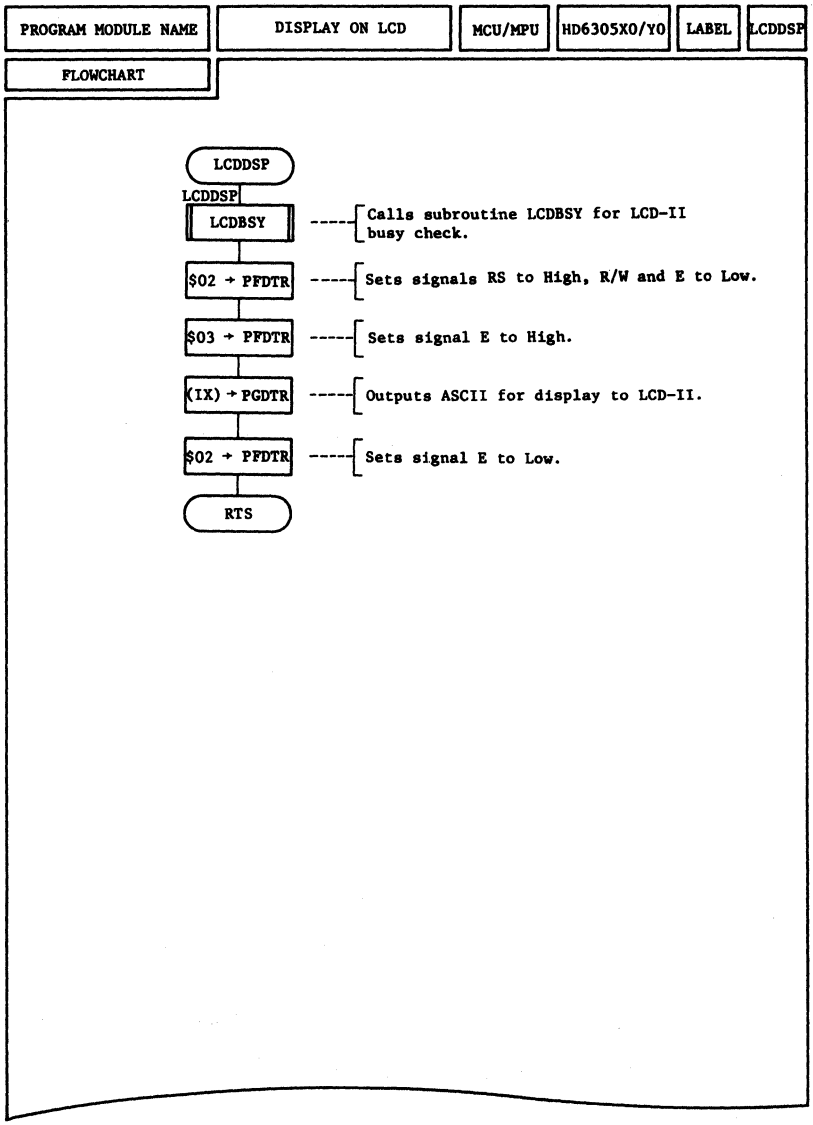
(1) PROGRAM MODULE NAME
(2) MCU/MPU
(3) LABEL

} Same as SPECIFICATION Format.

(4) FLOWCHART:

Gives program module flowchart. Flowchart comments are enterted to right.

Example:



1.5 4TH SECTION (SUBROUTINE)

The subroutine format is represented in Fig. 1.6. It gives subroutines used by program module referring to Fig. 1.6.

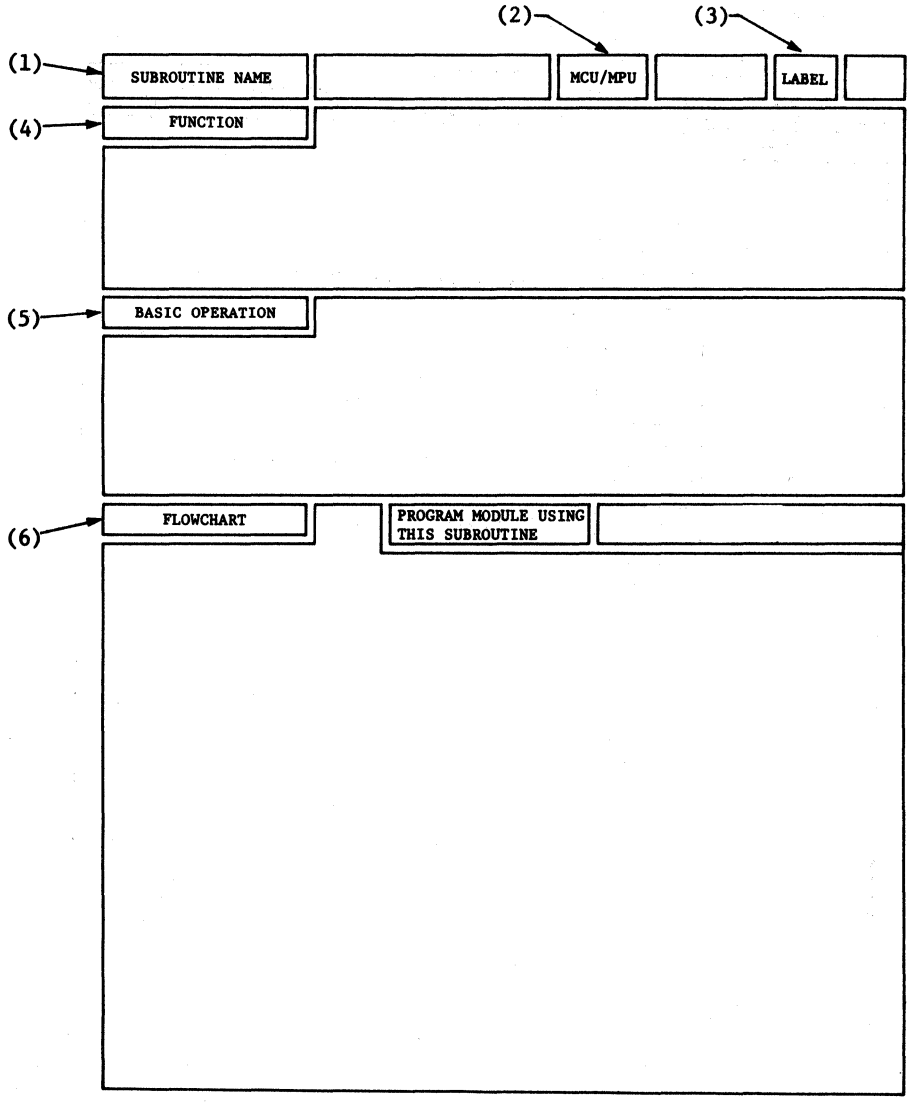


Fig. 1.6 Subroutine Format

(1) SUBROUTINE NAME:

Example:

SUBROUTINE NAME	CHECK BUSY FLAG

(2) MCU/MPU:

Indicates names of microcomputer and microprocessor family applicable to a program.

Example:

MCU/MPU	HD6305X0/Y0

(3) LABEL:

Indicates the subroutine entry point name. Call the subroutine with this label.

Example:

LABEL	LCDBSY

(4) FUNCTION:

Describes subroutine functions.

Example:

FUNCTION
(1) Tests LCD-II status. (2) Loops subroutine LCDBSY until LCD-II becomes READY.

(5) BASIC OPERATION:

Describes subroutine basic operation.

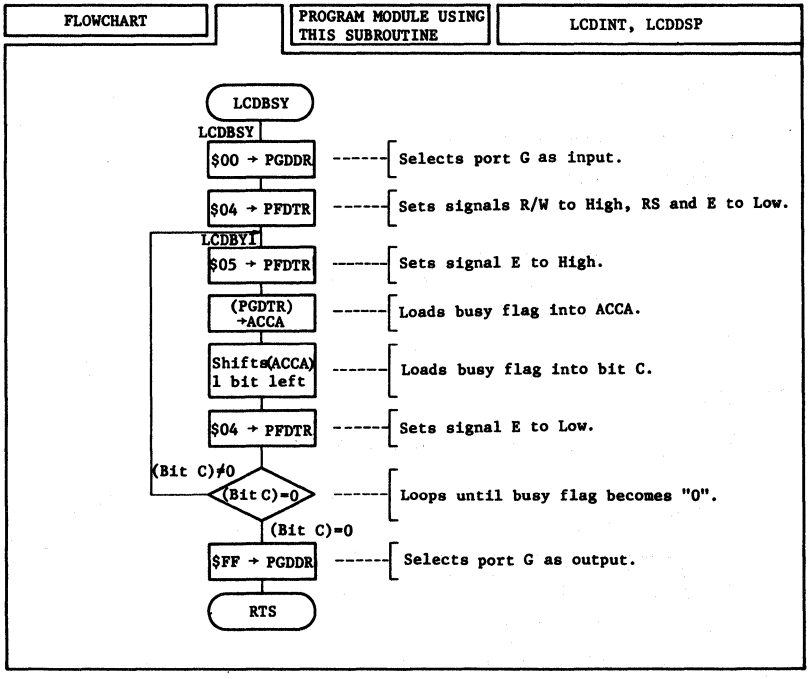
Example:

BASIC OPERATION
The MSB of data bus becomes busy flag if LCD-II data bus is read during RS=0, R/W=1 and E=1.

(6) FLOWCHART:

Gives the subroutine flowchart. Comments are described in the column on the right. The program modules using this subroutine are shown in the upper right of the format.

Example:



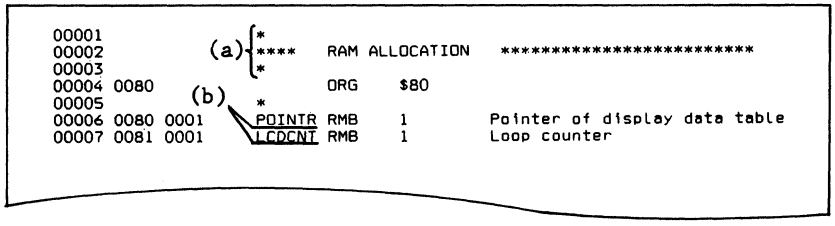
1.6 5TH SECTION (PROGRAM LISTING)

RAM allocation, CPU register allocation, program module and subroutine listing are described as follows.

(1) RAM Allocation:

RAM used in a program module or subroutine is allocated as below.

Example:



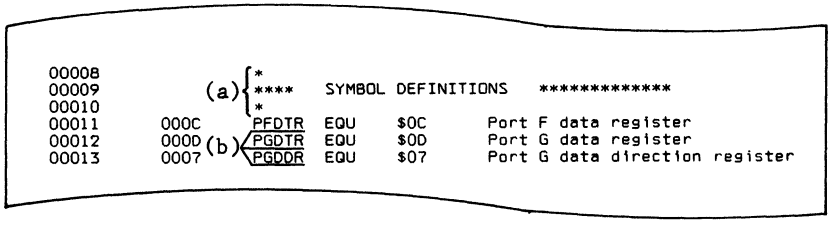
(a) The title "RAM ALLOCATION" is followed by the actual RAM allocation used.

(b) Label of allocated RAM.

(2) Definition of Symbols

Symbols used in a program module or subroutine are defined as below.

Example:



(a) The title is always "SYMBOL DEFINITIONS".

(b) Symbol definitions.

(3) Main Program

Describes main program listing of a sample application.

Example:

```
00014                    ************************************************
00015                    *                                              *
00016              (a)   *              MAIN PROGRAM : LCDMN            *
00017                    *                                              *
00018                    ************************************************
00019                    *
00020 1000         (b)             ORG     $1000
00021                    *
00022 1000 4F            LCDMN   CLR     A           Clear pointer
00023 1001 B7 80                 STA     POINTR
00024 1003 B7 0D                 STA     PGDTR       Initialize port G
00025 1005 A6 FF                 LDA     #$FF
00026 1007 B7 07                 STA     PGDDR       Select port G as output
00027 1009 CD 1021               JSR     LCDRES      Reset LCD-II
00028 100C CD 1041               JSR     LCDINT      Initialize LCD-II
00029 100F BE 80         LCDM1   LDX     POINTR      Load data table pointer
00030 1011 DE 108A               LDX     DDATA,X     Ascii data -> IX
00031 1014 A3 FF                 CPX     #$FF        Test if IX=$FF
00032 1016 27 07                 BEQ     PEND        Branch if IX=#$FF
00033 1018 CD 1059               JSR     LCDDSP      Display data
00034 101B 3C 80                 INC     POINTR      Increment pointer
00035 101D 20 F0                 BRA     LCDM1       Branch always LCDM1
00036 101F 20 FE         PEND    BRA     PEND        End of main program
```

(a) The title "MAIN PROGRAM" is always used followed by the entry point label in parenthesis.

(b) Entry point label.

(4) Program Module:

Describes program module listing of a sample application.

Example:

```
00037                 ************************************************
00038                 *                                              *
00039                 *          NAME : LCDRES (RESET LCD-II)        *
00040                 *                                              *
00041           (a)   ************************************************
00042                 *                                              *
00043                 *       ENTRY : NOTHING                        *
00044                 *     RETURNS : NOTHING                        *
00045           (b)   *                                              *
00046                 ************************************************
00047 1021 A6 03    LCDRES LDA    #3        Initialize loop counter
00048 1023 B7 81           STA    LCDCNT
00049 1025 A6 0C    LCDRS1 LDA    #$0C      Initialize 15ms counter
00050 1027 AE FF    LCDRS2 LDX    #$FF      Initialize inner counter
00051 1029 5A       LCDRS3 DEC    X         Decrement inner counter
00052 102A 26 FD           BNE    LCDRS3    Loop until inner counter=0
00053 102C 4A              DEC    A         Decrement 15ms counter
00054 102D 26 F8           BNE    LCDRS2    Loop until 15ms counter=0
00055 102F B7 0C           STA    PFDTR     Set RS=0,R/W=0,E=0
00056 1031 A6 01           LDA    #$01
00057 1033 B7 0C           STA    PFDTR     Set RS=0,R/W=0,E=1
00058 1035 A6 30           LDA    #$30
00059 1037 B7 0D           STA    PGDTR     Write instruction data
00060 1039 4F              CLR    A         Set E=0
00061 103A B7 0C           STA    PFDTR
00062 103C 3A 81           DEC    LCDCNT    Decrement loop counter
00063 103E 26 E5           BNE    LCDRS1    Loop until loop counter=0
00064 1040 81              RTS
```

(a) Program module title followed by the entry point label in parenthesis and description of entry and return arguments.

(b) Entry point label.

(5) Subroutine:

Describes listing of subroutines used in the program module.

Example:

```
00107                 ************************************************
00108                 *                                              *
00109           (a)   *          NAME : LCDBSY (CHECK BUSY FLAG)     *
00110                 *                                              *
00111           (b)   ************************************************
00112 106B 4F       LCDBSY CLR    A         Select port G as input
00113 106C B7 07           STA    PGDDR
00114 106E A6 04           LDA    #$04      Set RS=0,R/W=1,E=0
00115 1070 B7 0C           STA    PFDTR
00116 1072 A6 05    LCDBY1 LDA    #$05      Set E=1
00117 1074 B7 0C           STA    PFDTR
00118 1076 B6 0D           LDA    PGDTR     Read busy flag
00119 1078 48              LSL    A         Set busy flag to bit C
00120 1079 A6 04           LDA    #$04      Set E=0
00121 107B B7 0C           STA    PFDTR
00122 107D 25 F3           BCS    LCDBY1    Loop until busy flag=0
00123 107F A6 FF           LDA    #$FF      Select port G as output
00124 1081 B7 07           STA    PGDDR
00125 1083 81              RTS
```

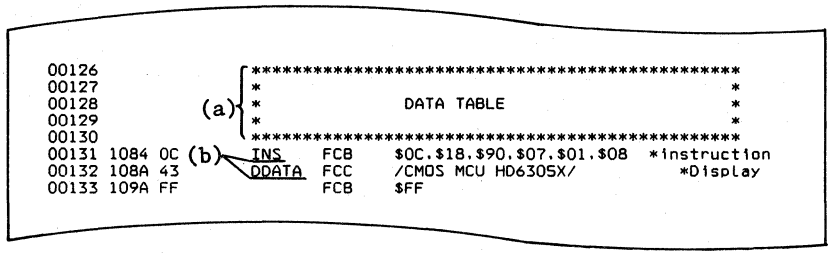
(a) Subroutine title followed by the entry point label in parenthesis.

(b) Entry point label.

(6) Data Table:

Describes data table used in the main program, program module and subroutines.

Example:



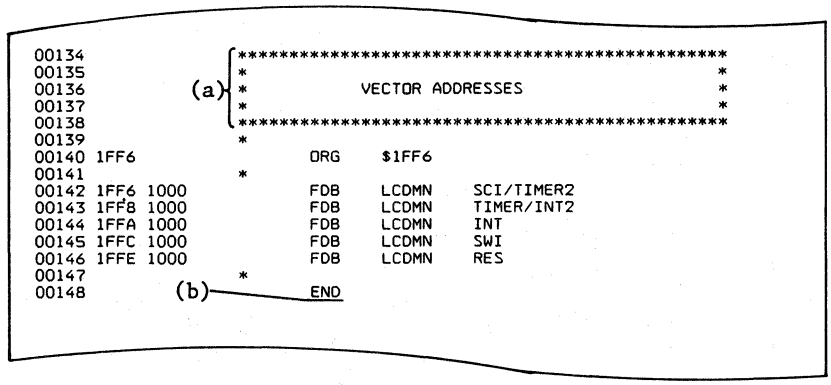
(a) The title is always "DATA TABLE".

(b) Data table label.

(7) Vector Addresses:

Describes vector address allocation.

Example:



(a) The title is always "VECTOR ADDRESSES".

(b) Denotes end of entire program. This can be moved if necessary.

1.7 PROGRAM MODULE USAGE

This section explains how to execute program modules described in APPLICATION NOTES.

Fig. 1.7 shows relation between these program modules and user programs. All program modules are used as subroutines and should be called as shown in Fig. 1.7.

Fig. 1.8 shows an example of a user program in which a program module is used as a subroutine.

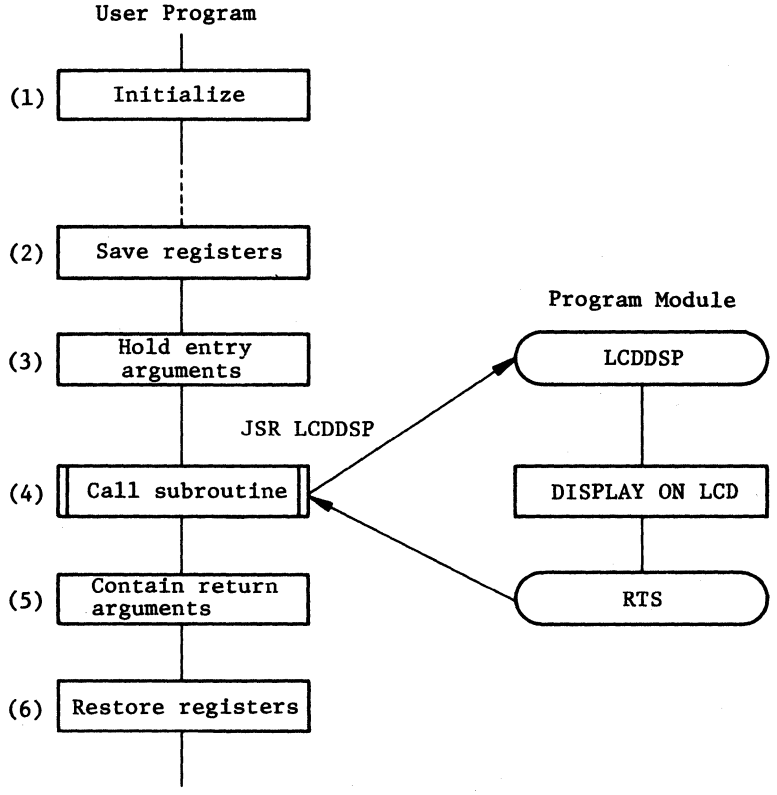


Fig. 1.7 Relation Between User Program and Program Module

```
User Program
    :
    :
    :
LDA   #$FF
STA   PGDDR
JSR   LCDRES      ..... Initializes before executing program module.
JSR   LCDINT
    :
    :
LDX   #$41        ..... Holds entry arguments.
JSR LCDDSP        ..... Calls program module
    :
    :
```

Fig. 1.8 Example of How to Execute a Program Module

Explanation of Fig. 1.7.

(1) Initialize program module

Examples of items requiring initialization are input/output ports, control registers and counters used by the program module. Refer to Program Module Sample Application for data details.

(2) Save registers

CPU registers used in the programs may return to user program while destroying the original contents. Thus register contents should be saved if necessary. Refer to the "CHANGES OF CPU REGISTERS AND FLAGS" column in SPECIFICATIONS (Format 1 in 3rd Section - Program Module) for register status after a program module is executed.

(3) Hold entry arguments

Define arguments in CPU registers or memory before calling a program module in the user program. Refer to "ARGUMENTS" in SPECIFICATIONS (Format 1 in 3rd Section - Program Module) for entry arguments to be held.

(4) Call subroutine

Program module is called.

(5) Contain return arguments

After a program module is executed, the results returned in the return arguments must be processed as needed. Refer to "ARGUMENTS" in SPECIFICATIONS (Format 1 in 3rd Section - Program Module) for details.

(6) Restore registers

Registers saved in (2) should be restored here. Note that when a program module is used as a subroutine, the stack area shown in SPECIFICATIONS (Format 1 in 3rd Section - Program Module) is necessary in addition to the stack area required the subroutine call in the user program. When any subroutine is called this stack area must be reserved.

1.8 SYMBOLS

Symbols and abbreviations used in APPLICATION NOTES are defined as follows.

(1) Operation

( ) = Contents
(( )) = Index address
→ = Transfer direction
+ = Addition
- = Subtraction
× = Multiplication
/ = Division
∧ = AND
∨ = OR
⊕ = Exclusive OR
$\bar{x}$ = NOT

(2) Register symbols in MCU/MPU

ACCA = Accumulator A
CCR = Condition code register
IX = 8-bit index register

(3) Bit 0 to bit 4 in the condition code register

C = Carry or borrow	bit 0
Z = Zero	bit 1
N = Negative	bit 2
I = Interrupt mask	bit 3
H = Carry from bit 3 to bit 4	bit 4

(4) Others

= = Equal sign
≠ = Not-equal sign
>, <, ≧, ≦ = Comparison signs
' ' = ASCII code inside
$ = Hexadecimal number
: = Labels of sequential addresses

SCI = Serial Communication Interface

DDR = Data Direction Register

TCR = Timer Control Register

TDR = Timer Data Register

SCR = SCI Control Register

SSR = SCI Status Register

SDR = SCI Data Register

MR = Miscellaneous Register

APPLICATION EXAMPLES

No.	Item	Micro-computer	Function	Devices	Reference Page
1	HD61830(LM200) GRAPHIC MODE	HD6305X0 (HD6305Y0)	I/O port	HD6305X0 HD61830 HM6116 LM200	37
2	LIQUID CRYSTAL MODULE (H2570) CONTROL	HD6305X0 (HD6305Y0)	I/O port	HD6305X0 H2570	61
3	DUTY CONTROL OF PULSE OUTPUT	HD6305X0 (HD6305Y0)	I/O port Timer	HD6305X0	79
4	PULSE WIDTH MEASUREMENT	HD6305X0 (HD6305Y0)	$\overline{INT}$ pin Timer	HD6305X0	93
5	INPUT PULSE COUNT	HD6305X0 (HD6305Y0)	Timer	HD6305X0	102
6	ZERO CROSS	HD6305X0 (HD6305Y0)	$\overline{INT}$ pin Timer	HD6305X0	110
7	KEY MATRIX (8 × 4)	HD6305X0 (HD6305Y0)	I/O port Timer 2	HD6305X0 8×4 KEY MATRIX	121
8	FLUORESCENT DISPLAY CONTROL	HD6305X0 (HD6305Y0)	I/O port Timer 2	HD6305X0 8-digit 8-segment fluorescent display tube	135
9	STEPPING MOTOR CONTROL	HD6305X0 (HD6305Y0)	I/O port Timer	HD6305X0 Stepping motor	147
10	WITH A COM-MERCIALLY AVAILABLE KEYBOARD	HD6305X0 (HD6305Y0)	I/O port $\overline{INT_2}$ pin	HD6305X0 ASCII Keyboard	172
11	SCI CLOCK SYNCHRONOUS (EXTERNAL CLOCK)	HD6305X0 (HD6305Y0)	SCI	HD6305X0	186
12	SCI CLOCK SYNCHRONOUS (INTERNAL CLOCK)	HD6305X0 (HD6305Y0)	SCI	HD6305X0	198
13	LIQUID CRYSTAL DRIVER (HD61100A) CONTROL	HD6305X0 (HD6305Y0)	I/O port SCI	HD6305X0 HD61100A 10-digit 8-segment LCD	210
14	EXTERNAL EXPANSION	HD6305Y2	External Expansion	HD6305Y2 HN27C64 HM6117 HD6321 HD6350 H2571	222
15	LOW POWER DISSIPATION MODE AND HA1835P CONTROL	HD6305X0 (HD6305Y0)	Low power dissipation mode (STANDBY, WAIT, STOP) I/O port	HD6305X0 HA1835P	254

1. HD61830 (LM200) GRAPHIC MODE

1.1 HARDWARE DESCRIPTION

(1) Function

(a) Permits graphic display on liquid crystal module LM200 through dot matrix liquid crystal graphic display controller HD61830 (hereinafter, LCTC). LCTC is controlled by the HD6305X0.

(b) LM200 display resolution is 64 × 240 pixels.

(c) Sends display data from the HD6305X0 to LCTC. Display RAM and LM200 are controlled automatically by LCTC.

(2) Microcomputer Applications

Displays graphics on LM200 by controlling LCTC data bus ($DB_0 \sim DB_7$) and control signals (E, RS and R/W) through port A and port C.

(3) Circuit Diagram

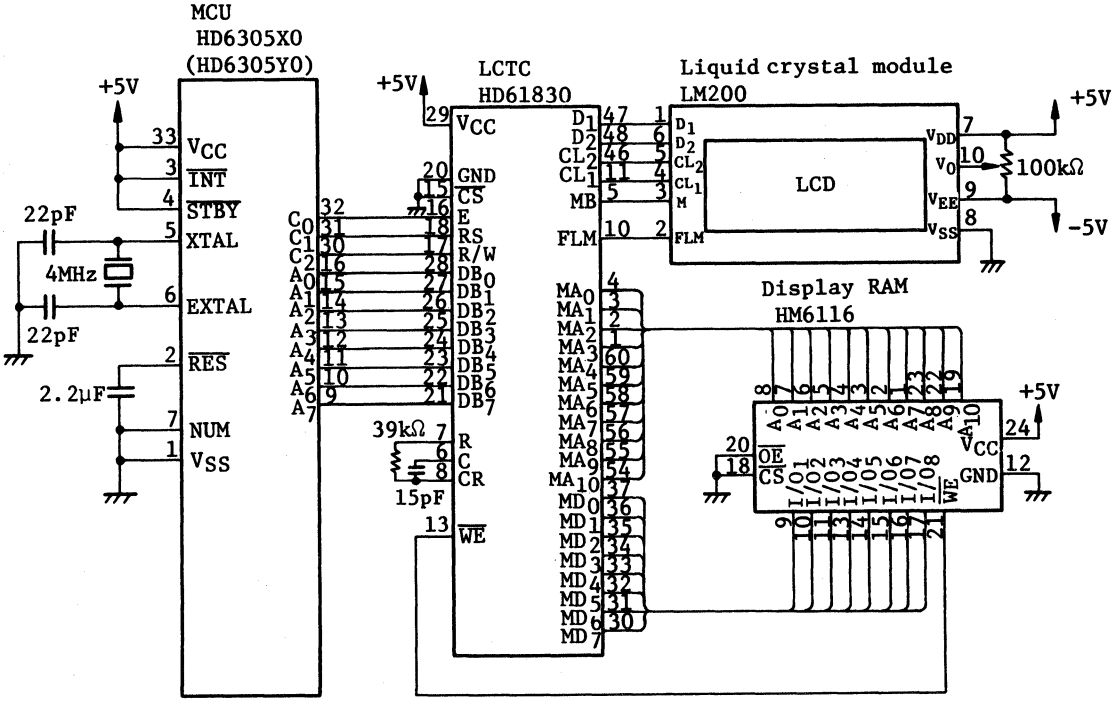


Fig. 1 LCTC Control Circuit

(4) Pin Functions

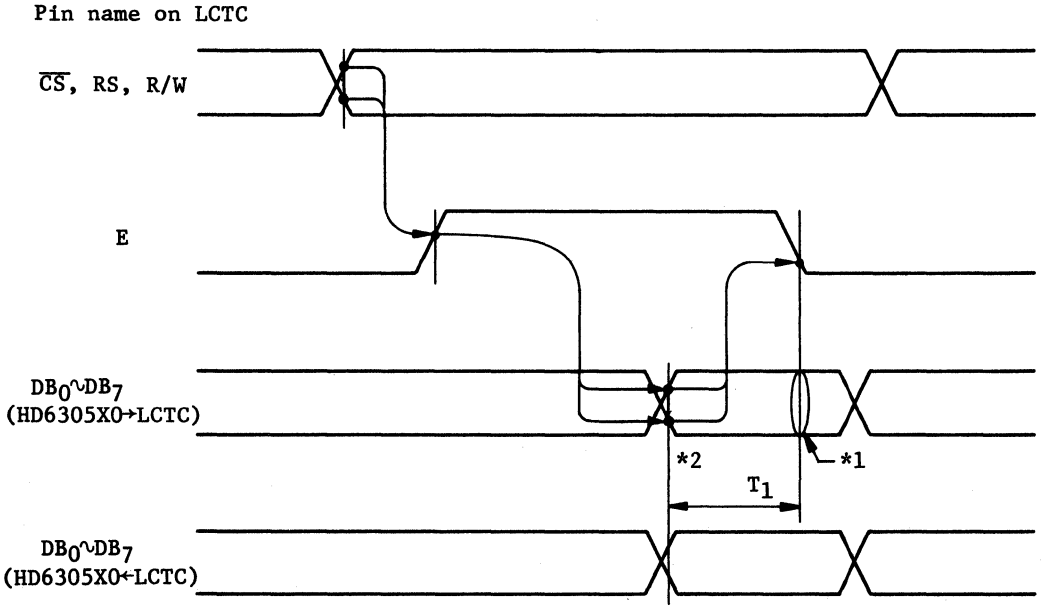
Pin functions at the interface between the HD6305X0 and LCTC are shown in Table 1.

Table 1 Pin Functions

Pin Name (HD6305X0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (LCTC)	Program Label
C_2	Output	High	Reads data	R/W	PCDTR
		Low	Writes data		
C_1	Output	High	Selects instruction register	RS	
		Low	Selects data register		
C_0	Output	High	Enable signal	E	
A_0	Input/ Output	—	Data lines	DB_0	PADTR
A_1	Input/ Output	—		DB_1	
A_2	Input/ Output	—		DB_2	
A_3	Input/ Output	—		DB_3	
A_4	Input/ Output	—		DB_4	
A_5	Input/ Output	—		DB_5	
A_6	Input/ Output	—		DB_6	
A_7	Input/ Output	—		DB_7	

(5) Hardware Operation

LCTC control signals ($\overline{CS}$, RS, R/W, E) are controlled by the program using the HD6305X0 I/O port. LCTC signal control timing is shown in Fig.2. The HD6305 family utilizes I/O ports rather than a bus to control LCTC to eliminate any timing restrictions on LCTC.



*1 Data is written into LCTC at the falling edge of E.

*2 Data from LCTC can be read during period T_1.

Fig. 2 HD6305X0 ↔ LCTC Interface

1.2 SOFTWARE DESCRIPTION

(1) Program Module Configuration

The program module configuration for graphic display on the liquid crystal module is shown in Fig. 3.

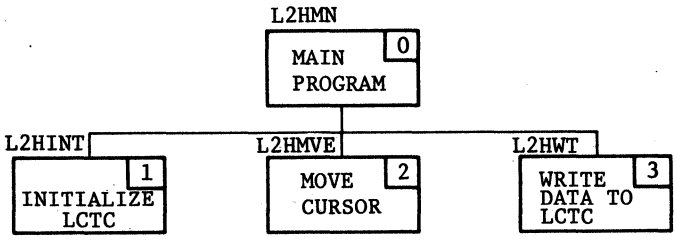


Fig. 3 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	L2HMN	Demonstrates graphic display on LM200.
1	INITIALIZE LCTC	L2HINT	Initializes LCTC for graphic display.
2	MOVE CURSOR	L2HMVE	Specifies LCTC cursor address.
3	WRITE DATA TO LCTC	L2HWT	Writes instruction and display data to LCTC.

(3) Program Module Sample Application (Main Program)

The flowchart in Fig. 4 is an example of the display process performed by the program module in Fig. 3.

The program in Fig. 4 creates the display on liquid crystal shown in Fig. 5.

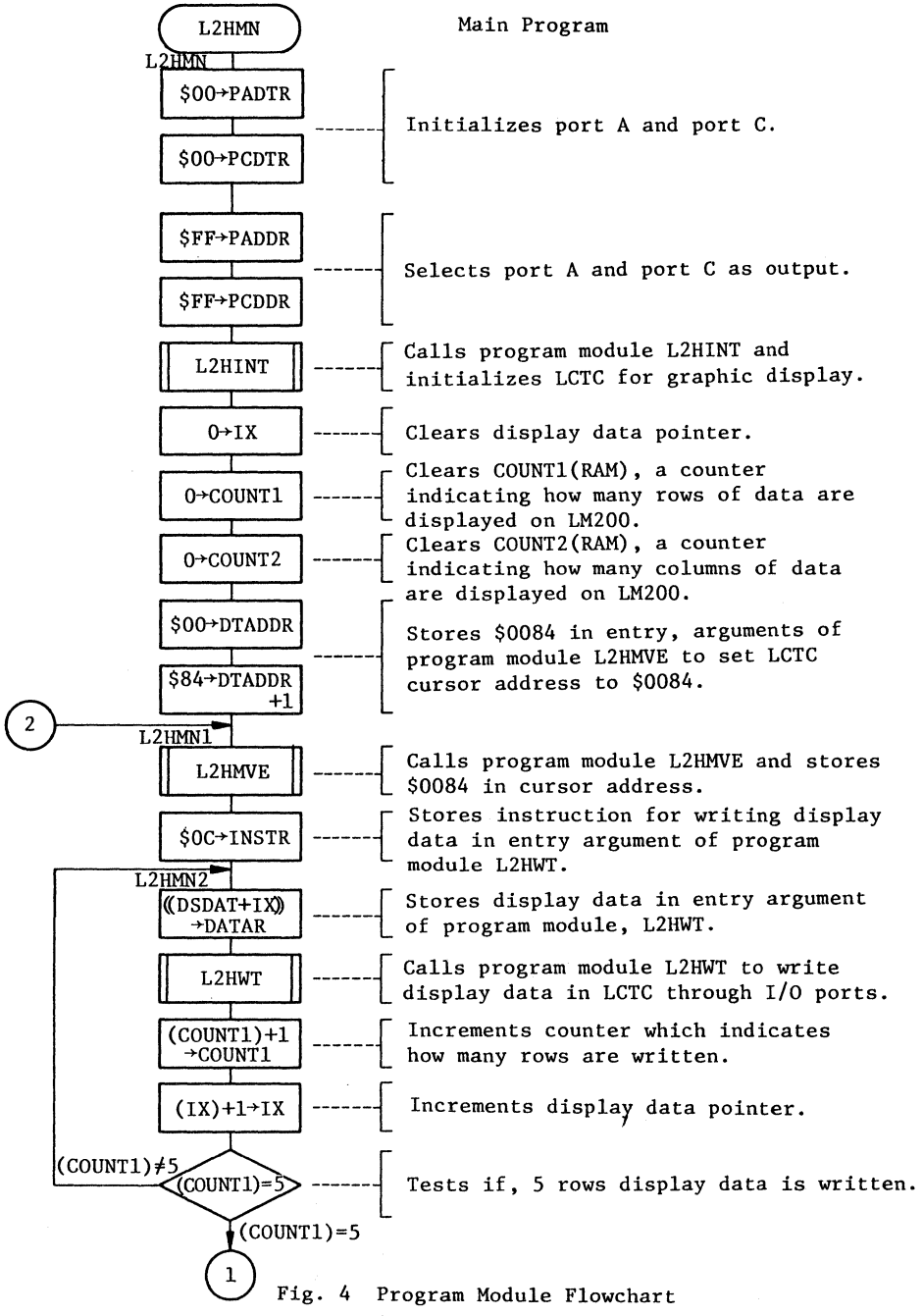


Fig. 4 Program Module Flowchart

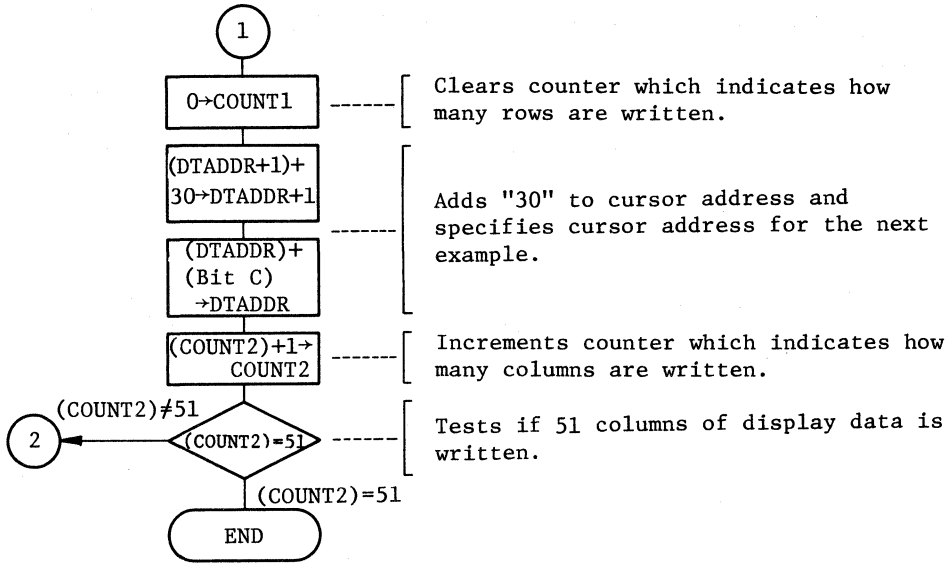


Fig. 4 (cont.) Program Module Flowchart

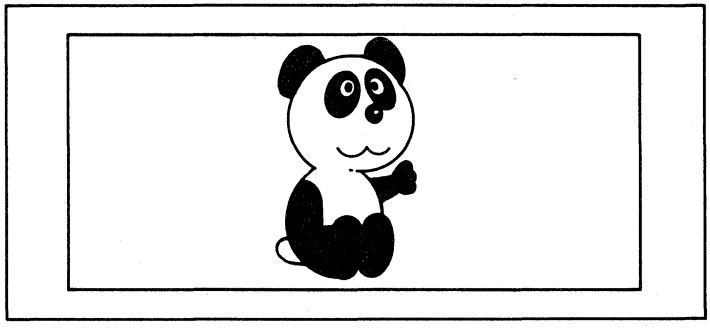


Fig. 5 Example of L2HMN Execution

PROGRAM MODULE NAME	INITIALIZE LCTC	MCU/MPU	HD6305X0/Y0	LABEL	L2HINT

FUNCTION

(1) Initializes LCTC for graphic display.

(2) Clears display RAM using for LM200.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	—	—	—
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	×

C	Z
×	×
N	I
×	●
H	
●	

SPECIFICATIONS

ROM (Bytes)
135
RAM (Bytes)
2
Stack (Bytes)
4
No. of cycles
509350
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Program module L2HINT has no arguments.

(b) Program module L2HINT execution places LCTC in graphic display mode and clears display on LM200.

(c) Program module L2HINT calls other program modules and subroutines shown in Table 3.

SPECIFICATIONS NOTES

(1) "SPECIFICATIONS" includes values in other program modules or subroutines called in program module L2HINT.

(2) "No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed when subroutine L2HBSY is executed by the minimum cycles.

DESCRIPTION

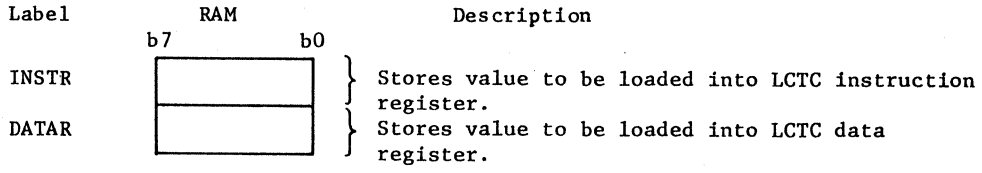
Table 3 Program Modules and Subroutines Called in L2HINT

Program Module/ Subroutine Name	Label	Function
WRITE DATA TO LCTC	L2HWT	Writes data to LCTC through I/O port of the HD6305X0.
CLEAR DISPLAY	L2HCLR	Clears display RAM to clear display.
MOVE CURSOR	L2HMVE	Loads cursor address into LCTC cursor address counter.
CHECK BUSY FLAG	L2HBSY	Checks LCTC busy flag.

(2) User Notes

Selects DDRs of port A and port C as output.

(3) RAM Description



(4) Sample Application

Program module L2HINT is called after selecting I/O ports.

```
        LDA   #$FF
        STA   PADDR      ---Selects port A and port C as output ports.
        STA   PCDDR
        JSR   L2HINT     ---Calls program module L2HINT.
```

(5) Basic Operation

(a) The instruction and data elements shown in Table 4 must be loaded into the instruction and data registers in corresponding pains to initialize LCTC. Program module L2HINT loads data shown in Table 4.

(b) RS signal is used to switch between the two registers;
High: Instruction register, Low: Data register

(c) These instructions and data are initially stored in INSTR(RAM) and DATAR(RAM) respectively using index addressing mode. If program module L2HWT is called, data is loaded into the instruction and data registers in LCTC.

(d) Program module L2HWT controls signals R/W, E and RS through port C.

(e) Subroutine L2HCLR is executed to clear display RAM for the liquid crystal module. Subroutine L2HCLR calls two program modules L2HMVE and L2HWT.

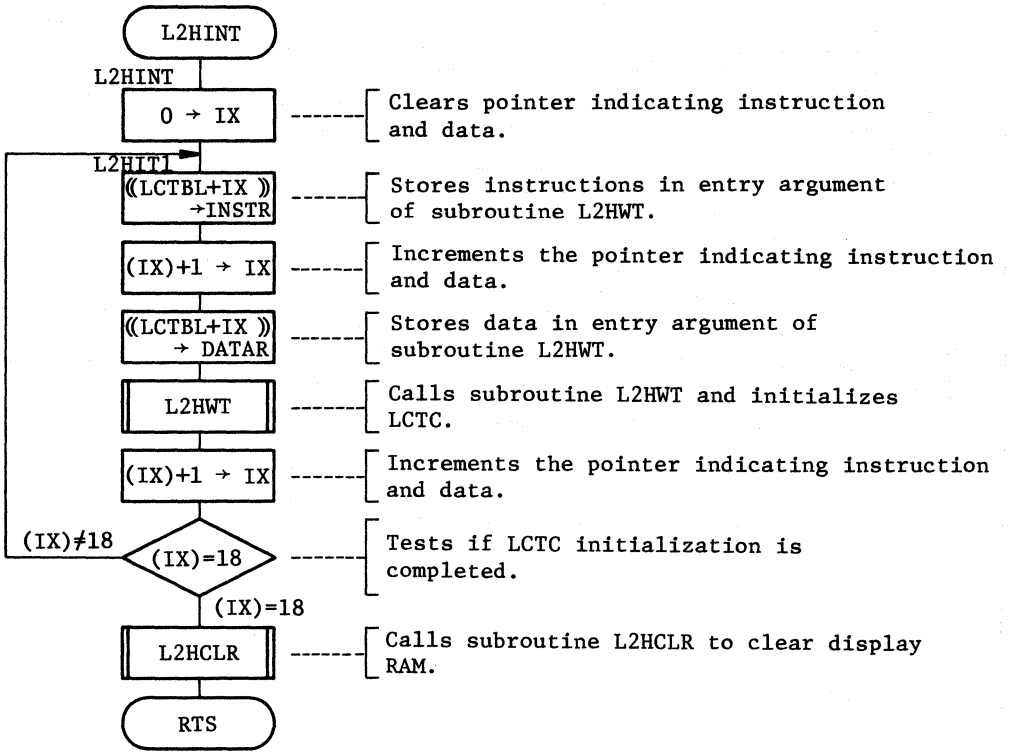
Table 4 Data to be stored in LCTC

Instruction	Data	Function
$00	$32	Selects Display ON, Master mode and Graphic mode.
$01	$07	Selects 8-bit of horizontal dots per charactor in display.
$02	$1D	Selects 30-byte of horizontal bytes in the graphic mode.
$03	$1F	Selects 1/32 duty in multiplex display.
$04	$00	Selects cursor position (note).
$08	$00	Selects display starting address to $0000.
$09	$00	
$0A	$00	Selects cursor address to $0000.
$0B	$00	

(Note) Cursor is not displayed in graphic mode.

PROGRAM MODULE NAME	INITIALIZE LCTC	MCU/MPU	HD6305X0/Y0	LABEL	L2HINT

FLOWCHART



PROGRAM MODULE NAME	MOVE CURSOR	MCU/MPU	HD6305X0/Y0	LABEL	L2HMVE

FUNCTION

Loads cursor address value stored in DTADDR(RAM) into LCTC cursor address counter.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Cursor Address Value	DTADDR (RAM)	2
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
●	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
70
RAM (Bytes)
4
Stack (Bytes)
4
No. of cycles
249
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details
DTADDR(RAM) : Holds cursor address value to be loaded into cursor address counter as 2-byte hexadecimal number.

(b) Program module L2HMVE loads cursor address value into cursor address counter to change cursor address on display.

(c) Program module L2HMVE calls other program modules and subroutines shown in Table 5.

SPECIFICATIONS NOTES

(1) "SPECIFICATIONS" includes values in other program modules or subroutines called in program module L2HINT.

(2) "No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed when subroutine L2HBSY is executed by the minimum cycles.

DESCRIPTION

Table 5 Program Modules and Subroutines called in L2HMVE

Program Module/ Subroutine Name	Label	Function
WRITE DATA TO LCTC	L2HWT	Writes data to LCTC through I/O port of the HD6305X0.
CHECK BUSY FLAG	L2HBSY	Checks LCTC busy flag.

(2) User Notes

Program module L2HMVE is called after selecting port A and port C.

(3) RAM Description

Label	RAM	Description
INSTR	b7 b0	Stores value to be transferred into LCTC instruction register.
DATAR		Stores value to be transferred into LCTC data register.
DTADDR	Upper / Lower	Stores cursor address value to be moved to LCTC cursor address counter as 2-byte hexadecimal number.

(4) Sample Application

Program module L2HMVE is called after selecting cursor address value and I/O ports.

```
WORK1   RMB    2          ----Reserves memory byte for 2-byte hexadecimal
                              address value.
        LDA    #$FF
        STA    PADDR      }--- Selects ports A and C as output
        STA    PCDDR      }
        LDA    WORK1
        STA    DTADDR     }--- Loads 2-byte hexadecimal cursor address value
        LDA    WORK1+1           set in user program into entry argument.
        STA    DTADDR+1   }
        JSR    L2HMVE     --- Calls program module L2HMVE.
```

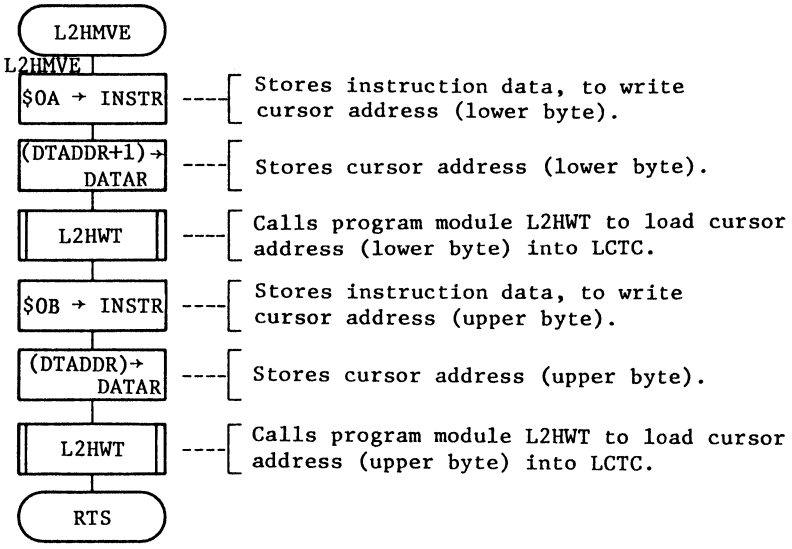
(5) Basic Operation

(a) To effect display at any location on LM200, the cursor address must be set before writing display data.

(b) The cursor address consists of two bytes. Program module L2HWT is used to set, first, the lower byte and then, the upper byte.

PROGRAM MODULE NAME	MOVE CURSOR	MCU/MPU	HD6305X0/Y0	LABEL	L2HMVE

FLOWCHART



7

PROGRAM MODULE NAME	WRITE DATA TO LCTC	MCU/MPU	HD6305X0/Y0	LABEL	L2HWT

FUNCTION

Writes instruction and data to LCTC, controlling LCTC control signals (RS, R/W and E) through I/O ports on the HD6305X0.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Instruction to LCTC	INSTR (RAM)	1
		Data to LCTC	DATAR (RAM)	1
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
●	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
47
RAM (Bytes)
2
Stack (Bytes)
2
No. of cycles
105
Reentrant
No
Relocation
No
Interrupt
Yes

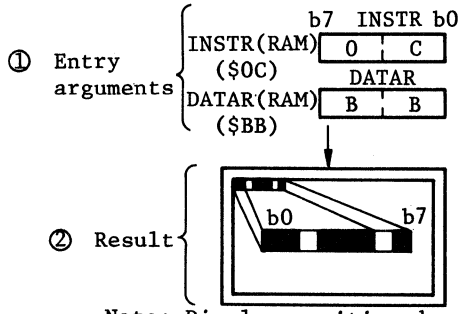
DESCRIPTION

(1) Function Details

(a) Argument details

INSTR(RAM): Holds value to be loaded into LCTC instruction register.

DATAR(RAM): Holds value to be loaded into LCTC data register.



Note: Display position depends on the cursor address.

Fig 6. Example of L2HWT Execution

SPECIFICATIONS NOTES

(1) "SPECIFICATIONS" includes values in subroutine L2HBSY.

(2) "No. of cycles" in "SPECIFICATIONS" represents the number of cycles required when subroutine L2HBSY is executed by the minimum cycles.

DESCRIPTION

(b) Fig. 6 shows an example of program module L2HWT execution. If program module L2HWT is executed, display data is written into LCTC.

(c) Program module L2HWT calls other subroutine shown in Table 6.

Table 6 Subroutine Called in L2HWT

Subroutine Name	Label	Function
CHECK BUSY FLAG	L2HBSY	Checks LCTC busy flag.

(2) User Notes

Selects DDRs of port A and port C as output.

(3) RAM Description

Label	RAM (b7 … b0)	Description
INSTR		Stores value to be loaded into LCTC instruction register.
DATAR		Stores value to be loaded into LCTC data register.

(4) Sample Application

Program module L2HWT is called after selecting I/O ports and instructions storing data to be written.

```
WORK 1   RMB    1---  Reserves memory byte for value to be loaded into
                      instruction register.
WORK 2   RMB    1---  Reserves memory byte for value to be loaded into
                      data register.
         LDA    #$FF      }
         STA    PADDR     }--- Selects port A and port C as output.
         STA    PCDDR     }
         LDA    WORK1     }--- Loads value to be loaded into instruction
         STA    INSTR     }    register into entry argument.
         LDA    WORK2     }--- Loads value to be loaded into data register
         STA    DATAR     }    into entry argument.
         JSR    L2HWT     --- Calls program module L2HWT.
```

(5) Basic Operation

(a) To write data to LCTC, the microcomputer must load corresponding pairs of data and instruction elements into LCTC data and instruction registers respectively. RS signal is used to switch between the instruction (High) and data (Low) registers. A flowchart of data writing example is shown in Fig. 7.

PROGRAM MODULE NAME	WRITE DATA TO LCTC	MCU/MPU	HD6305X0/Y0	LABEL	L2HWT

DESCRIPTION

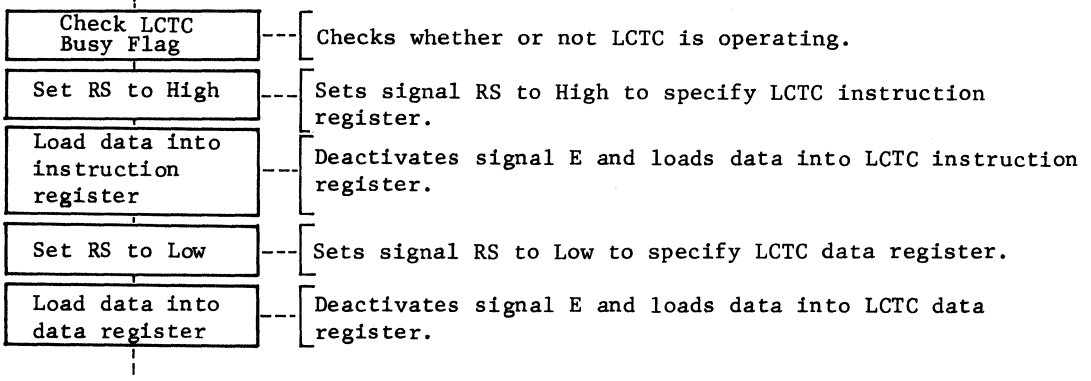


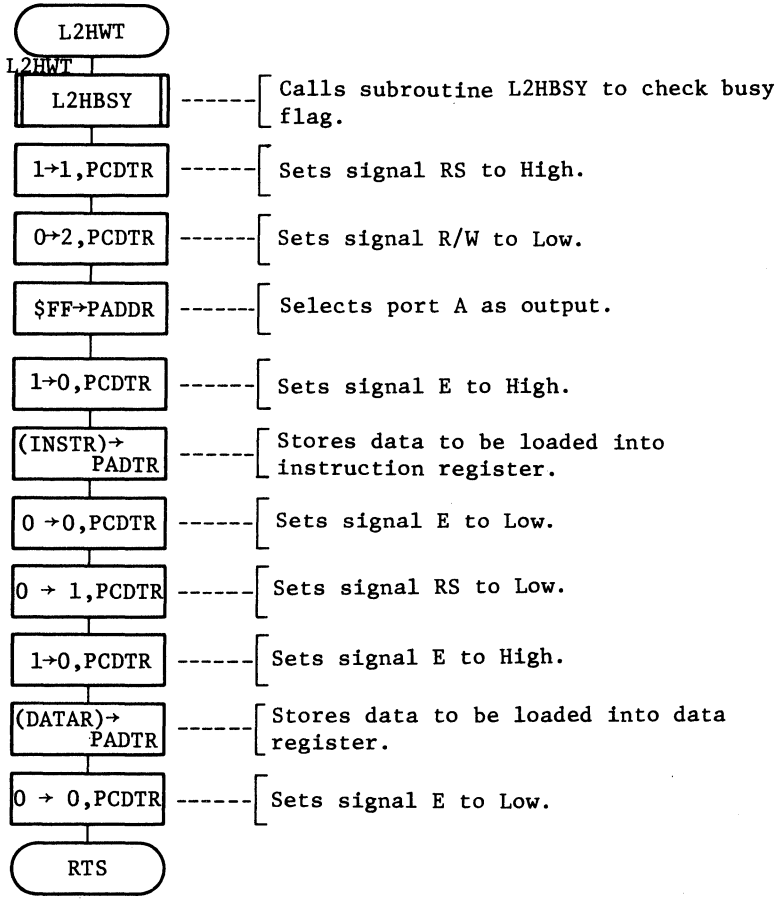
Fig. 7 Example of writing data into LCTC

(b) When LCTC is operating, data can not be written from the microcomputer. A busy flag indicates whether or not LCTC is in operation. Program module L2HWT checks LCTC status by calling subroutine L2HBSY.

Busy flag = 1 : Data can not be written.
Busy flag = 0 : Data can be written.

PROGRAM MODULE NAME	WRITE DATA TO LCTC	MCU/MPU	HD6305X0/Y0	LABEL	L2HWT

FLOWCHART



7

1.4 SUBROUTINE DESCRIPTION

SUBROUTINE NAME	CLEAR DISPLAY	MCU/MPU	HD6305X0/Y0	LABEL	L2HCLR

FUNCTION

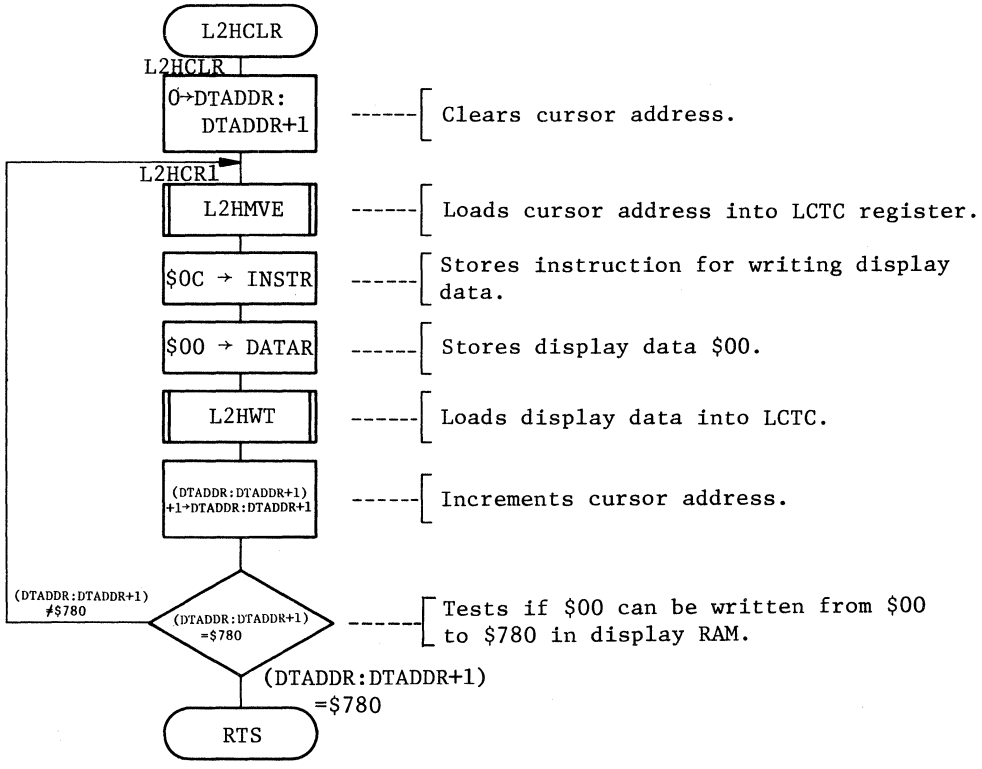
Clears display RAM and display on LM200.

BASIC OPERATION

(1) Calls program module L2HMVE to load cursor address into LCTC.

(2) Calls program module L2HWT to write $00 into cursor address specified in (1).

FLOWCHART | **PROGRAM MODULE USING THIS SUBROUTINE**: L2HINT



SUBROUTINE NAME	CHECK BUSY FLAG	MCU/MPU	HD6305X0/Y0	LABEL	L2HBSY

FUNCTION

(1) Checks busy flag to determine whether or not LCTC is in operation.

(2) Loops when LCTC is in operation.

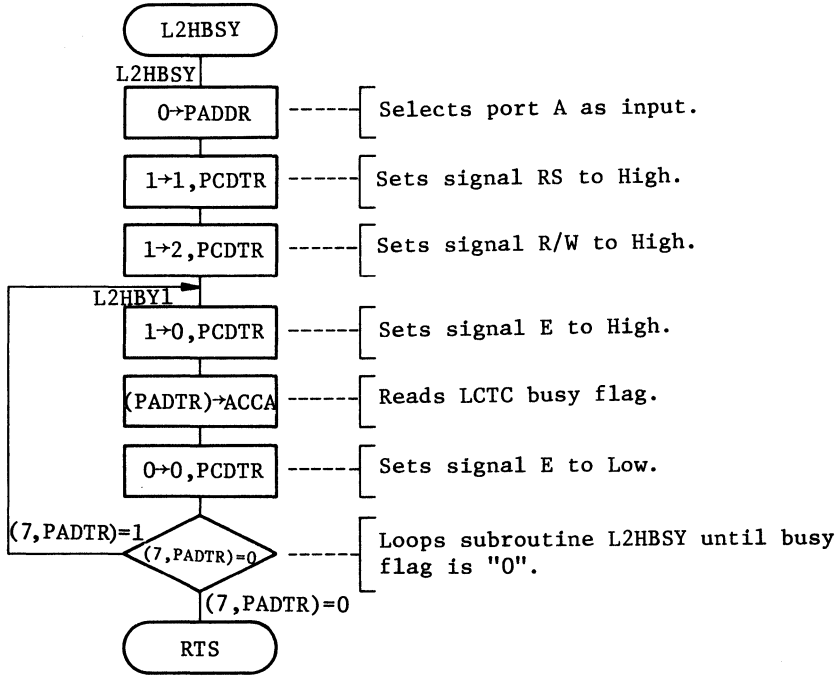
BASIC OPERATION

(1) Microcomputer must check LCTC state, because data from the microcomputer can not be accepted when LCTC is in operation.

(2) If signal RS is set to High, signal R/W to High and signal E to High through port C, the most significant bit (DB_7) in LCTC data bus ($DB_0 \sim DB_7$) serves as the busy flag.
DB_7 = 1 : Indicates LCTC is in operation.
DB_7 = 0 : Indicates data can be written to LCTC.

FLOWCHART

PROGRAM MODULE USING THIS SUBROUTINE: L2HWT



```
00001                    *
00002                    ****    RAM ALLOCATION    *************************
00003                    *
00004 0080               ORG     $80
00005                    *
00006 0080 0001  INSTR   RMB     1          LCTC instruction register data
00007 0081 0001  DATAR   RMB     1          LCTC data register data
00008 0082 0002  DTADDR  RMB     2          Cursor address
00009 0084 0001  COUNT1  RMB     1          Column counter
00010 0085 0001  COUNT2  RMB     1          Row counter
00011                    *
00012                    ****    SYMBOL DEFINITIONS **************
00013                    *
00014       0000 PADTR   EQU     $00        Port A data register
00015       0002 PCDTR   EQU     $02        Port C data register
00016       0004 PADDR   EQU     $04        Port A data direction register
00017       0006 PCDDR   EQU     $06        Port C data direction register
00018                    ************************************************
00019                    *                                              *
00020                    *             MAIN PROGRAM : L2HMN             *
00021                    *                                              *
00022                    ************************************************
00023                    *
00024 1000               ORG     $1000
00025                    *
00026 1000 4F    L2HMN   CLR     A
00027 1001 B7 00         STA     PADTR      Initialige port A
00028 1003 B7 02         STA     PCDTR      Initialige port C
00029 1005 A6 FF         LDA     #$FF
00030 1007 B7 04         STA     PADDR      Select port A as output
00031 1009 B7 06         STA     PCDDR      Select port C as output
00032 100B CD 104C       JSR     L2HINT     Initialize LCTC
00033 100E 5F            CLR     X          Clear pointer of display data
00034 100F BF 84         STX     COUNT1     Clear column counter
00035 1011 BF 85         STX     COUNT2     Clear row counter
00036 1013 A6 00         LDA     #$00
00037 1015 B7 82         STA     DTADDR     Store cursor address (upper)
00038 1017 A6 84         LDA     #$84
00039 1019 B7 83         STA     DTADDR+1   Store cursor address (lower)
00040 101B CD 1082 L2HMN1 JSR    L2HMVE     Write cursor address to LCTC
00041 101E A6 0C         LDA     #$0C       Store instruction
00042 1020 B7 80         STA     INSTR
00043 1022 D6 10E6 L2HMN2 LDA    DSDAT,X    Store data
00044 1025 B7 81         STA     DATAR
00045 1027 CD 1064       JSR     L2HWT      Write display data to LCTC
00046 102A 3C 84         INC     COUNT1     Increment column counter
00047 102C 5C            INC     X          Increment pointer
00048 102D B6 84         LDA     COUNT1     Test if column counter=5?
00049 102F A1 05         CMP     #5
00050 1031 26 EF         BNE     L2HMN2
00051 1033 4F            CLR     A          Clear column counter
00052 1034 B7 84         STA     COUNT1
00053 1036 B6 83         LDA     DTADDR+1   Add cursor address
00054 1038 AB 1E         ADD     #30
00055 103A B7 83         STA     DTADDR+1
```

```
00056 103C B6 82          LDA    DTADDR
00057 103E A9 00          ADC    #0
00058 1040 B7 82          STA    DTADDR
00059 1042 3C 85          INC    COUNT2    Increment row counter
00060 1044 B6 85          LDA    COUNT2    Test if row counter=51?
00061 1046 A1 33          CMP    #51
00062 1048 26 D1          BNE    L2HMN1    Loop until display end
00063 104A 20 FE    PEND  BRA    PEND      End of main program
00064                     ********************************************
00065                     *                                          *
00066                     *       NAME : L2HINT (INITIALIZE LCTC)    *
00067                     *                                          *
00068                     ********************************************
00069                     *                                          *
00070                     *      ENTRY : NOTHING                     *
00071                     *    RETURNS : NOTHING                     *
00072                     *                                          *
00073                     ********************************************
00074 104C 5F       L2HINT CLR    X         Clear data table pointer
00075 104D D6 10D4  L2HIT1 LDA    LCTBL,X   Store LCTC instruction
00076 1050 B7 80          STA    INSTR
00077 1052 5C             INC    X         Increment data table pointe
00078 1053 D6 10D4        LDA    LCTBL,X   Store LCTC data
00079 1056 B7 81          STA    DATAR
00080 1058 CD 1064        JSR    L2HWT     Write to LCTC
00081 105B 5C             INC    X         Increment data table pointe
00082 105C A3 12          CPX    #18       IX = 18 ?
00083 105E 26 ED          BNE    L2HIT1    Test if LCTC is initialized
00084 1060 CD 1099        JSR    L2HCLR    Clear display RAM
00085 1063 81             RTS
00086                     ********************************************
00087                     *                                          *
00088                     *       NAME : L2HWT (WRITE DATA TO LCTC)  *
00089                     *                                          *
00090                     ********************************************
00091                     *                                          *
00092                     *      ENTRY : INSTR (Instruction to LCTC) *
00093                     *              DATAR (Data to LCTC)        *
00094                     *    RETURNS : NOTING                      *
00095                     *                                          *
00096                     ********************************************
00097 1064 CD 10C2  L2HWT  JSR    L2HBSY    Check if LCTC is busy
00098 1067 12 02           BSET   1,PCDTR   Set RS=1
00099 1069 15 02           BCLR   2,PCDTR   Set RW=0
00100 106B A6 FF           LDA    #$FF      Select port A as output
00101 106D B7 04           STA    PADDR
00102 106F 10 02           BSET   0,PCDTR   Set E=1
00103 1071 B6 80           LDA    INSTR     Store instruction
00104 1073 B7 00           STA    PADTR
00105 1075 11 02           BCLR   0,PCDTR   Set E=0
00106 1077 13 02           BCLR   1,PCDTR   Set RS=0
00107 1079 10 02           BSET   0,PCDTR   Set E=1
00108 107B B6 81           LDA    DATAR     Store data
00109 107D B7 00           STA    PADTR
00110 107F 11 02           BCLR   0,PCDTR   Set E=0
```

```
00111 1081 81              RTS
00112                      ************************************************
00113                      *                                              *
00114                      *      NAME : L2HMVE (SET CURSOR ADDRESS)      *
00115                      *                                              *
00116                      ************************************************
00117                      *                                              *
00118                      *     ENTRY : DTADDR (CURSOR ADDRESS VALUE)    *
00119                      *   RETURNS : NOTING                           *
00120                      *                                              *
00121                      ************************************************
00122 1082 A6 0A    L2HMVE LDA    #$0A      Store instruction
00123 1084 B7 80           STA    INSTR
00124 1086 B6 83           LDA    DTADDR+1  Store lower cursor address
00125 1088 B7 81           STA    DATAR
00126 108A CD 1064         JSR    L2HWT     Write to LCTC
00127 108D A6 0B           LDA    #$0B      Store instruction
00128 108F B7 80           STA    INSTR
00129 1091 B6 82           LDA    DTADDR    Store upper cursor address
00130 1093 B7 81           STA    DATAR
00131 1095 CD 1064         JSR    L2HWT     Write to LCTC
00132 1098 81              RTS
00133                      ************************************************
00134                      *                                              *
00135                      *      NAME : L2HCLR (CLEAR DISPLAY)           *
00136                      *                                              *
00137                      ************************************************
00138 1099 3F 82    L2HCLR CLR    DTADDR    Clear cursor address
00139 109B 3F 83           CLR    DTADDR+1
00140 109D CD 1082  L2HCR1 JSR    L2HMVE    Write cursor address to LCTC
00141 10A0 A6 0C           LDA    #$0C      Store instruction
00142 10A2 B7 80           STA    INSTR
00143 10A4 3F 81           CLR    DATAR     Clear display data
00144 10A6 CD 1064         JSR    L2HWT     Write to LCTC
00145 10A9 B6 83           LDA    DTADDR+1  Increment cursor address
00146 10AB AB 01           ADD    #1
00147 10AD B7 83           STA    DTADDR+1
00148 10AF B6 82           LDA    DTADDR
00149 10B1 A9 00           ADC    #0
00150 10B3 B7 82           STA    DTADDR
00151 10B5 B6 82           LDA    DTADDR    Test if all display RAM is
00152 10B7 A1 07           CMP    #$7                                  cleared
00153 10B9 26 E2           BNE    L2HCR1
00154 10BB B6 82           LDA    DTADDR
00155 10BD A1 80           CMP    #$80
00156 10BF 26 DC           BNE    L2HCR1
00157 10C1 81              RTS
00158                      ************************************************
00159                      *                                              *
00160                      *      NAME : L2HBSY (CHECK BUSY FLAG)         *
00161                      *                                              *
00162                      ************************************************
00163 10C2 4F       L2HBSY CLR    A         Select port A as input
00164 10C3 B7 04           STA    PADDR
00165 10C5 12 02           BSET   1,PCDTR   Set RS=1
```

```
00166 10C7 14 02          BSET   2,PCDTR  Set RW=1
00167 10C9 10 02    L2HBY1 BSET   0,PCDTR  Set E=1
00168 10CB B6 00           LDA    PADTR    Read busy flag
00169 10CD 11 02           BCLR   0,PCDTR  Set E=0
00170 10CF A5 80           BIT    #$80     Test if busy flag = 0 ?
00171 10D1 26 F6           BNE    L2HBY1   Loop until busy flag=0
00172 10D3 81              RTS
00173                ***********************************************
00174                *                                             *
00175                *                DATA TABLE                   *
00176                *                                             *
00177                ***********************************************
00178 10D4 00       LCTBL  FCB    $00,$32  *Data for initialize LCTC
00179 10D6 01              FCB    $01,$07
00180 10D8 02              FCB    $02,$1D
00181 10DA 03              FCB    $03,$1F
00182 10DC 04              FCB    $04,$00
00183 10DE 08              FCB    $08,$00
00184 10E0 09              FCB    $09,$00
00185 10E2 0A              FCB    $0A,$00
00186 10E4 0B              FCB    $0B,$00
00187 10E6 00       DSDAT  FCB    $00,$00,$80,$0F,$00  *Data for display
00188 10EB C0              FCB    $C0,$00,$C0,$1F,$00
00189 10F0 E0              FCB    $E0,$01,$C0,$1F,$00
00190 10F5 F8              FCB    $F8,$03,$E7,$3F,$00
00191 10FA F8              FCB    $F8,$C7,$F8,$3F,$00
00192 10FF FC              FCB    $FC,$37,$C0,$7F,$00
00193 1104 FE              FCB    $FE,$0F,$80,$7F,$00
00194 1109 FE              FCB    $FE,$07,$80,$3F,$00
00195 110E FE              FCB    $FE,$03,$C3,$3F,$00
00196 1113 FF              FCB    $FF,$83,$E7,$3F,$00
00197 1118 FF              FCB    $FF,$C1,$2C,$1C,$00
00198 111D FE              FCB    $FE,$E0,$68,$2D,$00
00199 1122 7E              FCB    $7E,$70,$CE,$4C,$00
00200 1127 3E              FCB    $3E,$F0,$CC,$8E,$00
00201 112C 3C              FCB    $3C,$F0,$8F,$87,$00
00202 1131 10              FCB    $10,$F0,$0F,$0F,$01
00203 1136 10              FCB    $10,$E0,$87,$13,$01
00204 113B 10              FCB    $10,$E0,$83,$1B,$02
00205 1140 08              FCB    $08,$80,$01,$0F,$02
00206 1145 08              FCB    $08,$00,$00,$06,$02
00207 114A 08              FCB    $08,$00,$00,$00,$02
00208 114F 08              FCB    $08,$00,$00,$00,$01
00209 1154 10              FCB    $10,$00,$00,$00,$01
00210 1159 10              FCB    $10,$00,$40,$08,$01
00211 115E 20              FCB    $20,$80,$A0,$84,$00
00212 1163 20              FCB    $20,$00,$13,$C3,$01
00213 1168 40              FCB    $40,$00,$0C,$C0,$01
00214 116D 80              FCB    $80,$01,$00,$F8,$03
00215 1172 00              FCB    $00,$06,$C0,$F7,$07
00216 1177 00              FCB    $00,$02,$3C,$E0,$07
00217 117C 00              FCB    $00,$01,$40,$F8,$01
00218 1181 00              FCB    $00,$03,$80,$FF,$03
00219 1186 80              FCB    $80,$07,$00,$FF,$03
00220 118B 80              FCB    $80,$0F,$00,$9F,$03
```

```
00221 1190 C0              FCB     $C0,$0F,$00,$0E,$00
00222 1195 C0              FCB     $C0,$0F,$00,$02,$00
00223 119A C0              FCB     $C0,$1F,$00,$02,$00
00224 119F E0              FCB     $E0,$1F,$00,$03,$00
00225 11A4 E0              FCB     $E0,$0F,$C6,$07,$00
00226 11A9 E0              FCB     $E0,$0F,$DF,$1E,$00
00227 11AE E0              FCB     $E0,$AF,$FA,$1A,$00
00228 11B3 E0              FCB     $E0,$EF,$FA,$1A,$00
00229 11B8 D0              FCB     $D0,$77,$FB,$1A,$00
00230 11BD A8              FCB     $A8,$7B,$FB,$3F,$00
00231 11C2 64              FCB     $64,$FD,$FF,$3F,$00
00232 11C7 C2              FCB     $C2,$FE,$FF,$3F,$00
00233 11CC C2              FCB     $C2,$FF,$FF,$1F,$00
00234 11D1 84              FCB     $84,$FF,$FF,$1F,$00
00235 11D6 98              FCB     $98,$FF,$FF,$0F,$00
00236 11DB 60              FCB     $60,$FF,$FF,$0F,$00
00237 11E0 00              FCB     $00,$FE,$DF,$07,$00
00238                ************************************************
00239                *                                              *
00240                *              VECTOR ADDRESSES                *
00241                *                                              *
00242                ************************************************
00243                *
00244 1FF6                 ORG     $1FF6
00245                *
00246 1FF6 1000            FDB     L2HMN     SCI/TIMER2
00247 1FF8 1000            FDB     L2HMN     TIMER/INT2
00248 1FFA 1000            FDB     L2HMN     INT
00249 1FFC 1000            FDB     L2HMN     SWI
00250 1FFE 1000            FDB     L2HMN     RES
00251                *
00252                      END
```

2. LIQUID CRYSTAL MODULE (H2570) CONTROL

2.1 HARDWARE DESCRIPTION

(1) Function

(a) Controls dot matrix liquid crystal controller driver HD44780 (hereinafter, LCD-II) using the HD6305X0 and displays in character mode on liquid crystal module H2570.

(b) H2570 displays 5×7 dot characters in 1-column × 16-row.

(c) Transfers ASCII from the HD6305X0 as display data to LCD-II. LCD-II automatically controls liquid crystal driver HD44100 and LCD by controlling LCD-II with the HD6305X0.

(2) Microcomputer Applications

Displays characters on H2570 by controlling LCD-II data bus ($DB_0 \sim DB_7$) and control signals (E, RS and R/W) through port G and port F.

(3) Circuit Diagram

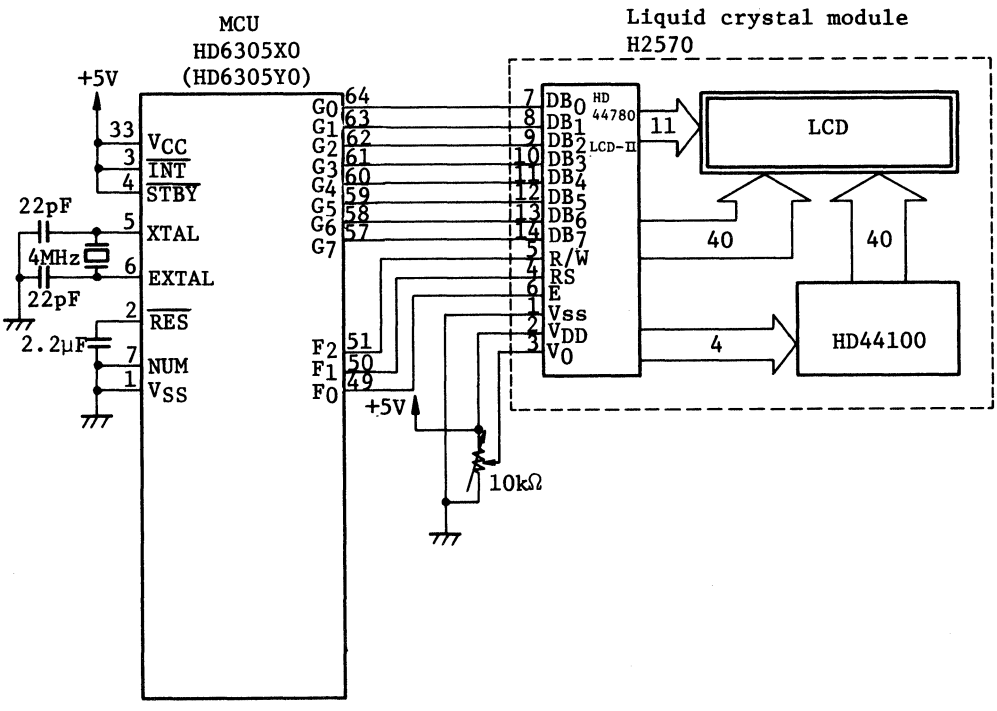


Fig. 1 H2570 Control Circuit

(4) Pin Functions

Pin functions at the interface between the HD6305X0 and LCD-II are shown in Table 1.

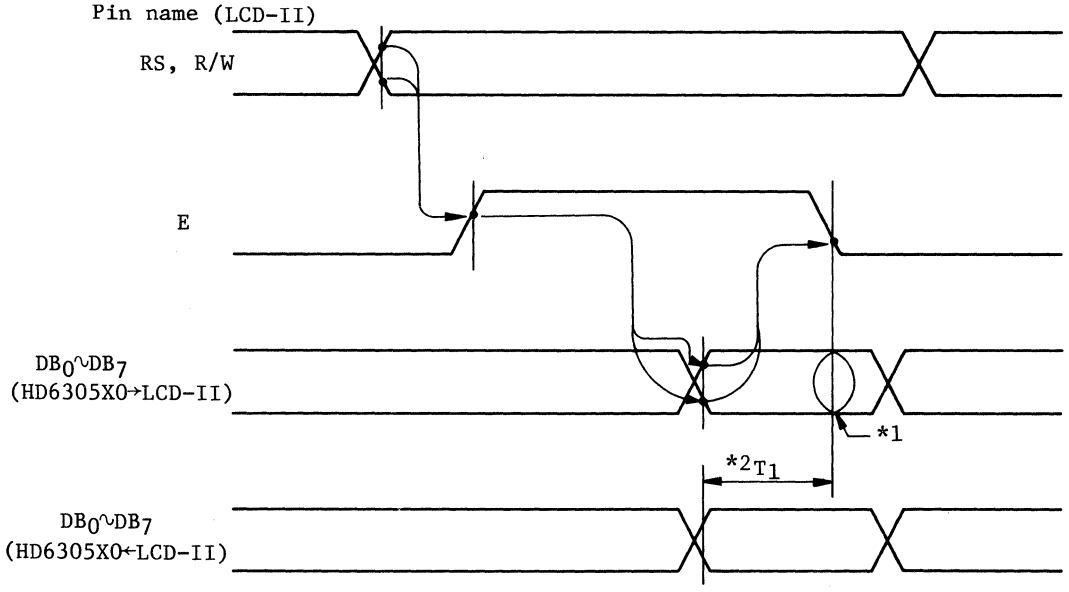
Table 1 Pin Functions

Pin Name (HD6305X0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (LCD-II)	Program Label
F_0	Output	High	Enable signal	E	PFDTR
F_1	Output	Low	Selects instruction register	RS	
		High	Selects data register		
F_2	Output	Low	Writes data (Microcomputer → LCD-II)	R/W	
		High	Reads data (Microcomputer ← LCD-II)		
G_0	Input/ Output	—	Data lines	DB_0	PGDTR
G_1	Input/ Output	—		DB_1	
G_2	Input/ Output	—		DB_2	
G_3	Input/ Output	—		DB_3	
G_4	Input/ Output	—		DB_4	
G_5	Input/ Output	—		DB_5	
G_6	Input/ Output	—		DB_6	
G_7	Input/ Output	—		DB_7	

(5) Hardware Operation

LCD-II control signals (RS, R/W, E) are controlled by the program. Each LCD-II control signal timing chart is shown in Fig. 2.

The HD6305 family utilizes I/O port rather than buses to control LCD-II to eliminate any timing restrictions on LCD-II.



*1 Data is written to LCTC at the falling edge of E.

*2 Data from LCD-II can be read during period T_1.

Fig. 2 HD6305X0 ↔ LCD-II Interface

2.2 SOFTWARE DESCRIPTION

(1) Program Module Configuration

The program module configuration for character display on the liquid crystal module is shown in Fig. 3.

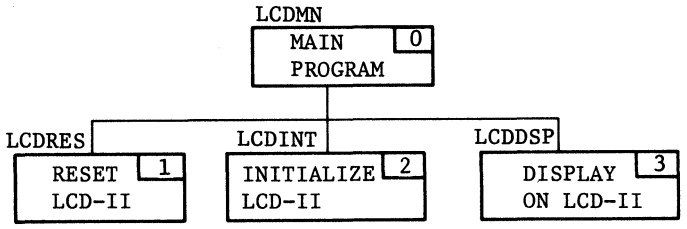


Fig. 3 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	LCDMN	Demonstrates character display on H2570.
1	RESET LCD-II	LCDRES	Resets LCD-II using instruction.
2	INITIALIZE LCD-II	LCDINT	Initializes LCD-II to display characters on LCD.
3	DISPLAY ON LCD	LCDDSP	Transfers ASCII code to LCD-II and displays on H2570.

(3) Program Module Sample Application (Main Program)

The flowchart in Fig. 4 is an example of character display on H2570 performed by program module in Fig. 3.

The program in Fig. 4 demonstrates the display on liquid crystal shown in Fig. 5.

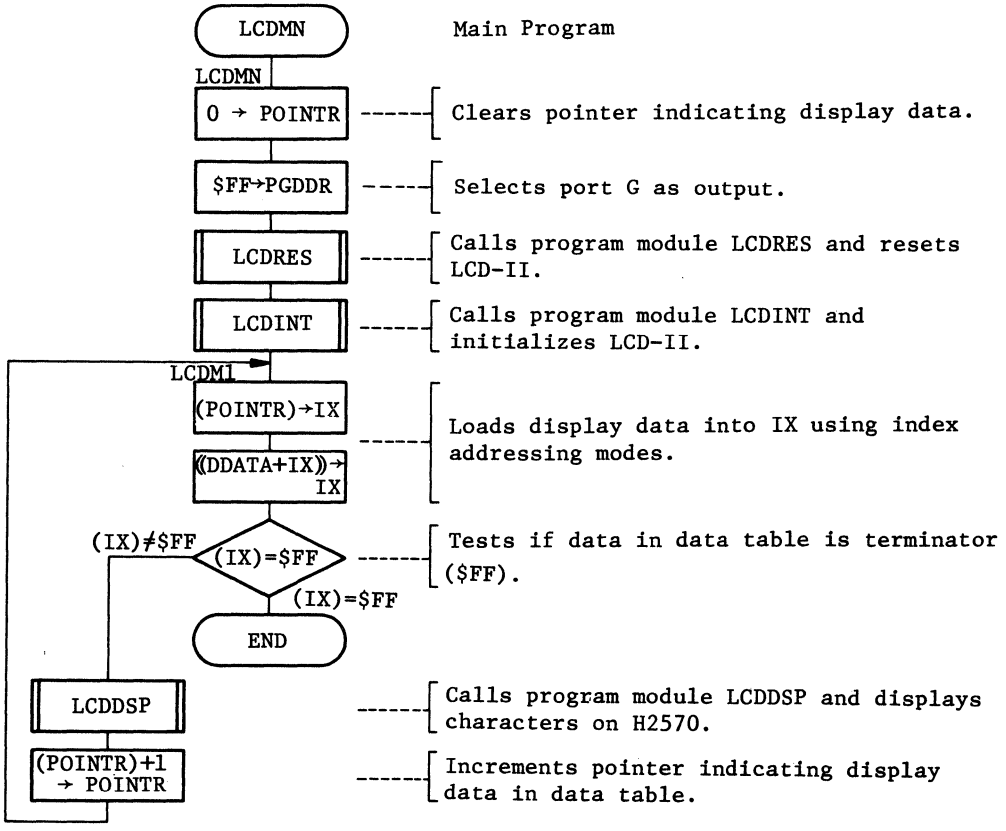


Fig. 4 Program Module Flowchart

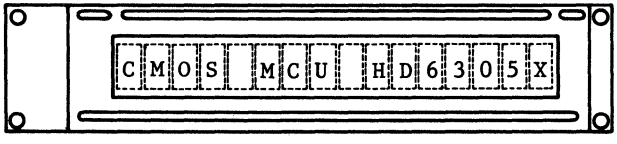


Fig. 5 Example of displaying on LCD

2.3 PROGRAM MODULE DESCRIPTION

PROGRAM MODULE NAME	RESET LCD-II	MCU/MPU	HD6305X0/Y0	LABEL	LCDRES

FUNCTION

Resets LCD-II using instruction.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	—	—	—
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	×

C	Z
●	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
32
RAM (Bytes)
1
Stack (Bytes)
0
No. of cycles
46426
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Program module LCDRES has no arguments.

(b) Resets LCD-II if power supply condition for built-in reset circuit is not suitable.

(c) Program module LCDRES calls neither program modules nor subroutines.

(2) User Notes

Selects DDR of port G as output.

SPECIFICATIONS NOTES

PROGRAM MODULE NAME	RESET LCD-II	MCU/MPU	HD6305X0/Y0	LABEL	LCDRES

DESCRIPTION

(3) RAM Description

Label	RAM	Description
LCDCNT	b7 b0	Stores number of LCD-II resets.

(4) Sample Application

Calls program module LCDRES after selecting I/O port.

```
        :
LDA     #$FF    }------ Selects port G as output.
STA     PGDDR   }
JSR     LCDRES  ------Calls program module LCDRES.
        :
```

(5) Basic operation

(a) Sets $30 to LCD-II with delay time as shown in Fig. 6.

(b) I/O port controls signals RS, R/W and E, and transfers $30.

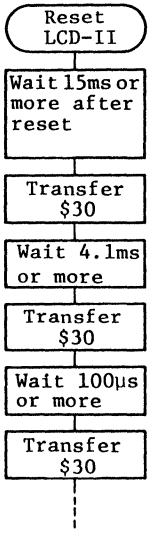
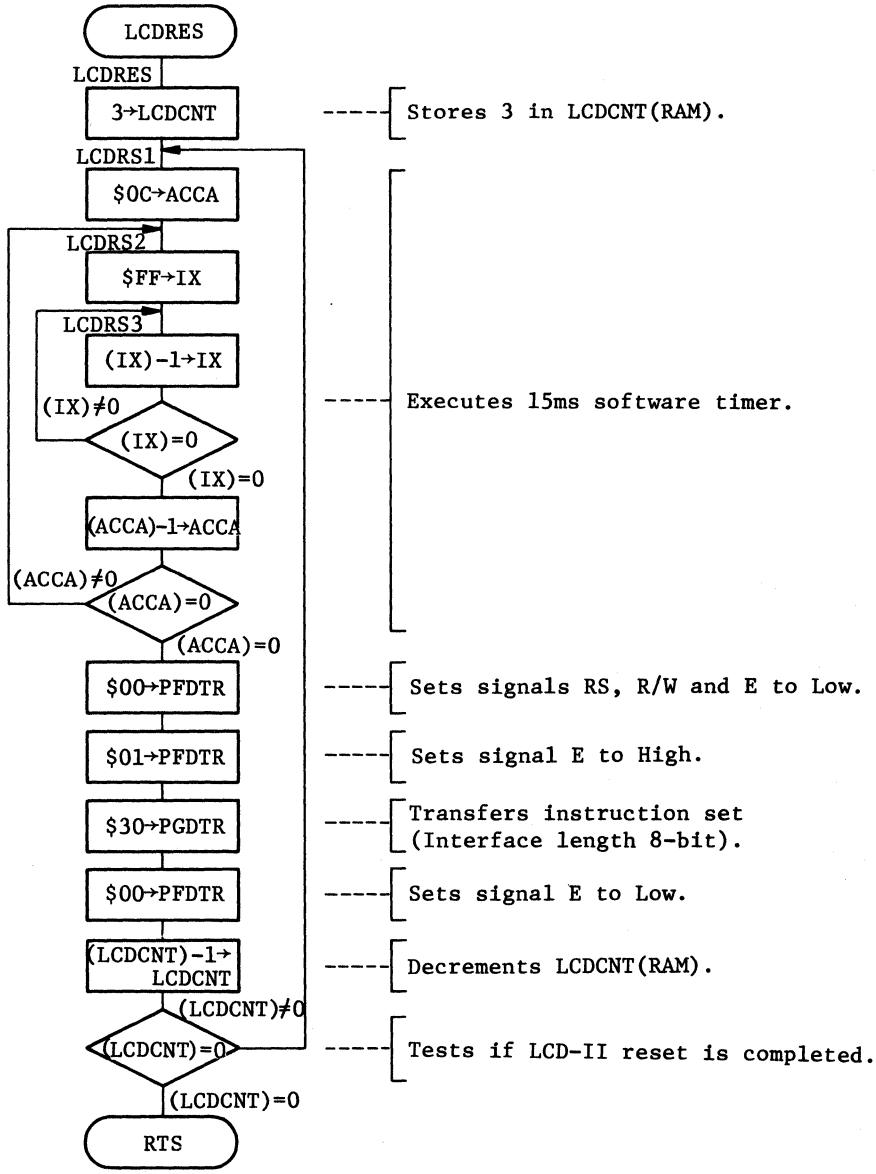


Fig. 6 LCD-II Reset Performed by Program

(c) Executes 15ms software timer and transferring $30 three times using loop process to reset LCD-II as shown in Fig. 6.

PROGRAM MODULE NAME	RESET LCD-II	MCU/MPU	HD6305X0/Y0	LABEL	LCDRES

FLOWCHART



PROGRAM MODULE NAME	INITIALIZE LCD-II	MCU/MPU	HD6305X0/Y0	LABEL	LCDINT

FUNCTION

Initializes LCD-II display mode.

ARGUMENTS

		Contents	Storage Location	Byte Lgth.
Arguments	Entry	—	—	—
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
×	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
48
RAM (Bytes)
0
Stack (Bytes)
2
No. of cycles
439
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Program module LCDINT has no arguments.

(b) Program module LCDINT clears display, selects shift left mode, auto-increments DDRAM address, defines DDRAM address and sets LCD "on".

(c) Program module LCDINT calls subroutines shown in Table 3.

SPECIFICATIONS NOTES

(1) "SPECIFICATIONS" includes subroutine LCDBSY.

(2) "No. of cycles" in "SPECIFICATIONS" represents the number of cycles required when subroutine LCDBSY is executed by the minimum cycles.

PROGRAM MODULE NAME	INITIALIZE LCD-II	MCU/MPU	HD6305X0/Y0	LABEL	LCDINT

DESCRIPTION

Table 3 Subroutine Called in LCDINT

Subroutine Name	Label	Function
CHECK BUSY FLAG	LCDBSY	Checks LCD-II busy.

(2) User Notes

(a) Selects DDR of port G as output.

(b) Calls program module LCDRES and reset LCD-II to initialize LCD-II.

(3) RAM Description

RAM is not used in program module LCDINT.

(4) Sample Application

Calls program module LCDINT after selecting I/O port and resetting LCD-II.

```
        LDA     #$FF
        STA     PGDDR     ---Selects port G as output.
        JSR     LCDRES    ---Calls program module LCDRES and resets LCD-II.
        JSR     LCDINT    ---Calls program module LCDINT.
```

(5) Basic Operation

(a) Program module LCDINT loads data shown in Table 4 into instruction register when initializing LCD-II.

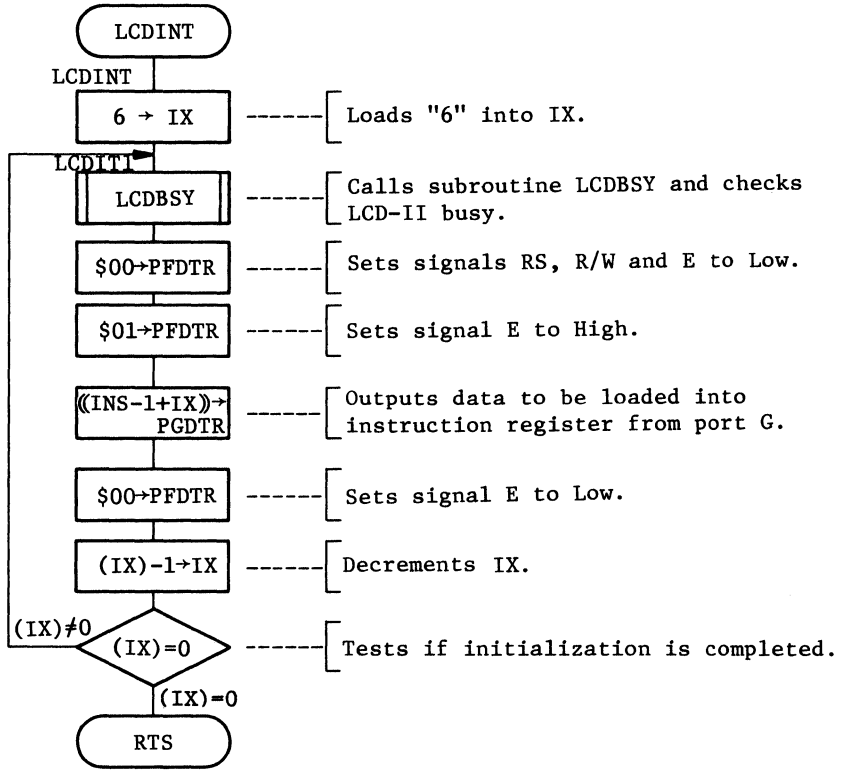
(b) Stores data shown in Table 4 in port G and controls signals E, R/W and RS of LCD-II by using port F, and sets them to LCD-II.

Table 4 LCD-II initialization data

Data	Function
$08	Turns off display.
$01	Clears all display. Loads $00 into DDRAM address.
$07	Selects cursor direction to right and shifts display.
$90	Loads $10 into DDRAM address.
$18	Selects cursor direction to left.
$0C	Turns on display.

PROGRAM MODULE NAME	INITIALIZE LCD-II	MCU/MPU	HD6305X0/Y0	LABEL	LCDINT

FLOWCHART



7

PROGRAM MODULE NAME	DISPLAY ON LCD	MCU/MPU	HD6305X0/Y0	LABEL	LCDDSP

FUNCTION

Stores ASCII code in DDRAM of LCD-II and displays characters on LCD.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Display data	IX	1
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
×	×
N	I
×	●
H	
●	

SPECIFICATIONS

ROM (Bytes)
42
RAM (Bytes)
0
Stack (Bytes)
2
No. of cycles
67
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details
IX: Holds 1-byte ASCII.

(b) Fig. 7 shows an example of program module LCDDSP execution. If entry argument is held in IX as shown in part ① of Fig. 7, ASCII is stored in current address of DDRAM, and characters are displayed on LCD.

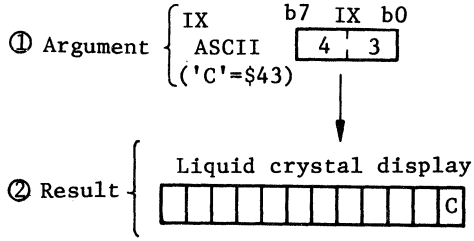


Fig. 7 Example of LCDDSP Execution

SPECIFICATIONS NOTES

(1) "SPECIFICATIONS" includes subroutine LCDBSY.

(2) "No. of cycles" in "SPECIFICATIONS" represents the number of cycles required when subroutine LCDBSY is executed by the minimum cycles.

(c) Program module LCDDSP calls subroutines shown in Table 5.

Table 5 Subroutine Called in LCDDSP

Subroutine Name	Label	Function
CHECK BUSY FLAG	LCDBSY	Checks busy flag.

(2) User Notes

(a) Selects DDR of port G as output.

(b) Calls program modules LCDRES and LCDINT and initializes LCD-II to make H2570 display characters using LCD-II.

(3) RAM Description

RAM is not used in program module LCDDSP.

(4) Sample Application

Program module LCDDSP is called after selecting I/O port, resetting LCD-II, initializing LCD-II and storing display data.

```
        :
    LDA     #$FF
    STA     PGDDR       }--Selects port G as output.
    JSR     LCDRES ----Calls program module LCDRES and resets LCD-II.
    JSR     LCDINT ----Calls program module LCDINT and initializes LCD-II.
    LDX     #$41   ----Stores display data in entry argument.
    JSR     LCDDSP  }--Calls program module LCDDSP.
        :
```

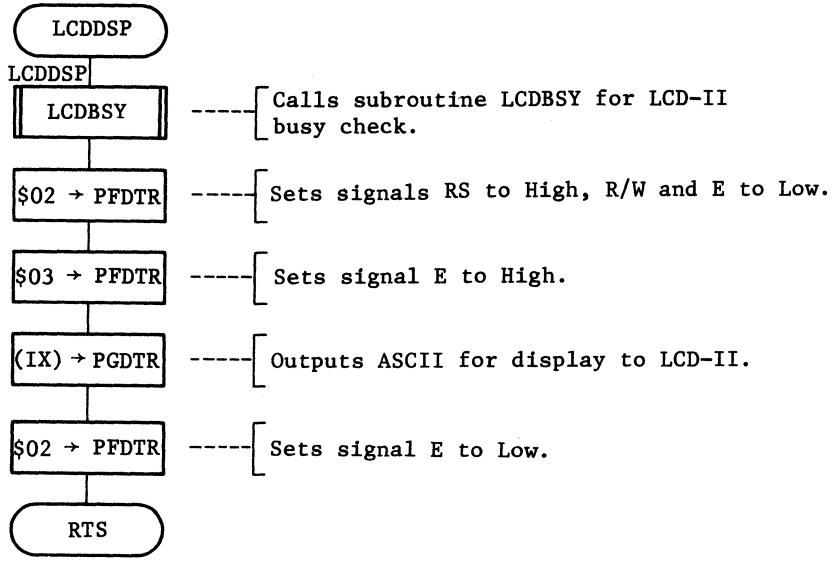
(5) Basic Operation

(a) Microcomputer cannot write data into LCD-II when LCD-II is in operation. Whether LCD-II is in operation or not can be checked by busy flag. Therefore, program module LCDDSP calls subroutine LCDBSY and checks LCD-II busy flag to test if LCD-II is in operation.
Busy flag = 1 : LCD-II is in operation. Data cannot be written.
Busy flag = 0 : Data can be written into LCD-II.

(b) Controls LCD-II control signal and writes data into LCD-II with timing illustrated in "2.1 HARDWARE DESCRIPTION, (5) Hardware Operation".

PROGRAM MODULE NAME	DISPLAY ON LCD	MCU/MPU	HD6305X0/Y0	LABEL	LCDDSP

FLOWCHART



2.4 SUBROUTINE DESCRIPTION

SUBROUTINE NAME	CHECK BUSY FLAG	MCU/MPU	HD6305X0/Y0	LABEL	LCDBSY

FUNCTION

(1) Tests LCD-II status.

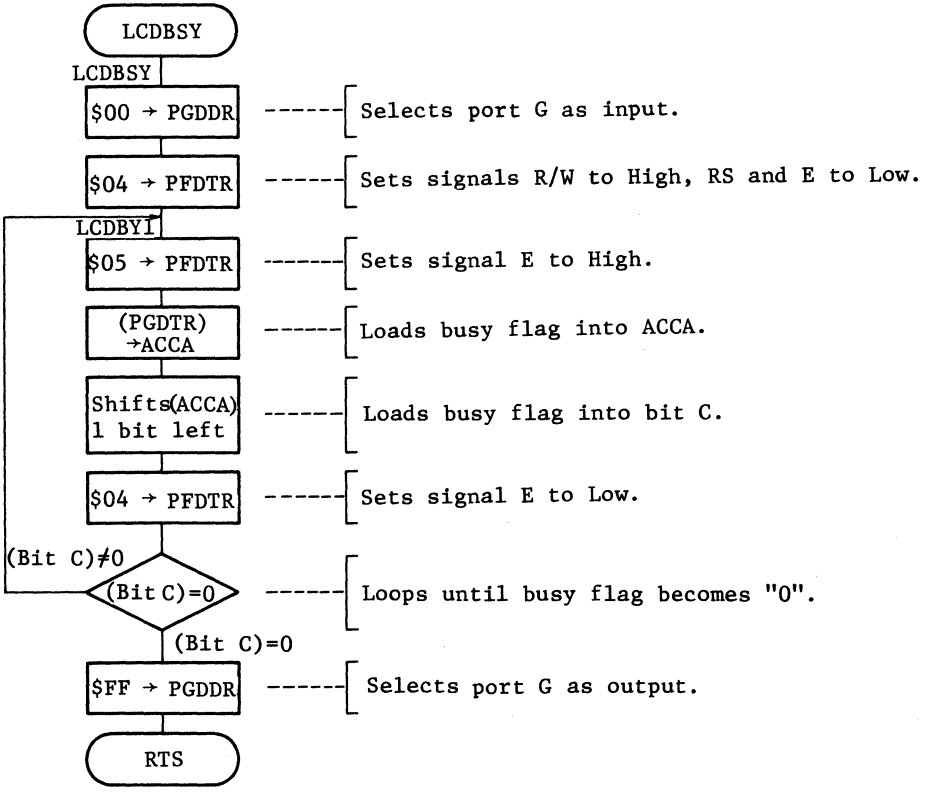
(2) Loops subroutine LCDBSY until LCD-II becomes READY.

BASIC OPERATION

The MSB of data bus becomes busy flag if LCD-II data bus is read during RS=0, R/W=1 and E=1.

FLOWCHART

PROGRAM MODULE USING THIS SUBROUTINE: LCDINT , LCDDSP



```
00001                    *
00002                    ****    RAM ALLOCATION    ************************
00003                    *
00004 0080               ORG     $80
00005                    *
00006 0080 0001     POINTR RMB    1          Pointer of display data table
00007 0081 0001     LCDCNT RMB    1          Loop counter
00008                    *
00009                    ****    SYMBOL DEFINITIONS    *************
00010                    *
00011      000C     PFDTR  EQU    $0C        Port F data register
00012      000D     PGDTR  EQU    $0D        Port G data register
00013      0007     PGDDR  EQU    $07        Port G data direction register
00014                    ************************************************
00015                    *                                              *
00016                    *             MAIN PROGRAM : LCDMN             *
00017                    *                                              *
00018                    ************************************************
00019                    *
00020 1000               ORG    $1000
00021                    *
00022 1000 4F       LCDMN  CLR    A          Clear pointer
00023 1001 B7 80         STA    POINTR
00024 1003 B7 0D         STA    PGDTR      Initialize port G
00025 1005 A6 FF         LDA    #$FF
00026 1007 B7 07         STA    PGDDR      Select port G as output
00027 1009 CD 1021       JSR    LCDRES     Reset LCD-II
00028 100C CD 1041       JSR    LCDINT     Initialize LCD-II
00029 100F BE 80    LCDM1  LDX    POINTR     Load data table pointer
00030 1011 DE 108A       LDX    DDATA,X    Ascii data -> IX
00031 1014 A3 FF         CPX    #$FF       Test if IX=$FF
00032 1016 27 07         BEQ    PEND       Branch if IX=#$FF
00033 1018 CD 1059       JSR    LCDDSP     Display data
00034 101B 3C 80         INC    POINTR     Increment pointer
00035 101D 20 F0         BRA    LCDM1      Branch always LCDM1
00036 101F 20 FE    PEND   BRA    PEND       End of main program
00037                    ************************************************
00038                    *                                              *
00039                    *          NAME : LCDRES (RESET LCD-II)        *
00040                    *                                              *
00041                    ************************************************
00042                    *                                              *
00043                    *       ENTRY : NOTHING                        *
00044                    *     RETURNS : NOTHING                        *
00045                    *                                              *
00046                    ************************************************
00047 1021 A6 03    LCDRES LDA    #3         Initialize loop counter
00048 1023 B7 81         STA    LCDCNT
00049 1025 A6 0C    LCDRS1 LDA    #$0C       Initialize 15ms counter
00050 1027 AE FF    LCDRS2 LDX    #$FF       Initialize inner counter
00051 1029 5A       LCDRS3 DEC    X          Decrement inner counter
00052 102A 26 FD         BNE    LCDRS3     Loop until inner counter=0
00053 102C 4A            DEC    A          Decrement 15ms counter
00054 102D 26 F8         BNE    LCDRS2     Loop until 15ms counter=0
00055 102F B7 0C         STA    PFDTR      Set RS=0,R/W=0,E=0
```

```
00056 1031 A6 01           LDA     #$01
00057 1033 B7 0C           STA     PFDTR     Set RS=0,R/W=0,E=1
00058 1035 A6 30           LDA     #$30
00059 1037 B7 0D           STA     PGDTR     Write instruction data
00060 1039 4F              CLR     A         Set E=0
00061 103A B7 0C           STA     PFDTR
00062 103C 3A 81           DEC     LCDCNT    Decrement loop counter
00063 103E 26 E5           BNE     LCDRS1    Loop until loop counter=0
00064 1040 81              RTS
00065                 ************************************************
00066                 *                                              *
00067                 *        NAME : LCDINT (INITIALIZE LCD-II)     *
00068                 *                                              *
00069                 ************************************************
00070                 *                                              *
00071                 *      ENTRY : NOTHING                         *
00072                 *    RETURNS : NOTHING                         *
00073                 *                                              *
00074                 ************************************************
00075 1041 AE 06    LCDINT LDX     #6        Load loop counter
00076 1043 CD 106B  LCDIT1 JSR     LCDBSY    Check busy flag
00077 1046 4F              CLR     A         Set RS=0,R/W=0,E=0
00078 1047 B7 0C           STA     PFDTR
00079 1049 A6 01           LDA     #$01      Set E=1
00080 104B B7 0C           STA     PFDTR
00081 104D D6 1083         LDA     INS-1,X   Instruction data->LCD-II
00082 1050 B7 0D           STA     PGDTR
00083 1052 4F              CLR     A         Set E=0
00084 1053 B7 0C           STA     PFDTR
00085 1055 5A              DEC     X         Decrement loop counter
00086 1056 26 EB           BNE     LCDIT1    Loop until loop counter=0
00087 1058 81              RTS
00088                 ************************************************
00089                 *                                              *
00090                 *      NAME : LCDDSP (DISPLAY ON LCD-II)       *
00091                 *                                              *
00092                 ************************************************
00093                 *                                              *
00094                 *      ENTRY : IX (DISPLAY DATA)               *
00095                 *    RETURNS : NOTHING                         *
00096                 *                                              *
00097                 ************************************************
00098 1059 CD 106B  LCDDSP JSR     LCDBSY    Check busy flag
00099 105C A6 02           LDA     #$02      Set RS=1,R/W=0,E=0
00100 105E B7 0C           STA     PFDTR
00101 1060 A6 03           LDA     #$03
00102 1062 B7 0C           STA     PFDTR     Set E=1
00103 1064 BF 0D           STX     PGDTR     Display data->LCD-II
00104 1066 A6 02           LDA     #$02      Set E=0
00105 1068 B7 0C           STA     PFDTR
00106 106A 81              RTS
00107                 ************************************************
00108                 *                                              *
00109                 *         NAME : LCDBSY (CHECK BUSY FLAG)      *
00110                 *                                              *
```

```
00111                   ************************************************
00112 106B 4F           LCDBSY CLR    A         Select port G as input
00113 106C B7 07               STA    PGDDR
00114 106E A6 04               LDA    #$04      Set RS=0,R/W=1,E=0
00115 1070 B7 0C               STA    PFDTR
00116 1072 A6 05        LCDBY1 LDA    #$05      Set E=1
00117 1074 B7 0C               STA    PFDTR
00118 1076 B6 0D               LDA    PGDTR     Read busy flag
00119 1078 48                  LSL    A         Set busy flag to bit C
00120 1079 A6 04               LDA    #$04      Set E=0
00121 107B B7 0C               STA    PFDTR
00122 107D 25 F3               BCS    LCDBY1    Loop until busy flag=0
00123 107F A6 FF               LDA    #$FF      Select port G as output
00124 1081 B7 07               STA    PGDDR
00125 1083 81                  RTS
00126                   ************************************************
00127                   *                                              *
00128                   *              DATA TABLE                      *
00129                   *                                              *
00130                   ************************************************
00131 1084 0C           INS    FCB    $0C,$18,$90,$07,$01,$08  *instruction
00132 108A 43           DDATA  FCC    /CMOS MCU HD6305X/          *Display
00133 109A FF                  FCB    $FF
00134                   ************************************************
00135                   *                                              *
00136                   *           VECTOR ADDRESSES                   *
00137                   *                                              *
00138                   ************************************************
00139                   *
00140 1FF6                     ORG    $1FF6
00141                   *
00142 1FF6 1000                FDB    LCDMN     SCI/TIMER2
00143 1FF8 1000                FDB    LCDMN     TIMER/INT2
00144 1FFA 1000                FDB    LCDMN     INT
00145 1FFC 1000                FDB    LCDMN     SWI
00146 1FFE 1000                FDB    LCDMN     RES
00147                   *
00148                          END
```

3. DUTY CONTROL OF PULSE OUTPUT

3.1 HARDWARE DESCRIPTION

(1) Function

(a) Outputs pulse with 0∿100% duty rate from the HD6305X0.

(b) Performs DA conversion of output pulse with external integration circuit.

(2) Microcomputer Applications

(a) Executes the interrupt routine with the built-in 8-bit timer with 7-bit prescaler in the HD6305X0 (hereinafter timer).

(b) Switches port C and outputs pulse with the interrupt routine.

(c) Changes High and Low period of pulse with TDR.

(3) Circuit Diagram

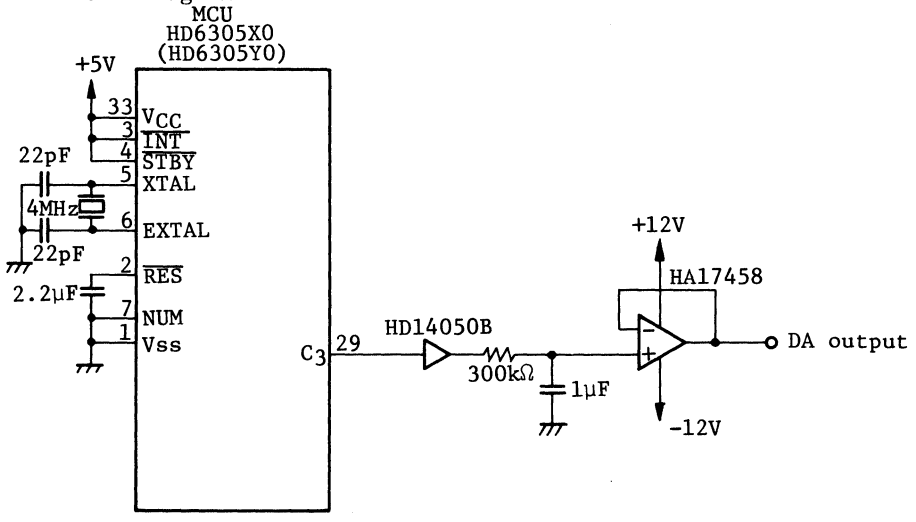


Fig. 1 Duty Pulse Output Circuit

(4) Pin Functions

Pin function for pulse output is shown in Table 1.

Table 1 Pin Function

Pin Name (HD6305X0)	Input/ Output	Active Level (High or Low)	Function	Program Label
C_3	Output	—	Outputs pulse	PCDTR

(5) Hardware Operation

Outputs pulse with 0∿100% duty rate at each 0.3s, increasing the duty rate 4% each time from port on the HD6305X0.

The pulse output and DA conversion are shown in Fig. 2.

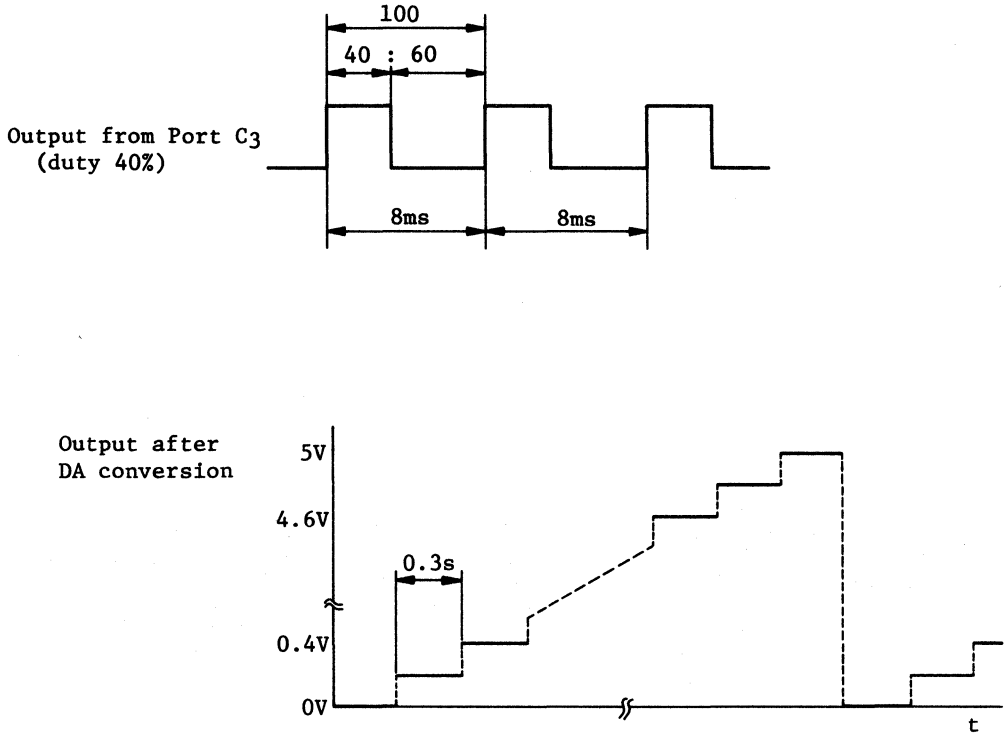


Fig. 2 Pulse Output and Waveform after DA Conversion

(Note) The output voltage after DA conversion is proportional to the duty.
The operational amplifier is used to prevent the flactuation of analog output voltage caused by the load in user system.

3.2 SOFTWARE DESCRIPTION

(1) Program Module Configuration

The program module configuration for pulse output is shown in Fig. 3.

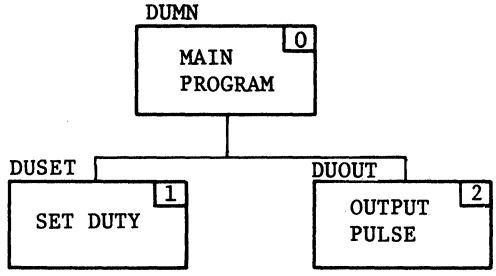


Fig. 3 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	DUMN	Outputs pulse with 0~100% duty rate.
1	SET DUTY	DUSET	Sets duty.
2	OUTPUT PULSE	DUOUT	Outputs pulse from I/O port.

(3) Program Module Sample Application (Main Program)

The flowchart in Fig. 4 is an example of DA conversion by controlling pulse, performed by the program module in Fig. 3. Main program in Fig. 4 changes pulse output with 0~100% duty rate at each 0.3 second, increasing the duty rate 4% each time.

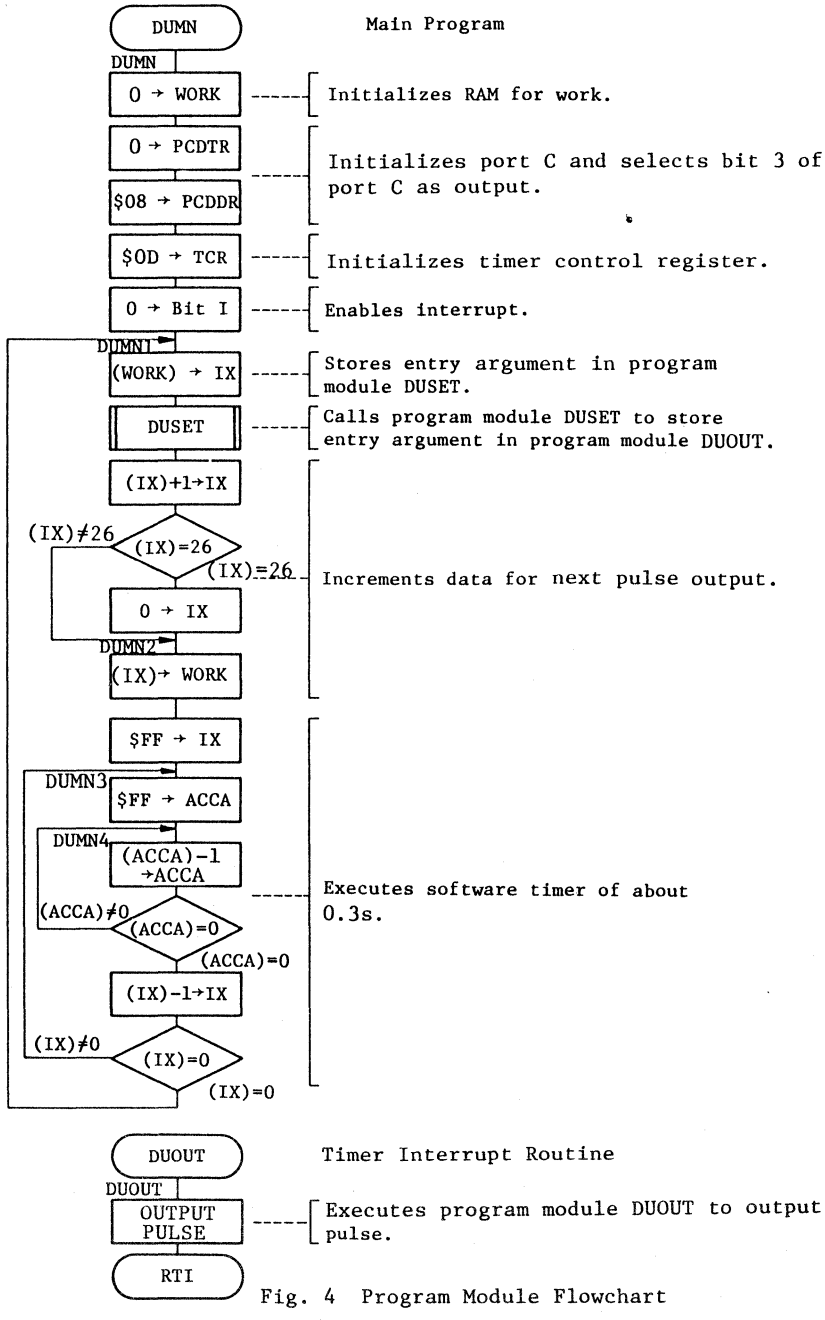


Fig. 4 Program Module Flowchart

PROGRAM MODULE NAME	SET DUTY	MCU/MPU	HD6305X0/Y0	LABEL	DUSET

FUNCTION

(1) Stores High and Low output period corresponding to 1-byte hexadecimal duty stored in entry argument.

(2) Specifies duty each 4%.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Duty	IX	1
	Returns	High output period	HTIME (RAM)	1
		Low output period	LTIME (RAM)	1
		Output status flag	HLOUT (RAM)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
×	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
40
RAM (Bytes)
3
Stack (Bytes)
0
No. of cycles
33
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details
IX: Holds duty as 1-byte hexadecimal number.
HTIME(RAM): Contains High output period.
LTIME(RAM): Contains Low output period.
HLOUT(RAM): Contains Flag indicating what output is performed; Low consecutive output, High consecutive output, or pulse output. Table 3 shows flag functions.

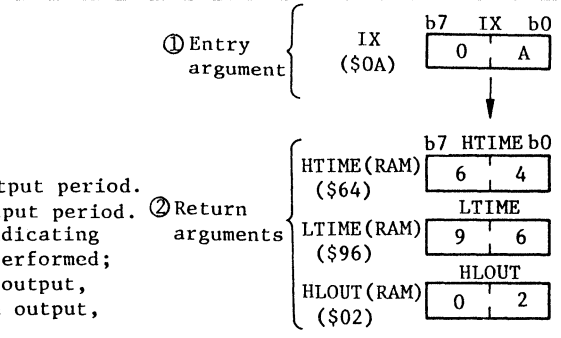


Fig. 5 Example of DUSET Execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles required in case of duty 40%.

DESCRIPTION

Table 3 Flag Functions

Label	Bit/Label		Function
	Bit 1	Bit 0	
HLOUT	0	0	Outputs Low consectively from bit 3 of Port C.
	1	0	Outputs pulse from bit 3 of port C.
	1	1	Outputs High consectively from bit 3 of port C.

(b) Fig. 5 shows an examples of program module DUSET execution. When entry argument is as shown in part ① of Fig. 5, High and Low period of duty 40% are contained in return arguments as shown in part ② of Fig. 5.

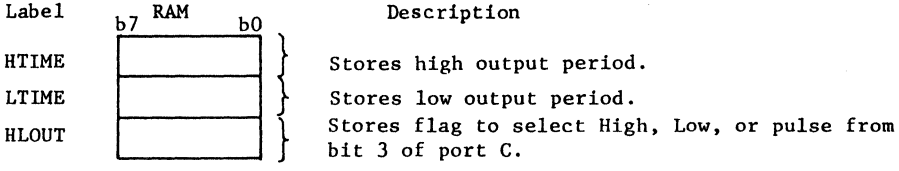
(c) Program module DUSET calls neither program modules nor subroutines.

(2) User Notes

(a) 1/4 of actual duty should be loaded into entry argument, otherwise exact High and Low output periods can not be obtained.

(b) Loads data within range of $00 and $19 into IX. If data outside this range is loaded, High and Low output period can not be obtained.

(3) RAM Description



(4) Sample Application

Program module DUSET is called after set duty.

```
LDX     #10     ---Defines duty.
JSR     DUSET   ---Calls program module DUSET.
```

(5) Basic Operations

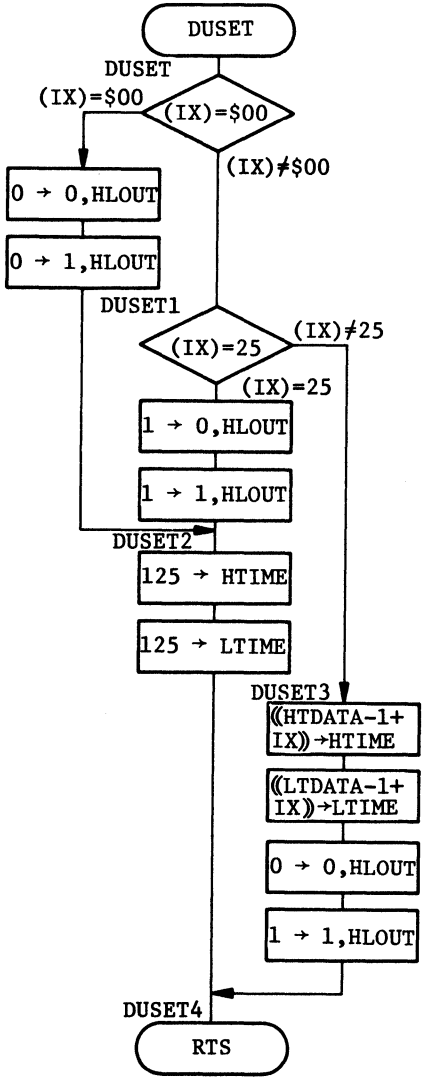
(a) IX is used as pointer to indicate High and Low output period corresponding to duty.

(b) When duty in IX is 0% or 100%, 125 is stored in HTIME(RAM) and LTIME(RAM).

(c) When duty is other than 0% and 100%, High and Low output periods are stored in HTIME(RAM) and LTIME(RAM), and "1" is stored in bit 1 of HLOUT(RAM), using index addressing mode respectively.

PROGRAM MODULE NAME	SET DUTY	MCU/MPU	HD6305X0/Y0	LABEL	DUSET

FLOWCHART



Tests if Low consecutive output is performed from bit 3 of port C.

Clears flag to output Low consecutively from bit 3 of port C.

Tests if High consecutive output is performed from bit 3 of port C.

Sets flag to output High consecutively from bit 3 of port C.

Stores data to output High or Low from bit 3 of port C.

Stores High output period data.

Stores Low output period data.

Sets flag to output pulse from bit 3 of port C.

7

PROGRAM MODULE NAME	OUTPUT PULSE	MCU/MPU	HD6305X0/Y0	LABEL	DUOUT

FUNCTION

Outputs pulse from bit 3 of port C.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	High output period	HTIME (RAM)	1
		Low output period	LTIME (RAM)	1
		Output status flag	HLOUT (RAM)	1
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
×	×
N	I
×	●
H	
●	

SPECIFICATIONS

ROM (Bytes)
26
RAM (Bytes)
3
Stack (Bytes)
2
No. of cycles
31
Reentrant
No
Relocation
No
Interrupt
No

DESCRIPTION

(1) Function Details

(a) Argument details

HTIME(RAM): Holds High output period.

LTIME(RAM): Holds Low output period.

HLOUT(RAM): Holds flag indicating what output is performed; Low consecutive output, High consecutive output, or pulse output.

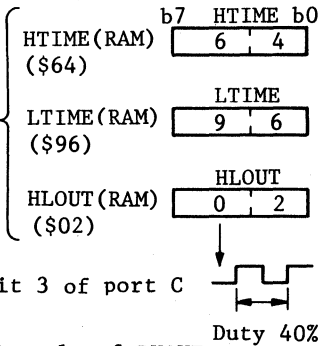


Fig. 6 Example of DUOUT Execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles needed to execut output with data in Sample Application.

Table 4 shows flag functions.

Table 4 Flag Functions

Label	Bit/Label		Function
	bit 1	bit 0	
HLOUT	0	0	Outputs Low consecutively from bit 3 of port C.
	1	0	Outputs pulse from bit 3 of port C.
	1	1	Outputs High consecutively from bit 3 of port C.

(b) Fig. 6 shows an example of program module DUOUT execution. If entry arguments are as shown in part ① of Fig. 6, pulse with 40% duty rate is output as shown in part ② of Fig. 6.

(c) Program module DUOUT calls neither program modules nor subroutines.

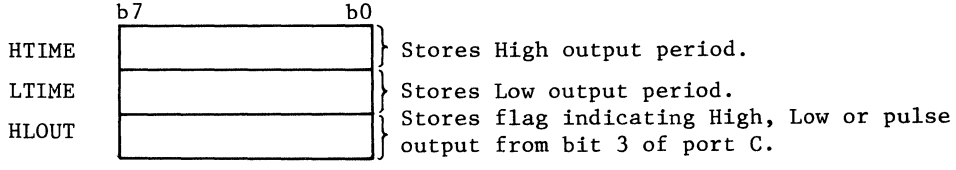
(2) User Notes

(a) Selects DDR of port C as output.

(b) Initializes timer.

(c) Clear bit I to enable timer interrupt.

(3) RAM Description



(4) Sample Application

Program module DUOUT is called to output pulse from bit 3 of port C after initializing port C, timer and duty, and enabling interrupt.

```
        :
    LDA   #$08    }
    STA   PCDDR   }---Selects bit 3 of port C as output.
    LDA   #$00    }
    STA   TCR     }---Sets timer dividing rate at ÷ 32.
    LDX   #10     --- Loads duty into entry argument of program module DUSET.
    JSR   DUSET   ---Calls program module DUSET to store High and Low output
                     periods in entry arguments of program module DUOUT.
    CLI           ---Enables interrupt.
        :
```

PROGRAM MODULE NAME	OUTPUT PULSE	MCU/MPU	HD6305X0/Y0	LABEL	DUOUT

DESCRIPTION

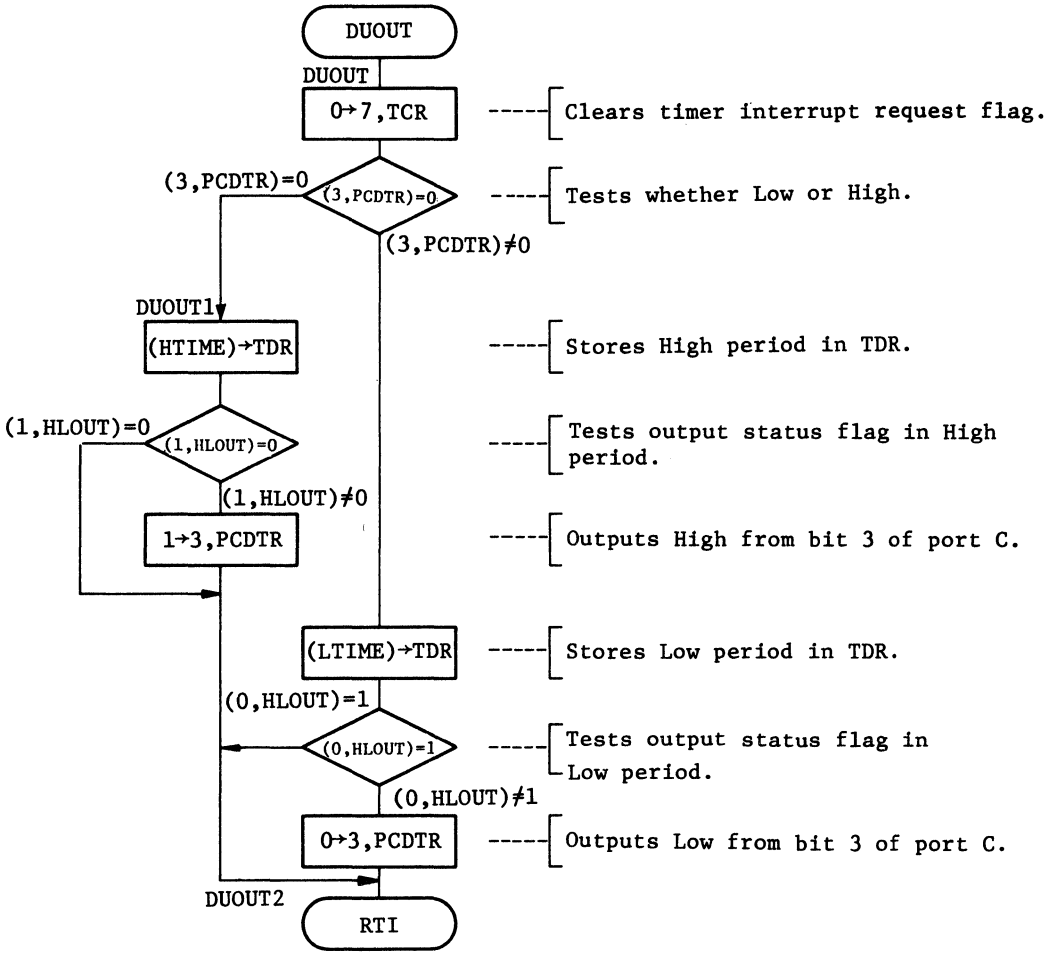
(5) Basic Operation

(a) Determines the previous output at every timer interrupt, and changes output port, High to Low, Low to High. The period of High or Low is stored in TDR.

(b) When duty is 0% or 100%, TDR is set to 50% output and the output port remains in High or Low state.

PROGRAM MODULE NAME	OUTPUT PULSE	MCU/MPU	HD6305X0/Y0	LABEL	DUOUT

FLOWCHART



7

3.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

3.5 PROGRAM LISTING

```
00001                   *
00002                   ****    RAM ALLOCATION    **************************
00003                   *
00004 0080              *       ORG     $80
00005                   *
00006 0080 0001 HTIME   RMB     1       High pulse data
00007 0081 0001 LTIME   RMB     1       Low pulse data
00008 0082 0001 HLOUT   RMB     1       Output data ststus
00009 0083 0001 WORK    RMB     1       Work for entry argument
00010                   *
00011                   ****    SYMBOL DEFINITIONS    ************
00012                   *
00013       0002 PCDTR  EQU     $02     Port C data register
00014       0006 PCDDR  EQU     $06     Port C data direction register
00015       0008 TDR    EQU     $08     Timer data register
00016       0009 TCR    EQU     $09     Timer control register
00017                   ***********************************************
00018                   *                                             *
00019                   *           MAIN PROGRAM : DUMN               *
00020                   *                                             *
00021                   ***********************************************
00022                   *
00023 1000              *       ORG     $1000
00024                   *
00025 1000 4F   DUMN    CLR     A       Clear entry argument
00026 1001 B7 83        STA     WORK
00027 1003 B7 02        STA     PCDTR   Initialize port C
00028 1005 A6 08        LDA     #$08    Select port C bit 3 as output
00029 1007 B7 06        STA     PCDDR
00030 1009 A6 0D        LDA     #$0D    Set prescaler 1/32
00031 100B B7 09        STA     TCR
00032 100D 9A           CLI             Enable inturrupts
00033 100E BE 83 DUMN1  LDX     WORK    Load entry argument of DUSET
00034 1010 AD 14        BSR     DUSET   Store entry argument of DUOUT
00035 1012 5C           INC     X       DUTY+4%->entry argument
00036 1013 A3 1A        CPX     #26     DUTY=104% ?
00037 1015 26 01        BNE     DUMN2
00038 1017 5F           CLR     X       Store 0% duty
00039 1018 BF 83 DUMN2  STX     WORK    Store duty in entry argument
00040 101A AE FF        LDX     #$FF    Execute 0.3s software timer
00041 101C A6 FF DUMN3  LDA     #$FF
00042 101E 4A    DUMN4  DEC     A
00043 101F 26 FD        BNE     DUMN4
00044 1021 5A           DEC     X
00045 1022 26 F8        BNE     DUMN3
00046 1024 20 E8        BRA     DUMN1
00047                   ***********************************************
00048                   *                                             *
00049                   *       NAME : DUSET (SET DUTY)               *
00050                   *                                             *
00051                   ***********************************************
00052                   *                                             *
00053                   *       ENTRY : IX    (DUTY DATA)             *
00054                   *     RETURNS : HTIME (HIGH PULSE PERIOD)       *
00055                   *               LTIME (LOW PULSE PERIOD)      *
```

```
00056                    *              HLOUT (OUTPUT STATUS FLAG)       *
00057                    *                                               *
00058                    ************************************************
00059 1026 5D            DUSET  TST     X          Test if duty=0% ?
00060 1027 26 06                BNE     DUSET1
00061 1029 11 82                BCLR    0,HLOUT    Set for low
00062 102B 13 82                BCLR    1,HLOUT    Set for low pulse
00063 102D 20 08                BRA     DUSET2
00064 102F A3 19         DUSET1 CPX     #25        Test if duty=100%?
00065 1031 26 0C                BNE     DUSET3     Branch for pulse output
00066 1033 10 82                BSET    0,HLOUT    Set for high
00067 1035 12 82                BSET    1,HLOUT    Set for high pulse
00068 1037 A6 7D         DUSET2 LDA     #125       Set 50% duty timming
00069 1039 B7 80                STA     HTIME
00070 103B B7 81                STA     LTIME
00071 103D 20 0E                BRA     DUSET4
00072 103F D6 1067       DUSET3 LDA     HTDATA-1,X  Set period of high pulse
00073 1042 B7 80                STA     HTIME
00074 1044 D6 107F              LDA     LTDATA-1,X  Set period of low pulse
00075 1047 B7 81                STA     LTIME
00076 1049 11 82                BCLR    0,HLOUT    Set for low pulse
00077 104B 12 82                BSET    1,HLOUT    Set for high pulse
00078 104D 81            DUSET4 RTS
00079                    ************************************************
00080                    *                                               *
00081                    *         NAME : DUOUT (OUTPUT PULSE)           *
00082                    *                                               *
00083                    ************************************************
00084                    *                                               *
00085                    *       ENTRY : HTIME (HIGH PULSE PERIOD)       *
00086                    *               LTIME (LOW PULSE PERIOD)        *
00087                    *               HLOUT (OUTPUT STATUS FLAG)      *
00088                    *     RETURNS : NOTHING                         *
00089                    *                                               *
00090                    ************************************************
00091 104E 1F 09         DUOUT  BCLR    7,TCR      Clear interrupt request bit
00092 1050 07 02 0B             BRCLR   3,PCDTR,DUOUT1  High or Low output?
00093 1053 B6 81                LDA     LTIME      Store for low pulse period
00094 1055 B7 08                STA     TDR
00095 1057 00 82 0D             BRSET   0,HLOUT,DUOUT2  Duty=0%?
00096 105A 17 02                BCLR    3,PCDTR    Output low pulse
00097 105C 20 09                BRA     DUOUT2
00098 105E B6 80         DUOUT1 LDA     HTIME      Store for high pulse period
00099 1060 B7 08                STA     TDR
00100 1062 03 82 02             BRCLR   1,HLOUT,DUOUT2  Duty=100% ?
00101 1065 16 02                BSET    3,PCDTR    Output high pulse
00102 1067 80            DUOUT2 RTI
00103                    ************************************************
00104                    *                                               *
00105                    *                   DATA TABLE                  *
00106                    *                                               *
00107                    ************************************************
00108 1068 0A            HTDATA FCB     10         4    *High pulse period
00109 1069 14                   FCB     20         8
00110 106A 1E                   FCB     30         12
```

```
00111 106B 28              FCB    40        16
00112 106C 32              FCB    50        20
00113 106D 3C              FCB    60        24
00114 106E 46              FCB    70        28
00115 106F 50              FCB    80        32
00116 1070 5A              FCB    90        36
00117 1071 64              FCB    100       40
00118 1072 6E              FCB    110       44
00119 1073 78              FCB    120       48
00120 1074 82              FCB    130       52
00121 1075 8C              FCB    140       56
00122 1076 96              FCB    150       60
00123 1077 A0              FCB    160       64
00124 1078 AA              FCB    170       68
00125 1079 B4              FCB    180       72
00126 107A BE              FCB    190       76
00127 107B C8              FCB    200       80
00128 107C D2              FCB    210       84
00129 107D DC              FCB    220       88
00130 107E E6              FCB    230       92
00131 107F F0              FCB    240       96
00132 1080 F0       LTDATA FCB    240       4     *Low pulse period
00133 1081 E6              FCB    230       8
00134 1082 DC              FCB    220       12
00135 1083 D2              FCB    210       16
00136 1084 C8              FCB    200       20
00137 1085 BE              FCB    190       24
00138 1086 B4              FCB    180       28
00139 1087 AA              FCB    170       32
00140 1088 A0              FCB    160       36
00141 1089 96              FCB    150       40
00142 108A 8C              FCB    140       44
00143 108B 82              FCB    130       48
00144 108C 78              FCB    120       52
00145 108D 6E              FCB    110       56
00146 108E 64              FCB    100       60
00147 108F 5A              FCB    90        64
00148 1090 50              FCB    80        68
00149 1091 46              FCB    70        72
00150 1092 3C              FCB    60        76
00151 1093 32              FCB    50        80
00152 1094 28              FCB    40        84
00153 1095 1E              FCB    30        88
00154 1096 14              FCB    20        92
00155 1097 0A              FCB    10        96
00156                *********************************************
00157                *                                           *
00158                *           VECTOR ADDRESSES                *
00159                *                                           *
00160                *********************************************
00161                *
00162 1FF6                 ORG    $1FF6
00163                *
00164 1FF6 1000            FDB    DUMN      SCI/TIMER2
00165 1FF8 104E            FDB    DUOUT     TIMER/INT2
00166 1FFA 1000            FDB    DUMN      INT
00167 1FFC 1000            FDB    DUMN      SWI
00168 1FFE 1000            FDB    DUMN      RES
00169                *
00170                      END
```

4. PULSE WIDTH MEASUREMENT

4.1 HARDWARE DESCRIPTION

(1) Function

(a) Measures the High period of the pulse using the HD6305X0.

(b) Measures pulse width from 100μs to 255μs.

(c) Converts measurement result into binary coded decimal (BCD) number.

(2) Microcomputer Applications

(a) Measures pulse width using TIMER pin and $\overline{INT}$ pin of the HD6305X0.

(b) The HD6305X0 counts down TDR while the TIMER pin is High.

(c) Executes interrupt routine on the $\overline{INT}$ falling edge, reads TDR value, and measures pulse width.

(3) Circuit Diagram

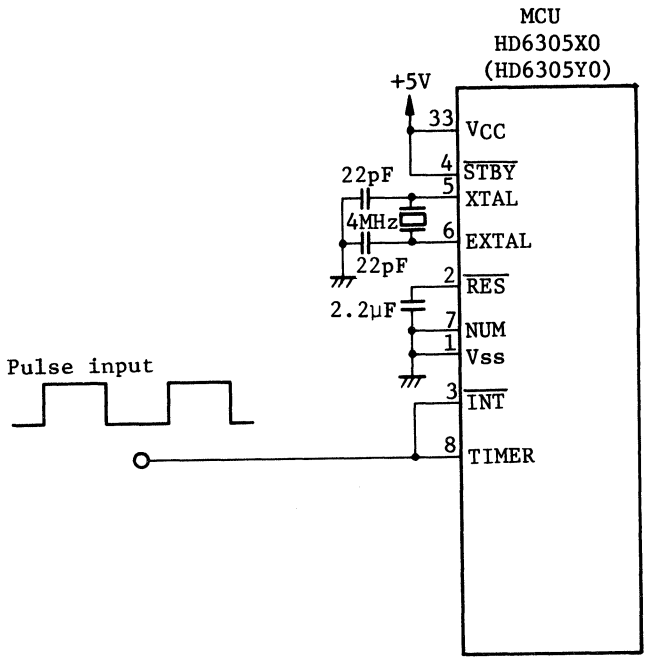


Fig. 1 Pulse Width Measurement Circuit

(4) Pin Functions

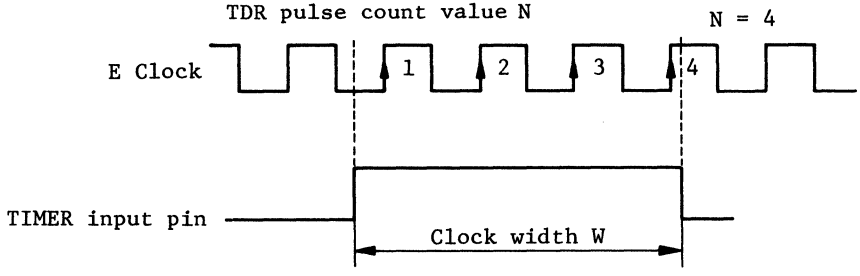
Pin functions for pulse width measurement are shown in Table 1.

Table 1 Pin Functions

Pin Name (HD6305X0)	Input/ Output	Active level (High or Low)	Function	Program Label
$\overline{\text{INT}}$	Input	Low	Detects falling edge of input pulse and executes the external interrupt routine.	
TIMER	Input	High	Counts down the TDR during "1" active period.	

(5) Hardware Operation

The HD6305X0 measures the pulse width by counting down the TDR using the E clock while TIMER pin is High, which is given in Fig. 2.



(Note) When E clock cycle is 1μs, the clock width W is 4μs. Measurement error is within 1μs.

Fig. 2 Pulse Width Measurement

4.2 SOFTWARE DESCRIPTION

(1) Program Module Configuration

The program module configuration for measuring pulse width and converting result into BCD number is shown in Fig. 3.

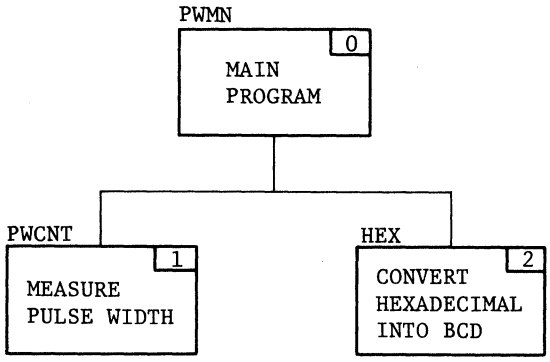


Fig. 3 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Lavel	Function
0	MAIN PROGRAM	PWMN	Obtains the pulse width in BCD number.
1	MEASURE PULSE WIDTH	PWCNT	Calculates the pulse width from TDR in 1-byte hexadecimal number.
2	CONVERT HEXADECIMAL INTO BCD	HEX	Converts 1-byte hexadecimal number into BCD. See subroutine HEX in HD6305 FAMILY APPLICATION NOTES (SOFTWARE) for details.

(3) Program Module Sample Application (Main Program)

The flowchart in Fig. 4 is an example of pulse width measurement performed by the program module in Fig. 3.

The main program in Fig. 4 obtains the pulse width as BCD number.

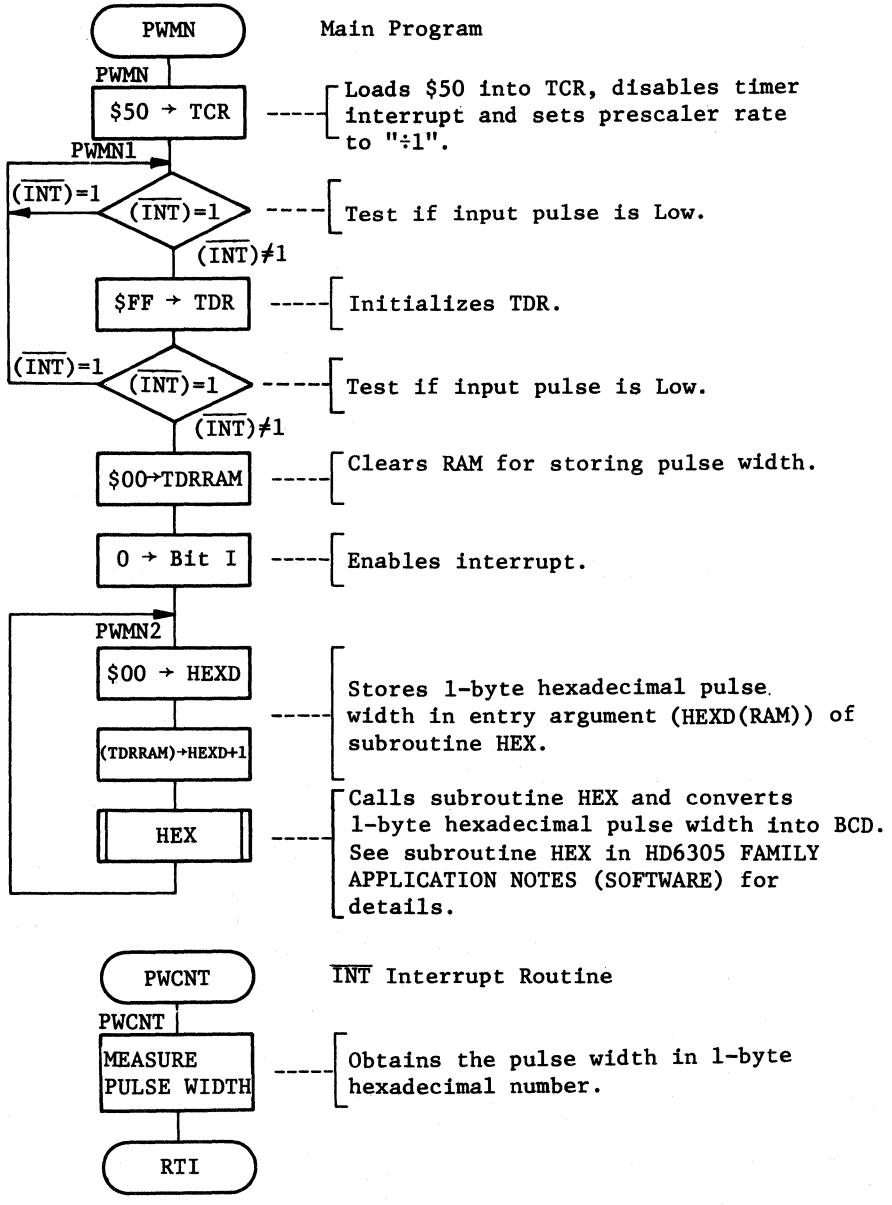


Fig. 4 Program Module Flowchart

PROGRAM MODULE NAME	MEASURE PULSE WIDTH	MCU/MPU	HD6305X0/Y0	LABEL	PWCNT

FUNCTION

(1) Calculates the pulse width in 1-byte hexadecimal number and stores result in TDRRAM(RAM).

(2) Sets oscillator frequency to 4MHz in program module PWCNT execution.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	—	—	—
	Returns	Pulse Width	TDRRAM (RAM)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
×	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
10
RAM (Bytes)
1
Stack (Bytes)
0
No. of cycles
18
Reentrant
No
Relocation
No
Interrupt
No

DESCRIPTION

(1) Function Details

(a) Argument details

TDRRAM (RAM): Contains pulse width in 1-byte hexadecimal number.

(b) Fig. 5 shows an example of program module PWCNT esecution.

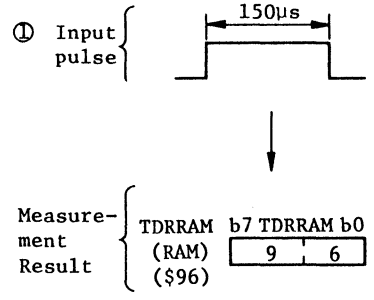


Fig. 5 Example of PWCNT Exécution

SPECIFICATIONS NOTES

PROGRAM MODULE NAME	MEASURE PULSE WIDTH	MCU/MPU	HD6305X0/Y0	LABEL	PWCNT

DESCRIPTION

If 150 µs pulse width is input as shown in part ① of Fig. 5, the measurement result is contained in TDRRAM(RAM) as shown in part ② of Fig. 5.

(c) Program module PWCNT calls neither program modules nor subroutines.

(2) User Notes

(a) Only pulse width between 100 µs and 200 µs can be correctly measured.

(b) Initializes timer.

(c) Clears bit I to enable $\overline{\text{INT}}$ interrupt.

(d) Sets oscillator frequency to 4MHz, because pulse width measurement is performed by 1µs E clock.

(3) RAM Description

Label	RAM	Description
	b7 b0	
TDRRAM	☐	Stores pulse width in 1-byte hexadecimal number.

(4) Sample Application

After initializing timer, enable interrupt.
Program module PWCNT is executed on the falling edge of $\overline{\text{INT}}$ pin signal.

```
LA1   BIH     LA1      ---Test if input pulse is Low.
      LDA     #$50     }
      STA     TCR      }---Sets E prescaler division rate to "÷1" using timer
      LDA     #$FF     }   interrupt mask and TIMER pin.
      STA     TDR      }---Initializes TDR.
      BIH     LA1      ---Test if input pulse is Low.
      CLR     TDRRAM   ---Clears RAM for pulse width.
      CLI              ---Enables interrupt.
```

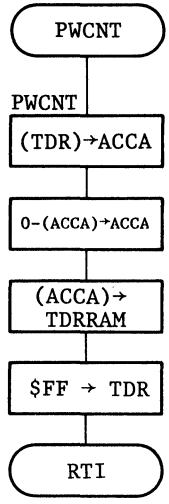
(5) Basic Operation

(a) Pulse width data in program module PWCNT is 2's complement of TDR because TDR decrements from $FF.

(b) To measure pulse width again after (a) execution, TDR is reinitialized to $FF.

PROGRAM MODULE NAME	MEASURE PULSE WIDTH	MCU/MPU	HD6305X0/Y0	LABEL	PWCNT

FLOWCHART



Calculates 2's complement of TDR value and stores it in TDRRAM(RAM).

Initializes TDR to measure next pulse width.

7

4.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

4.5 PROGRAM LISTING

```
00001                   *
00002                   ****    RAM ALLOCATION   ************************
00003                   *
00004 0080              *       ORG     $80
00005                   *
00006 0080 0001         TDRRAM  RMB     1       Pulse width data
00007 0081 0002         HEXD    RMB     2       2-byte hexadecimals
00008 0083 0003         DECD    RMB     3       5-digit BCD
00009 0086 0001         HCNTR   RMB     1       Subtraction counter
00010                   *
00011                   ****    SYMBOL DEFINITIONS   ************
00012                   *
00013       0008        TDR     EQU     $08     Timer Data Reg
00014       0009        TCR     EQU     $09     Timer CRTL Reg
00015                   ************************************************
00016                   *                                              *
00017                   *           MAIN PROGRAM : PWMN                *
00018                   *                                              *
00019                   ************************************************
00020                   *
00021 1000              *       ORG     $1000
00022                   *
00023 1000 A6 50        PWMN    LDA     #$50    Initialize TCR
00024 1002 B7 09                STA     TCR
00025 1004 2F FE        PWMN1   BIH     PWMN1   Test if INT=0
00026 1006 A6 FF                LDA     #$FF    Initialize TDR
00027 1008 B7 08                STA     TDR
00028 100A 2F F8                BIH     PWMN1   Test if INT=0
00029 100C 3F 80                CLR     TDRRAM  Clear pulse width data
00030 100E 9A                   CLI             Enable inturrupts
00031 100F 3F 81        PWMN2   CLR     HEXD    Clear upper byte of HEXD
00032 1011 B6 80                LDA     TDRRAM  Store pulse width result
00033 1013 B7 82                STA     HEXD+1
00034 1015 CD 101A              JSR     HEX     Convert result into BCD
00035 1018 20 F5                BRA     PWMN2
00036                   ************************************************
00037                   *                                              *
00038                   *  NAME : HEX (CONVERTING 2-BYTE HEXADECIMALS  *
00039                   *               INTO 5-DIGIT BCD)              *
00040                   *                                              *
00041                   ************************************************
00042                   *                                              *
00043                   *          ENTRY : HEXD (2-BYTE HEX)           *
00044                   *        RETURNS : DECD (5-FIGURE DECIMAL)     *
00045                   *                                              *
00046                   ************************************************
00047 101A 3F 83        HEX     CLR     DECD    Clear 5-digit BCD area
00048 101C 3F 84                CLR     DECD+1
00049 101E 3F 85                CLR     DECD+2
00050 1020 A6 10                LDA     #16     Set shift counter
00051 1022 B7 86                STA     HCNTR
00052 1024 38 82        HEX1    ASL     HEXD+1  Shift MSB of HEXD to carry
00053 1026 39 81                ROL     HEXD
00054 1028 AE 03                LDX     #3   Set ADDR pointer (addition counter
00055 102A E6 82        HEX2    LDA     DECD-1,X DECD * 2 + C -> ACCA
```

```
00056 102C E9 82          ADC    DECD-1,X
00057 102E 8D             DAA           Convert into BCD
00058 102F E7 82          STA    DECD-1,X Store 5-digit BCD area
00059 1031 5A             DEC    X        Decrement ADDR pointer
00060 1032 26 F6          BNE    HEX2     Branch until ADDR pointer=0
00061 1034 3A 86          DEC    HCNTR    Decrement shift counter
00062 1036 26 EC          BNE    HEX1     Branch until shift counter=0
00063 1038 81             RTS
00064                *************************************************
00065                *                                               *
00066                *      NAME : PWCNT (MEASURE PULSE WIDTH)       *
00067                *                                               *
00068                *************************************************
00069                *                                               *
00070                *      ENTRY : NOTHING                          *
00071                *    RETURNS : TDRRAM (PULSE WIDTH)             *
00072                *                                               *
00073                *************************************************
00074 1039 B6 08    PWCNT  LDA    TDR      Load TDR (high pulse width)
00075 103B 40              NEG    A        Calculate 2's complement of TDR
00076 103C B7 80           STA    TDRRAM   Store pulse width data
00077 103E A6 FF           LDA    #$FF     Initialize TDR to measure next
00078 1040 B7 08           STA    TDR                                pulse
00079 1042 80              RTI
00080                *************************************************
00081                *                                               *
00082                *             VECTOR ADDRESSES                  *
00083                *                                               *
00084                *************************************************
00085                *
00086 1FF6                 ORG    $1FF6
00087                *
00088 1FF6 1000            FDB    PWMN     SCI/TIMER2
00089 1FF8 1000            FDB    PWMN     TIMER/INT2
00090 1FFA 1039            FDB    PWCNT    INT
00091 1FFC 1000            FDB    PWMN     SWI
00092 1FFE 1000            FDB    PWMN     RES
00093                *
00094                      END
```

5. INPUT PULSE COUNT

5.1 HARDWARE DESCRIPTION

(1) Function

(a) Counts number of input pulses using the HD6305X0.

(b) Counts up to 255 pulses and indicates value as 1-byte hexadecimal number.

(2) Microcomputer Applications

(a) Inputs pulses from the HD6305X0 TIMER pin.

(b) The HD6305X0 counts down TDR with pulses input from TIMER pin and counts input pulses.

(3) Circuit Diagram

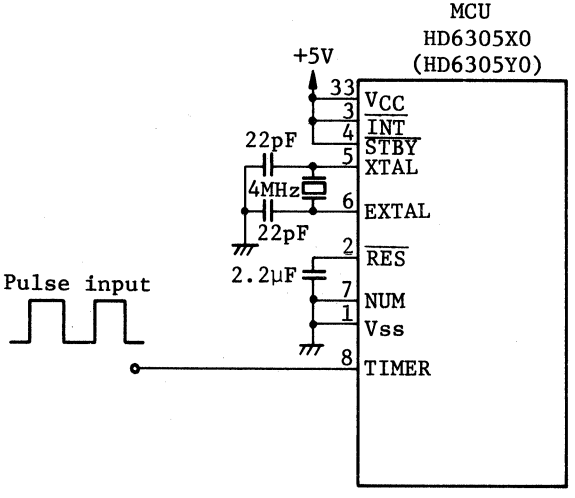


Fig. 1 Input Pulse Counting Using TIMER Pin

(4) Pin Functions

Pin function for input pulse count is shown in Table 1.

Table 1 Pin Functions

Pin Name (HD6305X0)	Input/ Output	Active Level (High or Low)	Function	Program Label
TIMER	Input	—	External event input	

(5) Hardware Operation

Input pulse count with use of the HD6305X0 TIMER pin is shown in Fig. 2.

After setting flag in STRTF(RAM) in main program, pulse count start/end timings are set by clearing the flag in STRTF(RAM).

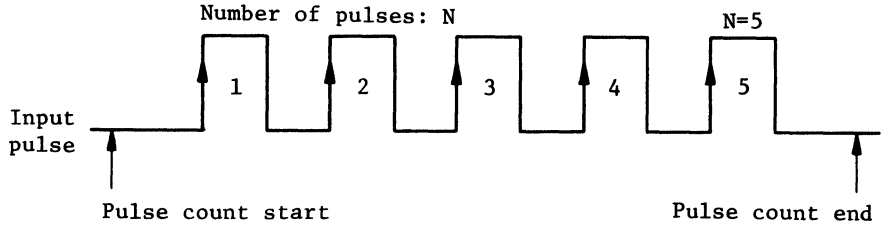


Fig. 2 Input Pulse Count

5.2 SOFTWARE DESCRIPTION

(1) Program Module Configuration

The program module configuration for input pulse count is shown in Fig. 3.

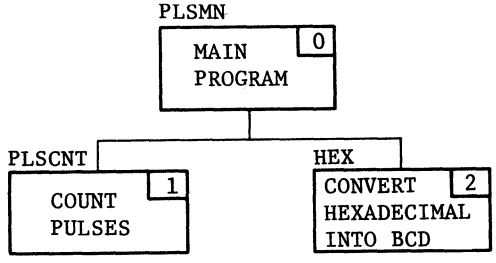


Fig. 3 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	PLSMN	Obtains input pulse count result as BCD number.
1	COUNT PULSES	PLSCNT	Obtains number of input pulses from TDR value.
2	CONVERT HEXADECIMAL INTO BCD	HEX	Converts 1-byte hexadecimal into BCD. See subroutine HEX in HD6305 FAMILY APPLICATION NOTES (SOFTWARE) for details.

(3) Program Module Sample Application (Main Program)

The flowchart in Fig. 4 is an example of counting input pulses and converting count result to BCD number, performed by the program module in Fig. 3.

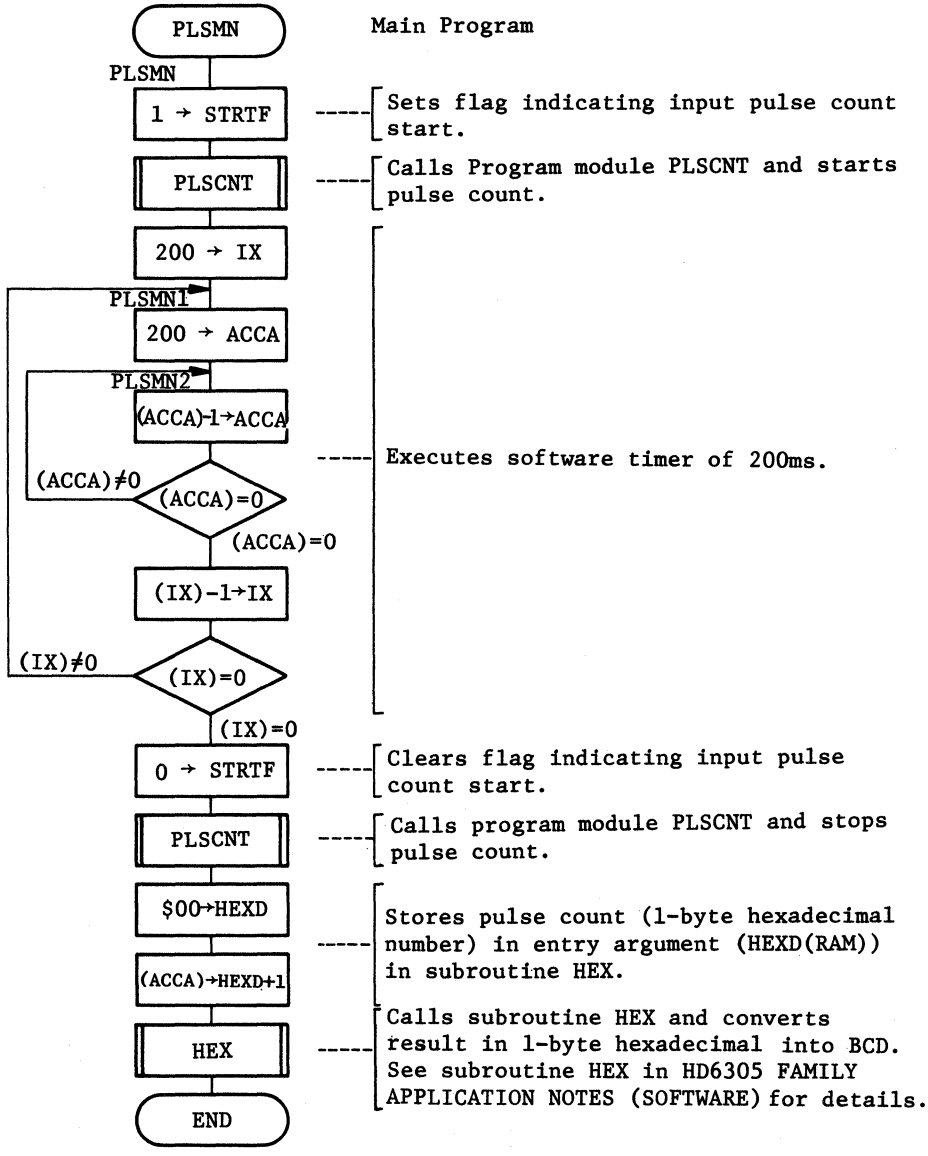


Fig. 4 Program Module Flowchart

PROGRAM MODULE NAME	COUNT PULSES	MCU/MPU	HD6305X0/Y0	LABEL	PLSCNT

FUNCTION

Counts pulses input from TIMER pin and stores result in ACCA.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Flag indicating start/stop	STRTF (RAM)	1
	Returns	Result of pulse count	ACCA	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
×	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
21
RAM (Bytes)
1
Stack (Bytes)
0
No. of cycles
20
Reentrant
No
Relocation
No
Interrupt
No

DESCRIPTION

(1) Function Details

(a) Arguments details
STRTF(RAM): Holds flag indicating start/stop of pulse count. Table 3 shows flag functions.
ACCA: Contains result of pulse count as 1-byte hexadecimal number.

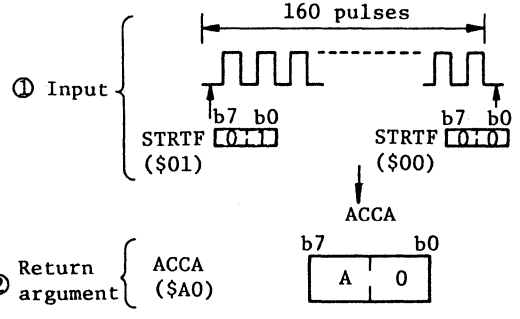


Fig. 5 Example of PLSCNT Execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles required to start pulse count.

DESCRIPTION

Table 3 Flag Functions

Label	Bit/Label bit 0	Function
STRTF	0	Stops pulse count
	1	Starts pulse count

(b) Fig. 5 shows an example of program module PLSCNT execution. If 160 pulses are input from TIMER pin as shown in part ① of Fig. 5, the count result is contained in ACCA as shown in part ② of Fig. 5.

(c) Program module PLSCNT calls neither program modules nor subroutines.

(2) User Notes

Counts up to 255 pulses because TDR is used to count input pulse.

(3) RAM Description

Label	RAM	Description
STRTF	b7 b0	Stores flag indicating start/stop of pulse count.

(4) Sample Application

Program module PLSCNT is called after timer is initialized and flag indicating start/stop of pulse count is held.

```
WORK1   RMB     1       ----Reserves memory byte for count result in
                                1-byte hexadecimal number.
        BSET    0,STRTF ----Sets flag to start pulse count.
        JSR     PLSCNT  ----Calls program module PLSCNT and starts pulse
                                count.
        BCLR    0,STRTF ----Clears flag to stop pulse count.
        JSR     PLSCNT  ----Calls program module PLSCNT and stop pulse
                                count.
        STA     WORK1   ----Stores count result in 1-byte hexadecimal
                                number in return argument.
```

(5) Basic Operation

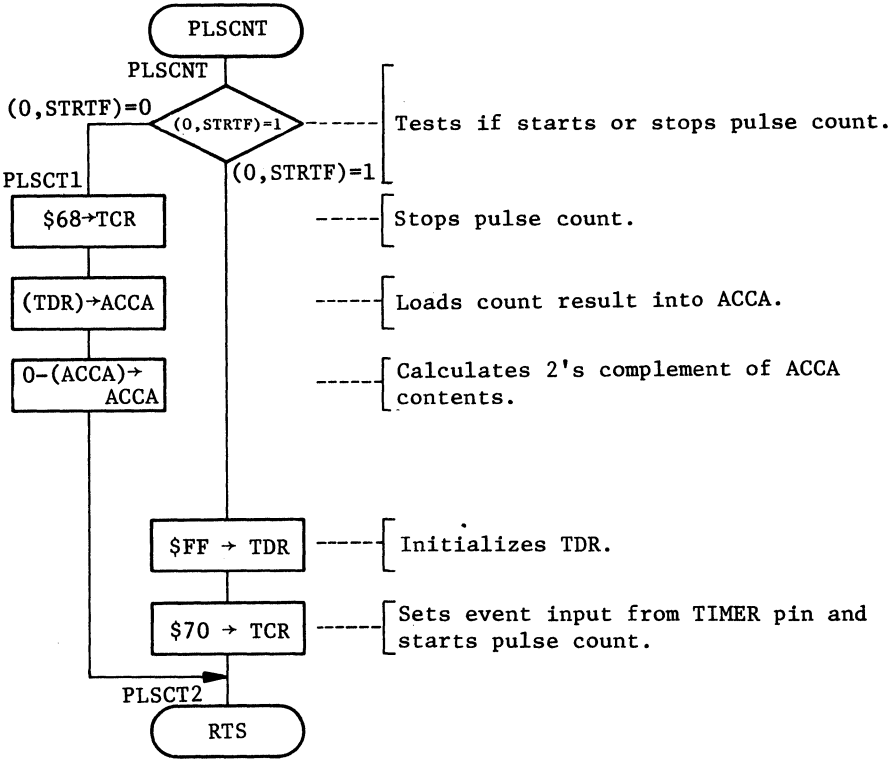
(a) The HD6305X0 timer decrements TDR from $FF immediately after clock is input from TIMER pin.

(b) TCR designates event input from TIMER pin as clock input and sets prescaler diving rate to "÷1"

(c) Count result is obtained by taking the 2's complement of TDR's contents when clock input stops.

PROGRAM MODULE NAME	COUNT PULSES	MCU/MPU	HD6305X0/Y0	LABEL	PLSCNT

FLOWCHART



5.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

5.5 PROGRAM LISTING

```
00001                   *
00002                   ****    RAM ALLOCATION    *************************
00003                   *
00004 0080                      ORG     $80
00005                   *
00006 0080 0001         STRTF   RMB     1         Flag indicating start and stop
00007 0081 0002         HEXD    RMB     2         2-byte hexadecimals    counter
00008 0083 0003         DECD    RMB     3         5-digit BCD
00009 0086 0001         HCNTR   RMB     1         Subtraction counter
00010                   *
00011                   ****    SYMBOL DEFINITIONS    ***********
00012                   *
00013       0008        TDR     EQU     $08       Timer data register
00014       0009        TCR     EQU     $09       Timer control register
00015                   ***********************************************
00016                   *                                             *
00017                   *          MAIN PROGRAM : PLSMN               *
00018                   *                                             *
00019                   ***********************************************
00020                   *
00021 1000                      ORG     $1000
00022                   *
00023 1000 A6 01        PLSMN   LDA     #$1       Set pulse count start flag
00024 1002 B7 80                STA     STRTF
00025 1004 CD 101F              JSR     PLSCNT    Start pulse count
00026 1007 AE C8                LDX     #200      Execute software timer of 200ms
00027 1009 A6 C8        PLSMN1  LDA     #200
00028 100B 4A           PLSMN2  DEC     A
00029 100C 26 FD                BNE     PLSMN2
00030 100E 5A                   DEC     X
00031 100F 26 F8                BNE     PLSMN1
00032 1011 3F 80                CLR     STRTF     Reset pulse count start flag
00033 1013 CD 101F              JSR     PLSCNT    Stop pulse count
00034 1016 3F 81                CLR     HEXD      Clear upper byte of HEXD
00035 1018 B7 82                STA     HEXD+1    Store pulse count result
00036 101A CD 1034              JSR     HEX       Convert result into BCD
00037 101D 20 FE        PEND    BRA     PEND      End of program
00038                   ***********************************************
00039                   *                                             *
00040                   *       NAME : PLSCNT (COUNT PULSES)          *
00041                   *                                             *
00042                   ***********************************************
00043                   *                                             *
00044                   *   ENTRY : STRTF (FLAG INDICATING START/STOP) *
00045                   * RETURNS : ACCA  (2's COMPLEMENT OF TDR)       *
00046                   *                                             *
00047                   ***********************************************
00048 101F 01 80 0A PLSCNT BRCLR  0,STRTF,PLSCT1  Test if count start or
00049 1022 A6 FF                LDA     #$FF      Initialize TDR          stop ?
00050 1024 B7 08                STA     TDR
00051 1026 A6 70                LDA     #$70      Start pulse count
00052 1028 B7 09                STA     TCR
00053 102A 20 07                BRA     PLSCT2
00054 102C A6 68        PLSCT1  LDA     #$68      Stop pulse count
00055 102E B7 09                STA     TCR
```

```
00056 1030 B6 08          LDA    TDR      Calculates 2's complement of
00057 1032 40             NEG    A                                  TDR
00058 1033 81       PLSCT2 RTS
00059                     ***********************************************
00060                     *                                             *
00061                     *  NAME : HEX (CONVERTING 2-BYTE HEXADECIMALS  *
00062                     *               INTO 5-DIGIT BCD)             *
00063                     *                                             *
00064                     ***********************************************
00065                     *                                             *
00066                     *          ENTRY : HEXD (2-BYTE HEXADECIMALS) *
00067                     *        RETURNS : DECD (5-DIGIT BCD)         *
00068                     *                                             *
00069                     ***********************************************
00070 1034 3F 83    HEX   CLR    DECD     Clear 5-digit BCD area
00071 1036 3F 84          CLR    DECD+1
00072 1038 3F 85          CLR    DECD+2
00073 103A A6 10          LDA    #16      Set shift counter
00074 103C B7 86          STA    HCNTR
00075 103E 38 82    HEX1  ASL    HEXD+1   Shift MSB of HEXD to carry
00076 1040 39 81          ROL    HEXD
00077 1042 AE 03          LDX    #3       Set ADDR pointer (addition
00078 1044 E6 82    HEX2  LDA    DECD-1,X DECD * 2 + C -> ACCA     count)
00079 1046 E9 82          ADC    DECD-1,X
00080 1048 8D             DAA             Convert into BCD
00081 1049 E7 82          STA    DECD-1,X Store 5-digit BCD area
00082 1048 5A             DEC    X        Decrement ADDR pointer
00083 104C 26 F6          BNE    HEX2     Branch until ADDR pointer=0
00084 104E 3A 86          DEC    HCNTR    Decrement shift counter
00085 1050 26 EC          BNE    HEX1     Branch until shift counter=0
00086 1052 81             RTS
00087                     ***********************************************
00088                     *                                             *
00089                     *              VECTOR ADDRESSES               *
00090                     *                                             *
00091                     ***********************************************
00092                     *
00093 1FF6                ORG    $1FF6
00094                     *
00095 1FF6 1000           FDB    PLSMN    SCI/TIMER2
00096 1FF8 1000           FDB    PLSMN    TIMER/INT2
00097 1FFA 1000           FDB    PLSMN    INT
00098 1FFC 1000           FDB    PLSMN    SWI
00099 1FFE 1000           FDB    PLSMN    RES
00100                     *
00101                     END
```

6. ZERO CROSS

6.1 HARDWARE DESCRIPTION

(1) Function

(a) Measures AC power frequency (50Hz or 60Hz) using the HD6305X0.

(b) Permits pulse conversion circuit to convert single 100V AC waveform into pulse waveform at double the frequency.

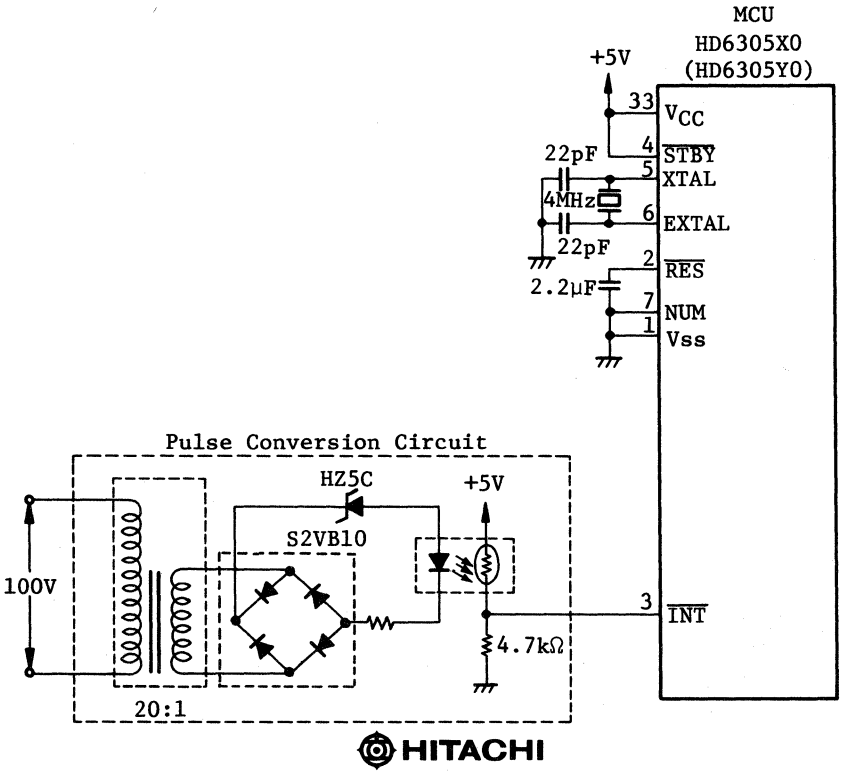
(c) Inputs AC power (100V) to pulse conversion circuit (photo coupler) and obtains frequency (50Hz or 60Hz).

(2) Microcomputer Applications

(a) Executes interrupt routine using built-in 8-bit timer with 7-bit prescaler (hereinafter, timer).

(b) Counts pulses of waveform converted from AC waveform by input from $\overline{INT}$ pin.

(3) Circuit Diagram



(4) Pin Functions

Pin function at the interface between the HD6305X0 and pulse conversion circuit is shown in Table 1.

Table 1 Pin Functions

Pin Name (HD6305X0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Pulse conversion circuit)	Program Label
$\overline{INT}$	Input	Low	Inputs power pulse signal.	Photo Coupler output	

(5) Hardware Operation

Timing charts of AC power frequency and pulse conversion are shown in Fig. 2.

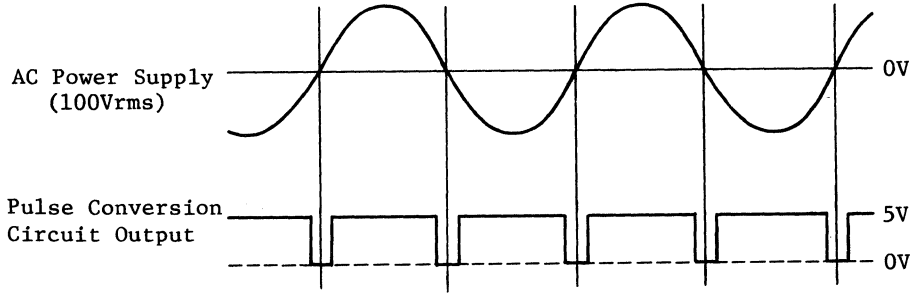


Fig. 2 AC Power Supply and Pulse Conversion Timing

6.2 SOFTWARE DESCRIPTION

(1) Program Module Configuration

The program module configuration for AC power frequency measurement is shown in Fig. 3.

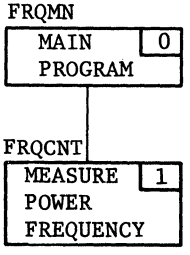


Fig. 3 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	FRQMN	Measures AC power frequency.
1	MEASURE POWER FREQUENCY	FRQCNT	Measures AC power frequency and outputs result (50Hz, 60Hz, or other frequencies outside of 45 - 65Hz)

(3) Program Module Sample Application (Main Program)

The flowchart in Fig. 4 is an example of AC power frequency measurement performed by the program module in Fig. 3.

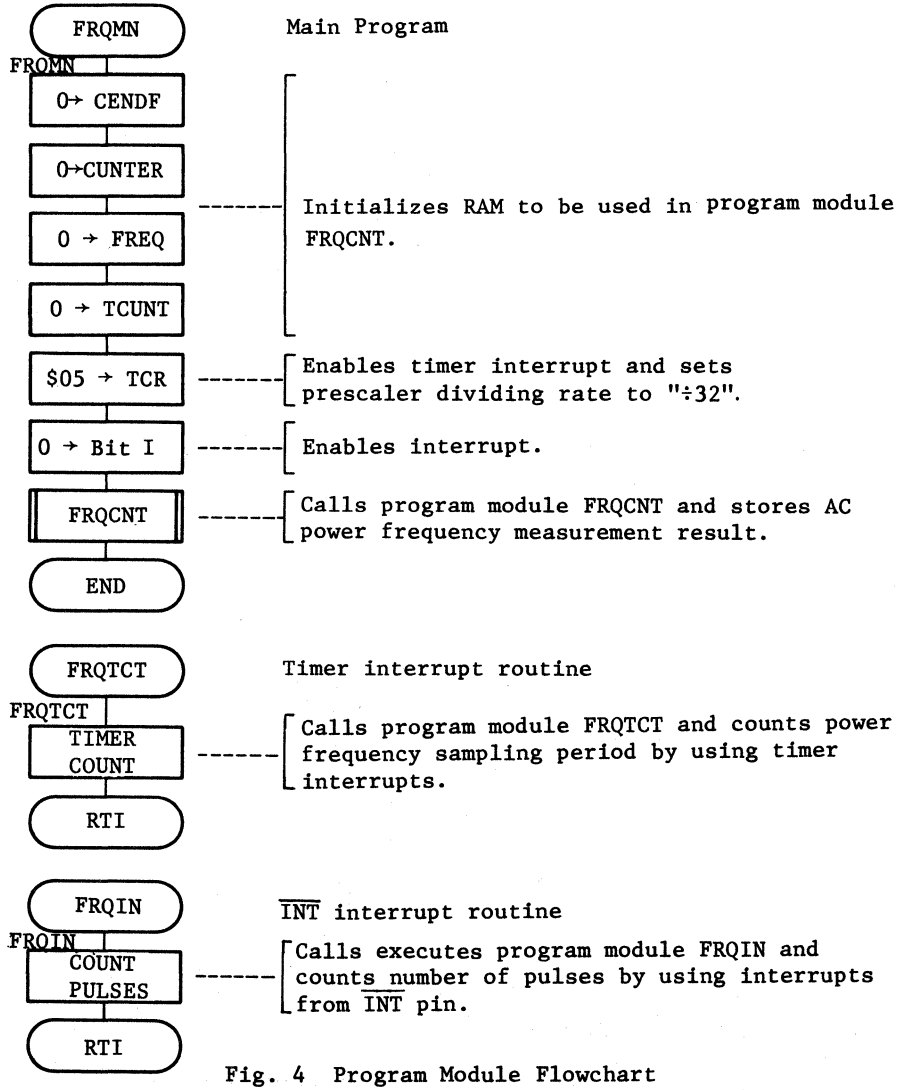


Fig. 4 Program Module Flowchart

PROGRAM MODULE NAME	MEASURE POWER FREQUENCY	MCU/MPU	HD6305X0/Y0	LABEL	FRQCNT

FUNCTION

Checks whether power frequency supplied is 50Hz or 60Hz depending on pulse width input to $\overline{INT}$ pin.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	—	—	—
	Returns	Frequency measurement result	FREQ (RAM)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	×

C	Z
×	×
N	I
×	×
H	
×	

SPECIFICATIONS

ROM (Bytes)
49
RAM (Bytes)
3
Stack (Bytes)
0
No. of cycles
35
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details
FREQ(RAM): Contains flag indicating result of frequency measurement. Table 3 shows flag functions.

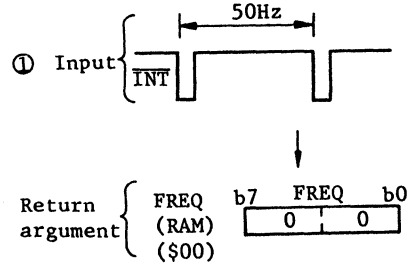


Fig. 5 Example of FRQCNT Execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles required to measure frequency of 60 Hz.

DESCRIPTION

Table 3 Flag Functions

Label	Bit/Label		Function
	bit 1	bit 0	
FREQ	0	0	Frequency is 50Hz.
	0	1	Frequency is 60Hz.
	1	1	Frequency is other than 50Hz or 60Hz.

(b) Fig. 5 shows an example of program module FRQCNT execution. If power supply of 50Hz is measured as shown in part ① of Fig. 5, result is contained in FREQ(RAM) as shown in part ② of Fig. 5.

(c) Program module FRQCNT calls subroutines shown in Table 4.

Table 4 Subroutines Called in FRQCNT

Program module/ subroutine name	Label	Function
TIMER COUNT	FRQCTC	Counts sampling period of frequency supplied.
COUNT PULSES	FRQIN	Counts number of pulses.

(2) User Notes

(a) Initializes timer.

(b) Clears bit I to enable timer interrupt.

(3) RAM Description

Label	RAM b7 b0	Description
CENDF		Stores flag indicating end of sampling.
CUNTER		Stores counter indicating number of pulses input from $\overline{\text{INT}}$ pin.
FREQ		Stores frequency measurement result.

PROGRAM MODULE NAME	MEASURE POWER FREQUENCY	MCU/MPU	HD6305X0/Y0	LABEL	FRQCNT

DESCRIPTION

(4) Sample Application

Program module FRQCNT is called after timer is initialized and interrupt is enabled.

```
        CLR     A
        STA     CENDF
        STA     CUNTER      }--Initializes RAM to be used in program module FRQCNT.
        STA     FREQ
        STA     TCUNT
        LDA     $05         }--Enables timer interrupt and sets prescaler dividing
        STA     TCR         }  rate to "÷32".
        CLI                 ---Enables interrupt.
    +-------------------+
    |  JSR    FRQCNT    |   ---Calls program module FRQCNT.
    +-------------------+

        BRSET   1,FREQ,ERR  ---Branches to service routine in case of
                                frequency is outside 45 - 65 Hz.
        BRSET   0,FREQ,HZ60 ---Branches to service routine in case of 60 Hz
                                frequency.
    +-------------------------+
    | Service routine in      |
    | case of 50Hz            |
    +-------------------------+
        BRA     USER
HZ60+-------------------------+
    | Service routine in      |
    | case of 60Hz            |
    +-------------------------+
        BRA     USER
ERR +-------------------------+
    | Service routine in case |
    | of outside 45 - 60Hz    |
    +-------------------------+
USER+-------------------------+
    | User program            |
    +-------------------------+
```

(5) Basic Operation

(a) CUNTER(RAM) is used in subroutine FRQIN.

(b) Counts pulses output from pulse conversion circuit using CUNTER(RAM).

(c) Counts pulses during 0.8s using subroutine FRQTCT.

(d) After having completed to count pulses, subroutine FRQTCT sets flag CENDF(RAM).

(e) Program module FRQCNT tests if counting pulses is completed by CENDF(RAM) flag.

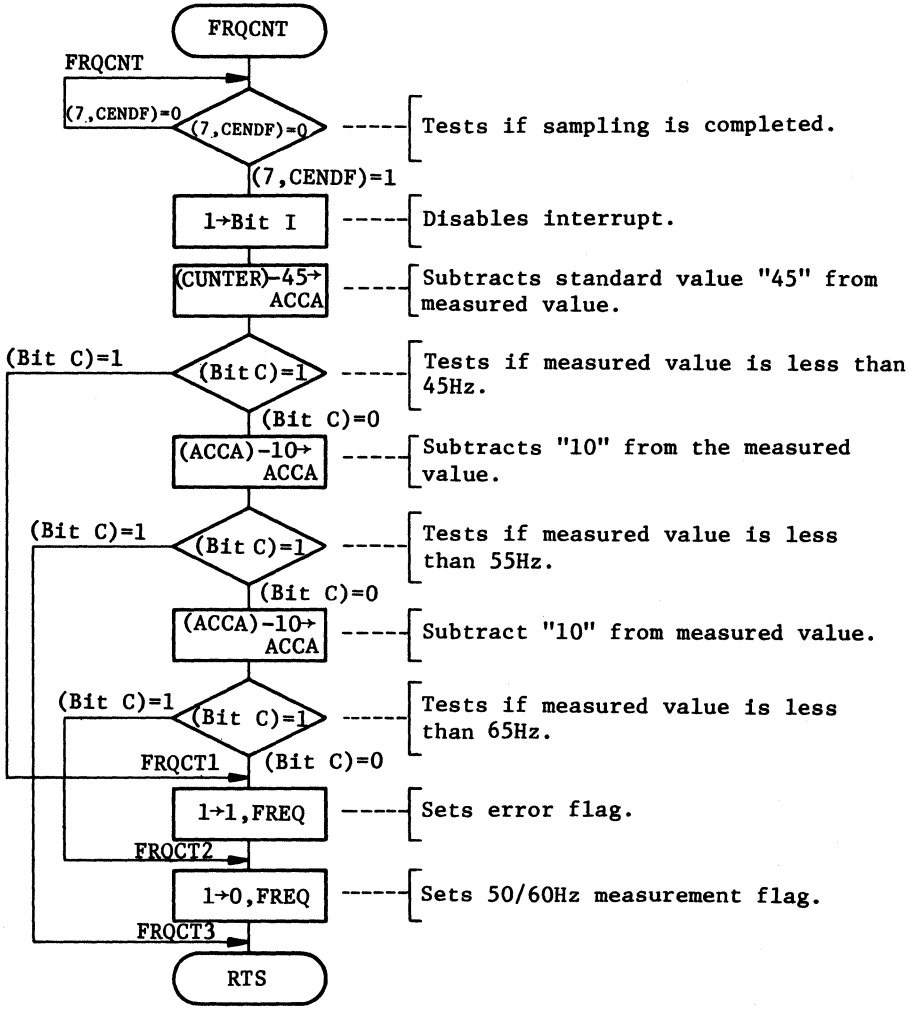
(f) When counting pulses is completed, test if content of CUNTER(RAM) is within range of 45 ∿ 60 using subtraction instruction.

(g) Tests whether 50Hz or 60Hz from subtraction result in (f) using subtraction instruction.

(h) After testing (f) and (g), set bits 0 and 1 of FREQ(RAM) corresponding to each result.

PROGRAM MODULE NAME	MEASURE POWER FREQUENCY	MCU/MPU	HD6305X0/Y0	LABEL	FRQCNT

FLOWCHART



6.4 SUBROUTINE DESCRIPTION

SUBROUTINE NAME	TIMER COUNT	MCU/MPU	HD6305X0/Y0	LABEL	FRQTCT

FUNCTION

Counts power frequency sampling period using timer interrupt.

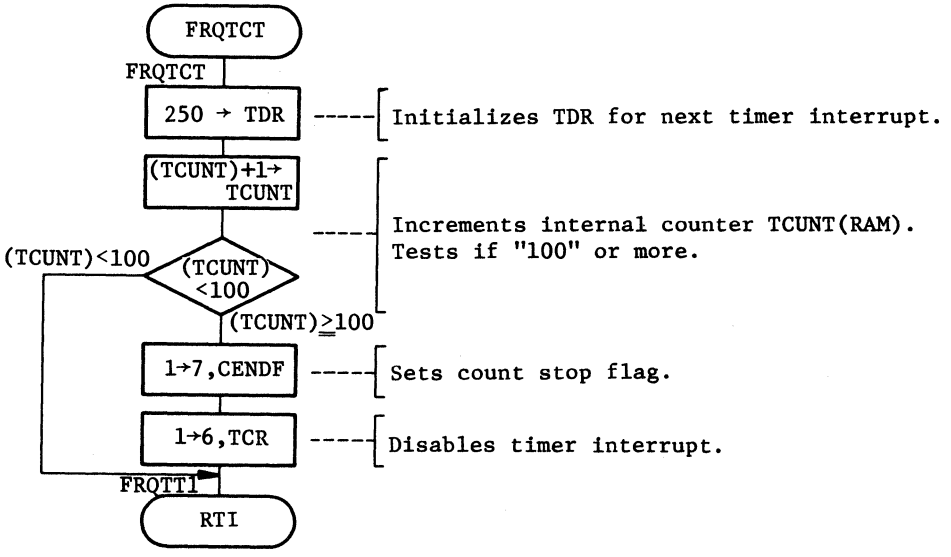
BASIC OPERATION

(1) Increments TCUNT(RAM) at every timer interrupt.

(2) Loops step (1) until TCUNT(RAM) is "100".

(3) Stops timer interrupt and sets count stop flag if TCUNT(RAM) is "100" or more.

FLOWCHART | **PROGRAM MODULE USING THIS SUBROUTINE** —



SUBROUTINE NAME	COUNT PULSES	MCU/MPU	HD6305X0/Y0	LABEL	FRQIN

FUNCTION

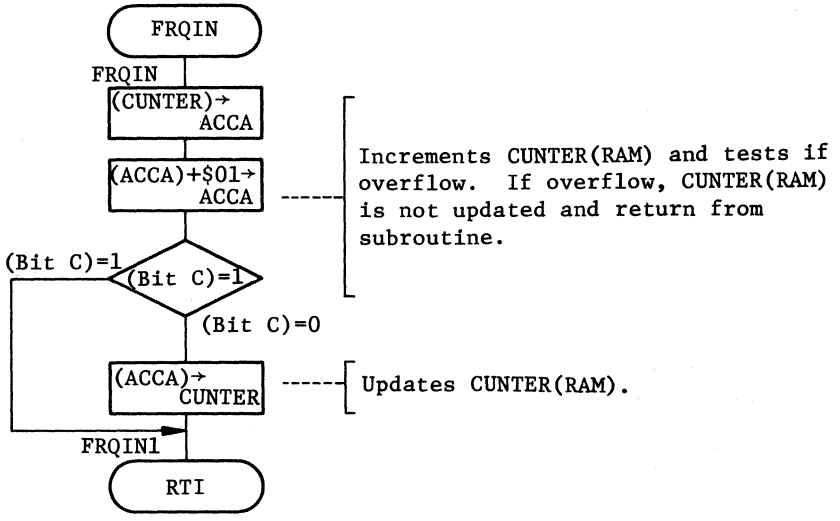
Counts number of pulses using interrupts from $\overline{\text{INT}}$ pin.

BASIC OPERATION

Increments CUNTER(RAM) until $FF.

FLOWCHART

PROGRAM MODULE USING THIS SUBROUTINE —



```
00001                    *
00002                    ****    RAM ALLOCATION ******************************
00003                    *
00004 0080                       ORG     $80
00005                    *
00006 0080 0001          CENDF   RMB     1         Zero cross check/stop flag
00007 0081 0001          CUNTER  RMB     1         Frequency counter
00008 0082 0001          FREQ    RMB     1         Frequency result
00009 0083 0001          TCUNT   RMB     1         Timer counter
00010                    *
00011                    ****    SYMBOL DEFINITIONS   ************************
00012                    *
00013          0008      TDR     EQU     $08       Timer data register
00014          0009      TCR     EQU     $09       Timer control register
00015                    ********************************************************
00016                    *                                                      *
00017                    *              MAIN PROGRAM : FRQMN                    *
00018                    *                                                      *
00019                    ********************************************************
00020                    *
00021 1000                       ORG     $1000
00022                    *
00023 1000 4F            FRQMN   CLR     A         Initialize RAM
00024 1001 B7 80                 STA     CENDF
00025 1003 B7 81                 STA     CUNTER
00026 1005 B7 82                 STA     FREQ
00027 1007 B7 83                 STA     TCUNT
00028 1009 A6 05                 LDA     #$5       Initialize TCR
00029 100B B7 09                 STA     TCR
00030 100D 9A                    CLI               Enable inturrupts
00031 100E AD 02                 BSR     FRQCNT    Measure power frequency
00032 1010 20 FE         PEND    BRA     PEND      End of main program
00033                    ********************************************************
00034                    *                                                      *
00035                    *       NAME : FRQCNT (MEASURE POWER FREQUENCY)        *
00036                    *                                                      *
00037                    ********************************************************
00038                    *                                                      *
00039                    *      ENTRY : NOTHING                                 *
00040                    *    RETURNS : FREQ (50Hz=0,60Hz=1,ERROR=3)            *
00041                    *                                                      *
00042                    ********************************************************
00043 1012 0F 80 FD      FRQCNT  BRCLR   7,CENDF,FRQCNT  Timer interrupt end ?
00044 1015 9B                    SEI               Disable interrupts
00045 1016 B6 81                 LDA     CUNTER    Test if under 45Hz
00046 1018 A0 2D                 SUB     #45
00047 101A 25 08                 BCS     FRQCT1    Test if 50Hz
00048 101C A0 0A                 SUB     #10
00049 101E 25 08                 BCS     FRQCT3    Test if 60Hz
00050 1020 A0 0A                 SUB     #10
00051 1022 25 02                 BCS     FRQCT2
00052 1024 12 82         FRQCT1  BSET    1,FREQ    Set enor flag
00053 1026 10 82         FRQCT2  BSET    0,FREQ    Set flag of 60Hz
00054 1028 81            FRQCT3  RTS
00055                    ********************************************************
```

```
00056                   *                                                     *
00057                   *       NAME : FRQIN (COUNT PULS)                     *
00058                   *                                                     *
00059                   *******************************************************
00060 1029 B6 81  FRQIN  LDA     CUNTER   CUNTER+1->ACCAP
00061 102B AB 01         ADD     #1
00062 102D 25 02         BCS     FRQIN1   Branch if overflow
00063 102F B7 81         STA     CUNTER   Update CUNTER
00064 1031 80     FRQIN1 RTI
00065                   *******************************************************
00066                   *                                                     *
00067                   *           NAME : FRQTCT (COUNT TIMER)               *
00068                   *                                                     *
00069                   *******************************************************
00070 1032 A6 FA  FRQTCT LDA     #250     Initialize TDR
00071 1034 B7 08         STA     TDR
00072 1036 3C 83         INC     TCUNT    TCUNT+1->TCUNT
00073 1038 B6 83         LDA     TCUNT
00074 103A A1 64         CMP     #100     TCUNT < 100 ?
00075 103C 25 04         BCS     FRQTT1   Branch if TCUNT<100
00076 103E 1E 80         BSET    7,CENDF  Set count stop flag
00077 1040 1D 09         BCLR    6,TCR    Disable timer interrupt
00078 1042 80     FRQTT1 RTI
00079                   *******************************************************
00080                   *                                                     *
00081                   *           VECTOR ADDRESSES                          *
00082                   *                                                     *
00083                   *******************************************************
00084                   *
00085 1FF6                ORG     $1FF6
00086                   *
00087 1FF6 1000           FDB     FRQMN    SCI/TIMER2
00088 1FF8 1032           FDB     FRQTCT   TIMER/INT2
00089 1FFA 1029           FDB     FRQIN    INT
00090 1FFC 1000           FDB     FRQMN    SWI
00091 1FFE 1000           FDB     FRQMN    RES
00092                   *
00093                     END
```

7. KEY MATRIX (8 × 4)

7.1 HARDWARE DESCRIPTION

(1) Function

(a) Executes key scan of 8×4 key matrix using the HD6305X0.

(b) Key data received is converted into ASCII of 'A' - 'Z', '1' - '6'.

(c) If two keys are depressed at the same time, data is invalid.

(2) Microcomputer Applications

(a) Executes interrupt routine every 8ms by using built-in 15-bit timer (hereinafter, timer 2) of HD6305X0.

(b) Executes key scan by outputting strobe signal from port C in interrupt routine.

(c) Cancels key chatter using interrupt routine.

(d) Strobe signal for key scan is controlled by switching input/output directions using DDR of Port C. In this method, ports which output no signals are input ports (high-impedance) therefore, diode for preventing collision between input and output signals is unnecessary.

(3) Circuit Diagram

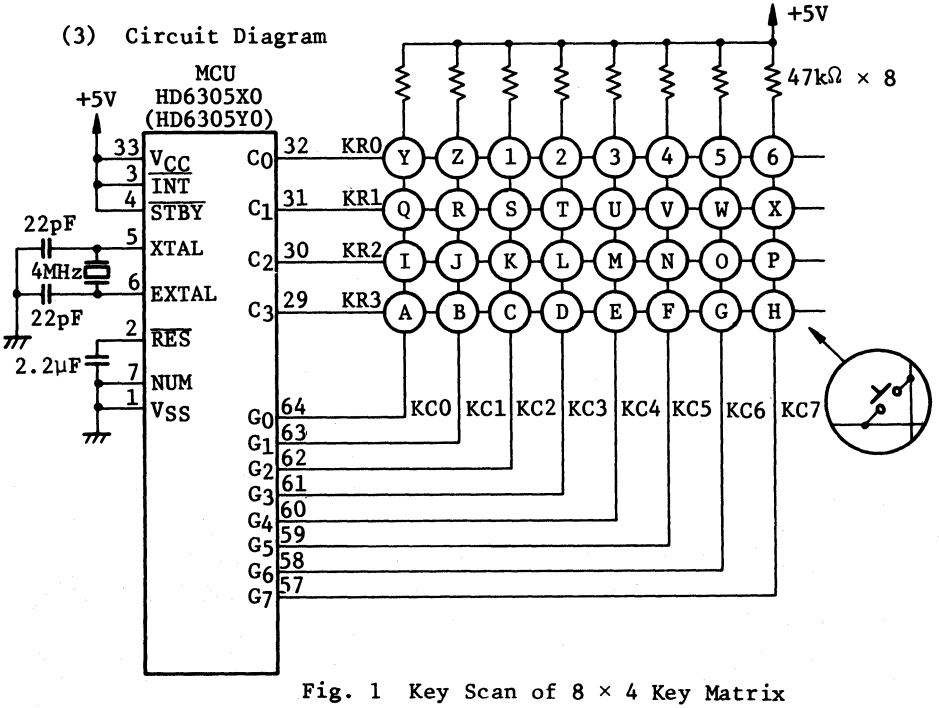


Fig. 1 Key Scan of 8 × 4 Key Matrix

(4) Pin Functions

Pin functions of the HD6305X0 and key matrix are shown in Table 1.

Table 1 Pin Functions

Pin Name (HD6305X0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Key Matrix)	Program Label
C_3	Output	Low	Outputs strobe signal for 8 × 4 key matrix retrieval.	KR3	PCDTR
C_2	Output	Low		KR2	
C_1	Output	Low		KR1	
C_0	Output	Low		KR0	
G_0	Input	—	Inputs key data of 8 × 4 key matrix.	KC0	PGDTR
G_1	Input	—		KC1	
G_2	Input	—		KC2	
G_3	Input	—		KC3	
G_4	Input	—		KC4	
G_5	Input	—		KC5	
G_6	Input	—		KC6	
G_7	Input	—		KC7	

(5) Hardware Operation

Timing chart for preventing chatter is shown in Fig. 2.

Key input signals are sampled every 8ms. If the same key input signal is sampled three times consecutively, key input data is established.

If the same key signal is sampled twice or less, the signal is canceled as chatter.

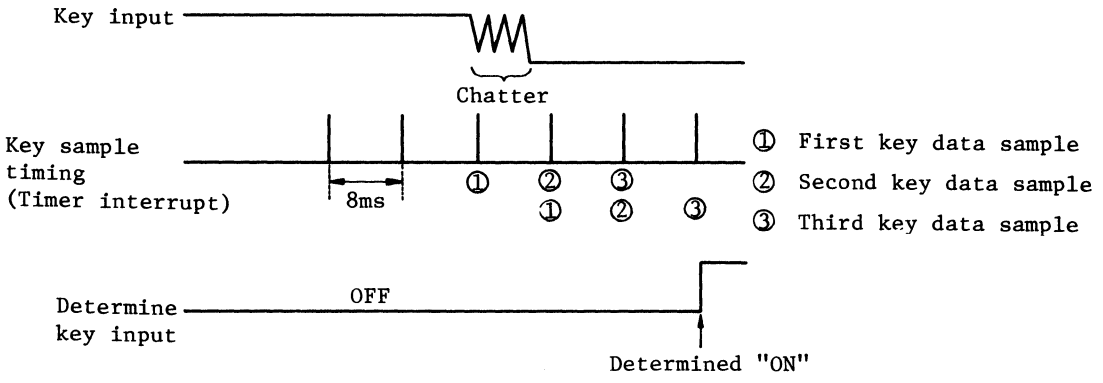


Fig. 2 Chatter Prevention Timing

7.2 SOFTWARE DESCRIPTION

(1) Program Module Configuration

The program module configuration for executing key scan of 8 × 4 key matrix is shown in Fig. 3.

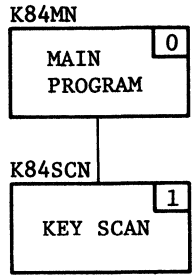


Fig. 3 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	K84MN	Converts 8 × 4 key matrix key scan data into ASCII.
1	KEY SCAN	K84SCN	Executes key scan of 8 × 4 key matrix.

(3) Program Module Sample Application (Main Program)

The flowchart in Fig. 4 is an example of key scan of 8 × 4 key matrix performed by the program module in Fig. 3.
The program in Fig. 4 stores key scan data in KEYSET(RAM).

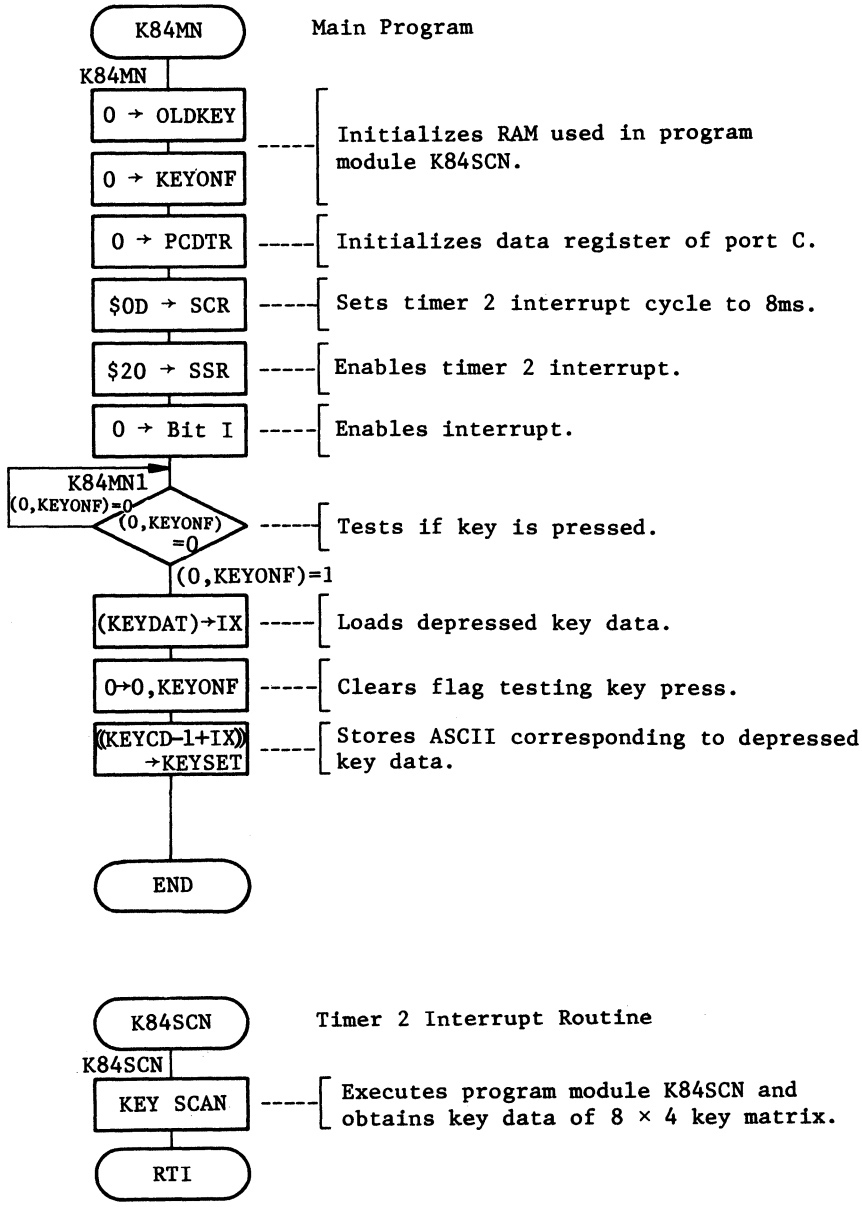


Fig. 4 Program Module Flowchart

PROGRAM MODULE NAME	KEY SCAN	MCU/MPU	HD6305X0/Y0	LABEL	K84SCN

FUNCTION

Executes key scan of 8 × 4 key matrix and stores key scan data in KEYDAT(RAM).

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	—	—	—
	Returns	Key data	KEYDAT (RAM)	1
		Establishment of key data	KEYONF (RAM)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	×

C	Z
×	×
N	I
×	●
H	
×	

SPECIFICATIONS

ROM (Bytes)
102
RAM (Bytes)
9
Stack (Bytes)
0
No. of cycles
428
Reentrant
No
Relocation
No
Interrupt
No

DESCRIPTION

(1) Function Details

(a) Argument details
KEYDAT(RAM) : Contains key scan data.
KEYONF(RAM) : Contains program module K84SCN completion status.
KEYONF(RAM)=1 : Indicates that key scan data is stored in KEYDAT(RAM).

① Key depressed { D key in "Circuit Diagram" is pressed.

② Return arguments {
KEYDAT(RAM) ($04) — b7 KEYDAT b0: 0 | 4
KEYONF(RAM) ($01) — KEYONF: 0 | 1

Fig. 5 Example of K84SCN Execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles required until key data is valid.

PROGRAM MODULE NAME	KEY SCAN	MCU/MPU	HD6305X0/Y0	LABEL	K84SCN

DESCRIPTION

KEYONF(RAM)=0 : Indicates that key scan data is not stored in KEYDAT(RAM).

(b) Fig. 5 shows an example of program module K84SCN execution. If key in part ① of Fig. 5 is depressed, key scan data is contained in KEYDAT(RAM) as shown in part ② of Fig. 5.

(c) Program module K84SCN calls neither program modules nor subroutines.

(2) User Notes

(a) Initializes timer 2.

(b) Clears bit I and enables interrupt.

(3) RAM Description

Label	RAM b7 b0	Description
OLDKEY		Stores previous key input data.
NEWKEY		Stores present key input data.
STBDAT		Stores data for strobe signal output.
KEYNUM		Stores key number.
TOTLKY		Stores total number of depressed key during present key scan.
KEYONF		Stores flag indicating key being defined during key scan.
KEYDAT		Stores defined key number by key scan.
SCOUNT		Stores counter for retrieving depressed key among key input data.
CHATFL		Bit 3-0; Stores counter for counting number of key scan data comparison. Bit 7 ; Stores flag for indicating whether chatter elimination has been completed.

(4) Sample Application

Program module K84SCN is called every 8ms after initializing timer 2 and enabling interrupt.

```
          CLR     OLDKEY      }
          CLR     KEYONF      }--- Clears RAM to be used.
          CLR     PCDTR        --- Initializes port C.
          LDA     #$0D        }
          STA     SCR         }
          LDA     #$20        }---Sets timer 2 interrupt cycle to 8ms, and
          STA     SSR         }   enables timer 2 interrupt.
          CLI                  --- Enables interrupt.
LOOP      BRCLR   0,KEYONF,LOOP---Tests key press.,
          LDA     KEYDAT      }
          STA     KEYSET      }---Stores key scan data in RAM.
          BCLR    0,KEYONF     ----Clears key press flag.
```

PROGRAM MODULE NAME	KEY SCAN	MCU/MPU	HD6305X0/Y0	LABEL	K84SCN

DESCRIPTION

(5) Basic Operation.

(a) Executes key scan with every 8ms timer interrupt. Checks key scan execution flag (KEYONF(RAM)) at the beginning of key scan, and decides whether present key scan data will be executed or not.

(b) Outputs strobe signal from bit 3 to 0 of port C one by one, and fetches key press data from port G.

(c) Tests if key scan data loaded into (b) is $FF.

(i) $FF key scan data means that key of the column supplying current strobe signal was not depressed. Key scan for next column is started.

(ii) Key scan data other than $FF means key of the column supplying current strobe signal is depressed. Test which row is pressed.

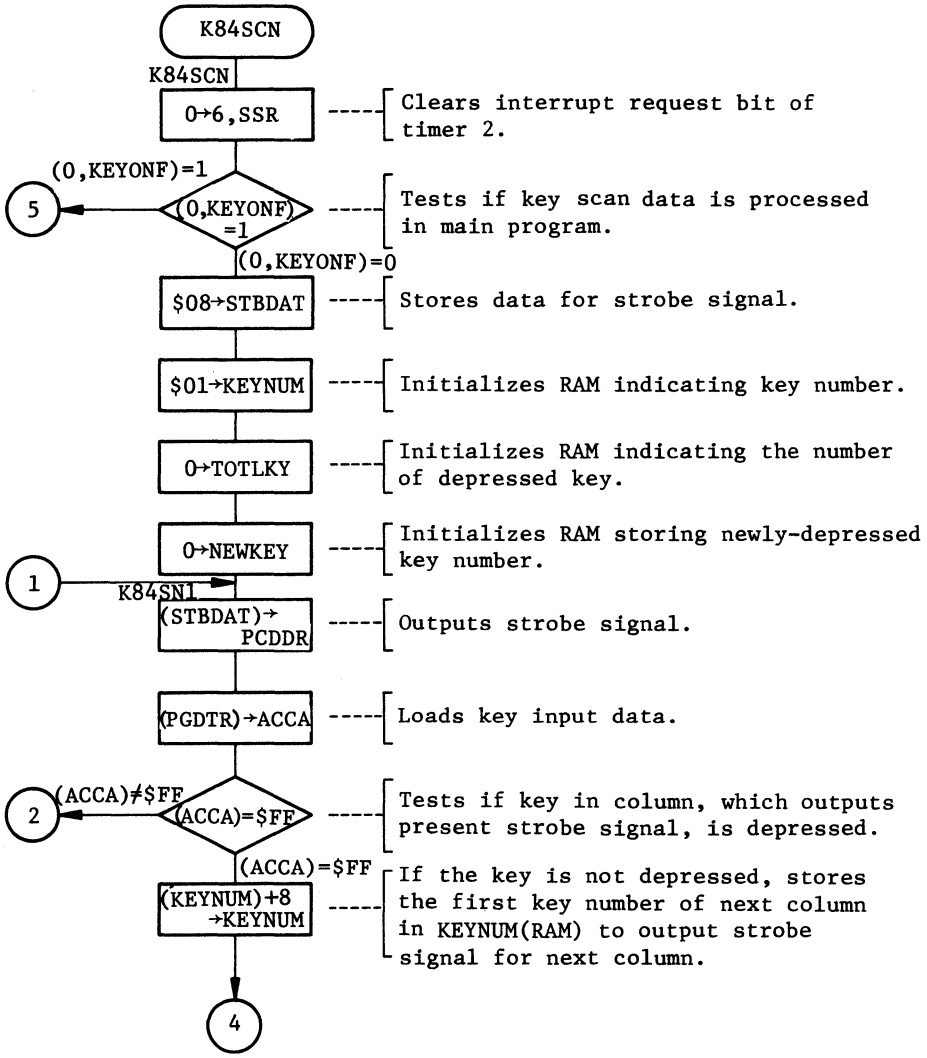
① Tests if bit C is 1 or 0 by shifting ACCA content where key scan data is stored, by 1 bit to right 8 times. In case of bit C is 0, the key was depressed.

② To test double keys depressed, if key scan data includes 0, TOTLKY(RAM) is incremented. In case of TOTLKY=1, KEYNUM(RAM) content is stored in NEWKEY(RAM). In case of TOTLKY>1, double keys were depressed and return from program module is performed.

(d) Compares key data (NEWKEY(RAM)) detected in (c) with previous key data (OLDKEY(RAM)). If they are the same, counts-up lower 4 bits of chatter counter CHATFL(RAM). When the counter indicates "3", the key is defined. This is indicated by setting MSB of chatter cancel flag CHATFL(RAM) to "1". The CHATFL(RAM) indicates both counter and flag. The chatter cancel flag is cleared when NEWKEY(RAM) and OLDKEY(RAM) data are different, and when no key is depressed.

PROGRAM MODULE NAME	KEY SCAN	MCU/MPU	HD6305X0/Y0	LABEL	K84SCN

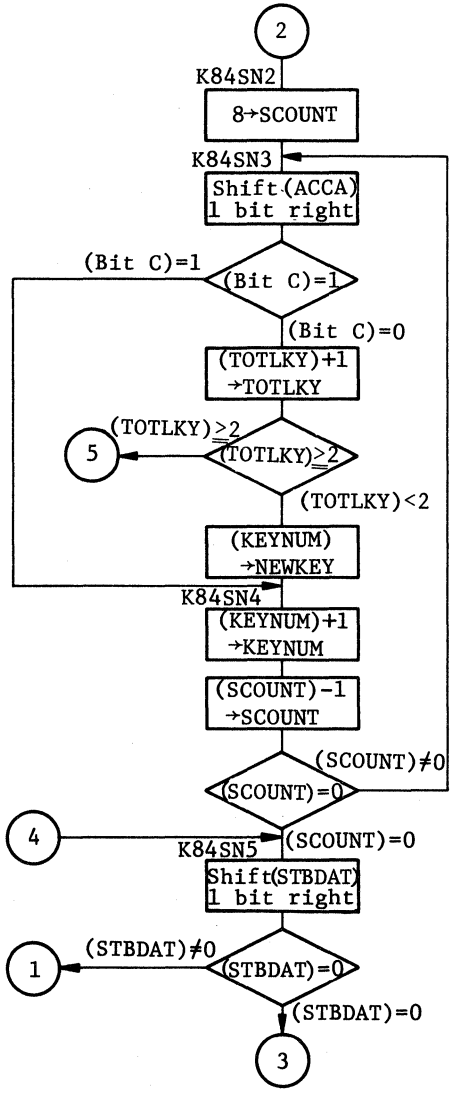
FLOWCHART



7

PROGRAM MODULE NAME	KEY SCAN	MCU/MPU	HD6305X0/Y0	LABEL	K84SCN

FLOWCHART



Stores shift counter to check what key is depressed among 8 bits.

Shifts key input data 1 bit right.

Tests if key is depressed.

Tests if 2 keys are depressed at the same time.

Stores depressed key number in NEWKEY(RAM).

Increments RAM indicating key number.

Decrements shift counter.

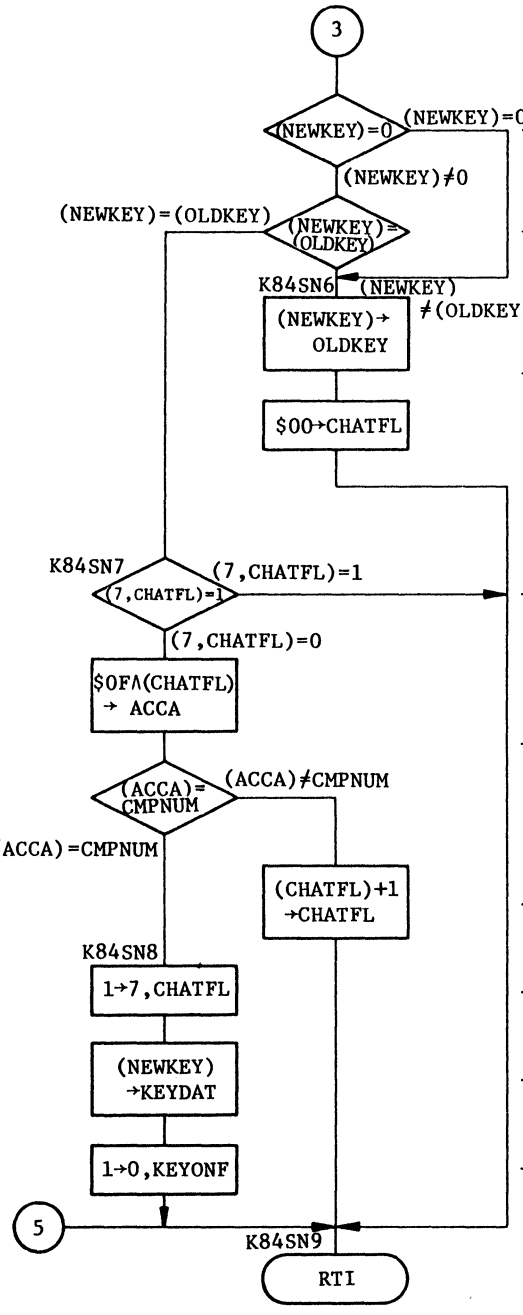
Tests if one row key scan is completed.

Shift strobe signal data 1 bit right to execute key scan for next column.

Tests if all column key scan is completed.

PROGRAM MODULE NAME	KEY SCAN	MCU/MPU	HD6305X0/Y0	LABEL	K84SCN

FLOWCHART



Tests if present key input data is 0.

Compares present key input data with previous key input data.

If present key input data and previous data are different, stores NEWKEY(RAM) contents in OLDKEY(RAM), and clears RAM indicating the number of key scan and continuous key press.

Tests if key is continuously pressed.

Compares the number of chatter (CHATFL(RAM)) with constant number. If they are the same, determines as key press and fetches key data.

Increments the number of chatter.

Sets flag indicating the termination of chatter prevention.

Stores depressed key data in RAM for processing with main program.

Sets flag indicating depressed key data being defined.

7

7.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

7.5 PROGRAM LISTING

```
00001                    *
00002                    ****    RAM ALLOCATION    ************************
00003                    *
00004 0080                       ORG     $80
00005                    *
00006 0080 0001          KEYSET  RMB     1         ASCII data
00007 0081 0001          OLDKEY  RMB     1         Old key data
00008 0082 0001          NEWKEY  RMB     1         New key data
00009 0083 0001          CHATFL  RMB     1         Chatter counter and flag
00010 0084 0001          KEYONF  RMB     1         Key on flag
00011 0085 0001          KEYDAT  RMB     1         Key data
00012 0086 0001          TOTLKY  RMB     1         Total key number
00013 0087 0001          KEYNUM  RMB     1         Key number
00014 0088 0001          STBDAT  RMB     1         Strobe data
00015 0089 0001          SCOUNT  RMB     1         Sift counter
00016                    *
00017                    ****    SYMBOL DEFINITION   **********************
00018                    *
00019       0002         PCDTR   EQU     $02       Port C data direction register
00020       0006         PCDDR   EQU     $06       Port C data register
00021       0007         PGDDR   EQU     $07       Port G data direction register
00022       000D         PGDTR   EQU     $0D       Port G data register
00023       0010         SCR     EQU     $10       SCI control register
00024       0011         SSR     EQU     $11       SCI status register
00025       0003         CMPNUM  EQU     $03       Chatter number
00026                    ************************************************
00027                    *                                              *
00028                    *           MAIN PROGRAM : K84MN               *
00029                    *                                              *
00030                    ************************************************
00031                    *
00032 1000                       ORG     $1000
00033                    *
00034 1000 4F            K84MN   CLR     A         Initialize RAM
00035 1001 B7 81                 STA     OLDKEY
00036 1003 B7 84                 STA     KEYONF
00037 1005 B7 02                 STA     PCDTR     Initialize port C
00038 1007 A6 0D                 LDA     #$0D      Initialize SCR
00039 1009 B7 10                 STA     SCR
00040 100B A6 20                 LDA     #$20      Initialize SSR
00041 100D B7 11                 STA     SSR
00042 100F 9A                    CLI               Enable interrupt
00043 1010 01 84 FD      K84MN1  BRCLR   0,KEYONF,K84MN1  Test if key on ?
00044 1013 BE 85                 LDX     KEYDAT    Load key data
00045 1015 11 84                 BCLR    0,KEYONF  Clear key on flag
00046 1017 D6 1083               LDA     KEYCD-1,X  Store ASCII correspond to
00047 101A B7 80                 STA     KEYSET                    key data
00048 101C 20 FE         PEND    BRA     PEND      End of program
00049                    ************************************************
00050                    *                                              *
00051                    *          NAME : K84SCN (KEY SCAN)            *
00052                    *                                              *
00053                    ************************************************
00054                    *                                              *
00055                    *     NTRY    : NOTHING                        *
```

```
00056                    *    RETURNS : KEYDAT (KEY DATA)                *
00057                    *              KEYONF (ESTAFLISHMENT OF KEY ON) *
00058                    *                                               *
00059                    ************************************************
00060 101E 1D 11       K84SCN BCLR   6,SSR     Clear interrupt request flag
00061 1020 00 84 60            BRSET  0,KEYONF,K84SN9  Test if key on flag=1 ?
00062 1023 A6 08               LDA    #$08     Initialize strobe data
00063 1025 B7 88               STA    STBDAT
00064 1027 A6 01               LDA    #$01     Initialize key number
00065 1029 B7 87               STA    KEYNUM
00066 102B 3F 86               CLR    TOTLKY   Initialize total key
00067 102D 3F 82               CLR    NEWKEY   Initialize new key
00068 102F B6 88       K84SN1  LDA    STBDAT   Output strobe
00069 1031 B7 06               STA    PCDDR
00070 1033 B6 0D               LDA    PGDTR    Load key data
00071 1035 A1 FF               CMP    #$FF     Test if key on ?
00072 1037 26 08               BNE    K84SN2   Branch if key on
00073 1039 A6 08               LDA    #8       Store next key number
00074 103B BB 87               ADD    KEYNUM
00075 103D B7 87               STA    KEYNUM
00076 103F 20 19               BRA    K84SN5
00077 1041 AE 08       K84SN2  LDX    #8       Store sift counter
00078 1043 BF 89               STX    SCOUNT
00079 1045 47          K84SN3  ASR    A        Test if key on
00080 1046 25 0C               BCS    K84SN4   Branch if not key on
00081 1048 3C 86               INC    TOTLKY   Increment total key
00082 104A BE 86               LDX    TOTLKY   Test if 2 key on
00083 104C A3 01               CPX    #1
00084 104E 25 33               BCS    K84SN9   Branch if 2 key on
00085 1050 BE 87               LDX    KEYNUM   Store key data in new key
00086 1052 BF 82               STX    NEWKEY
00087 1054 3C 87       K84SN4  INC    KEYNUM   Increment key number
00088 1056 3A 89               DEC    SCOUNT   Decrement sift counter
00089 1058 26 EB               BNE    K84SN3   Test if sift 8 bits
00090 105A 37 88       K84SN5  ASR    STBDAT   Output next strobe
00091 105C 26 D1               BNE    K84SN1   Test if all scan is compleated
00092 105E B6 82               LDA    NEWKEY   Test if new key=0?
00093 1060 27 04               BEQ    K84SN6   Branch if new key is 0
00094 1062 B1 81               CMP    OLDKEY   New key = old key ?
00095 1064 27 06               BEQ    K84SN7   Branch if equal
00096 1066 B7 81       K84SN6  STA    OLDKEY   Store new key in old key
00097 1068 3F 83               CLR    CHATFL   Clear chatter counter
00098 106A 20 17               BRA    K84SN9
00099 106C 0E 83 14    K84SN7  BRSET  7,CHATFL,K84SN9  Test chatter end flag
00100 106F A6 0F               LDA    #$0F     Check chatter number
00101 1071 B4 83               AND    CHATFL
00102 1073 B1 03               CMP    CMPNUM
00103 1075 27 04               BEQ    K84SN8   Branch if chatter number=3
00104 1077 3C 83               INC    CHATFL   Increment chatter counter
00105 1079 20 08               BRA    K84SN9
00106 107B 1E 83       K84SN8  BSET   7,CHATFL Set chatter end flag
00107 107D B6 82               LDA    NEWKEY   Store new key in key data
00108 107F B7 85               STA    KEYDAT
00109 1081 10 84               BSET   0,KEYONF Set key on flag
00110 1083 80          K84SN9  RTI
```

```
00111                   ************************************************
00112                   *                                              *
00113                   *       DATA TABLE                             *
00114                   *                                              *
00115                   ************************************************
00116 1084 41           KEYCD   FCC     "ABCDEFGH"
00117 108C 49                   FCC     "IJKLMNOP"
00118 1094 51                   FCC     "QRSTUVWX"
00119 109C 59                   FCC     "YZ123456"
00120                   ************************************************
00121                   *                                              *
00122                   *       VECTOR ADDRESSES                       *
00123                   *                                              *
00124                   ************************************************
00125                   *
00126 1FF6                      ORG     $1FF6
00127                   *
00128 1FF6 101E                 FDB     K84SCN    SCI/TIMER2
00129 1FF8 1000                 FDB     K84MN     INT2
00130 1FFA 1000                 FDB     K84MN     INT
00131 1FFC 1000                 FDB     K84MN     SWI
00132 1FFE 1000                 FDB     K84MN     RES
00133                   *
00134                           END
```

8. FLUORESCENT DISPLAY CONTROL

8.1 HARDWARE DESCRIPTION

(1) Function

(a) Displays on 8-segment × 8-digit fluorescent display tube using the HD6305X0.

(b) Controls fluorescent display tube by dynamic drive.

(2) Microcomputer Applications

(a) Executes interrupt routine every 1ms using a 15-bit timer contained in the HD6305X0 (hereinafter, timer 2).

(b) Drives fluorescent display tube by supplying segment signal from port A and digit signal from port C in interrupt routine.

(3) Circuit Diagram

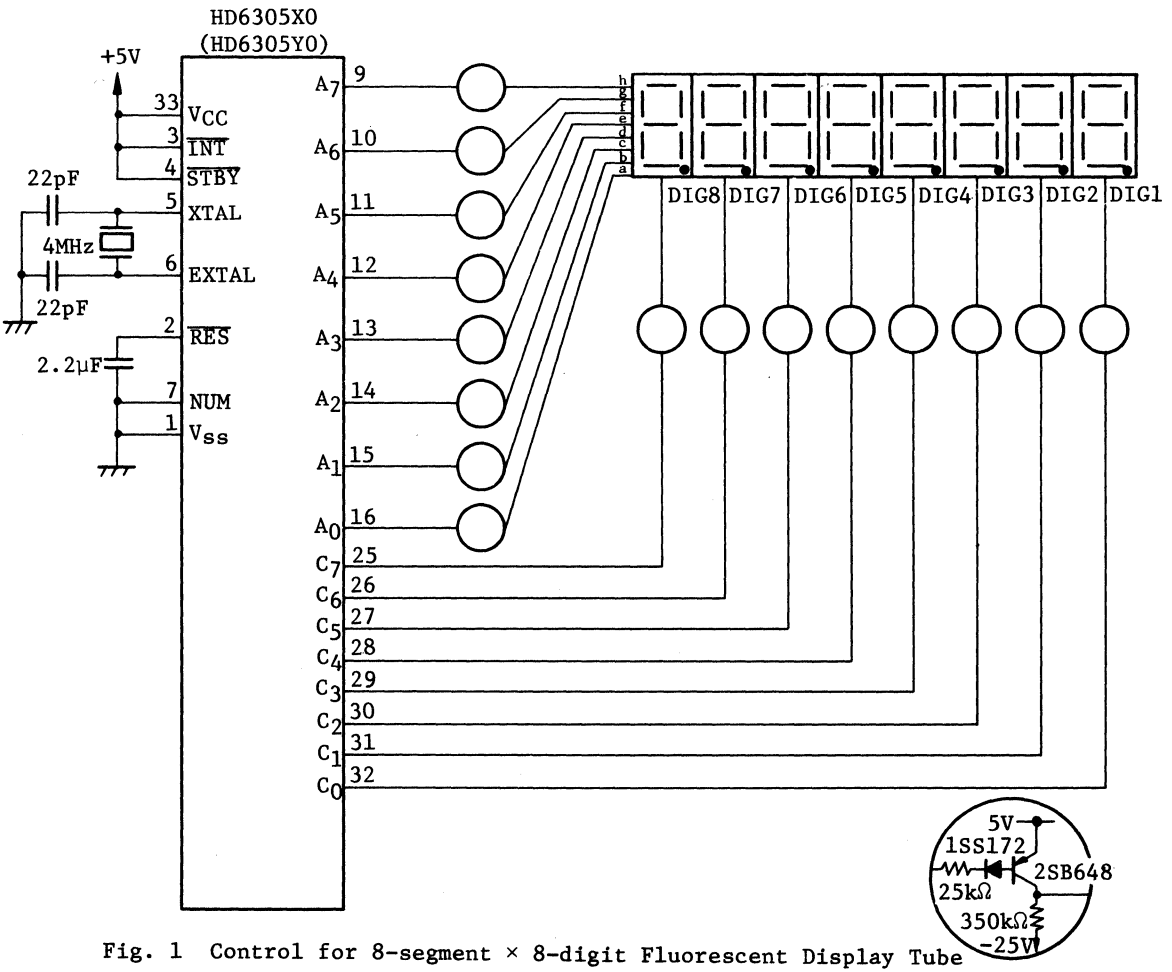
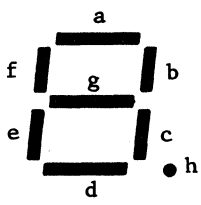


Fig. 1 Control for 8-segment × 8-digit Fluorescent Display Tube

(4) Pin Functions

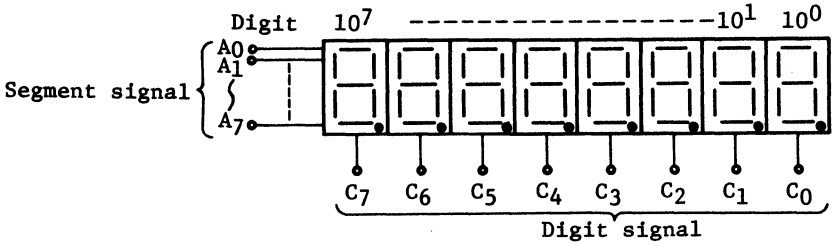
Pin functions at the interface between the HD6305X0 and fluorescent display tube are shown in Table 1.

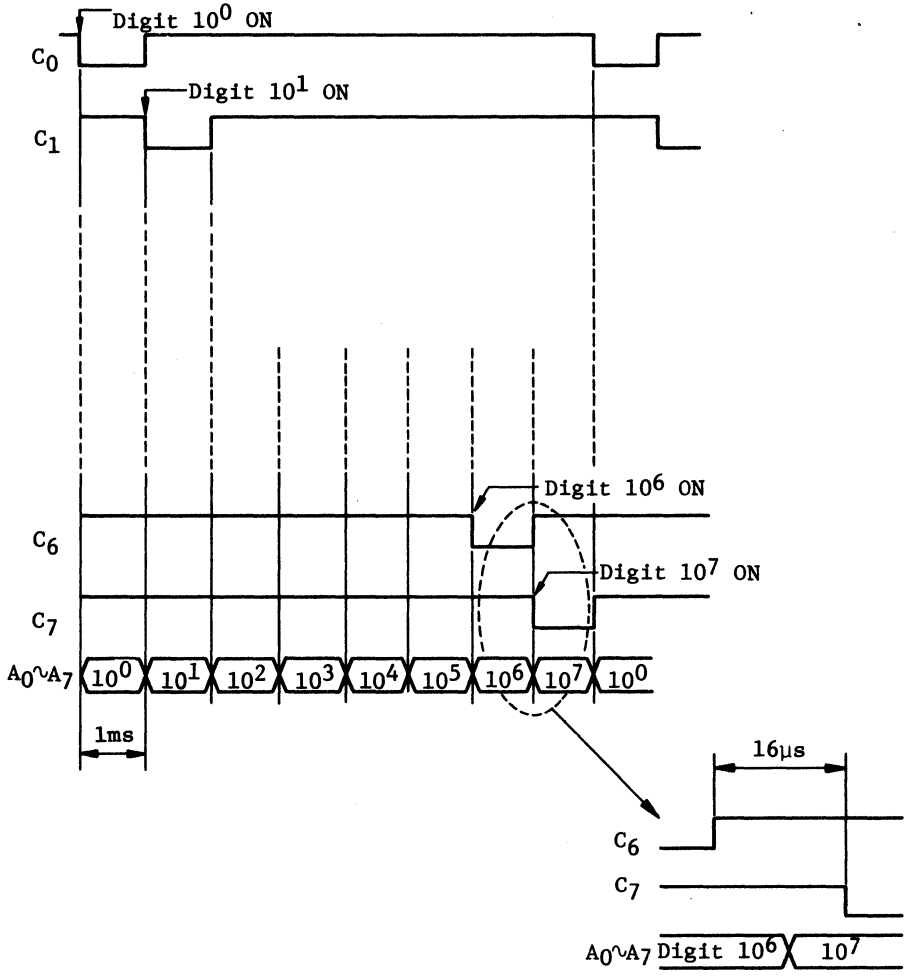
Table 1 Pin Functions

Pin Name (HD6305X0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Fluorescent display)	Program Label
C_0	Output	Low	Outputs digit signal of 8-segment × 8-digit fluorescent display tube.	DIG1	PCDTR
C_1	Output	Low		DIG2	
C_2	Output	Low		DIG3	
C_3	Output	Low		DIG4	
C_4	Output	Low		DIG5	
C_5	Output	Low		DIG6	
C_6	Output	Low		DIG7	
C_7	Output	Low		DIG8	
A_0	Output	Low	Outputs segment signal of 8-segment × 8-digit fluorescent display tube pins "a" to "h" are connected to segments as below.  Segment	a	PADTR
A_1	Output	Low		b	
A_2	Output	Low		c	
A_3	Output	Low		d	
A_4	Output	Low		e	
A_5	Output	Low		f	
A_6	Output	Low		g	
A_7	Output	Low		h	

(5) Hardware Operation

Fluorescent display tube timing at frame frequency 122Hz, duty 1/8 is shown in Fig. 2.





8.2 SOFTWARE DESCRIPTION

(1) Program Module Configuration

Program module configuration for display on 8-segment×8-digit fluorescent display tube is shown in Fig. 3.

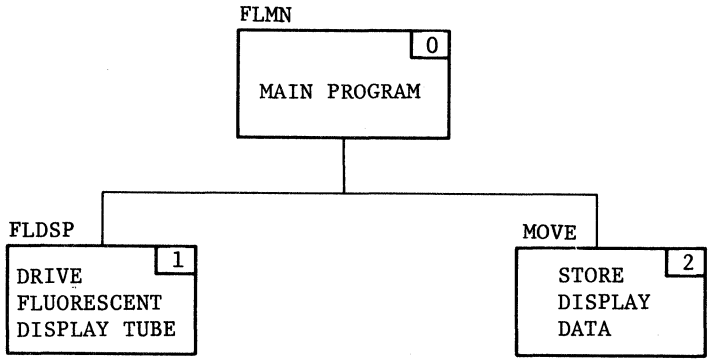


Fig. 3 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	FLMN	Demonstrates display on 8-segment×8-digit fluorescent display tube.
1	DRIVE FLUORESCENT DISPLAY TUBE	FLDSP	Drives 8-segment×8-digit fluorescent display tube for display.
2	STORE DISPLAY DATA	MOVE	Stores display data in display RAM. See subroutine MOVE in HD6305 FAMILY APPLICATION NOTES (SOFTWARE) for details.

(3) Program Module Sample Application (Main Program)

The flowchart in Fig. 4 is an example of a fluorescent display using the program module in Fig. 3. The program in Fig. 4 demonstrates the fluorescent display shown in Fig. 5.

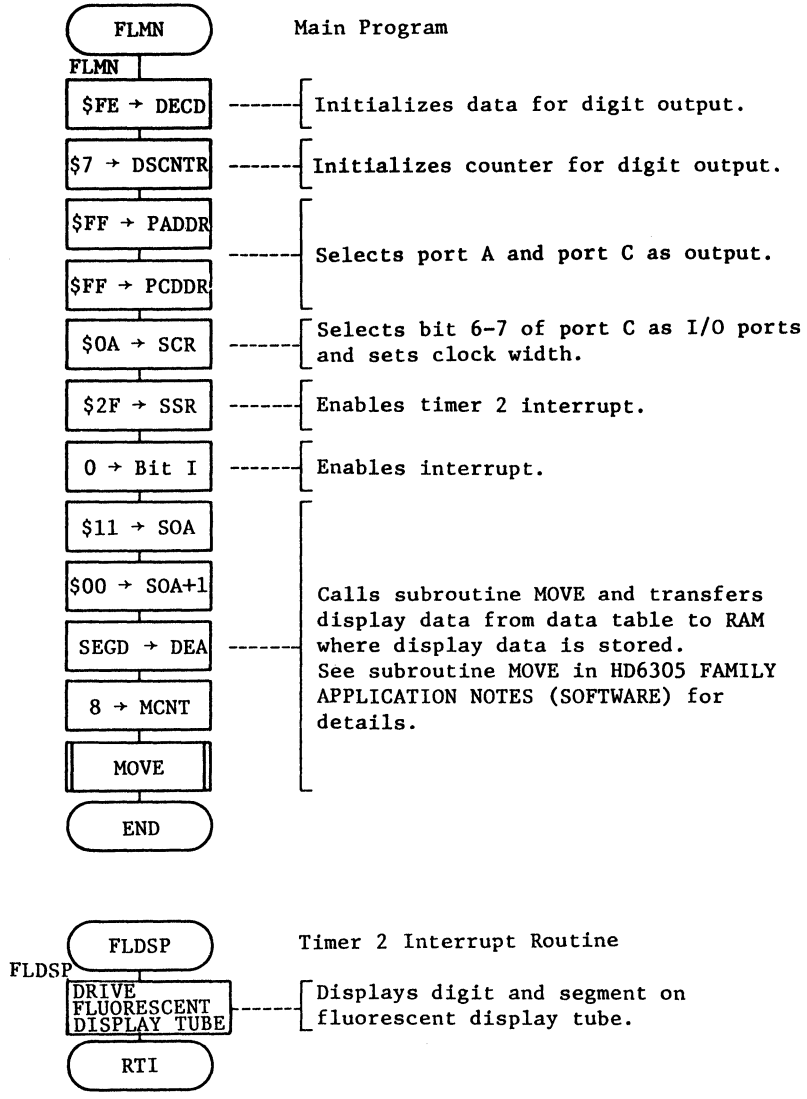


Fig. 4 Program Module Flowchart

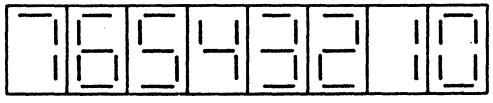


Fig. 5 Fluorescent Display Example

PROGRAM MODULE NAME	DRIVE FLUORESCENT DISPLAY TUBE	MCU/MPU	HD6305X0/Y0	LABEL	FLDSP

FUNCTION

Displays on 8-segment × 8-digit fluorescent display tube.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Display data	SEGD (RAM)	8
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	×

C	Z
×	×
N	I
×	●
H	
●	

SPECIFICATIONS

ROM (Bytes)
35
RAM (Bytes)
10
Stack (Bytes)
0
No. of cycles
43
Reentrant
No
Relocation
No
Interrupt
No

DESCRIPTION

(1) Function Details

(a) Argument details
SEGD(RAM) : Holds display data for each digit.

(b) Fig-5 shows an example of program module FLDSP execution. If entry argument is held as shown in part ① of Fig. 6, fluorescent displays as shown in part ② of Fig. 6.

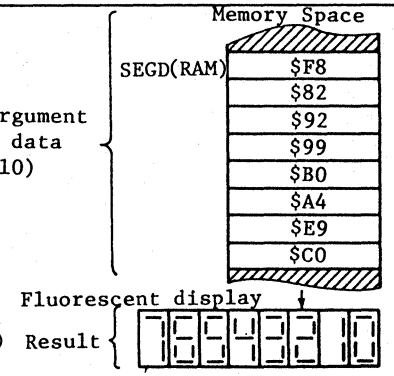


Fig. 6 Example of FLDSP Execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles required to display 1 digit.

DESCRIPTION

(c) Table 3 shows relation between display data and actual display.

Table 3 Display Data and Corresponding Display

Display Data	Display	Display Data	Display
$C0	0	$92	5
$E9	1	$82	6
$A4	2	$F8	7
$B0	3	$80	8
$99	4	$90	9

(d) Program module FLDSP calls neither program modules nor subroutines.

(2) User Notes

(a) Reserves data shown in Table 3 as data table.

(b) Selects DDRs of port A and port C as output.

(c) Initializes timer 2.

(d) Clears bit I to enable timer 2 interrupt.

(3) RAM Description

Label	RAM	Description
	b7 b0	
SEGD	10^7 digit 10^6 digit 10^5 digit 10^4 digit 10^3 digit 10^2 digit 10^1 digit 10^0 digit	Stores 8-digit display data shown in Table 3.
DSCNTR		Stores counter indicating digit to be displayed.
DECD		Stores data indicating digit turned on.

7

DESCRIPTION

(4) Sample Application

Program module FLDSP is executed every 1ms after clearing bit I, initializing digit output data, counter, port and timer 2. Fluorescent displays when display data is stored in display RAM.

```
        :
        :
        :
LDA     #$FE    }------ Initializes digit output data.
STA     DECD    }
LDA     #$07    }------ Initializes digit output counter.
STA     DSCNTR  }
LDA     #$FF    }
STA     PADDR   }------ Selects ports A and C as output.
STA     PCDDR   }
LDA     #$0A    }------ Sets interval of timer 2 interrupt to 1ms.
STA     SCR     }
LDA     #$2F    }------ Enables timer 2 interrupt.
STA     SSR     }
CLI             ------ Enables interrupt.
LDA     #$11    }
STA     SOA     }
LDA     #$00    }
STA     SOA+1   }------ Calls subroutine MOVE and transfers display
LDA     #SEGD   }       data from data table to SEGD(RAM) where display
STA     DEA     }       data is stored.
LDA     #$08    }
STA     MCNT    }
JSR     MOVE    }
        :
        :
        :
ORG     $1100
FCB     $F8, $82, $92, $99, $B0, $A4, $E9, $C0 -------- Display data
```

(5) Basic Operation

(a) Previous digit display is turned off and next digit is displayed every timer interrupt.

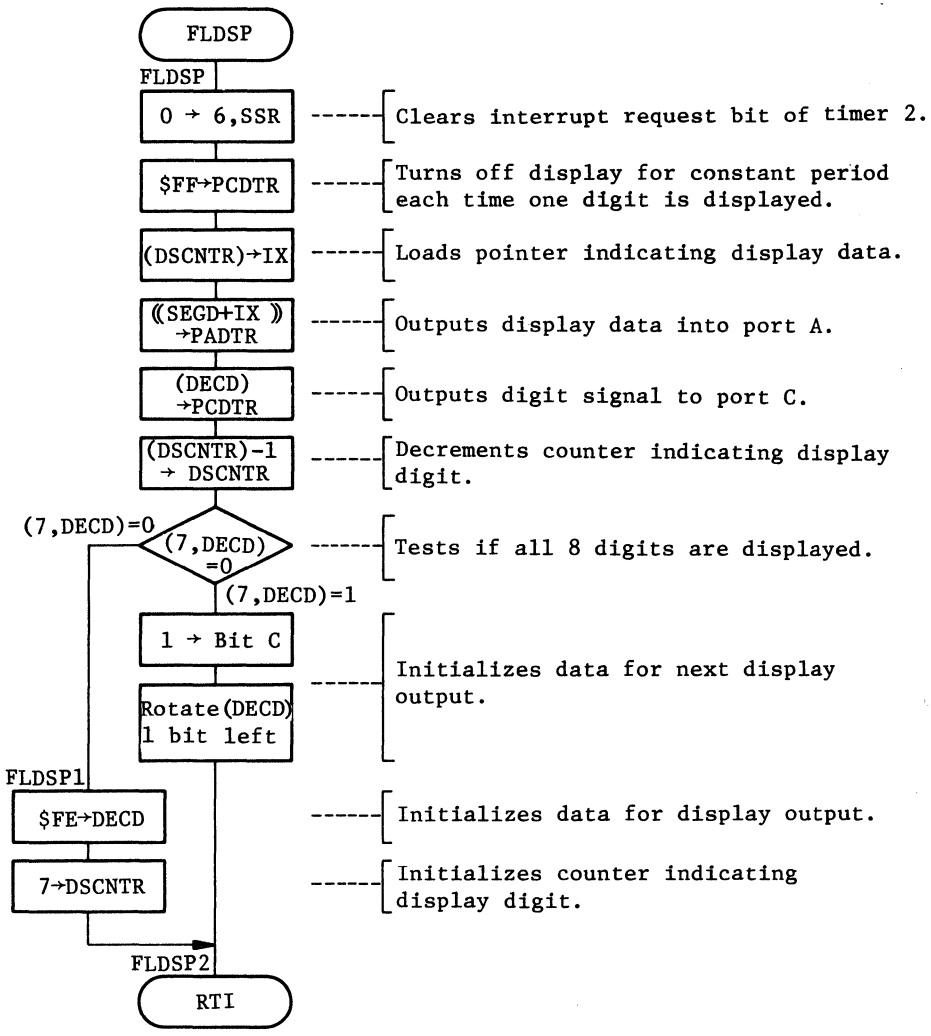
(b) DSCNTR(RAM) is used as counter for display digits and pointer for display data and digit data.

(c) Outputs display data and digit data to port using index addressing mode.

(d) Decrements counter for display digits and pointer for digit data.

PROGRAM MODULE NAME	DRIVE FLUORESCENT DISPLAY TUBE	MCU/MPU	HD6305X0/Y0	LABEL	FLDSP

FLOWCHART



7

8.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

8.5 PROGRAM LISTING

```
00001                   *
00002                   ****    RAM ALLOCATION    **************************
00003                   *
00004 0080                      ORG     $80
00005                   *
00006 0080 0008         SEGD    RMB     8         Display data
00007 0088 0001         DSCNTR  RMB     1         Digit counter
00008 0089 0001         DECD    RMB     1         Digit data
00009 008A 0002         SOA     RMB     2         Source address
00010 008C 0001         DEA     RMB     1         Destination address
00011 008D 0001         MCNT    RMB     1         Transfer counter
00012 008E 0004         MSUB    RMB     4         Work area for subroutine
00013 0092 0001         SPNT    RMB     1         Relative data of source address
00014                   *
00015                   ****    SYMBOL DEFINITIONS    ************
00016                   *
00017       0000        PADTR   EQU     $00       Port A data register
00018       0002        PCDTR   EQU     $02       Port C data register
00019       0004        PADDR   EQU     $04       Port A data direction register
00020       0006        PCDDR   EQU     $06       Port C data direction register
00021       0010        SCR     EQU     $10       SCI control register
00022       0011        SSR     EQU     $11       SCI status register
00023                   ************************************************
00024                   *                                              *
00025                   *            MAIN PROGRAM : FLMN               *
00026                   *                                              *
00027                   ************************************************
00028                   *
00029 1000                      ORG     $1000
00030                   *
00031 1000 A6 FE        FLMN    LDA     #$FE      Initialize digit data
00032 1002 B7 89                STA     DECD
00033 1004 A6 07                LDA     #7        Initialize digit counter
00034 1006 B7 88                STA     DSCNTR
00035 1008 A6 FF                LDA     #$FF
00036 100A B7 04                STA     PADDR     Select port A as output
00037 100C B7 06                STA     PCDDR     Select port C as output
00038 100E A6 0A                LDA     #$0A      Initialize SCR
00039 1010 B7 10                STA     SCR
00040 1012 A6 2F                LDA     #$2F      Initialize SSR
00041 1014 B7 11                STA     SSR
00042 1016 9A                   CLI               Enable interrupts
00043 1017 A6 11                LDA     #$11      Stores source start address
00044 1019 B7 8A                STA     SOA       into entry argument
00045 101B A6 00                LDA     #$00
00046 101D B7 8B                STA     SOA+1
00047 101F A6 80                LDA     #SEGD     Stores destinatio start address
00048 1021 B7 8C                STA     DEA       into entry argument
00049 1023 A6 08                LDA     #8        Stores length of byte to be
00050 1025 B7 8D                STA     MCNT      moved into entry argument
00051 1027 AD 25                BSR     MOVE      Move Segment data
00052 1029 20 FE        PEND    BRA     PEND      End of main program
```

```
00054                    ********************************************
00055                    *                                          *
00056                    * NAME : FLDSP (DRIVE FLOURSCENT DISPLAY   *
00057                    *               TUBE)                      *
00058                    *                                          *
00059                    ********************************************
00060                    *                                          *
00061                    *      ENTRY : DEGD (DISPLAY DATA)         *
00062                    *    RETURNS : NOTHING                     *
00063                    *                                          *
00064                    ********************************************
00065 102B 1D 11        FLDSP  BCLR   6,SSR     Clear interrupt request flag
00066 102D A6 FF               LDA    #$FF      Turn off display
00067 102F B7 02               STA    PCDTR
00068 1031 BE 88               LDX    DSCNTR    Load digit counter
00069 1033 E6 80               LDA    SEGD,X    Output display data
00070 1035 B7 00               STA    PADTR
00071 1037 B6 89               LDA    DECD      Output digit data
00072 1039 B7 02               STA    PCDTR
00073 103B 3A 88               DEC    DSCNTR    Decrement digit counter
00074 103D 0F 89 05            BRCLR  7,DECD,FLDSP1  Display 8-digit ?
00075 1040 99                  SEC              Store next digit data
00076 1041 39 89               ROL    DECD
00077 1043 20 08               BRA    FLDSP2
00078 1045 A6 FE        FLDSP1 LDA    #$FE      Initialize digit data
00079 1047 B7 89               STA    DECD
00080 1049 A6 07               LDA    #7        Initialize digit counter
00081 104B B7 88               STA    DSCNTR
00082 104D 80           FLDSP2 RTI
00083                    ************************************************
00084                    *                                              *
00085                    *      NAME : MOVE (MOVING MEMORY BLOCKS)      *
00086                    *                                              *
00087                    ************************************************
00088                    *                                              *
00089                    *      ENTRY : SOA  (SOURCE ADDR)              *
00090                    *              DEA  (DESTINATION ADDR)         *
00091                    *              MCNT (TRNSFER COUNTER)          *
00092                    *    RETURNS : NOTHING                         *
00093                    *                                              *
00094                    ************************************************
00095 104E A6 D6        MOVE   LDA    #$D6
00096 1050 B7 8E               STA    MSUB      Store instruction code (LDA
00097 1052 B6 8A               LDA    SOA                              Disp,X)
00098 1054 B7 8F               STA    MSUB+1    Store source ADDR (H)
00099 1056 B6 8B               LDA    SOA+1
00100 1058 B7 90               STA    MSUB+2    Store source ADDR (L)
00101 105A A6 81               LDA    #$81
00102 105C B7 91               STA    MSUB+3    Store instruction code (RTS)
00103 105E 3F 92               CLR    SPNT  Clear relative data of source ADDR
00104 1060 BE 92        MOVE1  LDX    SPNT  Load relative data of source  ADDR
00105 1062 BD 8E               JSR    MSUB      Load transfer data
00106 1064 BE 8C               LDX    DEA       Destination ADDR
00107 1066 F7                  STA    0,X       Store transfer data
00108 1067 3C 8C               INC    DEA       Increment destination ADDR
```

```
00109 1069 3C 92          INC     SPNT      Increment relative data of
                                            source ADDR
00110 106B 3A 8D          DEC     MCNT      Decrement transfer counter
00111 106D 26 F1          BNE     MOVE1     Branch until transfer counter=0
00112 106F 81             RTS
00113                *************************************************
00114                *                                               *
00115                *               DATA  TABLE                     *
00116                *                                               *
00117                *************************************************
00118                *
00119 1100                ORG     $1100
00120                *
00121 1100 F8             FCB     $F8,$82,$92,$99,$B0,$A4,$E9,$C0
00122                *************************************************
00123                *                                               *
00124                *               VECTOR ADDRESSES                *
00125                *                                               *
00126                *************************************************
00127                *
00128 1FF6                ORG     $1FF6
00129                *
00130 1FF6 102B           FDB     FLDSP     SCI/TIMER2
00131 1FF8 1000           FDB     FLMN      TIMER/INT2
00132 1FFA 1000           FDB     FLMN      INT
00133 1FFC 1000           FDB     FLMN      SWI
00134 1FFE 1000           FDB     FLMN      RES
00135                *
00136                     END
```

9. STEPPING MOTOR CONTROL

9.1 HARDWARE DESCRIPTION

(1) Function

(a) Drives stepping motor using the HD6305X0.

(b) Uses 4-phase 2 exciting stepping motor.

(c) 1 to 255 steps can be selected.

(d) Controls slue pulse rate (slow-up, operating and slow-down) when 12 steps or more are selected.

(e) Selects clockwise (normal) slue and counterclockwise (reverse) slue with stepping motor. Operates backlash when reverse slue is selected.

(2) Microcomputer Applications

(a) Executes interrupt routine using 8-bit timer with 7-bit prescaler contained in the HD6305X0 (hereinafter, timer).

(b) Drives stepping motor by outputting pulses from port B by interrupt routine.

(c) Generates slow-up and slow-down pulse rate by changing TDR value.

(3) Circuit Diagram

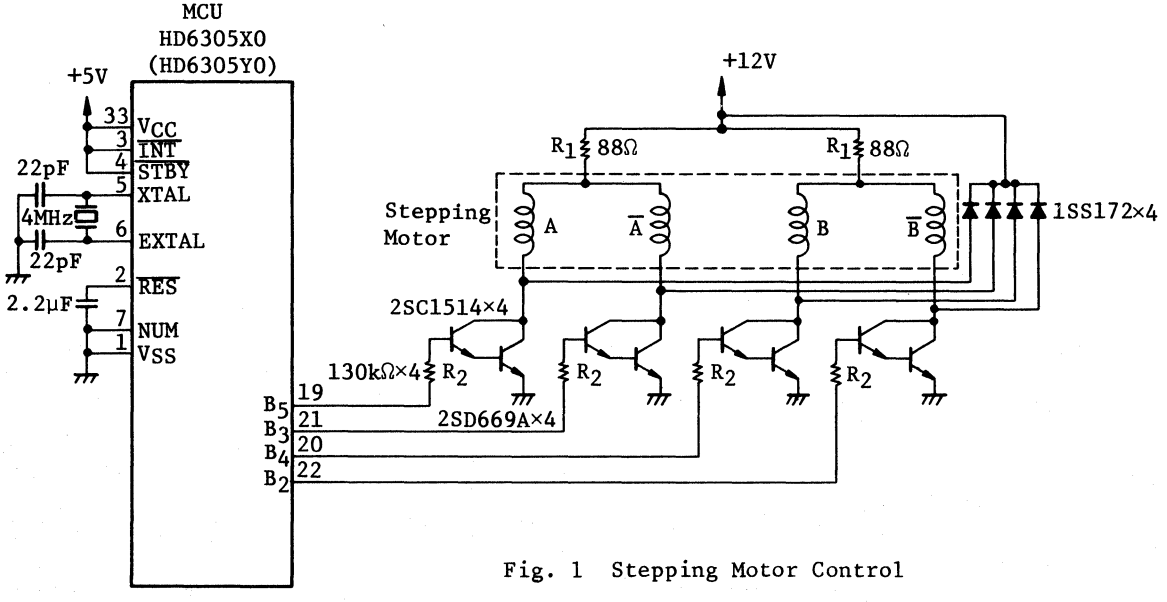


Fig. 1 Stepping Motor Control

(4) Pin Functions

Pin functions at the interface between the HD6305X0 and stepping motor are shown in Table 1.

Table 1 Pin Functions

Pin Name (HD6305X0)	Input/ Output	Active level (High or Low)	Function	Pin Name (Motor)	Program Label
B_2	Output	—	Connects stepping motor.	$\overline{B}$	PBDTR
B_3	Output	—		$\overline{A}$	
B_4	Output	—		B	
B_5	Output	—		A	

(5) Hardware Operation

The stepping motor supplies pulses at the HD6305X0 I/O port as shown in Fig. 2.

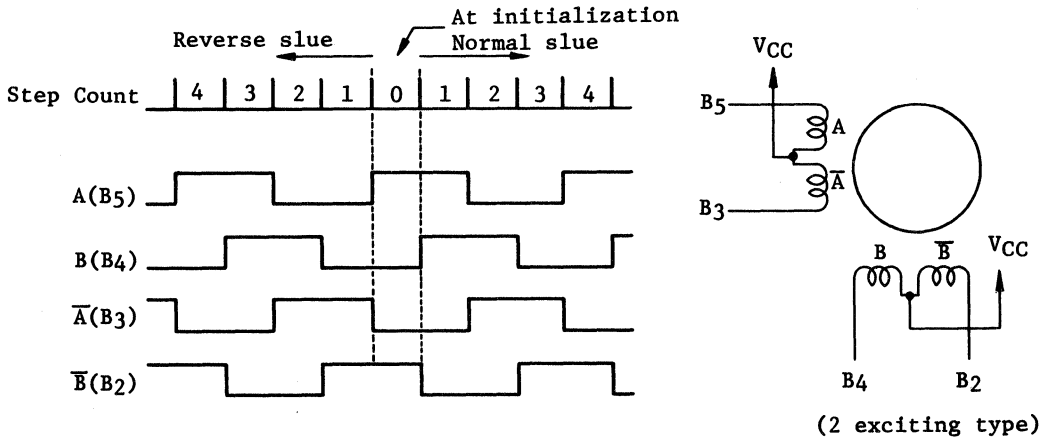


Fig. 2 Outline of Stepping Motor Operation

Slow-up and slow-down pulse rate are supplied to the stepping motor every four steps as shown in Fig. 3.

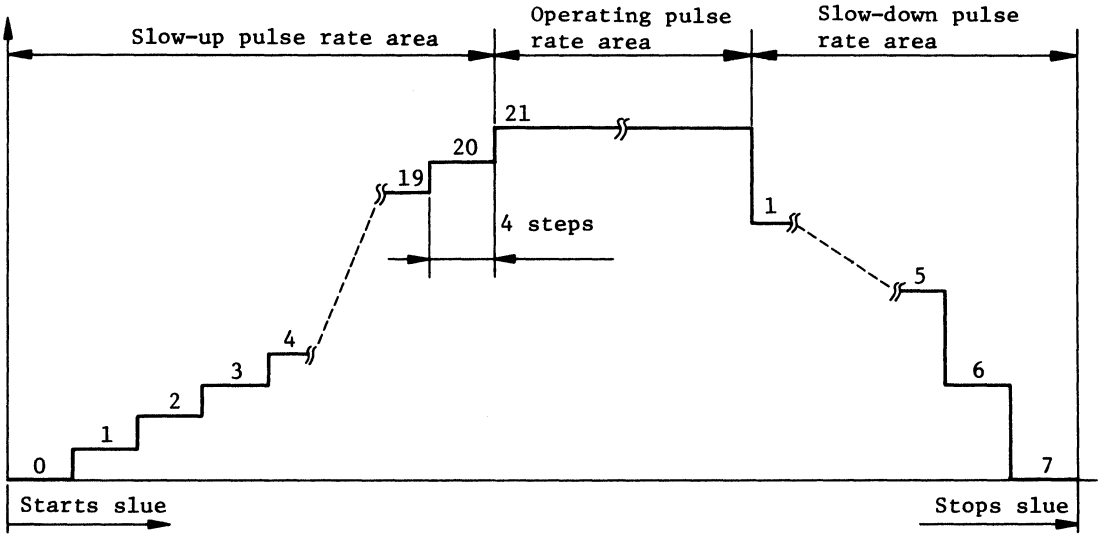


Fig. 3 Outputs to the Stepping Motor

(a) The pulse rate changes every 4 steps for slow-up and slow-down.

(b) In case of slow-up area to the operating pulse area, the pulse rate changes in 21 steps to gradually increase the slue speed.

(c) In case of slow-down area to stop, the pulse rate changes in 7 steps to gradually reduce the slue speed.

(d) When reverse slue is selected, an additional one-step rotation, followed by a normal one-step rotation, is executed.

7

9.2 SOFTWARE DESCRIPTION

(1) Program Module Configuration

The program module configuration for stepping motor control is shown in Fig. 4.

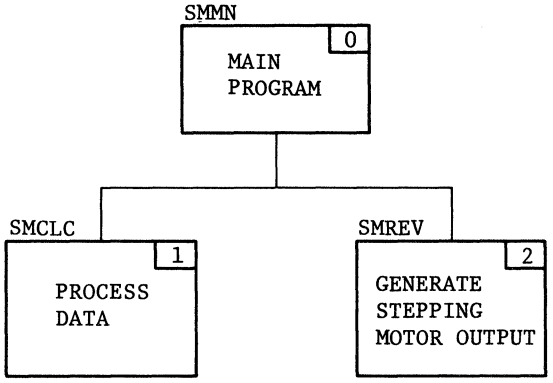


Fig. 4 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	SMMN	Rotates stepping motor.
1	PROCESS DATA	SMCLC	Calculates output data for slow-up, operating, slow-down and backlash by supplying total step count.
2	GENERATE STEPPING MOTOR OUTPUT	SMREV	Supplies pulses to the stepping motor.

(3) Program Module Sample Application (Main program)

The flowchart in Fig. 5 is an example of the stepping motor rotation performed by the program module in Fig. 4. When the program module of Fig. 5 is executed, the stepping motor makes 201 reverse slue, then one normal slue.

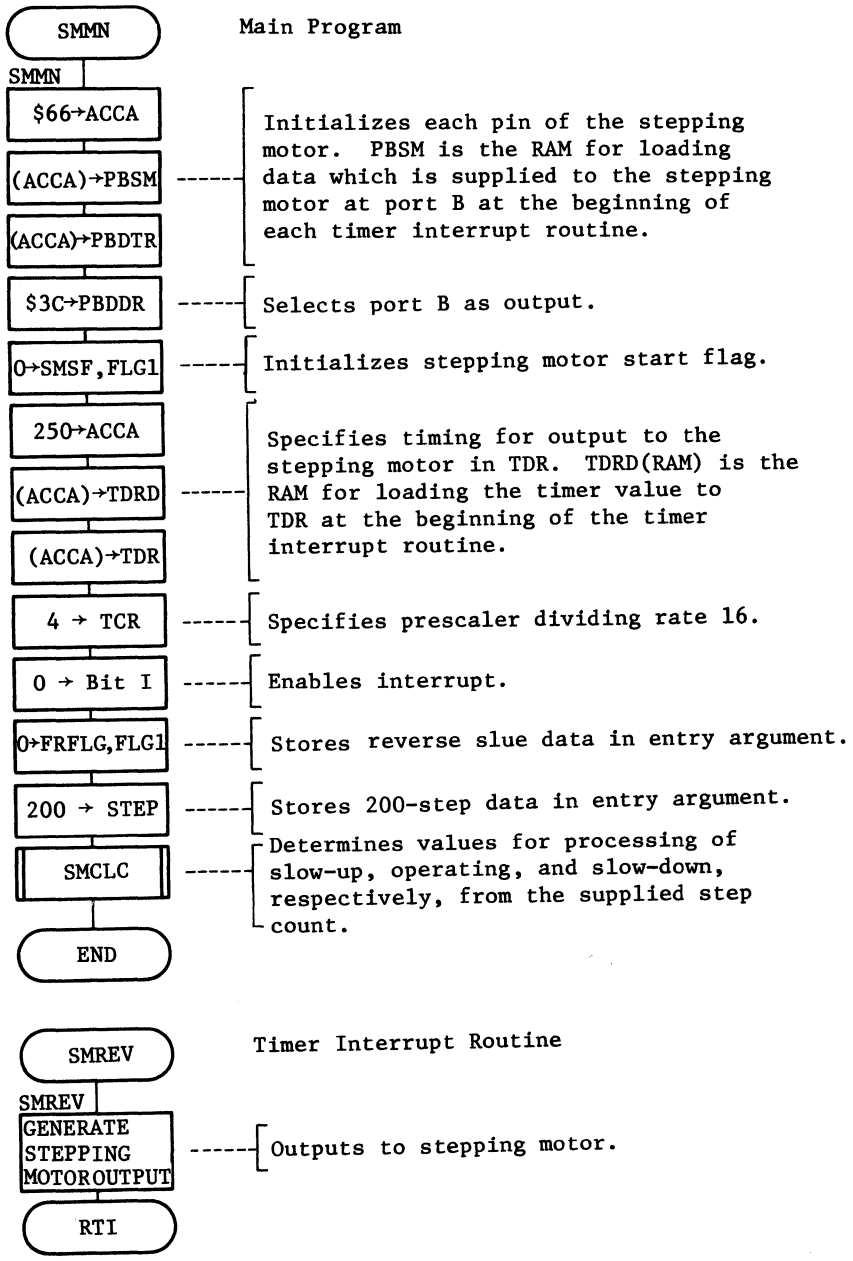


Fig. 5 Program Module Flowchart

PROGRAM MODULE NAME	PROCESS DATA	MCU/MPU	HD6305X0/Y0	LABEL	SMCLC

FUNCTION

Generates data for outputting to the stepping motor by the timer routine. Sets data for slow-up, operating and slow-down after determining the backlash requirement.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Normal/Reverse slue flag	FRFLG, FLG1	1
		Total Step Count	STEP	1
	Returns	Slow-up data	SUP	1
		Operating data	SHOLD	1
		Slow-down data	SDWN	1
		Remainder of step	STEPE	1
		Normal/Reverse slue flag	FRFLG, FLG1	1
		Slue start flag	SMSF, FLG1	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	×

C	Z
×	×

N	I
×	●

H
×

SPECIFICATIONS

ROM (Bytes)
140
RAM (Bytes)
9
Stack (Bytes)
2
No. of cycles
333
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

FLG1(RAM) : Holds flag indicating slue direction and start of stepping motor rotation. Flag functions are shown in Table 3.
STEP(RAM) : Holds total step count.
SUP(RAM) : Contains slow-up data.
SHOLD(RAM) : Contains operating data.
SDWN(RAM) : Contains slow-down data.
STEPE(RAM) : Contains remainder of total step count divided by 4.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles required to execute data in the sample application.

Table 3 Flag Functions

Label	Bit/Label		Function
	SMSF	FRFLG	
FLG1	-	1	Rotates stepping motor clockwise (normal).
	-	0	Rotates stepping motor counterclockwise (reverse).
	0	-	Stops slue.
	1	-	Starts slue.

(b) Fig. 6 shows an example of program module SMCLC execution.
If entry arguments are held as shown in part ① of Fig. 6, the remainders of slow-up, operating, and slow-down are contained as shown in part ② of Fig. 6.

(c) Program module SMCLC calls other program modules and subroutines shown in Table 4.

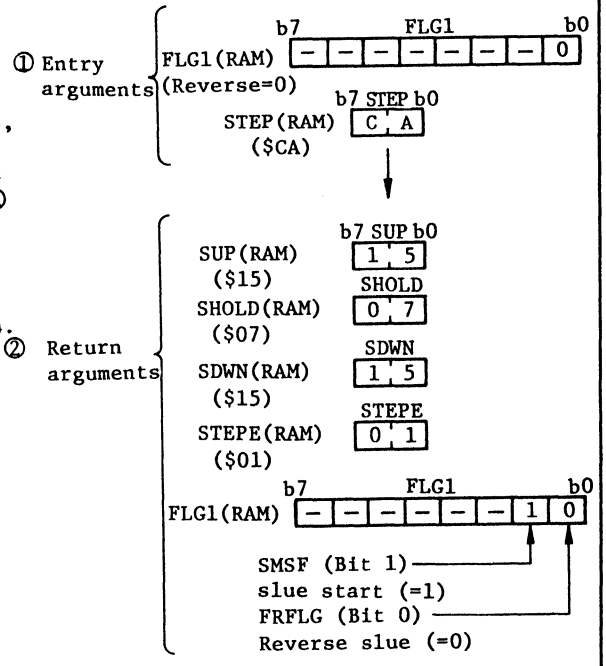


Fig. 6 Example of SMCLC Execution

Table 4 Program Modules and Subroutines Called in SMCLC

Program Module/ Subroutine Name	Label	Function
CALCULATE NORMAL/REVERSE DATA	SMFR	Tests whether the slue is normal or reverse and sets the next data to the stepping motor.

(2) User Notes

For reverse rotation, the maximum step count is $FE. If $FF is loaded, correct rotation data can not be obtained.

(3) RAM Description

Label	RAM b7 b0	Description
STEP		Stores total step count.
SUP		Stores one-fourth of the step count due to slow-up.
SHOLD		Stores one-fourth of the step count due to operating.
SDWN		Stores one-fourth of the step count due to slow-down.
STEPE		Stores remainder of a total step divided by 4.
SCNTR		Stores 4-step counter.
SDWNW		Stores SDWN work area.
STEPW		Stores STEP work area.
FLG1		Stores flag indicating normal/reverse slue and start/stop.

(4) Sample Application

Calls program module SMCLC after selecting normal/reverse rotation flag and total step count.

```
WORK1   RMB     1           ----- Reserves memory byte for total step
                                  count.
        BSET    FRFLG, FLG1 ----- Sets normal/reverse flag to normal
                                  rotation.
        LDA     WORK1       }---- Loads the total step count into entry
        STA     STEP        }     argument.
        JSR     SMCLC       ----- Calls program module SMCLC and
                                  processes the data.
```

(5) Basic Operation

(a) Tests whether slue is normal or reverse. If it is reverse, "1" is added to the total step count for backlush processing.

(b) Slow-up and slow-down operations are executed every 4 steps. The total step count is multiplied by one-fourth.
If the result is 2 or less, slow-up and slow-down are not executed.

(c) If one-fourth the step count is 3 or more, values of SUP(RAM), SHOLD (RAM), and SDWN(RAM) are determined for slow-up and slow-down processing.

(d) Slow-up of 21 steps and slow-down of 7 steps are executed.

(e) SUP(RAM), SHOLD(RAM), and SDWN(RAM) are shown in (i) to (iii), by one-fourth the step count "n" obtained in (c) and (d).

(i) At n = 3 to 29, n = 2+4m (m = 1 to 6)

SUP(RAM) = one-fourth the total step count - (m+2)
SHOLD(RAM) = 1
SDWN(RAM) = m

(ii) At n = 3 to 29, n < 2+4m (m = 1 to 6)

SUP(RAM) = one-fourth the total step count - (m+1)
SHOLD(RAM) = 0
SDWN(RAM) = m

(iii) At $n \geq 30$

SUP(RAM) = 21
SHOLD(RAM) = one-fourth the total count - 29
SDWN(RAM) = 7

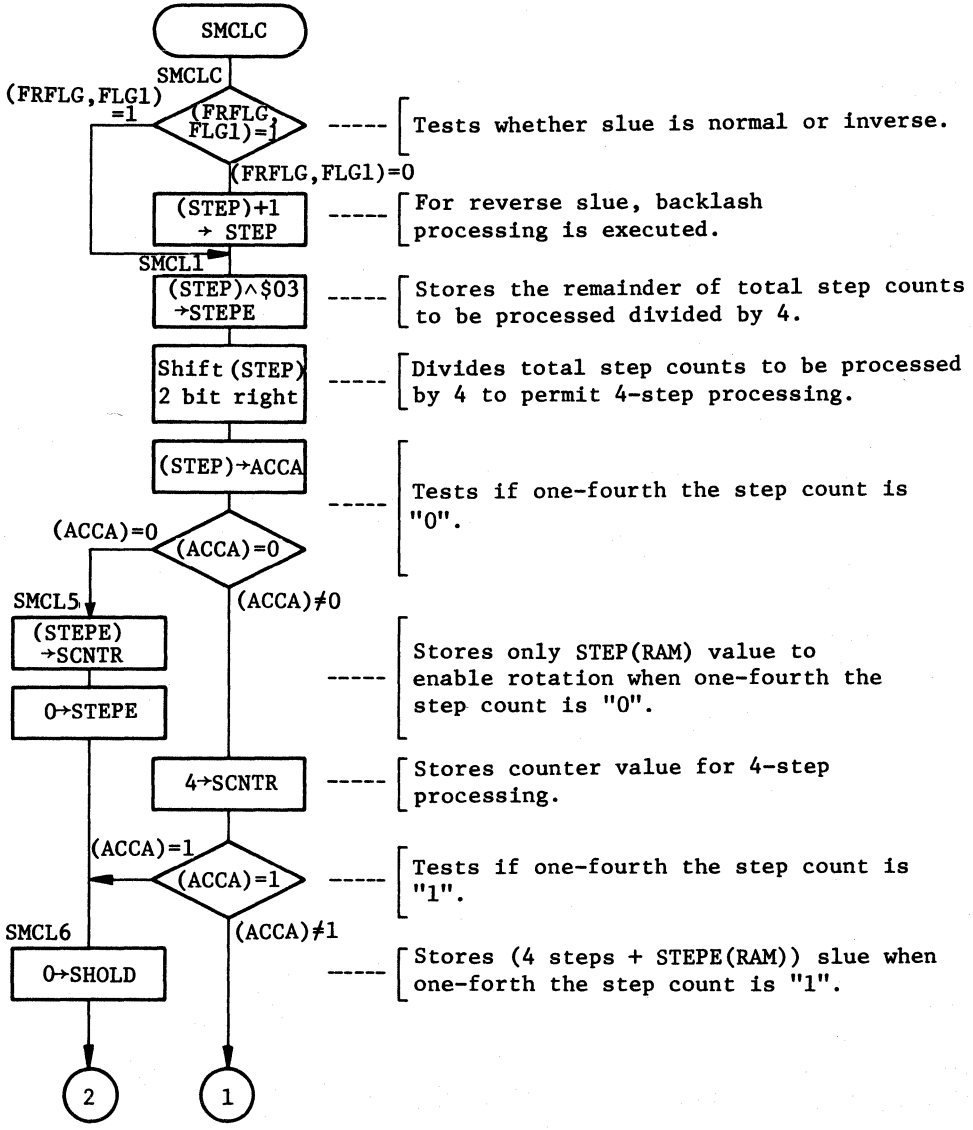
(f) If one-fourth the step count is "0", STEPE(RAM), the remainder of one-fourth the step count only is output.

(g) If one-fourth the step count is n(1 to 2), a step count of "n×4+STEPE (RAM)" is output.

7

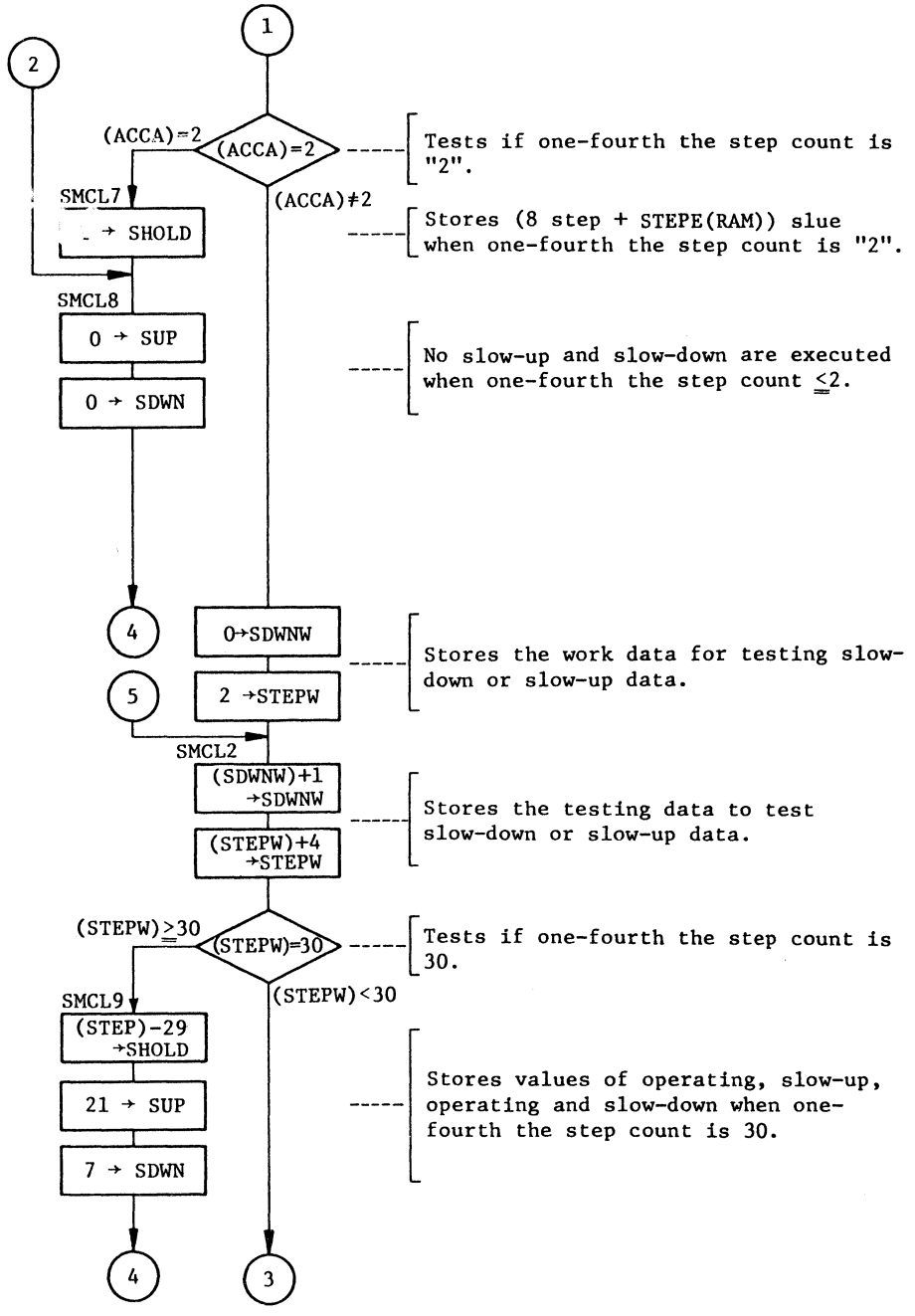
PROGRAM MODULE NAME	PROCESS DATA	MCU/MPU	HD6305X0/Y0	LABEL	SMCLC

FLOWCHART



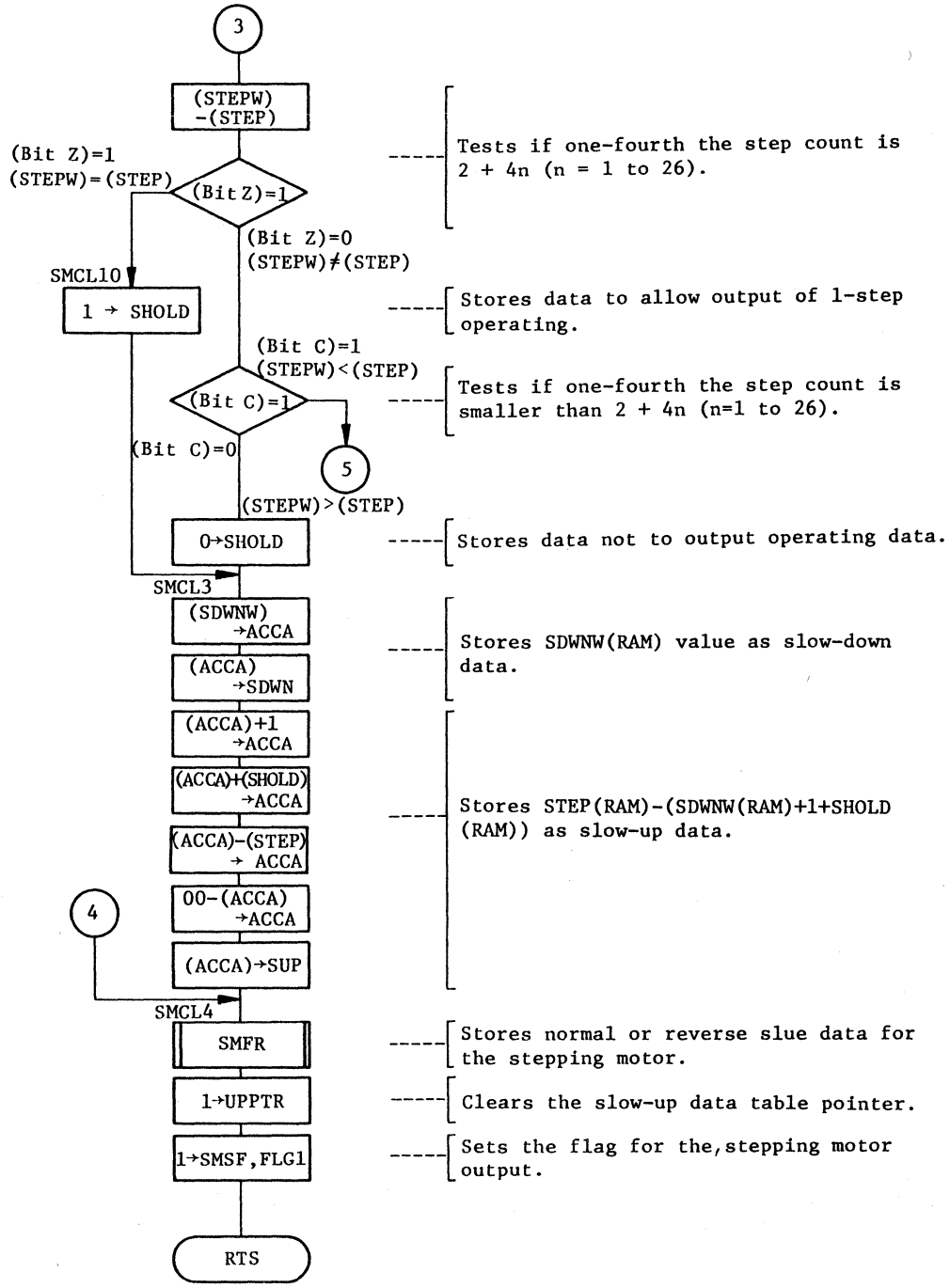
PROGRAM MODULE NAME	PROCESS DATA	MCU/MPU	HD6305X0/Y0	LABEL	SMCLC

FLOWCHART



PROGRAM MODULE NAME	PROCESS DATA	MCU/MPU	HD6305X0/Y0	LABEL	SMCLC

FLOWCHART



PROGRAM MODULE NAME	GENERATE STEPPING MOTOR OUTPUT	MCU/MPU	HD6305X0/Y0	LABEL	SMREV

FUNCTION

Outputs pulse to the stepping motor.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Slow-up data	SUP	1
		Operating data	SHOLD	1
		Slow-down data	SDWN	1
		Remainder of the step	STEPE	1
		Normal/ Reverse slue flag	FRFLG, FLG1	1
		Slue start flag	SMSF, FLG1	1
	Returns	Slue start flag	SMSF, FLG1	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	×

C	Z
×	×
N	I
×	●
H	
●	

SPECIFICATIONS

ROM (Bytes)
105
RAM (Bytes)
6
Stack (Bytes)
2
No. of cycles
76
Reentrant
No
Relocation
No
Interrupt
No

DESCRIPTION

(1) Function Details

(a) Argument details

FLG1 (RAM) : Holds flag indicating rotation direction and rotation start of the stepping motor. Flag functions are shown in Table 5.
STEP (RAM) : Holds total slue step count.
SUP (RAM) : Holds slow-up data.
SHOLD (RAM): Holds operating data.
SDWN (RAM) : Holds slow-down data.
STEPE (RAM): Holds remainder of the total step count divided by 4.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles required to execute with data in the sample application.

PROGRAM MODULE NAME	GENERATE STEPPING MOTOR OUTPUT	MCU/MPU	HD6305X0/Y0	LABEL	SMREV

DESCRIPTION

Table 5 Flag Functions

Label	Bit/Label		Function
	SMSF	FRFLG	
FLG1	-	1	Rotates stepping motor clockwise (normal).
	-	0	Rotates stepping motor counterclockwise (reverse).
	0	-	Stops slue.
	1	-	Starts slue.

(b) Fig. 7 shows an example of program module SMREV execution.
If the entry argument is held as shown in part ① of Fig. 7, outputs pulse to the stepping motor.
Then, the slue start flag SMSF(RAM) is contained as shown in part ② of Fig. 7.

(c) Program module SMREV calls other program modules and subroutines shown in Table 6.

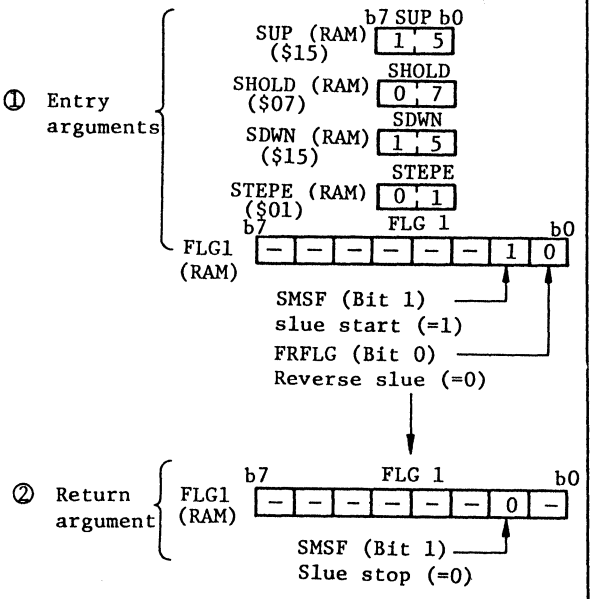


Fig. 7 Example of SMREV Execution

Table 6 Program Modules and Subroutines Called in SMREV

Program module/ Subroutine Name	Label	Function
LOAD NORMAL/ REVERSE SLUE DATA	SMFR	Tests whether slue is normal or reverse and stores next data in stepping motor.

(2) User Notes

(a) Selects bits 2-5 of port B as output.

(b) Initializes timer.

(c) Clears bit I to enable timer interrupt.

DESCRIPTION

(3) RAM Description

Label	RAM	Description
	b7 b0	
STEP		Stores total step counts.
SUP		Stores one-fourth the step count for slow-up.
SHOLD		Stores one-fourth the step count for operating.
SDWN		Stores one-fourth the step count for slow-down.
STEPE		Stores the remainder of the total step count divided by 4.
FLG1		Stores flag indicating normal/reverse slue and start/stop for the stepping motor.
PBSM		Holds data output to each pin of stepping motor.
TDRD		Holds data to be stored in TDR.

(4) Sample Application

Program module SMREV is called after selecting I/O port, initializing timer and enabling interrupt.

```
WORK1   RMB     1           ----Reserves memory byte for the total step
                                count.
        LDA     #$66
        STA     PBSM        }--- Initializes the stepping motor output pin.
        STA     PBDTR
        LDA     #$3C        }
        STA     PBDDR       } --- Selects bit 2-5 of port B as outputs.
        BCLR    SMSF, FLG1  ----Sets rotation stop.
        LDA     #250
        STA     TDRD
        STA     TDR         }----Initializes output timing to the stepping
        LDA     #4                  motor.
        STA     TCR
        CLI                 ----Enables interrupt.
        BCLR    FRFLG, FLG1
        LDA     WORK1       }--- Loads entry argument.
        STA     STEP
        JSR     SMCLC       ---- Calculates entry argument.
```

7

PROGRAM MODULE NAME	GENERATE STEPPING MOTOR OUTPUT	MCU/MPU	HD6305X0/Y0	LABEL	SMREV

DESCRIPTION

(5) Basic Operation

(a) Program module SMREV is called by the timer routine.

(b) Outputs pulses one step to stepping motor every timer interrupt.

(c) At the beginning of timer interrupt, timer timing (slow-up, operating and slow-down slue) which is held in advance and output to the stepping motor.

(d) Following (c), data to be output at the next timer interrupt is loaded.

(e) Only the timer timing is held and no output is provided to the stepping motor, when the rotation start flag SMSF(RAM) is cleared.

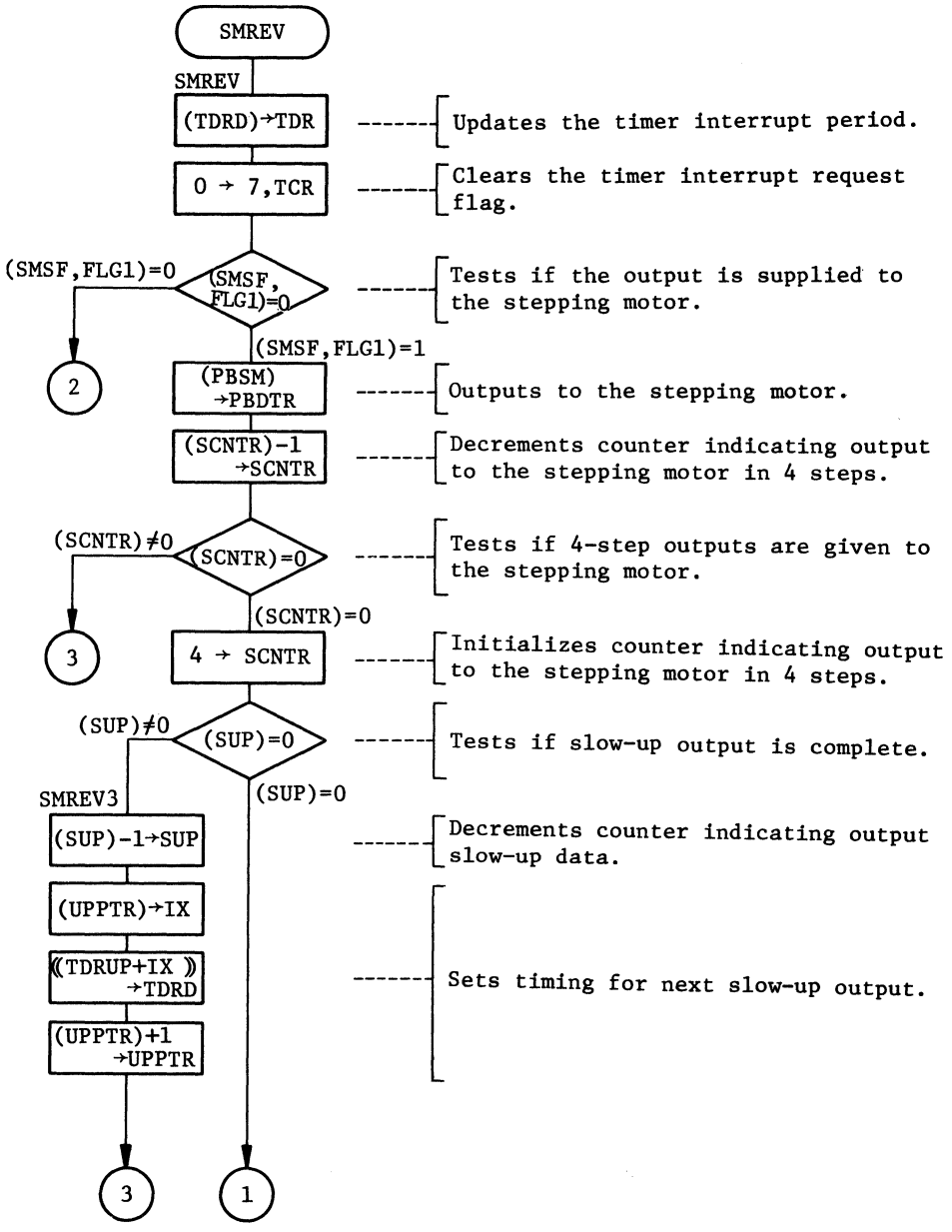
(f) Slow-up or slow-down output data is supplied to the stepping motor every 4-step.

(g) To implement (f), a test is performed to determine if the 4-step output is supplied within the same timing. Then, outputs are supplied in the sequence of slow-up, operating and slow-down.
Normal/reverse slue is tested and if it is reverse, output for backlash is provided.

(h) Timing data for slow-up and slow-down should be set in the data table in advance.

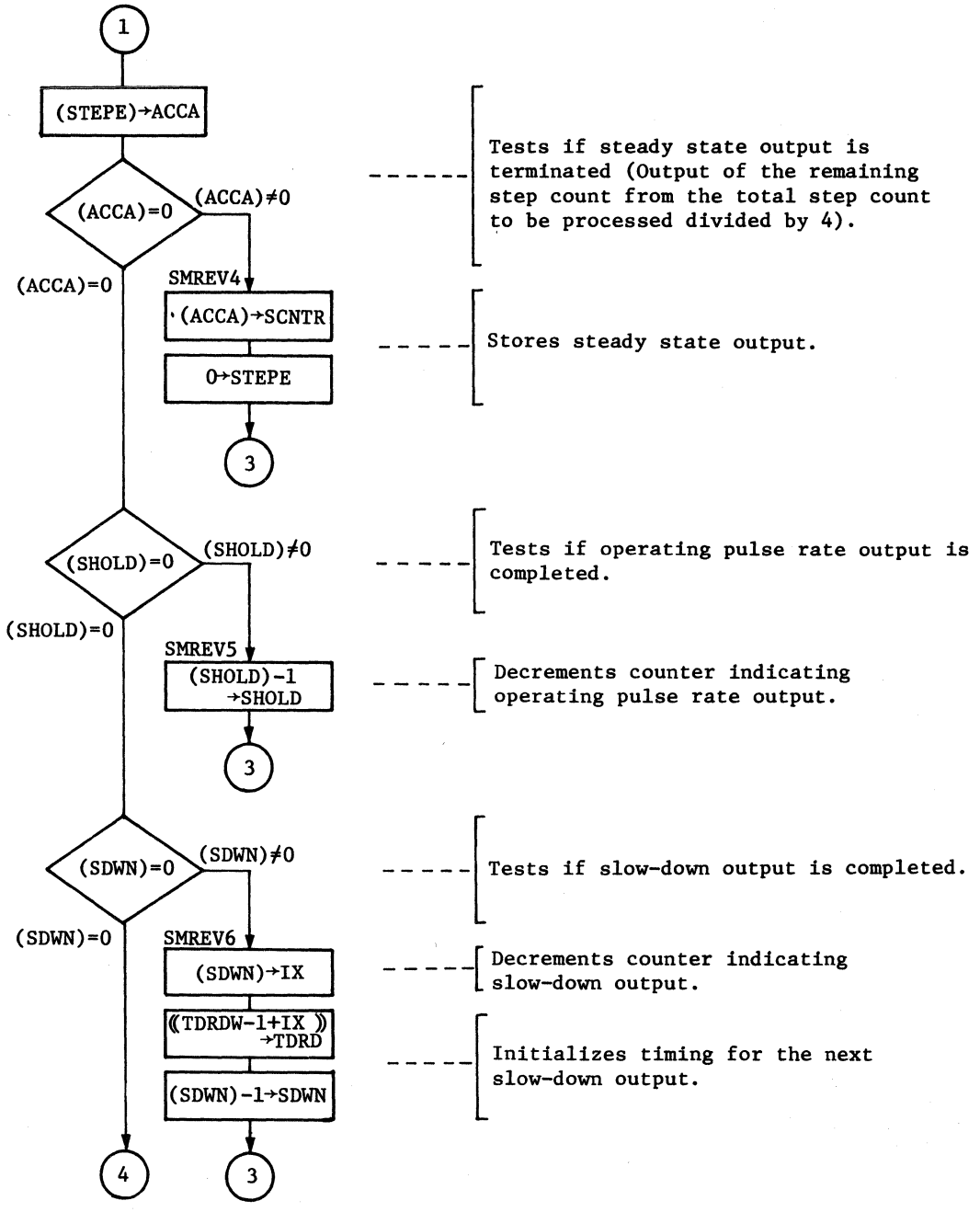
PROGRAM MODULE NAME	GENERATE STEPPING MOTOR OUTPUT	MCU/MPU	HD6305X0/Y0	LABEL	SMREV

FLOWCHART



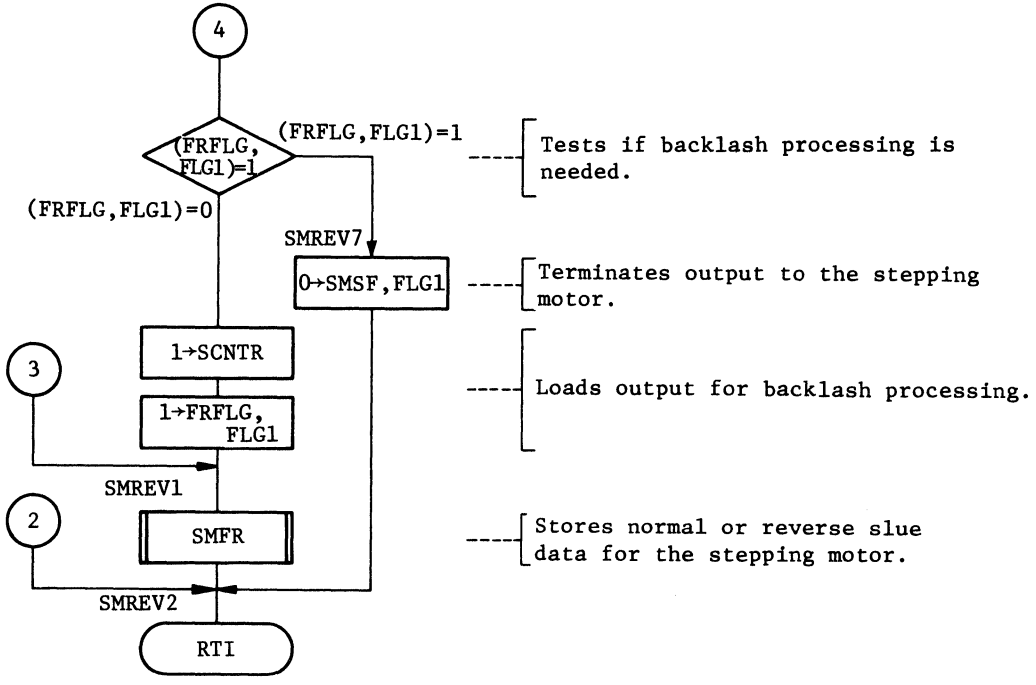
PROGRAM MODULE NAME	GENERATE STEPPING MOTOR OUTPUT	MCU/MPU	HD6305X0/Y0	LABEL	SMREV

FLOWCHART



PROGRAM MODULE NAME	GENERATE STEPPING MOTOR OUTPUT	MCU/MPU	HD6305X0/Y0	LABEL	SMREV

FLOWCHART



9.4 SUBROUTINE DESCRIPTION

SUBROUTINE NAME	CALCULATE NORMAL/REVERSE ROTATION DATA	MCU/MPU	HD6305X0/Y0	LABEL	SMFR

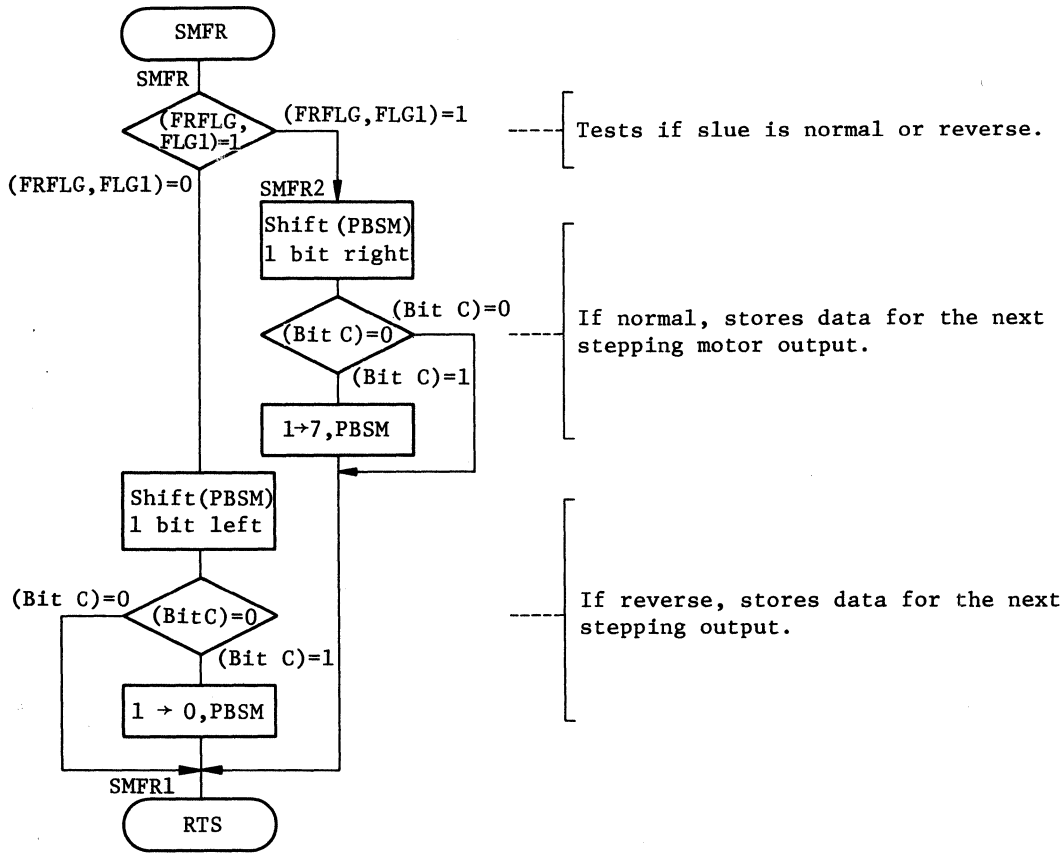
FUNCTION

Tests whether slue flag (FRFLG(RAM)) is normal or reverse, stores data for the next stepping motor output in PBSM(RAM).

BASIC OPERATION

Tests whether slue is normal or reverse, and then performs shift. As a result, output data for one-step normal/reverse slue is obtained.

FLOWCHART | **PROGRAM MODULE USING THIS SUBROUTINE**: SMCLC, SMREV



```
00001                    *
00002                    ****    RAM ALLOCATION    ********************
00003                    *
00004 0080               ORG     $80
00005                    *
00006 0080 0001  PBSM    RMB     1          Work to keep PBDTR data
00007 0081 0001  TDRD    RMB     1          Work to keep TDR data
00008 0082 0001  STEP    RMB     1          Total step count
00009 0083 0001  SUP     RMB     1          Slow-up data
00010 0084 0001  SHOLD   RMB     1          Operating data
00011 0085 0001  SDWN    RMB     1          Slow-down data
00012 0086 0001  STEPE   RMB     1          Remainder of step
00013 0087 0001  SCNTR   RMB     1          4 steps counter
00014 0088 0001  UPPTR   RMB     1          TDR up-table pointer
00015 0089 0001  STEPW   RMB     1          Work for slow-up data
00016 008A 0001  SDWNW   RMB     1          Work for slow-doun data
00017 008B 0001  FLG1    RMB     1          Flag area
00018                    *
00019                    ****    SYMBOL DEFINITIONS    ***************
00020                    *
00021       0001 PBDTR   EQU     $01        Port B data register
00022       0005 PBDDR   EQU     $05        Port B data direction register
00023       0008 TDR     EQU     $08        Timer data register
00024       0009 TCR     EQU     $09        Timer control register
00025       0000 FRFLG   EQU     0          Fore and reverse flag
00026       0001 SMSF    EQU     1          Slue start flag
00027                    ********************************************************
00028                    *                                                      *
00029                    *            MAIN PROGRAM : SMMN                       *
00030                    *                                                      *
00031                    ********************************************************
00032                    *
00033 1000               ORG     $1000
00034                    *
00035 1000 A6 66 SMMN    LDA     #$66       Initialize work for PBDTR
00036 1002 B7 80         STA     PBSM
00037 1004 B7 01         STA     PBDTR      Initialize PBDTR
00038 1006 A6 3C         LDA     #$3C       Select port B b2-b5 as output
00039 1008 B7 05         STA     PBDDR
00040 100A 13 8B         BCLR    SMSF,FLG1  Initialize SMSF
00041 100C A6 FA         LDA     #250
00042 100E B7 81         STA     TDRD       Initialize work for TDR
00043 1010 B7 08         STA     TDR        Initialize TDR
00044 1012 A6 04         LDA     #4         Initialize TCR
00045 1014 B7 09         STA     TCR
00046 1016 9A            CLI                Enable interrupts
00047 1017 11 8B         BCLR    FRFLG,FLG1 Load argument
00048 1019 A6 C8         LDA     #200       Store argument of 200 steps
00049 101B B7 82         STA     STEP
00050 101D AD 02         BSR     SMCLC      Calculate argument for SMREV
00051 101F 20 FE PEND    BRA     PEND       End of program
00052                    ************************************************
00053                    *                                              *
00054                    *       NAME : SMCLC (PROCESS DATA)            *
00055                    *                                              *
```

```
00056                    ************************************************
00057                    *                                              *
00058                    *       ENTRY :FRFLG(1:FORE SLUE,0:REVERSE SLUE)*
00059                    *             :STEP(TOTAL STEP COUNT)          *
00060                    *     RETURNS :SUP(SLOW-UP DATA)               *
00061                    *              SHOLD(OPERATING DATA)           *
00062                    *              SDWN(SLOW-DOWN DATA)            *
00063                    *              STEPE(REMAINDER OF STEP)        *
00064                    *              FRFLG(FORE/REVERSE SLUE FLAG)   *
00065                    *              SMSF(SLUE START FLAG )          *
00066                    *                                              *
00067                    ************************************************
00068 1021 00 8B 02 SMCLC  BRSET  FRFLG,FLG1,SMCL1  Fore or reverse slue
00069 1024 3C 82           INC    STEP      If reverse increment STEP
00070 1026 B6 82    SMCL1  LDA    STEP      Load lower 2 bit
00071 1028 A4 03           AND    #$03
00072 102A B7 86           STA    STEPE
00073 102C 34 82           LSR    STEP      Store 1/4 times of STEP
00074 102E 34 82           LSR    STEP
00075 1030 B6 82           LDA    STEP      Test if STEP=0 ?
00076 1032 27 3C           BEQ    SMCL5     Branch if STEP=0
00077 1034 AE 04           LDX    #4        Initialize 4 steps counter
00078 1036 BF 87           STX    SCNTR
00079 1038 A1 01           CMP    #1        Test if STEP=1?
00080 103A 27 3A           BEQ    SMCL6     Branch if STEP=1
00081 103C A1 02           CMP    #2        Test if STEP=2?
00082 103E 27 3A           BEQ    SMCL7     Branch if STEP=2
00083 1040 3F 8A           CLR    SDWNW     Initialize work area
00084 1042 A6 02           LDA    #2
00085 1044 B7 89           STA    STEPW
00086 1046 3C 8A    SMCL2  INC    SDWNW     Increment slow-down work
00087 1048 A6 04           LDA    #4        Add slow-up work
00088 104A BB 89           ADD    STEPW
00089 104C B7 89           STA    STEPW
00090 104E A1 1E           CMP    #30       Test if STEPW=>30?
00091 1050 24 33           BHS    SMCL9     Branch STEPW=>30
00092 1052 B1 82           CMP    STEP      Test if STEPW=STEP?
00093 1054 27 3F           BEQ    SMCL10    Branch STEPW=STEP
00094 1056 25 EE           BLO    SMCL2     Test if STEPW<STEP?
00095 1058 3F 84           CLR    SHOLD     Set no oprating data
00096 105A B6 8A    SMCL3  LDA    SDWNW     Store slow-down data
00097 105C B7 85           STA    SDWN
00098 105E 4C              INC    A         STEP-(SDWNW+1+SHOLD)->SUP
00099 105F BB 84           ADD    SHOLD
00100 1061 B0 82           SUB    STEP
00101 1063 40              NEG    A
00102 1064 B7 83           STA    SUP
00103 1066 CD 10F2  SMCL4  JSR    SMFR      Set fore or reverse data
00104 1069 A6 01           LDA    #1        Initialize pointer to slow-up
00105 106B B7 88           STA    UPPTR
00106 106D 12 8B           BSET   SMSF,FLG1  Set slue start flag
00107 106F 81              RTS
00108 1070 B6 86    SMCL5  LDA    STEPE     Store STEPE in slue-STEPE data
00109 1072 B7 87           STA    SCNTR
00110 1074 3F 86           CLR    STEPE
```

```
00111 1076 3F 84     SMCL6  CLR    SHOLD    Clear slue for(4steps+STEPE)
00112 1078 20 04            BRA    SMCL8
00113 107A A6 01     SMCL7  LDA    #1       Store slue for(8steps+STEPE)
00114 107C B7 84            STA    SHOLD
00115 107E 4F        SMCL8  CLR    A
00116 107F B7 83            STA    SUP      Store in other than slow-up
00117 1081 B7 85            STA    SDWN     Store in other than slow-down
00118 1083 20 E1            BRA    SMCL4
00119 1085 B6 82     SMCL9  LDA    STEP     Store operating data
00120 1087 A0 1D            SUB    #29
00121 1089 B7 84            STA    SHOLD
00122 108B A6 15            LDA    #21      Store slow-up data
00123 108D B7 83            STA    SUP
00124 108F A6 07            LDA    #7       Store slow-down data
00125 1091 B7 85            STA    SDWN
00126 1093 20 D1            BRA    SMCL4
00127 1095 A6 01     SMCL10 LDA    #1       Store operating data
00128 1097 B7 84            STA    SHOLD
00129 1099 20 BF            BRA    SMCL3
00130                ************************************************
00131                *                                              *
00132                *      NAME : SMREV (GENERATE STEPPING MOTOR   *
00133                *                    OUTPUT)                   *
00134                ************************************************
00135                *                                              *
00136                *      ENTRY : SUP(SLOW-UP DATA)               *
00137                *              SHOLD(OPERATING DATA)           *
00138                *              SDWN(SLOW-DOWN DATA)            *
00139                *              STEPE(REMAINDER OF STEP)        *
00140                *              FRFLG(FORE/REVERSE SLUE FLAG)   *
00141                *              SMSF(SLUE START FLAG)           *
00142                *    RETUNRS : SMSF(SLUE START FLAG)           *
00143                *                                              *
00144                ************************************************
00145 109B B6 81     SMREV  LDA    TDRD     Initialize TDR
00146 109D B7 08            STA    TDR
00147 109F 1F 09            BCLR   7,TCR    Clear interrupt request bit
00148 10A1 03 8B 27         BRCLR  SMSF,FLG1,SMREV2  Test if drive motor
00149 10A4 B6 80            LDA    PBSM     Drive stepping motor
00150 10A6 B7 01            STA    PBDTR
00151 10A8 3A 87            DEC    SCNTR    Decrement for every 4 steps
00152 10AA 26 1D            BNE    SMREV1   Output 4 steps ?
00153 10AC A6 04            LDA    #4       Initialize counter for every
00154 10AE B7 87            STA    SCNTR
00155 10B0 B6 83            LDA    SUP      Slow-up complete ?
00156 10B2 26 18            BNE    SMREV3
00157 10B4 B6 86            LDA    STEPE    STEPE complete ?
00158 10B6 26 21            BNE    SMREV4
00159 10B8 B6 84            LDA    SHOLD    Operate complete ?
00160 10BA 26 23            BNE    SMREV5
00161 10BC B6 85            LDA    SDWN     Slow-down complete ?
00162 10BE 26 23            BNE    SMREV6
00163 10C0 00 8B 2B         BRSET  FRFLG,FLG1,SMREV7  Back rush ?
00164 10C3 A6 01            LDA    #1       Initialize for back rush
00165 10C5 B7 87            STA    SCNTR
```

```
00166 10C7 10 8B          BSET   FRFLG,FLG1  Set fore slue flag
00167 10C9 AD 27    SMREV1 BSR    SMFR       Calculate forward or reverse
00168 10CB 80       SMREV2 RTI
00169 10CC 3A 83    SMREV3 DEC    SUP        Decrement slow-up data
00170 10CE BE 88           LDX    UPPTR      Store slow-up timer data
00171 10D0 D6 1104         LDA    TDRUP,X
00172 10D3 B7 81           STA    TDRD
00173 10D5 3C 88           INC    UPPTR      Increment slow-up data
00174 10D7 20 F0           BRA    SMREV1
00175 10D9 B7 87    SMREV4 STA    SCNTR      Store for operating
00176 10DB 3F 86           CLR    STEPE
00177 10DD 20 EA           BRA    SMREV1
00178 10DF 3A 84    SMREV5 DEC    SHOLD      Decrement slow-hold data
00179 10E1 20 E6           BRA    SMREV1
00180 10E3 BE 85    SMREV6 LDX    SDWN       Store slow-down timer data
00181 10E5 D6 1119         LDA    TDRDW-1,X
00182 10E8 B7 81           STA    TDRD
00183 10EA 3A 85           DEC    SDWN       Decrement slow-down data
00184 10EC 20 DB           BRA    SMREV1
00185 10EE 13 8B    SMREV7 BCLR   SMSF,FLG1  Clear slue start flag
00186 10F0 20 D9           BRA    SMREV2
00187                      ********************************************************
00188                      *                                                      *
00189                      *   NAME : SMFR (CALCULATE NOMAL/INVERSE ROTATION      *
00190                      *                DATA)                                 *
00191                      ********************************************************
00192 10F2 00 8B 07 SMFR   BRSET  FRFLG,FLG1,SMFR2  Reverse slue ?
00193 10F5 38 80           LSL    PBSM       Store next slue data
00194 10F7 24 02           BCC    SMFR1
00195 10F9 10 80           BSET   0,PBSM
00196 10FB 81       SMFR1  RTS
00197 10FC 34 80    SMFR2  LSR    PBSM       Store next slue data
00198 10FE 24 FB           BCC    SMFR1
00199 1100 1E 80           BSET   7,PBSM
00200 1102 20 F7           BRA    SMFR1
00201                      ********************************************************
00202                      *                                                      *
00203                      *              DATA TABLE                              *
00204                      *                                                      *
00205                      ********************************************************
00206 1104 FA       TDRUP  FCB    250        *Slow-up data
00207 1105 F2              FCB    242
00208 1106 EC              FCB    236
00209 1107 E0              FCB    224
00210 1108 D2              FCB    210
00211 1109 C6              FCB    198
00212 110A BC              FCB    188
00213 110B B4              FCB    180
00214 110C AC              FCB    172
00215 110D A4              FCB    164
00216 110E 9E              FCB    158
00217 110F 98              FCB    152
00218 1110 92              FCB    146
00219 1111 88              FCB    136
00220 1112 82              FCB    130
```

```
00221 1113 7E              FCB     126
00222 1114 7A              FCB     122
00223 1115 76              FCB     118
00224 1116 72              FCB     114
00225 1117 70              FCB     112
00226 1118 6C              FCB     108
00227 1119 6A              FCB     106
00228 111A FA       TDRDW  FCB     250       *Slow-down data
00229 111B CB              FCB     203
00230 111C AB              FCB     171
00231 111D 94              FCB     148
00232 111E 82              FCB     130
00233 111F 74              FCB     116
00234 1120 6E              FCB     110
00235              ************************************************
00236              *                                              *
00237              *       VECTOR ADDRESSES                       *
00238              *                                              *
00239              ************************************************
00240              *
00241 1FF6                 ORG     $1FF6
00242              *
00243 1FF6 1000            FDB     SMMN      SCI/TIMRE2
00244 1FF8 109B            FDB     SMREV     TIMER/INT2
00245 1FFA 1000            FDB     SMMN      INT
00246 1FFC 1000            FDB     SMMN      SWI
00247 1FFE 1000            FDB     SMMN      RES
00248              *
00249                      END
```

10. WITH A COMMERCIALLY AVAILABLE KEYBOARD

10.1 HARDWARE DESCRIPTION

(1) Function

(a) Receives key data from ASCII keyboard using the HD6305X0.

(b) Permits ASCII keyboard to output 7-bit ASCII code and $\overline{\text{STROBE}}$ signal when a key is pressed.

(2) Microcomputer Applications

(a) Inputs ASCII keyboard $\overline{\text{STROBE}}$ signal to the HD6305X0 $\overline{\text{INT}_2}$ pin and executes interrupt routine at the falling edge of $\overline{\text{STROBE}}$ pin.

(b) Reads key data in port B input by interrupt routine.

(3) Circuit Diagram

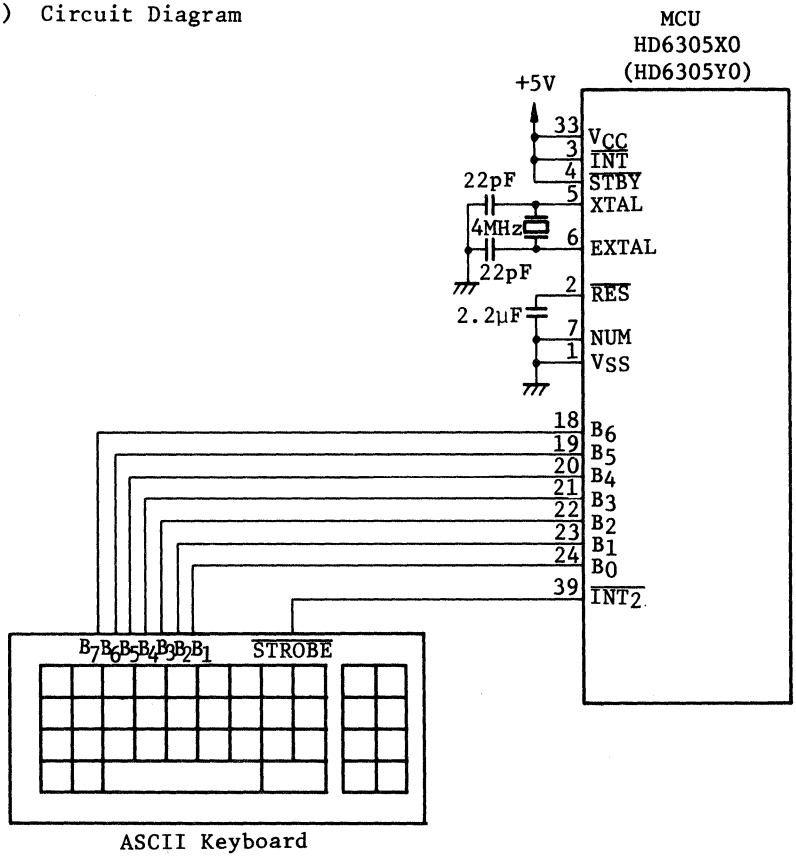


Fig. 1 Reading Data from ASCII Keyboard

(4) Pin Functions

Pin functions at the interface between the HD6305X0 and ASCII Keyboard are shown in Table 1.

Table 1 Pin Functions

Pin Name (HD6305X0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Key-board)	Program Label
$\overline{INT_2}$	Input	Low	Requests external interrupt to the HD6305X0.	$\overline{STROBE}$	
B_0	Input	—	Data lines	B_1	PBDTR
B_1	Input	—		B_2	
B_2	Input	—		B_3	
B_3	Input	—		B_4	
B_4	Input	—		B_5	
B_5	Input	—		B_6	
B_6	Input	—		B_7	

(5) Hardware Operation

Timing chart for $\overline{STROBE}$ signal and key data output are shown in Fig. 2. When $\overline{STROBE}$ signal is output from ASCII keyboard, $\overline{INT_2}$ interrupt is generated at the falling edge ① and the HD6305X0 reads port B data.

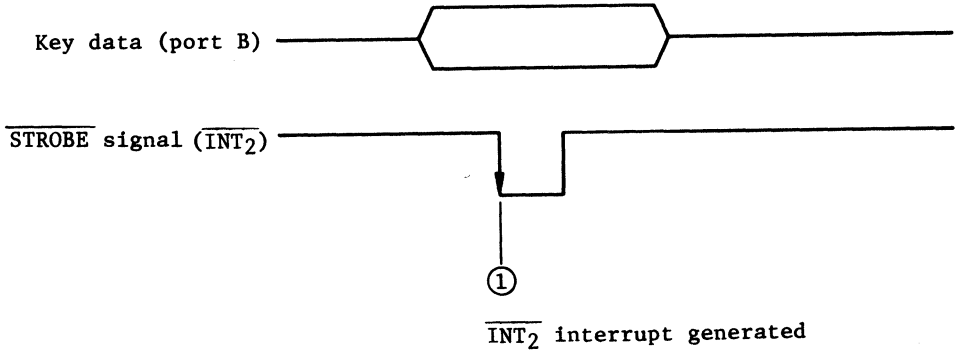


Fig. 2 ASCII Keyboard Timing Chart

10.2 SOFTWARE DESCRIPTION

(1) Program Module Configuration

The program module configuration for data input from ASCII keyboard is shown in Fig. 3.

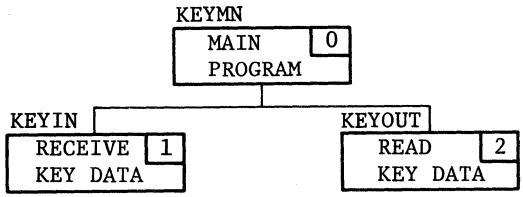


Fig. 3 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	KEYMN	Receives key data from ASCII keyboard.
1	RECEIVE KEY DATA	KEYIN	Stores key data in key buffer.
2	READ KEY DATA	KEYOUT	Reads key buffer data.

(3) Program Module Sample Application (Main Program)

The flowchart in Fig. 4 is an example of key data input from ASCII keyboard performed by the program module in Fig. 3.
Key data from ASCII keyboard is stored in KEYDAT(RAM) after main program of Fig. 4 is executed.

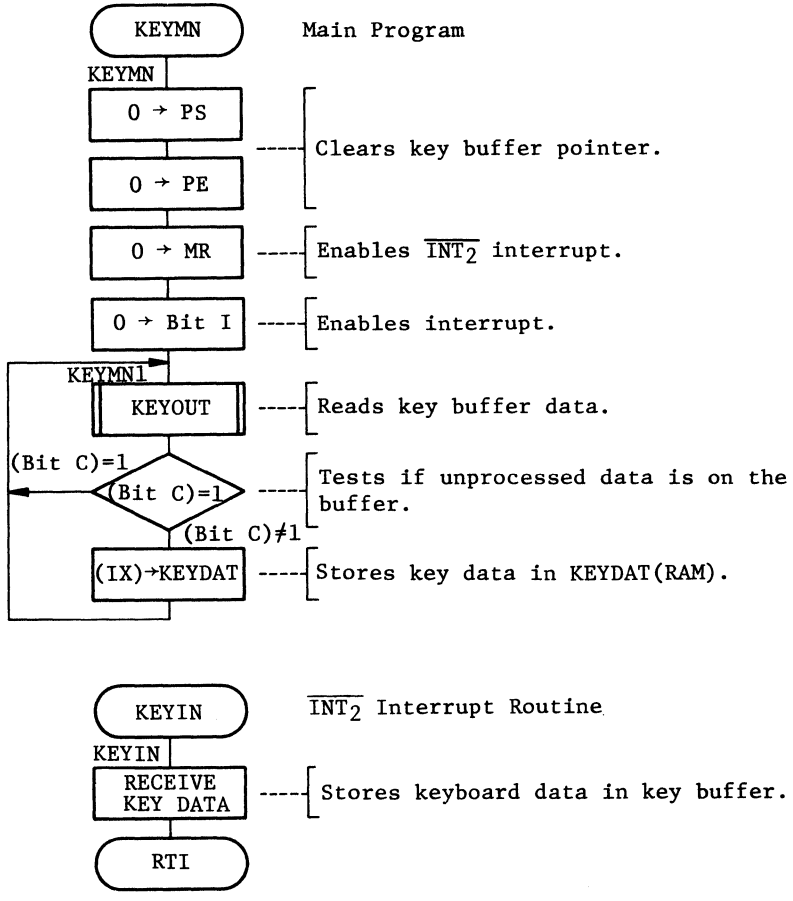


Fig. 4 Program Module Flowchart

PROGRAM MODULE NAME	RECEIVE KEY DATA	MCU/MPU	HD6305X0/Y0	LABEL	KEYIN

FUNCTION

Stores keyboard data in key buffer.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	—	—	—
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	×

C	Z
×	×
N	I
×	●
H	
●	

SPECIFICATIONS

ROM (Bytes)
23
RAM (Bytes)
18
Stack (Bytes)
0
No. of cycles
42
Reentrant
No
Relocation
No
Interrupt
No

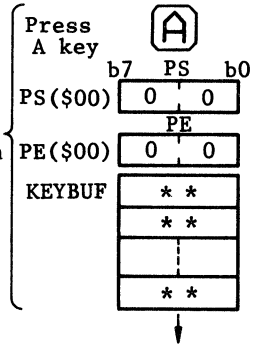
DESCRIPTION

(1) Function Details

(a) Program module KEYIN has no arguments.

(b) Fig. 5 shows an example of program module KEYIN execution. If key A is pressed as shown in part ① of Fig. 5, key data is stored in key buffer as shown in part ② of Fig. 5.

(c) Program module KEYIN calls neither program modules nor subroutines.



SPECIFICATIONS NOTES

PROGRAM MODULE NAME	RECEIVE KEY DATA	MCU/MPU	HD6305X0/Y0	LABEL	KEYIN

DESCRIPTION

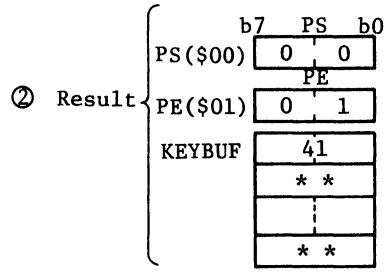
(2) User Notes

(a) Clears RAM since 2-byte RAM is used as pointer indicating key buffer.

(b) Selects DDR of port B as input.

(c) Clears $\overline{INT_2}$ interrupt mask bit.

(d) Clears bit I to enable $\overline{INT_2}$ interrupt.



**: hexadecimal

Fig. 5 Example of KEYIN Execution

(3) RAM Description

Label	RAM	Description
PS	b7 b0	Stores pointer indicating unprocessed key data in key buffer.
PE		Stores pointer indicating key data in key buffer.
KEYBUF		Used as key buffer storing 15-byte key data.

(4) Sample Application

Program module KEYIN is called if ASCII key is pressed after RAM to be used is cleard, $\overline{INT_2}$ interrupt is enabled, and after I/O port is selected and interrupt is enabled.

```
  ¦
CLR     A       }
STA     PS      }------ Clears RAM to be used.
STA     PE      }
STA     MR      ------ Clears INT2 interrupt mask.
STA     PBDDR   ------ Selects port B as input.
CLI             ------ Enables interrupt.
  ¦
```

7

PROGRAM MODULE NAME	RECEIVE KEY DATA	MCU/MPU	HD6305X0/Y0	LABEL	KEYIN

DESCRIPTION

(5) Basic Operation

(a) Input/output to/from key buffer.

(i) Calls program module KEYIN at every $\overline{INT_2}$ interrupt and fetches key data from key buffer. Then, calls program module KEYIN in main program and fetches key data from key buffer.

(ii) Clears starting point PS(RAM) and ending point PE(RAM) and stores key data in key buffer starting address.

(iii) Program module KEYIN stores 1 byte of key data in 16-byte buffer area pointed by PE(RAM), and increments PE(RAM).

(iv) Program module KEYOUT fetches 1 byte from 16-byte buffer area pointed by PS(RAM) and increments PS(RAM).

(v) PS(RAM) and PE(RAM) become "0" if they are incremented till 15 bytes because the buffer area is 16-byte long.

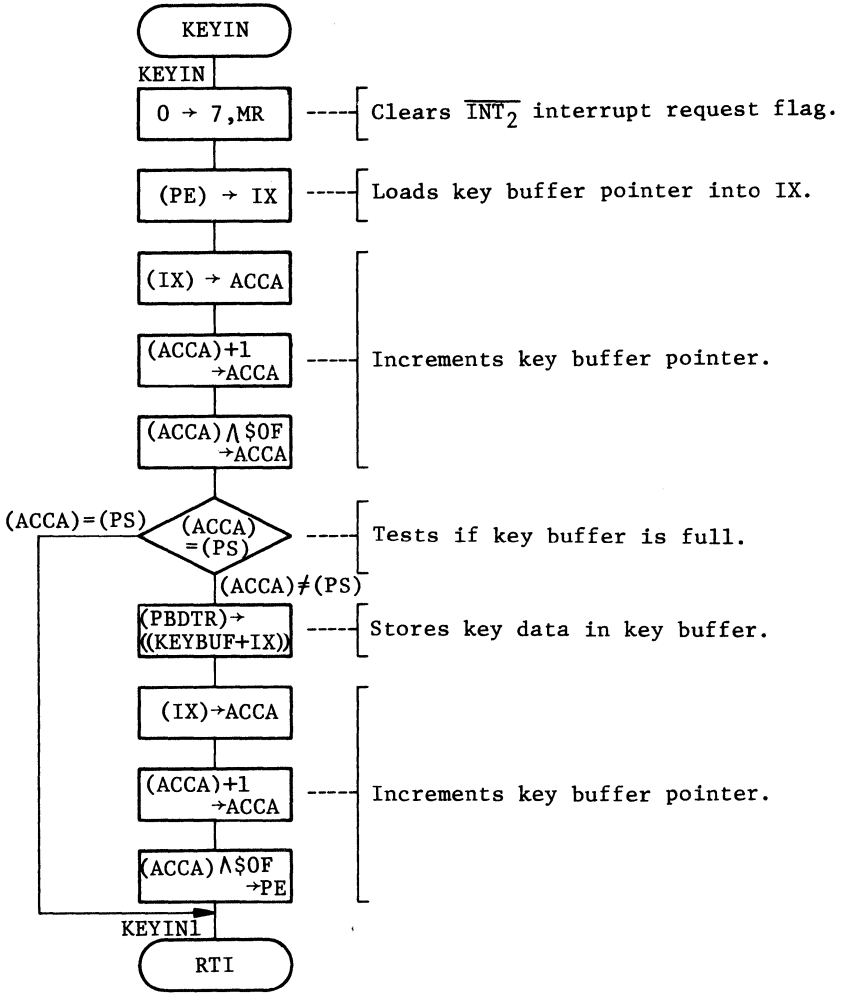
(b) Input to key buffer

(i) Program module KEYIN loads PE(RAM) into ACCA and increments ACCA. Then, compares ACCA content with PS(RAM). If (ACCA)=PS(RAM), key data is not stored in key buffer.
If (ACCA) $\neq$ (PS), key data is stored in key buffer and PE(RAM) is incremented.

(ii) Key buffer can be used up to 15 bytes.

PROGRAM MODULE NAME	RECEIVE KEY DATA	MCU/MPU	HD6305X0/Y0	LABEL	KEYIN

FLOWCHART



7

PROGRAM MODULE NAME	READ KEY DATA	MCU/MPU	HD6305X0/Y0	LABEL	KEYOUT

FUNCTION

Reads data from key buffer.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	—	—	—
	Returns	Unprocessed key data	IX	1
		Unprocessed key data existence	Bit C (CCR)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	×

C	Z
↕	×
N	I
×	●
H	
●	

SPECIFICATIONS

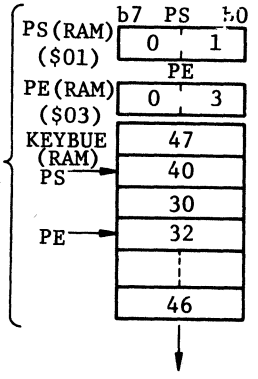
ROM (Bytes)
20
RAM (Bytes)
18
Stack (Bytes)
0
No. of cycles
27
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

IX: Contains unprocessed key data in key buffer.
Bit C (CCR): Indicates existence of unprocessed key data in key buffer.
Bit C = 0: Unprocessed data is in key buffer.



SPECIFICATIONS NOTES

PROGRAM MODULE NAME	READ KEY DATA	MCU/MPU	HD6305X0/Y0	LABEL	KEYOUT

DESCRIPTION

② Return arguments

IX ($40)

BitC	b7				IX			b0
0					4			0

BitC = 1: No unprocessed data is in key buffer.

Fig. 6 Example of KEYOUT Execution

(b) Fig. 6 shows an example of program module KEYOUT execution. Unprocessed data is contained into IX after program module KEYOUT execution.

(c) Program module KEYOUT calls neither program modules nor subroutine.

(2) User Notes

Program module KEYOUT should be used with program module KEYIN.

(3) RAM Description

Label	RAM	Description
PS	b7 b0	Stores pointer indicating unprocessed data in key buffer.
PE		Stores pointer indicating key data in key buffer.
KEYBUF		Used as key buffer storing 15 byte key data.

(4) Sample Application

First, clears RAM to be used, enables $\overline{INT_2}$ interrupt, and initializes I/O port. Second, enables interrupt and executes program module KEYIN. Then, execute program module KEYOUT.

```
WORK1   RMB   1        ---Reserves memory byte for key buffer unprocessed
                          data.
        CLR   A        )
        STA   PS       )-Clears RAM to be used.
        STA   PE       )
        STA   MR       ---Enables INT2 interrupt.
        STA   PBDDR    ---Selects port B as input.
        CLI            ---Enables interrupt.
LOOP    JSR   KEYOUT   --Calls program module KEYOUT.
        BCS   LOOP     ---Tests if unprocessed data is in key buffer.
        STX   WORK1    ---Stores unprocessed data in return argument in
                          RAM.
```

7

PROGRAM MODULE NAME	READ KEY DATA	MCU/MPU	HD6305X0/Y0	LABEL	KEYOUT

DESCRIPTION

(5) Basic Operation

(a) Input/output to/from key buffer.
See (a) of (5) Basic operation in program module KEYIN for details.

(b) Output from key buffer.

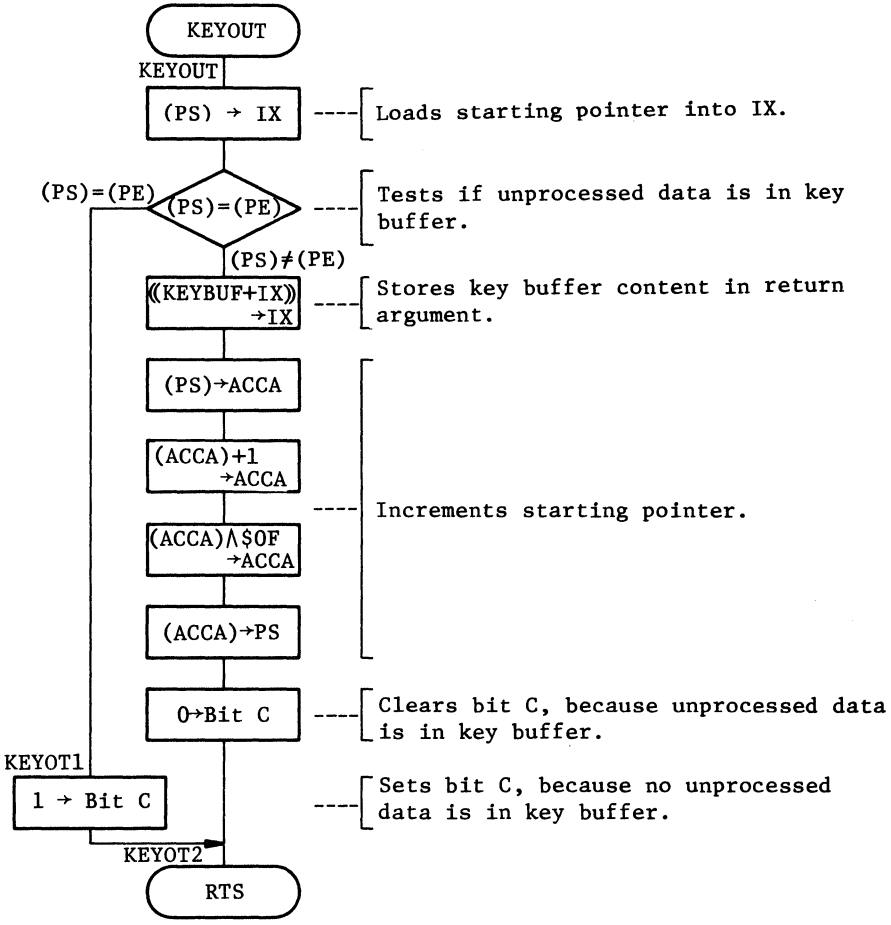
(i) Program module KEYOUT tests if values in starting pointer PS(RAM) and values in ending pointer PE(RAM) are equal.

(ii) In case of PS(RAM)=PE(RAM), no key data is in key buffer and bit C is set.

(iii) In case of PS(RAM)≠PE(RAM), key data is fetched from key buffer PS(RAM) indicates, and PS(RAM) is incremented.

PROGRAM MODULE NAME	READ KEY DATA	MCU/MPU	HD6305X0/Y0	LABEL	KEYOUT

FLOWCHART



10.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

10.5 PROGRAM LISTING

```
00001                    *
00002                    ****     RAM ALLOCATION     ************************
00003                    *
00004 0080                        ORG     $80
00005                    *
00006 0080 0001          PS       RMB     1          Roll buffer start pointer
00007 0081 0001          PE       RMB     1          Roll buffer end pointer
00008 0082 0010          KEYBUF   RMB     16         Roll buffer area
00009 0092 0001          KEYDAT   RMB     1          Work for key data
00010                    *
00011                    ****     SYNBOL DEFINITIONS    ************
00012                    *
00013       0005         PBDDR    EQU     $05        Port B data direction register
00014       0001         PBDTR    EQU     $01        Port B data register
00015       000A         MR       EQU     $0A        Miscellaneous register
00016                    ************************************************
00017                    *                                              *
00018                    *             MAIN PROGRAM : KEYMN             *
00019                    *                                              *
00020                    ************************************************
00021                    *
00022 1000                        ORG     $1000
00023                    *
00024 1000 4F            KEYMN    CLR     A
00025 1001 B7 80                  STA     PS         Initialize start pointer
00026 1003 B7 81                  STA     PE         Initialize end pointer
00027 1005 B7 0A                  STA     MR         Initialize INT2 interrupt
00028 1007 9A                     CLI                Enable interrupt
00029 1008 CD 1011       KEYMN1   JSR     KEYOUT     Read key data of roll buffer
00030 100B 25 FB                  BCS     KEYMN1     Test if read
00031 100D BF 92                  STX     KEYDAT     Store work RAM
00032 100F 20 F7                  BRA     KEYMN1
00033                    ************************************************
00034                    *                                              *
00035                    *        NAME : KEYOUT (READ KEY DATA)         *
00036                    *                                              *
00037                    ************************************************
00038                    *                                              *
00039                    *       ENTRY : NOTHING                        *
00040                    *     RETURNS : IX    (KEY DATA)               *
00041                    *               CARRY (C=0:TRUE,C=1:FALSE)     *
00042                    *                                              *
00043                    ************************************************
00044 1011 BE 80         KEYOUT   LDX     PS         Start pointer = end pointer ?
00045 1013 B3 81                  CPX     PE
00046 1015 27 0C                  BEQ     KEYOT1     Branch if equal
00047 1017 EE 82                  LDX     KEYBUF,X   Read key data
00048 1019 B6 80                  LDA     PS         Increment start pointer
00049 101B 4C                     INC     A
00050 101C A4 0F                  AND     #$0F       Clear upper 4 bit
00051 101E B7 80                  STA     PS
00052 1020 98                     CLC                Clear carry
00053 1021 20 01                  BRA     KEYOT2
00054 1023 99            KEYOT1   SEC                Set carry
00055 1024 81            KEYOT2   RTS
```

```
00056                   ************************************************
00057                   *                                              *
00058                   *        NAME : KEYIN (RECEIVE KEY DATA)       *
00059                   *                                              *
00060                   ************************************************
00061                   *                                              *
00062                   *              ENTRY : NOTHING                 *
00063                   *            RETURNS : NOTHING                 *
00064                   *                                              *
00065                   ************************************************
00066 1025 1F 0A       KEYIN  BCLR   7,MR      Enable INT2 inturrupt
00067 1027 BE 81              LDX    PE        Increment end pointer
00068 1029 9F                 TXA              Transfer IX to ACCA
00069 102A 4C                 INC    A         Increment ACCA
00070 102B A4 0F              AND    #$0F      Clear upper 4 bit
00071 102D B1 80              CMP    PS        Start pointer=end pointer ?
00072 102F 27 0A              BEQ    KEYIN1    Branch equal
00073 1031 B6 01              LDA    PBDTR     Load data from key boad
00074 1033 E7 82              STA    KEYBUF,X  Store in roll buffer
00075 1035 9F                 TXA              Transfer IX to ACCA
00076 1036 4C                 INC    A         Increment end pointer
00077 1037 A4 0F              AND    #$0F      Clear upper 4 bit
00078 1039 B7 81              STA    PE
00079 103B 80           KEYIN1 RTI
00080                   ************************************************
00081                   *                                              *
00082                   *              VECTOR ADDRESSES                *
00083                   *                                              *
00084                   ************************************************
00085                   *
00086 1FF6                    ORG    $1FF6
00087                   *
00088 1FF6 1000               FDB    KEYMN     SCI/TIMER2
00089 1FF8 1025               FDB    KEYIN     TIMER/INT2
00090 1FFA 1025               FDB    KEYIN     INT
00091 1FFC 1025               FDB    KEYIN     SWI
00092 1FFE 1025               FDB    KEYIN     RES
00093                   *
00094                         END
```

11. SCI CLOCK SYNCHRONOUS (EXTERNAL CLOCK)

11.1 HARDWARE DESCRIPTION

(1) Function

(a) Receives ASCII sent from master as clock synchronous serial data using the HD6305X0, converts the received data from ASCII lowercase into uppercase and sends it to master system.

(b) Converts ASCII lowercase into uppercase, if lowercase is received.

(c) Uses protocol sending data from master system first.

(2) Microcomputer Application

(a) Transfers data to/from master system using clocked SCI.

(b) Transfers data by sending $\overline{CK}$ request to master system and receiving transfer clock from master system using port C bit 4.

(3) Circuit Diagram

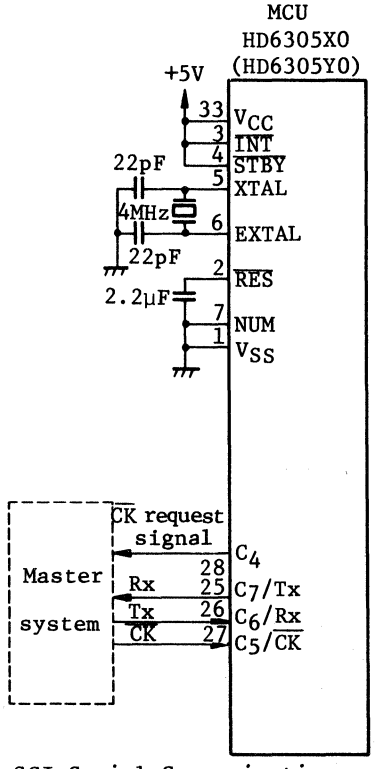


Fig. 1 SCI Serial Communication

(4) Pin Functions

Pin functions at the interface between the HD6305X0 SCI pins and master system pins.

Table 1 Pin Functions

Pin Name (HD6305X0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Master system)	Program Label
$\overline{CK}$	Input	Low	Inputs transfer clock when receiving/sending serial data	Serial clock	
Rx	Input	—	Inputs serial data	Serial data output	
Tx	Output	—	Outputs serial data	Serial data input	
C_4	Output	Low	Requests transfer clock to output to master system	$\overline{CK}$ request signal	PCDTR

(5) Hardware Operation

SCI timing chart is shown in Fig. 2.

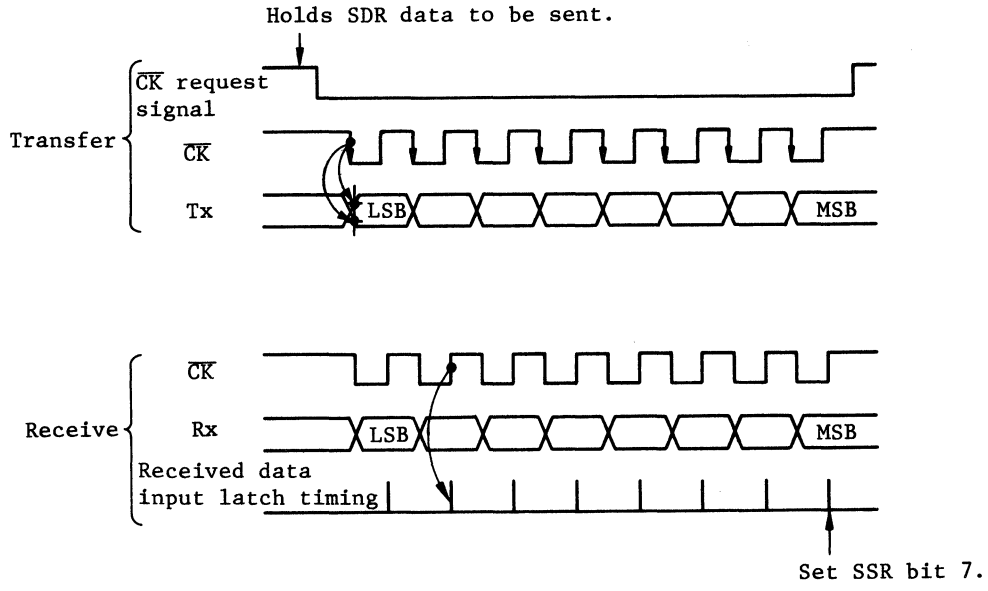


Fig. 2 SCI Timing Chart

11.2 SOFTWARE DESCRIPTION

(1) Program Module Configuration

The program module configuration for SCI communication with master system is shown in Fig. 3.

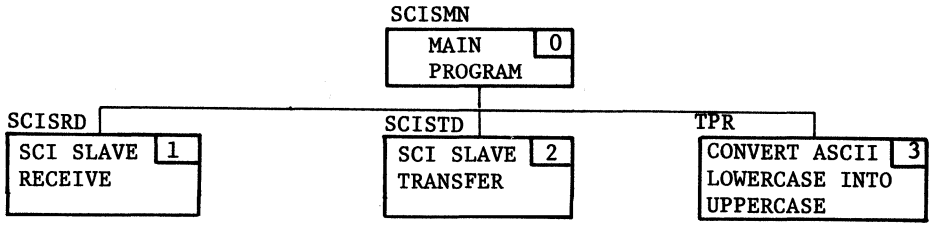


Fig. 3 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	SCISMN	Communicates with master system using clocked synchronous interface SCI.
1	SCI SLAVE RECEIVE	SCISRD	Receives data from master system using external clock.
2	SCI SLAVE TRANSFER	SCISTD	Sends serial data to master using external clock.
3	CONVERT ASCII LOWERCASE INTO UPPERCASE	TPR	Converts ASCII lowercase into uppercase. See subroutine TPR in HD6305 FAMILY APPLICATION NOTES (SOFTWARE) for details.

(3) Program Module Application (Main Program)

Flowchart in Fig. 3 is an example of receiving ASCII from master system, converting ASCII lowercase into uppercase and sending it to master system, performed by the program module in Fig. 3.

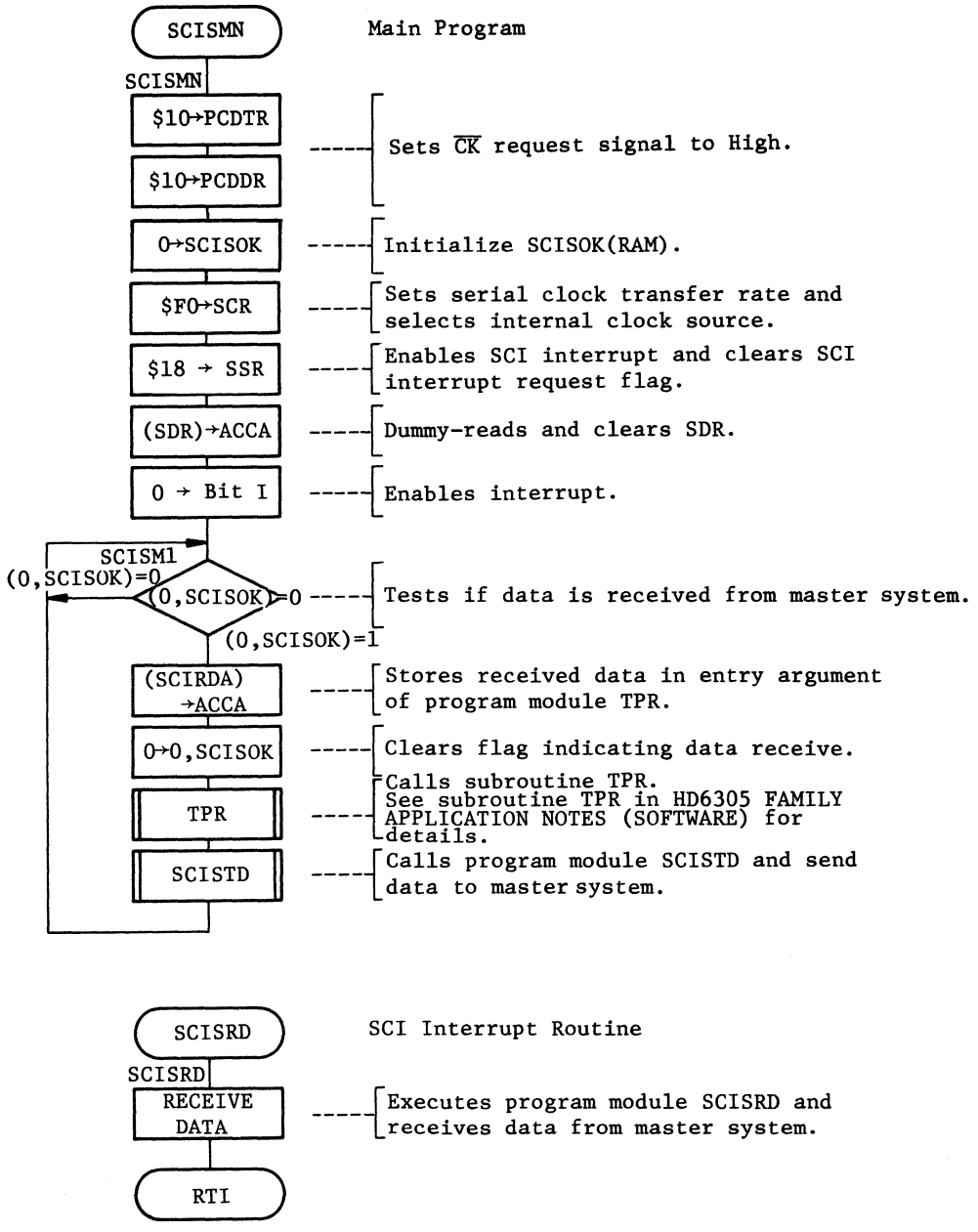


Fig. 4 Program Module Flowchart

PROGRAM MODULE NAME	SCI SLAVE RECEIVE	MCU/MPU	HD6305X0/Y0	LABEL	SCISRD

FUNCTION

Receives data sent from master system and stores it in SCIRDA(RAM).

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	—	—	—
	Returns	Received data	SCIRDA (RAM)	1
		Existence of received data	SCISOK (RAM)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
●	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
18
RAM (Bytes)
1
Stack (Bytes)
0
No. of cycles
26
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

SCIRDA(RAM): Contains data sent from master system.

SCISOK(RAM) : Indicates existence of received data.

SCISOK(RAM)=1: Data is received from master system.

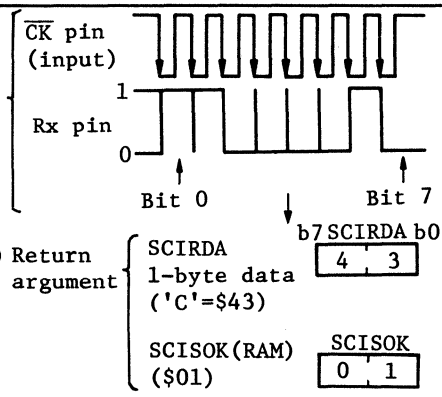


Fig. 5 Example of SCISRD Execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles are needed when having no wait time for receiving data.

SCISOK(RAM)=0: No data is received from master system.

(b) Program module SCISRD execution stores contents of SCI data register in SCIRDA (RAM).

(c) SCISRD calls neither program modules nor subroutines.

(2) User Notes

(a) Initializes SCR.

(b) When program module SCISRD is used, resetting system (in case of power on reset, supplying power) should be performed from master system.

(c) Program module SCISRD should be called before master system begins to send data.

(d) Program module SCISRD loops until sending data from master system is completed.

(3) RAM Description

Label	RAM b7 b0	Description
SCIRDA		Stores received data.
SCISOK		Stores existence of received data.

(4) Sample Application

Program module SCISRD is called if SDR is dummy-read, bit I is cleared, data is received after I/O port is designated and SCR and SSR are initialized.

```
          :
          LDA     #$10
          STA     PCDTR     ------ Sets port C bit 4 to High.
          STA     PCDDR
          LDA     #$F0
          STA     SCR       ------ Sets transfer rate and selects internal
          LDA     #$18              clock source.
          STA     SSR       ------ Enables SCI interrupt.
          LDA     SDR       ------ Dummy-reads and clears SDR.
          CLI               ------ Enables interrupt.
LOOP      BRCLR   0,SCISOK,LOOP --- Waits for receive completion.
          LDA     SCIRDA    ------ Loads received data into ACCA.
          BCLR    0,SCISOK  ------ Clears flag indicating receive completion
          :
```

(5) Basic Operation

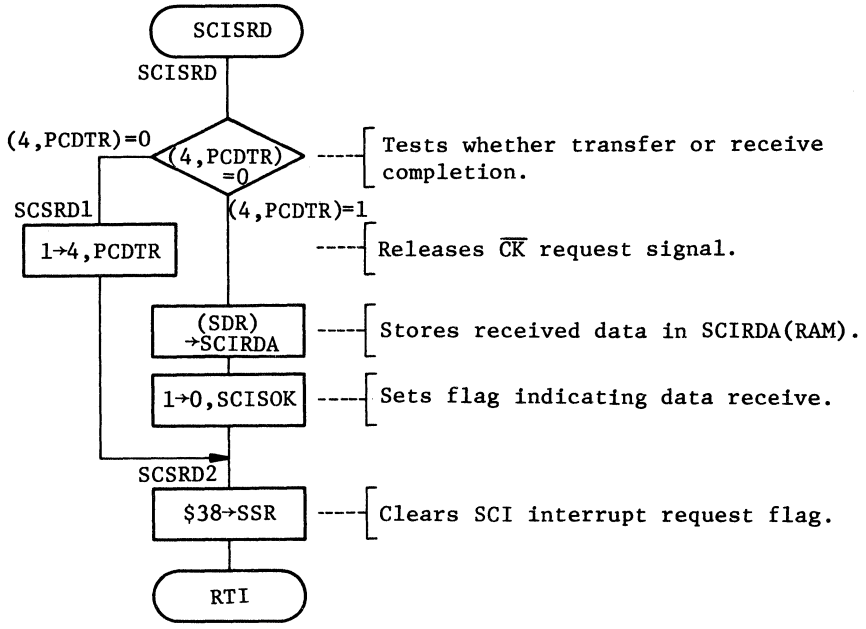
(a) Receives serial data using SCI interrupt routine.

(b) SCI interrupt routine is generated at serial data transfer/receive.

(c) If SCI interrupt is generated after serial data is transferred, either transfer or receive is decided by considering $\overline{CK}$ request signal being output.

PROGRAM MODULE NAME	SCI SLAVE RECEIVE	MCU/MPU	HD6305X0/Y0	LABEL	SCISRD

FLOWCHART



PROGRAM MODULE NAME	SCI SLAVE TRANSFER	MCU/MPU	HD6305X0/Y0	LABEL	SCISTD

FUNCTION

Sends data of ACCA to master system.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Data to be sent	ACCA	1
	Returns	-	-	-

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
●	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
5
RAM (Bytes)
0
Stack (Bytes)
0
No. of cycles
13
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

ACCA: Holds data to be sent to master system.

(b) Program module SCISTD execution transfers contents of ACCA to SDR and sends them to master.

(c) SCISTD calls neither program modules nor subroutines.

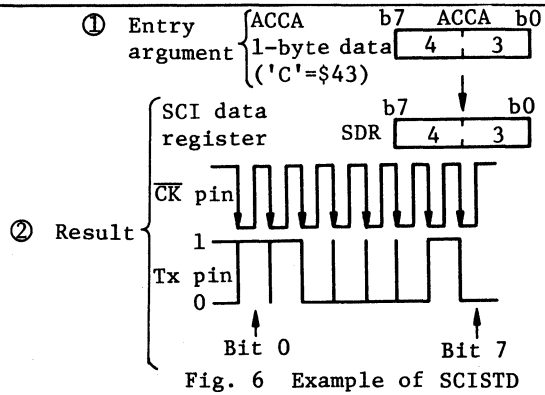


Fig. 6 Example of SCISTD Execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents number of cycles needed when having no wait time for transfer completion.

PROGRAM MODULE NAME	SCI SLAVE TRANSFER	MCU/MPU	HD6305X0/Y0	LABEL	SCISTD

DESCRIPTION

(2) User Notes

(a) Selects bit 4 of port C as output.

(b) Initialize SCI.

(3) RAM Description

RAM is not used by program module SCISTD.

(4) Sample Application

Call SCISTD after selecting I/O port, initializing SCR and SSR and storing data to be sent.

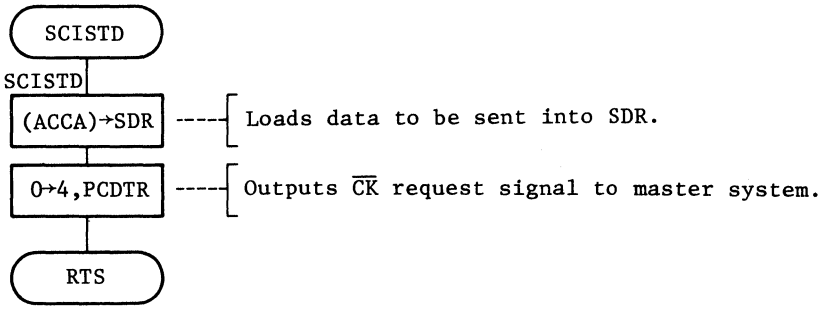
```
WORK1   RMB     1           ----- Reserves memory byte for loading
                                  data to be sent in user program.
        LDA     #$10        }
        STA     PCDTR       }---- Sets port C bit 4 to High.
        STA     PCDDR       }
        LDA     #$F0        }
        STA     SCR         }---- Selects external clock source.
        LDA     #$18        }
        STA     SSR         }---- Clears SCI interrupt request flag.
        LDA     WORK1       }---- Loads data to be sent into entry argument.
        JSR     SCISTD      ----- Calls program module SCISTD.
```

(5) Basic Operation

Loads data to be sent into SDR and outputs $\overline{CK}$ request signal.

PROGRAM MODULE NAME	SCI SLAVE TRANSFER	MCU/MPU	HD6305X0/Y0	LABEL	SCISTD

FLOWCHART



11.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

11.5 PROGRAM LISTING

```
00001                   *
00002                   ****    RAM ALLOCATION    ************************
00003                   *
00004 0080                      ORG     $80
00005                   *
00006 0080 0001         SCIRDA  RMB     1         Received data
00007 0081 0001         SCISOK  RMB     1         Existence of received data
00008                   *
00009                   ****    SYMBOL DEFINITIONS    ********************
00010                   *
00011       0002        PCDTR   EQU     $02       Port C data register
00012       0006        PCDDR   EQU     $06       Port C data direction register
00013       0010        SCR     EQU     $10       SCI contorol register
00014       0011        SSR     EQU     $11       SCI status register
00015       0012        SDR     EQU     $12       SCI data register
00016                   ************************************************
00017                   *                                              *
00018                   *        MAIN PROGRAM : SCISMN                 *
00019                   *                                              *
00020                   ************************************************
00021                   *
00022 1000                      ORG     $1000
00023                   *
00024 1000 A6 10        SCISMN  LDA     #$10
00025 1002 B7 02                STA     PCDTR     Initialize port C
00026 1004 B7 06                STA     PCDDR     Select port C bit 4 as output
00027 1006 4F                   CLR     A         Initialize SCISOK
00028 1007 B7 81                STA     SCISOK
00029 1009 A6 F0                LDA     #$F0      Initialize SCR
00030 100B B7 10                STA     SCR
00031 100D A6 18                LDA     #$18      Initialize SSR
00032 100F B7 11                STA     SSR
00033 1011 B6 12                LDA     SDR       Dummy read of SDR
00034 1013 9A                   CLI               Enable interrupt
00035 1014 01 81 FD SCISM1      BRCLR   0,SCISOK,SCISM1  Test if received data
00036 1017 B6 80                LDA     SCIRDA    Load received data
00037 1019 11 81                BCLR    0,SCISOK  Crear received data flag
00038 101B CD 103A              JSR     TPR       Convert into lowercase
00039 101E CD 1035              JSR     SCISTD    Output data to master
00040 1021 20 F1                BRA     SCISM1
00041                   ************************************************
00042                   *                                              *
00043                   *   NAME : SCISRD (SCI SLAVE RECEIVE)          *
00044                   *                                              *
00045                   ************************************************
00046                   *                                              *
00047                   *      ENTRY : NOTHING                         *
00048                   *    RETURNS : SCIRDA  (RECEIVED DATA)         *
00049                   *              SCISOK  (SCISOK=1;TRUE,         *
00050                   *                       SCISOK=0;FALSE)        *
00051                   *                                              *
00052                   ************************************************
00053 1023 09 02 08 SCISRD      BRCLR   4,PCDTR,SCSRD1  Test if Tx or Rx
00054 1026 B6 12                LDA     SDR       Store received data
00055 1028 B7 80                STA     SCIRDA
```

```
00056 102A 10 81            BSET    0,SCISOK Set received data flag
00057 102C 20 02            BRA     SCSRD2   Branch SCSRD2
00058 102E 18 02    SCSRD1  BSET    4,PCDTR  Set CK=1
00059 1030 A6 38    SCSRD2  LDA     #$38     Clear interrupt request flag
00060 1032 B7 11            STA     SSR
00061 1034 80               RTI
00062                ***************************************************
00063                *                                                 *
00064                *   NAME : SCISTD (SCI SLAVE TRNSFER)             *
00065                *                                                 *
00066                ***************************************************
00067                *                                                 *
00068                *      ENTRY : ACCA    (DATA TO BE SENT)          *
00069                *    RETURNS : NOTHING                            *
00070                *                                                 *
00071                ***************************************************
00072 1035 B7 12    SCISTD  STA     SDR      Store transfer data
00073 1037 19 02            BCLR    4,PCDTR  Set CK-pin=0
00074 1039 81               RTS
00075                ***************************************************
00076                *                                                 *
00077                *   NAME : TPR (CONVERTING ASCII LOWERCASE        *
00078                *               INTO ASCII)                       *
00079                *                                                 *
00080                ***************************************************
00081 103A A1 61    TPR     CMP     #'a      ACCA - 'a' ?
00082 103C 25 06            BCS     TPR1     Branch if ACCA < 'a'
00083 103E A1 7A            CMP     #'z      ACCA - 'z' ?
00084 1040 22 02            BHI     TPR1     Branch if ACCA > 'z'
00085 1042 A4 DF            AND     #$DF     Convert lowercase to Uppercase
00086 1044 81       TPR1    RTS
00087                ***************************************************
00088                *                                                 *
00089                *              VECTOR ADDRESSES                   *
00090                *                                                 *
00091                ***************************************************
00092                *
00093 1FF6                  ORG     $1FF6
00094                *
00095 1FF6 1023             FDB     SCISRD   SCI/TIMER2
00096 1FF8 1000             FDB     SCISMN   TIMER/INT2
00097 1FFA 1000             FDB     SCISMN   INT
00098 1FFC 1000             FDB     SCISMN   SWI
00099 1FFE 1000             FDB     SCISMN   RES
00100                       END
```

7

12.1 HARDWARE DESCRIPTION

(1) Function

(a) Transfers clock synchronous serial data to slave system, and receives data from slave.

(b) Handshakes using transfer/receive control signal to transfer or receive data.

(2) Microcomputer Applications

(a) Transfers data to/from slave system using clock synchronous SCI.

(b) Inputs $\overline{CK}$ request signal to bit 3 of port C from slave system. Outputs transfer rate clock to slave system considering input state and receives data.

(c) Outputs transfer request signal from bit 4 of port C to inform data transfer to slave system.

(3) Circuit Diagram

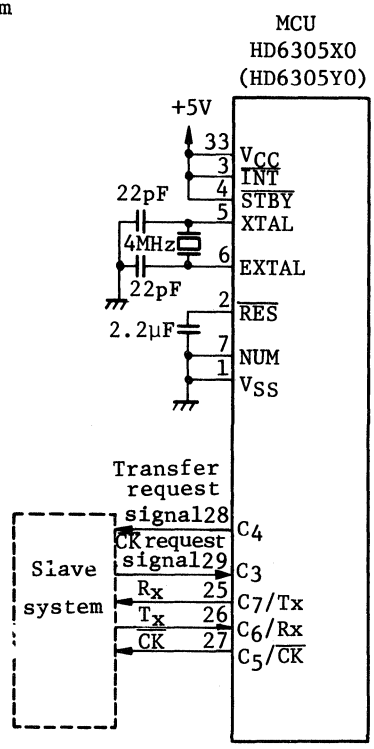


Fig. 1 Clock Synchronous SCI Circuit

(4) Pin Functions

Pin functions at the interface between the HD6305X0 SCI pins and slave system are shown in Table 1.

Table 1 Pin Functions

Pin Name (HD6305X0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Slave system)	Program Lavel
$\overline{CK}$	Output	Low	Outputs transfer clock.	Serial clock	
Rx	Input	—	Receives data.	Serial data output	
Tx	Output	—	Transfers data.	Serial data input	
C_3	Input	Low	Inputs transfer clock request from slave. Outputs clock if low.	$\overline{CK}$ request signal	PCDTR
C_4	Output	Low	Informs transfer start to slave.	Transfer request signal	

(5) Hardware Operation

SCI timing chart is shown in Fig. 2.

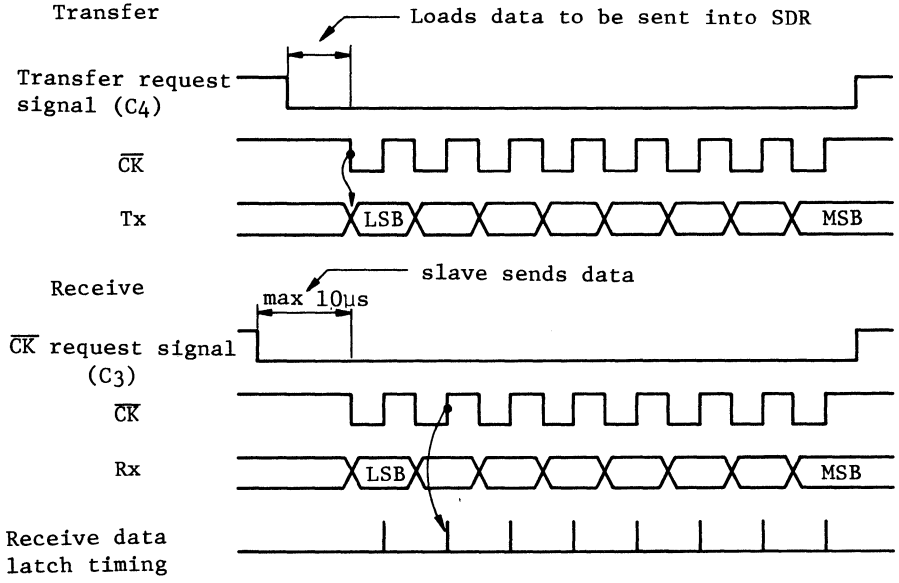


Fig. 2 SCI Timing Chart

12.2 SOFTWARE DESCRIPTION

(1) Program Module Configuration

The program module configuration for serial communication with slave system is shown in Fig. 3.

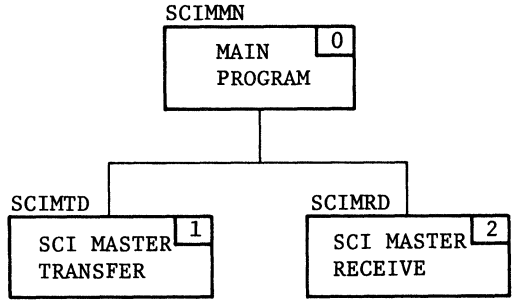


Fig. 3 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	SCIMMN	Sends data to slave system and receive it from slave system without change by using the HD6305X0 SCI.
1	SCI MASTER TRANSFER	SCIMTD	Sends serial data to slave system using internal clock.
2	SCI MASTER RECEIVE	SCIMRD	Receives data from slave system using internal clock.

(3) Program Module Sample Application (Main Program)

The flowchart in Fig. 4 is an example of sending serial data to slave system and receiving it from slave system, performed by the program module in Fig. 3.

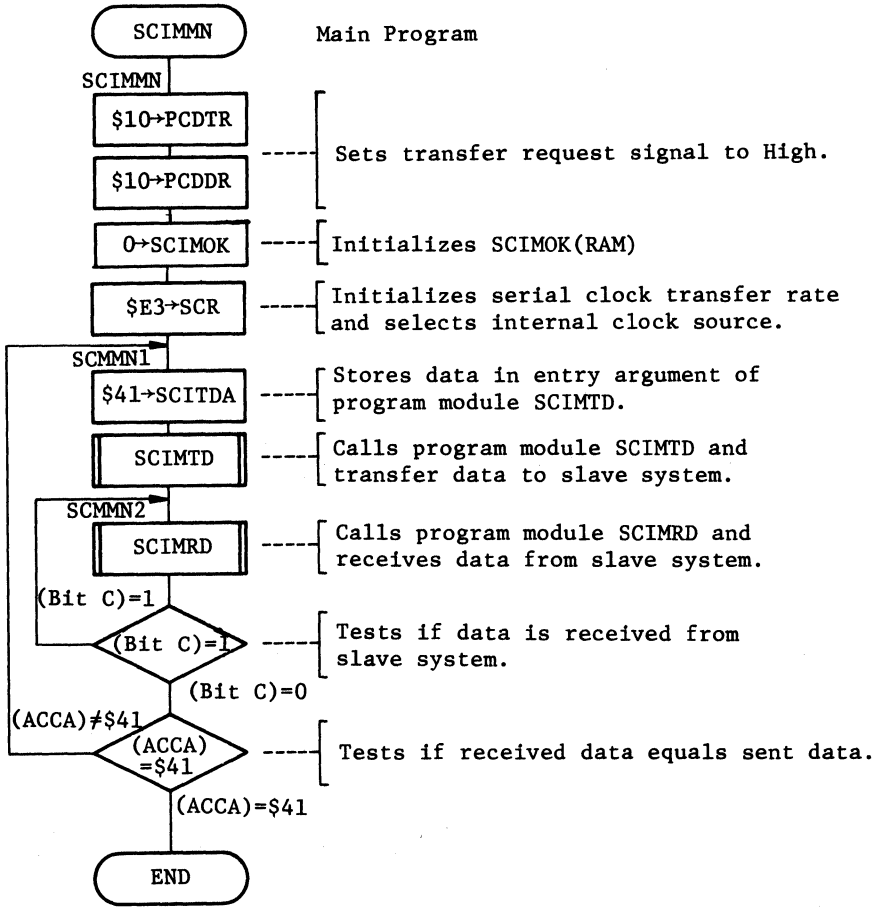


Fig. 4 Program Module Flowchart

PROGRAM MODULE NAME	SCI MASTER TRANSFER	MCU/MPU	HD6305X0/Y0	LABEL	SCIMTD

FUNCTION

Sends data in SCITDA(RAM) into slave system using internal clock.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Data to be sent	SCITDA (RAM)	1
	Returns	-	-	-

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	×

C	Z
●	×

N	I
×	●

H
●

SPECIFICATION

ROM (Bytes)
48
RAM (Bytes)
2
Stack (Bytes)
0
No. of cycles
50
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details
SCITDA(RAM): Holds data to be sent to slave system.

(b) Program module SCIMTD execution loads entry argument content into SDR and sends it to slave system.

(c) Program module SCIMTD calls neither program modules nor subroutines.

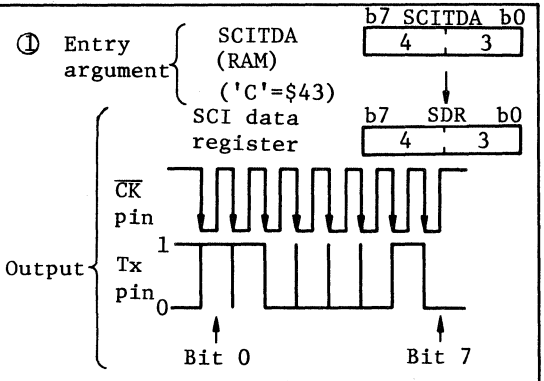


Fig. 5 Example of SCIMTD Execution

SPECIFICATIONS NOTES

(2) User Notes

(a) Initializes SCR.

(b) When using program module SCIMTD, resetting system (in case of power on reset, activating power) is performed master reset firstly, then reset slave system secondly.

(c) Sets slave to receiving state before program module SCIMTD execution.

(d) Program module SCIMTD loops until sending data is completed.

(3) RAM Description

Label	RAM (b0 ... b7)	Description
SCITDA		Stores data to be sent.
SCIMOK		Indicates data.

(4) Sample Application

Calls program module SCIMTD after selecting I/O port, loading data to be sent and initializing SCR.

```
        :
LDA     #$10
STA     PCDTR       ----- Sets bit 4 of port C to High.
STA     PCDDR
LDA     #$E3
STA     SCR         ----- Selects internal clock source.
LDA     #$41
STA     SCITDA      ----- Loads data to be sent into entry argument.
JSR     SCIMTD      ----- Calls program module SCIMTD.
        :
```

(5) Basic Operation

(a) When transfer request signal is output, $\overline{CK}$ request signal may output from slave system with the same timing. Then, 12μs software timer is executed and $\overline{CK}$ request signal is tested after $\overline{CK}$ request signal is output.

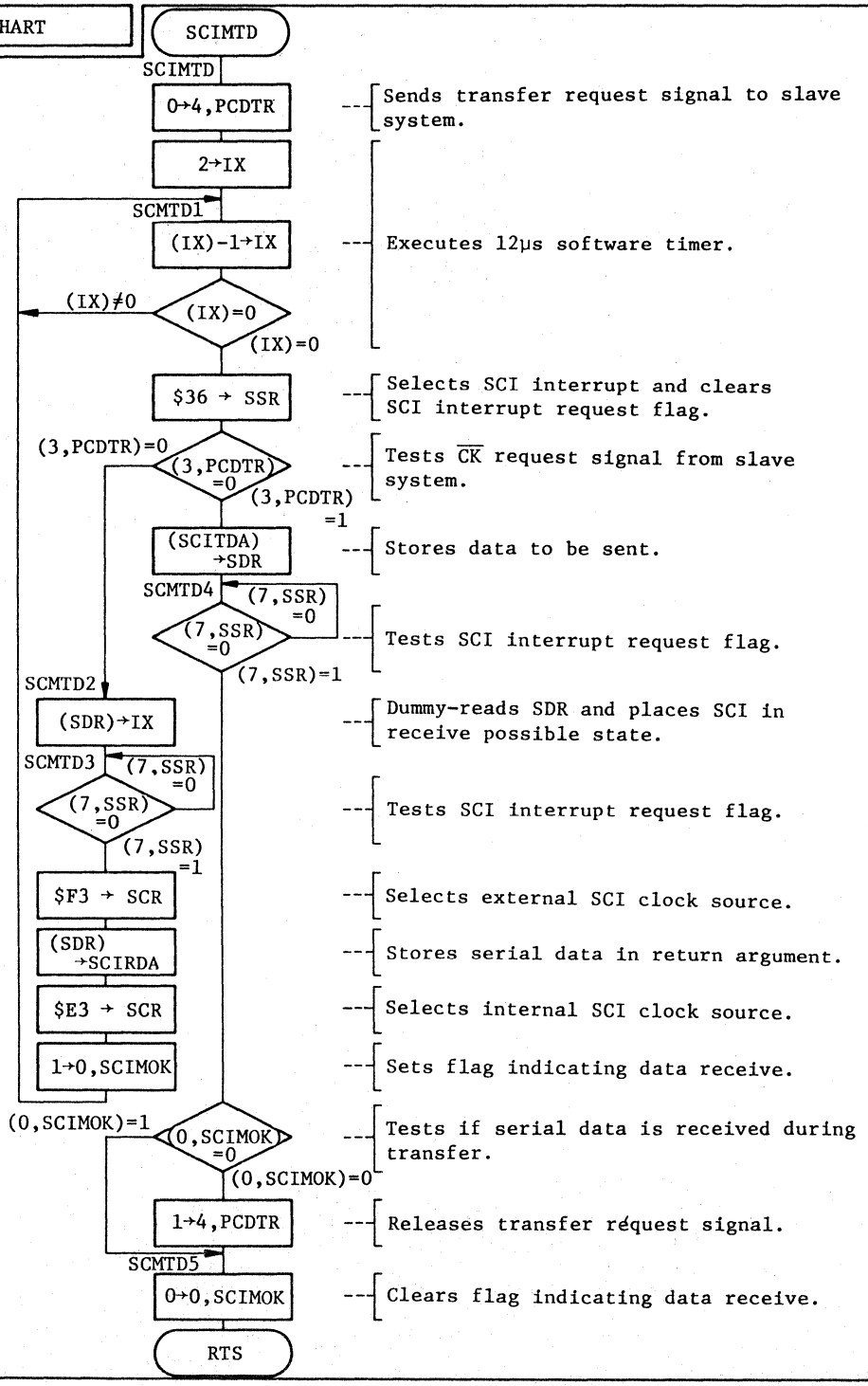
(b) Transfers received serial data if $\overline{CK}$ request signal is output.

(c) Goes to the next step after SCI interrupt request flag is set and transfer/receive completes.

(d) When serial data is received, retains transfer request in output state until main program processes received data so that next data can not be received.

PROGRAM MODULE NAME	SCI MASTER TRANSFER	MCU/MPU	HD6305X0/Y0	LABEL	SCIMTD

FLOWCHART



PROGRAM MODULE NAME	SCI MASTER RECEIVE	MCU/MPU	HD6305X0/Y0	LABEL	SCIMRD

FUNCTION

(1) Outputs serial clock and receives data from slave system.

(2) Permits outputting when slave system cannot output serial clock.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	—	—	—
	Returns	Received data	ACCA	1
		Existence of received data	Bit C (CCR)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	×

C	Z
●	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
36
RAM (Bytes)
0
Stack (Bytes)
0
No. of cycles
43
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details
ACCA: Contains data received from slave.

Bit C(CCR): Indicates existence of received data.

Bit C = 1 : No data is received from slave system.

Bit C = 0 : Data is received from slave system.

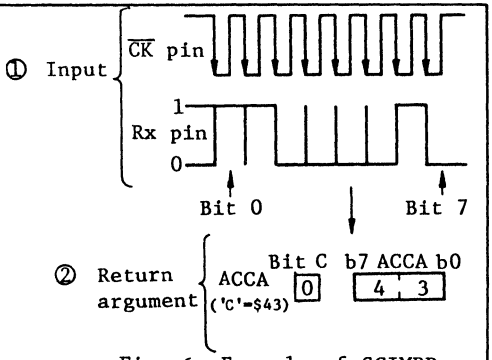


Fig. 6 Example of SCIMRD Execution

SPECIFICATIONS NOTES

PROGRAM MODULE NAME	SCI MASTER RECEIVE	MCU/MPU	HD6305X0/Y0	LABEL	SCIMRD

DESCRIPTION

(b) Program module SCIMRD execution contains SDR contents in return argument ACCA.

(c) Program module SCIMRD calls neither program modules nor subroutines.

(2) User Notes

(a) Initializes SCR.

(b) When program module SCIMRD is executed, set bit 3 and bit 4 of data register port C to "1".

(c) Goes to transfer possible state in slave system before program module SCIMRD execution.

(d) Program module SCIMRD loops until receiving data from slave system is completed.

(3) RAM Description

RAM is not used by program module SCIMRD

(4) Sample Application

Calls program module SCIMRD after selecting I/O port and initializing SCR.

```
            LDA     #$30    }
            STA     PCDTR   }--- Sets bit 3 and bit 4 of port C to High.
            STA     PCDDR   }
            LDA     #$E3    }---Selects internal clock source.
            STA     SCR     }
LOOP        JSR     SCIMRD  ---Calls program module SCIMRD.
            BCS     LOOP    ---Tests receive completion.
```

(5) Basic Operation

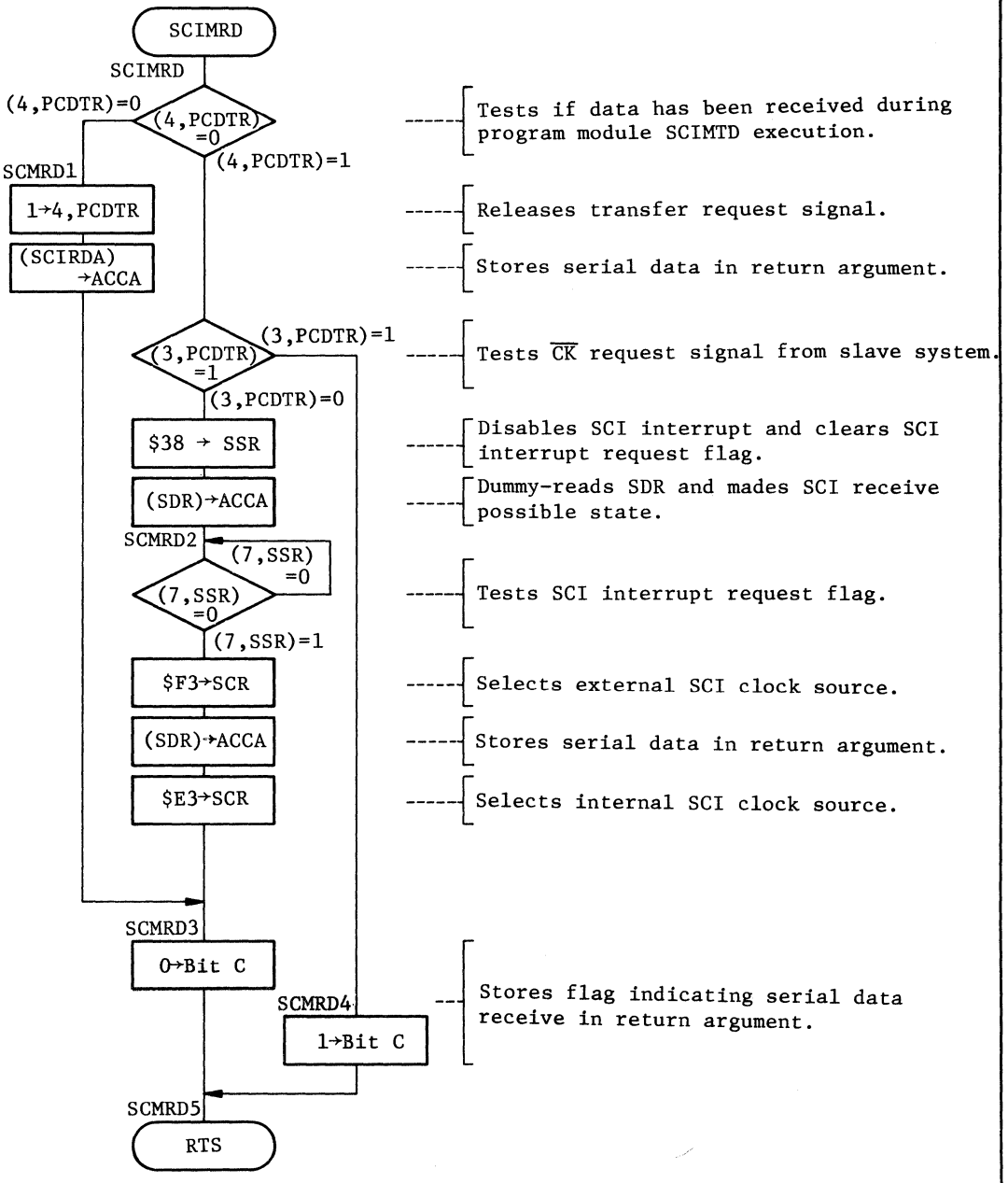
(a) Checks transfer request signal to test if data is received after outputting the signal.

(b) Tests if $\overline{CK}$ request signal has been output from slave system.

(c) If output, sets SCI interrupt request flag, and stores serial data in return argument after checking that serial data is received.

PROGRAM MODULE NAME	SCI MASTER RECEIVE	MCU/MPU	HD6305X0/Y0	LABEL	SCIMRD

FLOWCHART



12.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

12.5 PROGRAM LISTING

```
00001                   *
00002                   ****    RAM ALLOCATION    ************************
00003                   *
00004 0080                      ORG     $80
00005                   *
00006 0080 0001         SCIMOK  RMB     1         Indicate of receiving data
00007 0081 0001         SCIRDA  RMB     1         Received data
00008 0082 0001         SCITDA  RMB     1         SCI output data
00009                   *
00010                   ****    SYMBOL DEFINITIONS    *********************
00011                   *
00012       0001        PCDTR   EQU     $01       Port C data register
00013       0006        PCDDR   EQU     $06       Port C data direction register
00014       0010        SCR     EQU     $10       SCI control register
00015       0011        SSR     EQU     $11       SCI status register
00016       0012        SDR     EQU     $12       SCI data register
00017                   ************************************************
00018                   *                                              *
00019                   *             MAIN PROGRAM : SCIMMN            *
00020                   *                                              *
00021                   ************************************************
00022                   *
00023 1000                      ORG     $1000
00024                   *
00025 1000 A6 10        SCIMMN  LDA     #$10      Initialize port C
00026 1002 B7 01                STA     PCDTR
00027 1004 B7 06                STA     PCDDR     Select port C bit 4 as output
00028 1006 4F                   CLR     A         Initialize SCIMOK
00029 1007 B7 80                STA     SCIMOK
00030 1009 A6 E3                LDA     #$E3      Initialize SCR
00031 100B B7 10                STA     SCR
00032 100D A6 41        SCMMN1  LDA     #$41      Store output data
00033 100F B7 82                STA     SCITDA
00034 1011 CD 1043              JSR     SCIMTD    Output SCI data
00035 1014 CD 101F      SCMMN2  JSR     SCIMRD    Receive SCI data
00036 1017 25 FB                BCS     SCMMN2    Loop until receive data
00037 1019 A3 41                CPX     #$41      Test if Tx data=Rx data?
00038 101B 26 F0                BNE     SCMMN1    Branch if not equal
00039 101D 20 FE        PEND    BRA     PEND      End of program
00040                   ************************************************
00041                   *                                              *
00042                   *    NAME : SCIMRD (SCI MASTER RECEIVE)        *
00043                   *                                              *
00044                   ************************************************
00045                   *                                              *
00046                   *       ENTRY : NOTHING                        *
00047                   *     RETURNS : ACCA  (RECEIVED DATA)          *
00048                   *               CARRY (C=0;TRUE,C=1;FALSE)     *
00049                   *                                              *
00050                   ************************************************
00051 101F 09 01 18 SCIMRD BRCLR  4,PCDTR,SCMRD1  Branch if Rx=0
00052 1022 06 01 1C        BRSET  3,PCDTR,SCMRD4  Branch if CK=0
00053 1025 A6 38           LDA    #$38      Initialize SSR
00054 1027 B7 11           STA    SSR
00055 1029 B6 12           LDA    SDR       Execute dummy read
```

```
00056 102B 0F 11 FD SCMRD2 BRCLR  7,SSR,SCMRD2  Loop until received data
00057 102E A6 F3           LDA    #$F3      Select SCI as event input
00058 1030 B7 10           STA    SCR
00059 1032 B6 12           LDA    SDR       Load received data
00060 1034 AE E3           LDX    #$E3      Select SCI as no clock input
00061 1036 BF 10           STX    SCR
00062 1038 20 04           BRA    SCMRD3
00063 103A 18 01     SCMRD1 BSET   4,PCDTR   Set Rx=1
00064 103C B6 81           LDA    SCIRDA    Load SCI data
00065 103E 98        SCMRD3 CLC               Clear bit C
00066 103F 20 01           BRA    SCMRD5
00067 1041 99        SCMRD4 SEC               Set bit C
00068 1042 81        SCMRD5 RTS
00069                       ********************************************
00070                       *                                          *
00071                       *   NAME : SCIMTD (SCI MASTER TRANSFER)    *
00072                       *                                          *
00073                       ********************************************
00074                       *                                          *
00075                       *      ENTRY : SCITDA (TRANSFER DATA)      *
00076                       *    RETURNS : NOTHING                     *
00077                       *                                          *
00078                       ********************************************
00079 1043 19 01     SCIMTD BCLR   4,PCDTR   Set request low
00080 1045 AE 02           LDX    #2        Execute software timer
00081 1047 5A        SCMTD1 DEC    X
00082 1048 26 FD           BNE    SCMTD1
00083 104A A6 38           LDA    #$38      Mask SCI interrupt
00084 104C B7 11           STA    SSR
00085 104E 07 01 0F        BRCLR  3,PCDTR,SCMTD2  Branch if CK=0
00086 1051 B6 82           LDA    SCITDA    Store transfer data
00087 1053 B7 12           STA    SDR
00088 1055 0F 11 FD SCMTD4 BRCLR  7,SSR,SCMTD4  Loop until received data
00089 1058 00 80 02        BRSET  0,SCIMOK,SCMTD5  Set Tx=0
00090 105B 18 01           BSET   4,PCDTR   Set request high
00091 105D 11 80     SCMTD5 BCLR   0,SCIMOK Clear SCI receiving flag
00092 105F 81              RTS
00093 1060 BE 12     SCMTD2 LDX    SDR       Execute dummy read
00094 1062 0F 11 FD SCMTD3 BRCLR  7,SSR,SCMTD3  Loop until received data
00095 1065 A6 F3           LDA    #$F3      Select SCI as event input
00096 1067 B7 10           STA    SCR
00097 1069 B6 12           LDA    SDR       Load transfer data
00098 106B B7 81           STA    SCIRDA
00099 106D A6 E3           LDA    #$E3      Select SCI as no clock input
00100 106F B7 10           STA    SCR
00101 1071 10 80           BSET   0,SCIMOK Set SCI receiving flag
00102 1073 20 D2           BRA    SCMTD1
00103                       ********************************************
00104                       *                                          *
00105                       *              VECTOR ADDRESSES            *
00106                       *                                          *
00107                       ********************************************
00108                       *
00109 1FF6                  ORG    $1FF6
00110                       *
00111 1FF6 1000             FDB    SCIMMN    SCI/TIMER2
00112 1FF8 1000             FDB    SCIMMN    TIMER/INT2
00113 1FFA 1000             FDB    SCIMMN    INT
00114 1FFC 1000             FDB    SCIMMN    SWI
00115 1FFE 1000             FDB    SCIMMN    RES
00116                       END
```

13. LIQUID CRYSTAL DRIVER (HD61100A) CONTROL

13.1 HARDWARE DESCRIPTION

(1) Function

(a) Controls LCD driver HD61100A using the HD6305X0 to display numerals (from 0 to 9) on LCD.

(b) Liquid crystal displays with 8 segment × 10 digits using dynamic drive.

(c) Transfers 10 digits segment data from the HD6305X0 to HD61100A.

(2) Microcomputer Applications

(a) Port C controls HD61100A control signals (M and CL_1) and common signal of liquid crystal, and displays numerals on liquid crystal.

(b) Controls HD61100A control signals (CL_2 and DL) by using clocked synchronizing SCI within the HD6305X0 and transfers display data to HD61100A.

(3) Circuit Diagram

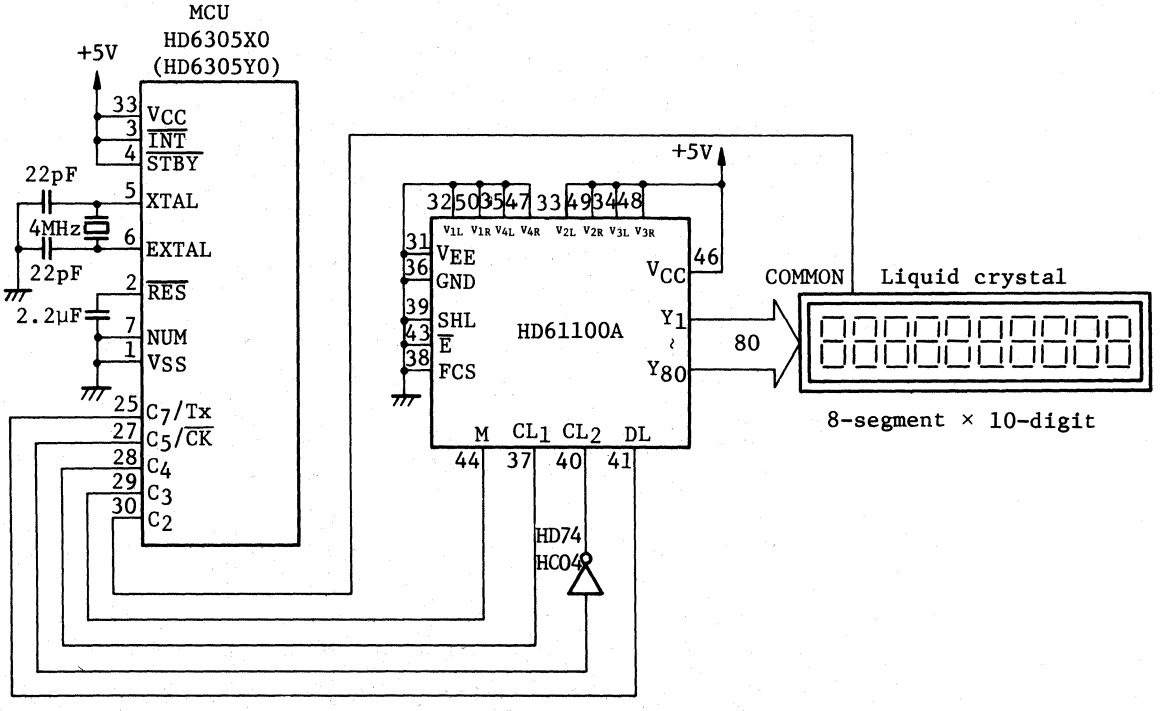


Fig. 1 HD61100A Control

(4) Pin Functions

Pin functions at the interface between the HD6305X0 and HD61100A are shown in Table 1.

Table 1 Pin Functions

Pin Name (HD6305X0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (HD61100A)	Program Label
C_3	Output	—	Alternate signal of LCD driving output.	M	PCDTR
C_4	Output	—	Outputs LCD driving signal at the falling edge.	CL_1	
$\overline{CK}$	Output	—	Triggers display data at the falling edge.	CL_2	
T_X	Output	—	Inputs display data.	DL	

(5) Hardware Operation

Control signals (M, CL_1, CL_2, DL) of HD61100A and LCD common signal are controlled by the HD6305X0 clocked synchronizing SCI and port. The timing chart of each signal is shown in Fig. 2.

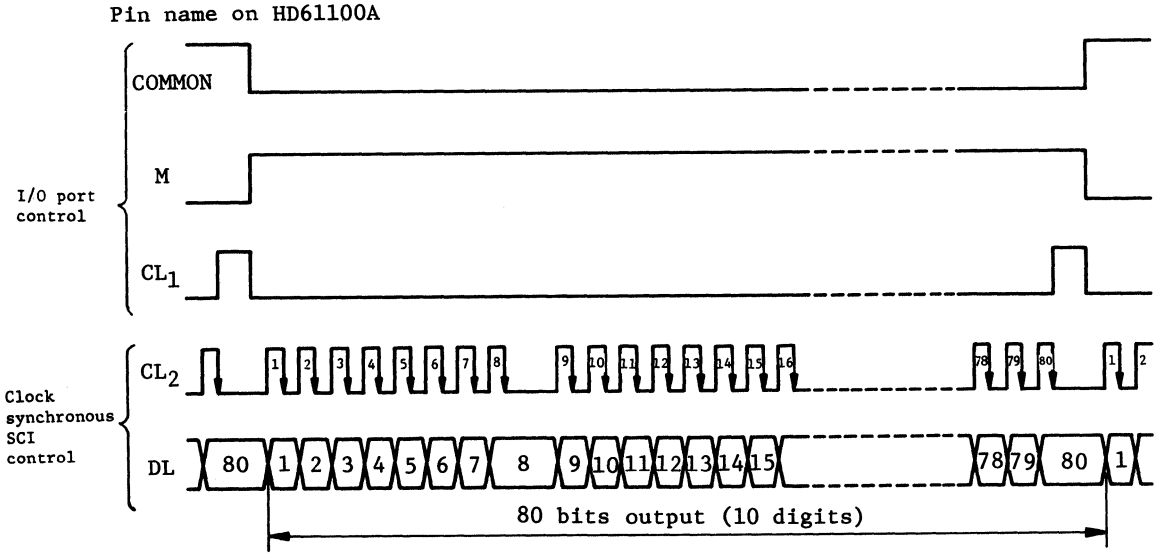


Fig. 2 HD6305X0 → HD61100A and LCD Interface

13.2 SOFTWARE DESCRIPTION

(1) Program Module Configuration

The program module configuration for character display on the liquid crystal module is shown in Fig. 3.

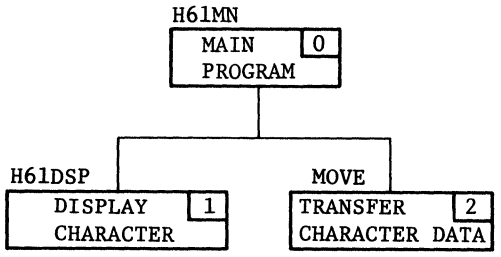


Fig. 3 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	H61MN	Displays 8 segment × 10 digits LCD.
1	DISPLAY CHARACTER	H61DSP	Transfers display data to HD61100A and displays on LCD.
2	TRANSFER CHARACTER DATA	MOVE	Stores display data in display RAM. See subroutine MOVE in HD6305 FAMILY APPLICATION NOTES (SOFTWARE) for details.

(3) Program Module Sample Application (Main Program)

The Flow chart in Fig. 4 is an example of character display on liquid crystal performed by the program module in Fig. 3. The main program in Fig. 4 demonstrates the display on liquid crystal as shown in Fig. 5.

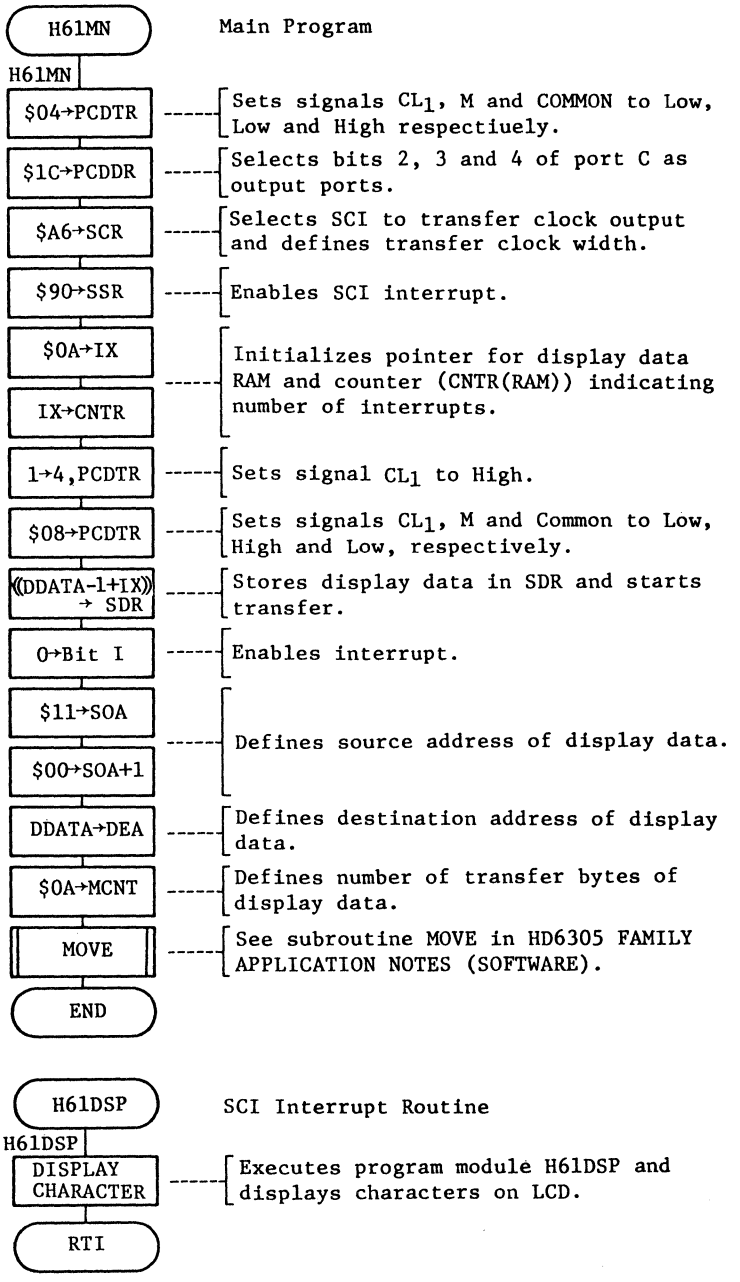


Fig. 4 Program Module Flowchart

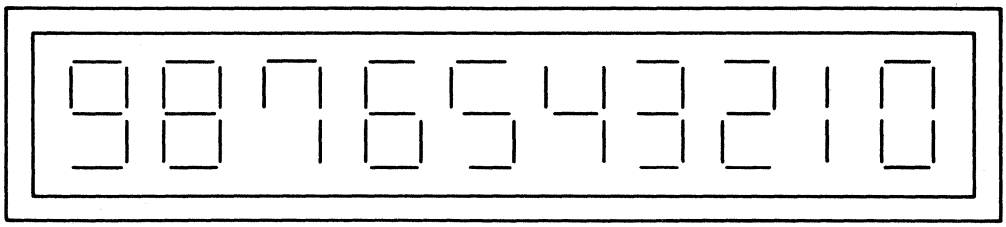


Fig. 5 LCD Example

PROGRAM MODULE NAME	DISPLAY CHARACTER	MCU/MPU	HD6305X0/Y0	LABEL	H61DSP

FUNCTION

Transfers display data to HD61100A and displays numerals on LCD.

ARGUMENTS

		Contents	Storage Location	Byte Lgth.
Arguments	Entry	Display data	DDATA (RAM)	10
	Returns	-	-	-

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	×

C	Z
×	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
30
RAM (Bytes)
11
Stack (Bytes)
5
No. of cycles
40
Reentrant
No
Relocation
No
Interrupt
No

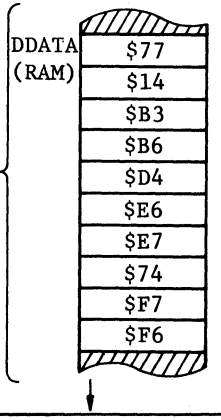
DESCRIPTION

(1) Function Details

(a) Argument details
DDATA: Holds 10-byte display data.

(b) Fig. 6 shows an example of program module H61DSP execution. If entry argument is held as shown in part ① of Fig. 6, display data is transferred to HD61100A and displays characters on LCD as shown in part ② of Fig. 6.

① Entry arguments Display data (9,8,7,6,5,4, 3,2,1,0)



SPECIFICATIONS NOTES

(c) Program module H61DSP calls neither program modules nor subroutines.

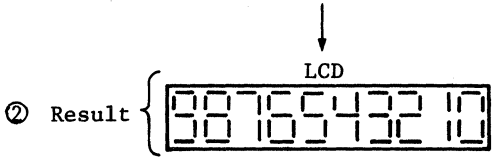


Fig. 6 Example of H61DSP Execution

(2) User Notes

(a) Selects DDR of port C as output.

(b) Initializes clock synchronous SCI for display data transfer.

(c) Clears bit I to enable SCI interrupt.

(d) Loads display data into SDR in main program to service SCI interrupt.

(3) RAM Description

Label	b7 RAM b0	Description
CNTR		Stores pointer for indicating display data and counter indicating number of interrupts.
DDATA	10^0 digit	Stores display data.
	10^1 digit	
	10^2 digit	
	10^3 digit	
	10^4 digit	
	10^5 digit	
	10^6 digit	
	10^7 digit	
	10^8 digit	
	10^9 digit	

PROGRAM MODULE NAME	DISPLAY CHARACTER	MCU/MPU	HD6305X0/Y0	LABEL	H61DSP

DESCRIPTION

(4) Sample Application

Program module H61DSP is called each time SDR becomes empty after selecting I/O port, initializing SCI, loading data into SDR and enabling interrupt.

```
        |
        |
        |
        LDA     #$04        }---- Initializes port C.
        STA     PCDTR       }
        LDA     #$1C        }
        STA     PCDDR       }---- Selects bit 2, 3 and 4 of port C as output.
        LDA     #$A6        }
        STA     SCR         }
        LDA     #$90        }---- Initializes SCI.
        STA     SSR         }
        LDX     #$0A        }
        STX     CNTR        }---- Initializes counter (CNTR).
        BSET    4,PCDTR     }
        LDA     #8          }
        STA     PCDTR       }---- Initializes port C.
        LDA     DDATA-1, X  }
        STA     SDR         }---- Stores display data in SCR.
        CLI                 ----- Enables interrupt.
        LDA     #$11        }
        STA     SOA         }
        LDA     #$00        }
        STA     SOA+1       }
        LDA     #$8A        }
        STA     DEA         }
        LDA     #$0A        }
        STA     MCNT        }
        BSR     MOVE        }---- Transfers display data to RAM using sub-
                                  routine MOVE.
                                  See subroutine MOVE in HD6305 FAMILY
                                  APPLICATION NOTES (SOFTWARE) for details.
        |
        |
```

(5) Basic Operation

(a) Transfers 10-byte display data to HD61100A shift-register for 8 segment × 10 digits LCD. Clocked synchronizing SCI of the HD6305X0 controls clock and data signal.

(b) Stores display data in 10-byte display RAM beforehand and outputs 1-byte display data by one SCI interrupt.

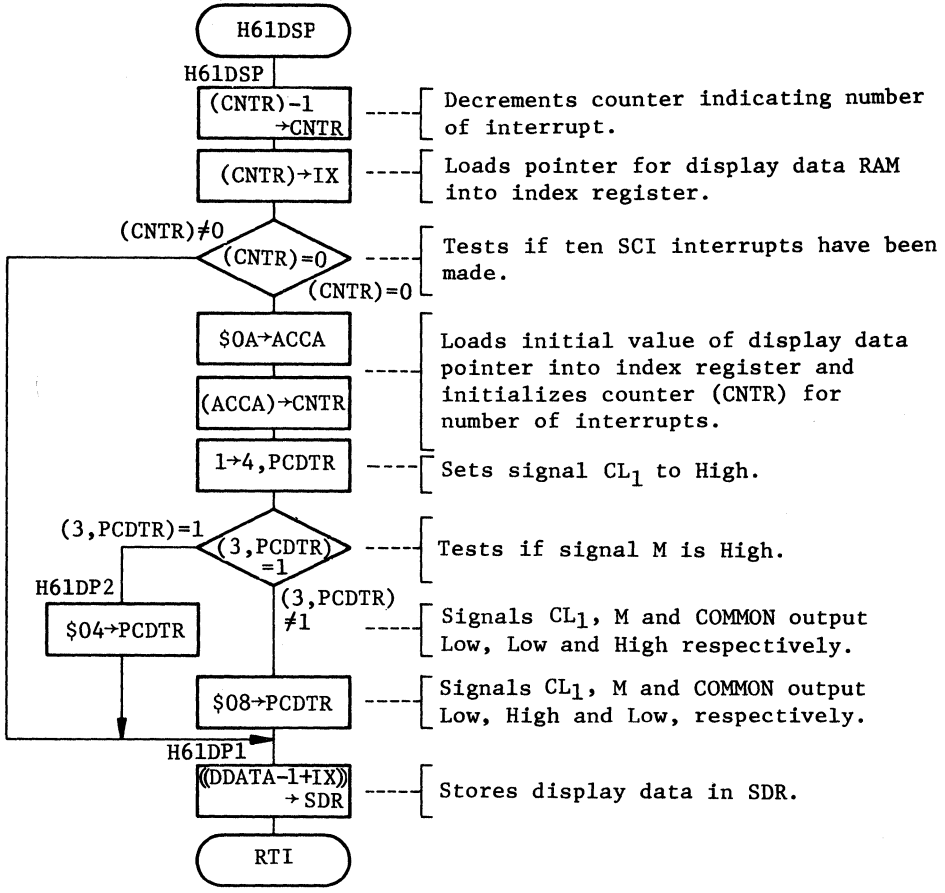
(c) Decrements display RAM pointer and counter CNTR(RAM) after each interrupt, and re-initializes CNTR(RAM) every ten interrupts.

(d) Executes only one SCI interrupt in main program, and then outputs display data with SCI interrupt automatically with each output.

7

PROGRAM MODULE NAME	DISPLAY CHARACTER	MCU/MPU	HD6305X0/Y0	LABEL	H61DSP

FLOWCHART



13.4 SUBROUTINE DESCRIPTION

This application example calls no subroutines.

13.5 PROGRAM LISTING

```
00001                    *
00002                    ********************RAM ALLOCATION*******************
00003                    *
00004                    *
00005 0080                       ORG     $80
00006                    *
00007 0080 000A          DDATA   RMB     10          Display RAM area
00008 008A 0001          CNTR    RMB     1           Counter of display data
00009 008B 0002          SOA     RMB     2           Source ADDR (MOVE)
00010 008D 0001          DEA     RMB     1           Destination ADDR (MOVE)
00011 008E 0001          MCNT    RMB     1           Transfer counter (MOVE)
00012 008F 0004          MSUB    RMB     4           Work area for subroutine (MOVE)
00013 0093 0001          SPNT    RMB     1           Relative data of source ADDR
00014                    *
00015                    ***************** SYMBOL DIFINITIONS ***************
00016                    *
00017       0002         PCDTR   EQU     $02         Port C data register
00018       0006         PCDDR   EQU     $06         Port C data direction register
00019       0010         SCR     EQU     $10         SCI control register
00020       0011         SSR     EQU     $11         SCI status register
00021       0012         SDR     EQU     $12         SCI data register
00022                    ************************************************
00023                    *                                              *
00024                    *           MAIN PROGRAM : H61MN               *
00025                    *                                              *
00026                    ************************************************
00027                    *
00028 1000                       ORG     $1000
00029                    *
00030 1000 A6 04         H61MN   LDA     #$04        Set CL1=0,M=0,COMMON=1
00031 1002 B7 02                 STA     PCDTR
00032 1004 A6 1C                 LDA     #$1C        Select port C bit 2-4 as output
00033 1006 B7 06                 STA     PCDDR
00034 1008 A6 A6                 LDA     #$A6        Initialize SCR
00035 100A B7 10                 STA     SCR         (data and clock output)
00036 100C A6 90                 LDA     #$90        Initialize SSR
00037 100E B7 11                 STA     SSR         (SCI interrupt enable)
00038 1010 AE 0A                 LDX     #$A         Initialize counter
00039 1012 BF 8A                 STX     CNTR
00040 1014 18 02                 BSET    4,PCDTR     Set CL1=1,M=0,COMMON=1
00041 1016 A6 08                 LDA     #$8         Set CL1=0,M=1,COMMON=0
00042 1018 B7 02                 STA     PCDTR
00043 101A E6 7F                 LDA     DDATA-1,X   Store display data in SDR
00044 101C B7 12                 STA     SDR
00045 101E 9A                    CLI                 Enable interrupt
00046 101F A6 11                 LDA     #$11        Store source address
00047 1021 B7 8B                 STA     SOA         (Source address=2 byte)
00048 1023 A6 00                 LDA     #$00
00049 1025 B7 8C                 STA     SOA+1
00050 1027 A6 80                 LDA     #DDATA      Store destination address
00051 1029 B7 8D                 STA     DEA
00052 102B A6 0A                 LDA     #$0A        Initialize transfer counter
00053 102D B7 8E                 STA     MCNT
00054 102F CD 1052               JSR     MOVE        Move data table to display RAM
00055 1032 20 FE         PEND    BRA     PEND        End of program
```

```
00056                   ***************************************************
00057                   *                                                 *
00058                   *       NAME : H61DSP (DISPLAY CHARACTER)         *
00059                   *                                                 *
00060                   ***************************************************
00061                   *                                                 *
00062                   *      ENTRY : DDATA (DISPLAY DATA)               *
00063                   *    RETURNS : NOTHING                            *
00064                   *                                                 *
00065                   ***************************************************
00066 1034 3A 8A       H61DSP DEC     CNTR      Decrement counter
00067 1036 BE 8A              LDX     CNTR      Test if CNTR=0?
00068 1038 26 0D              BNE     H61DP1    Branch if not CNTR=0
00069 103A A6 0A              LDA     #$0A      Initialize counter
00070 103C B7 8A              STA     CNTR
00071 103E 18 02              BSET    4,PCDTR   Set CL1=1
00072 1040 06 02 09           BRSET   3,PCDTR,H61DP2  Branch if M=1
00073 1043 A6 08              LDA     #8        Set CL1=0,M=1,COMMON=0
00074 1045 B7 02              STA     PCDTR
00075 1047 E6 7F       H61DP1 LDA     DDATA-1,X  Store display data in SDR
00076 1049 B7 12              STA     SDR
00077 104B 80                 RTI
00078 104C A6 04       H61DP2 LDA     #4        Set CL1=0,M=0,COMMON=1
00079 104E B7 02              STA     PCDTR
00080 1050 20 F5              BRA     H61DP1    Branch to H61DP1
00081                   ***************************************************
00082                   *                                                 *
00083                   *       NAME : MOVE (MOVING MEMORY BLOCKS)        *
00084                   *                                                 *
00085                   ***************************************************
00086                   *                                                 *
00087                   *      ENTRY : SOA  (SOURCE ADDR)                 *
00088                   *              DEA  (DESTINATION ADDR)            *
00089                   *              MCNT (TRANSFER COUNTER)            *
00090                   *    RETURNS : NOTHING                            *
00091                   *                                                 *
00092                   ***************************************************
00093 1052 A6 D6       MOVE   LDA     #$D6      Load instruction code
                                                 (LDA Disp,X)
00094 1054 B7 8F              STA     MSUB
00095 1056 B6 8B              LDA     SOA       Load source ADDR (H)
00096 1058 B7 90              STA     MSUB+1
00097 105A B6 8C              LDA     SOA+1     Load source ADDR (L)
00098 105C B7 91              STA     MSUB+2
00099 105E A6 81              LDA     #$81      Load instruction code (RTS)
00100 1060 B7 92              STA     MSUB+3
00101 1062 3F 93              CLR     SPNT      Clear relative data of source
                                                 ADDR
00102 1064 BE 93       MOVE1  LDX     SPNT      Load relative data of source
                                                 ADDR
00103 1066 BD 8F              JSR     MSUB      Load transfer data
00104 1068 BE 8D              LDX     DEA       Load destination ADDR
00105 106A F7                 STA     0,X       Store transfer data
00106 106B 3C 8D              INC     DEA       Increment destination ADDR
00107 106D 3C 93              INC     SPNT      Increment relative data of
                                                 source ADDR
00108 106F 3A 8E              DEC     MCNT      Decrement transfer counter
00109 1071 26 F1              BNE     MOVE1     Branch until transfer counter=0
00110 1073 81                 RTS
```

```
00111                   ************************************************
00112                   *                                              *
00113                   *              DATA TABLE                      *
00114                   *                                              *
00115                   ************************************************
00116                   *
00117 1100                        ORG     $1100
00118                   *
00119 1100 14                     FCB     $14,$B3,$B6,$D4,$E6  *Segment data
00120 1105 E7                     FCB     $E7,$74,$F7,$F6,$77
00121                   ************************************************
00122                   *                                              *
00123                   *       VECTOR ADDRESSES                       *
00124                   *                                              *
00125                   ************************************************
00126                   *
00127 1FF6                        ORG     $1FF6
00128                   *
00129 1FF6 1034                   FDB     H61DSP    SCI/TIMER2
00130 1FF8 1000                   FDB     H61MN     TIMER/INT2
00131 1FFA 1000                   FDB     H61MN     INT
00132 1FFC 1000                   FDB     H61MN     SWI
00133 1FFE 1000                   FDB     H61MN     RES
00134                   *
00135                             END
```

14. EXTERNAL EXPANSION

14.1 HARDWARE DESCRIPTION

(1) Function

(a) Controls externally expanded memory and peripheral LSIs using HD6305Y2.

(b) Performs asynchronous serial interface with a console typewriter using the HD6350 (hereinafter, ACIA).

(c) Controls liquid crystal module (hereinafter, H2571) through the HD6321 (hereinafter, PIA) and displays inputs from a console typewriter in character mode.

(2) Microcomputer Applications

Interfaces with externally expanded LSIs through address buses, data buses and control signals (R/$\overline{W}$, E) using the HD6305Y2 externally expanded functions.

(3) Circuit Diagram

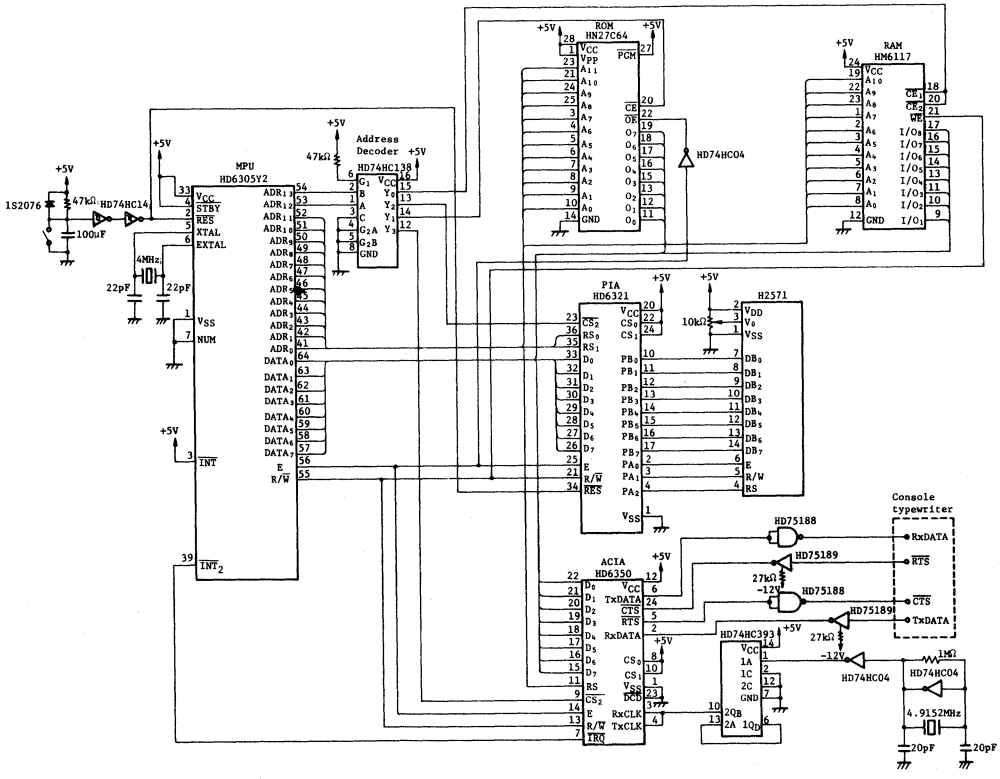


Fig. 1 External Expansion Circuit

(4) Memory Map

(a) Memories or peripheral LSIs are allocated in external memory space using address decoder (HD74HC138). ADR_{12} and ADR_{13} are connected to the HD74HC138 A and B pins. 16k byte memory space (from $0140 to $3FFF) are divided into 4 (4k bytes) and allocated. Table 1 shows system address decode.

Table 1 System Address Decode

HD74HC138										Address	Allocation
Input						Output					
G_1	G_2A	G_2B	C	B	A	Y_0	Y_1	Y_2	Y_3		
			–	ADR_{13}	ADR_{12}						
H	L	L	L	L	L	L	H	H	H	$0000 - $0FFF	RAM
H	L	L	L	L	H	H	L	H	H	$1000 - $1FFF	ROM
H	L	L	L	H	L	H	H	L	H	$2000 - $2FFF	PIA
H	L	L	L	H	H	H	H	H	L	$3000 - $3FFF	ACIA

(b) Fig. 2 shows system memory map

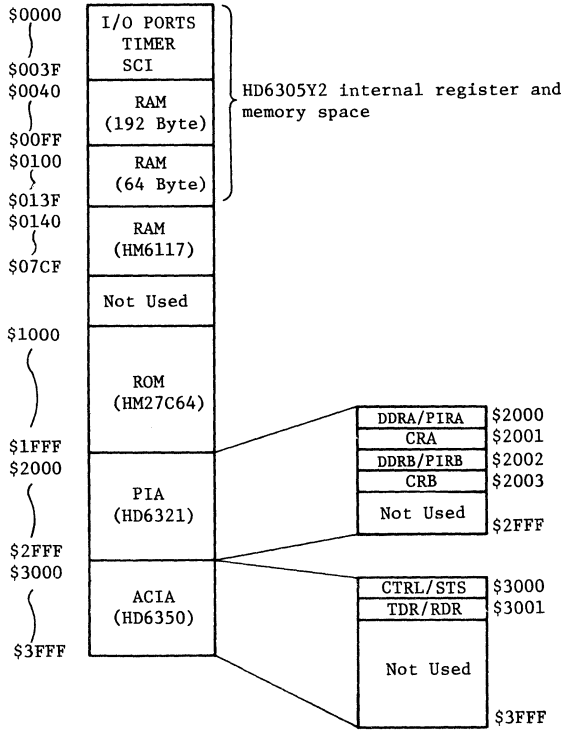
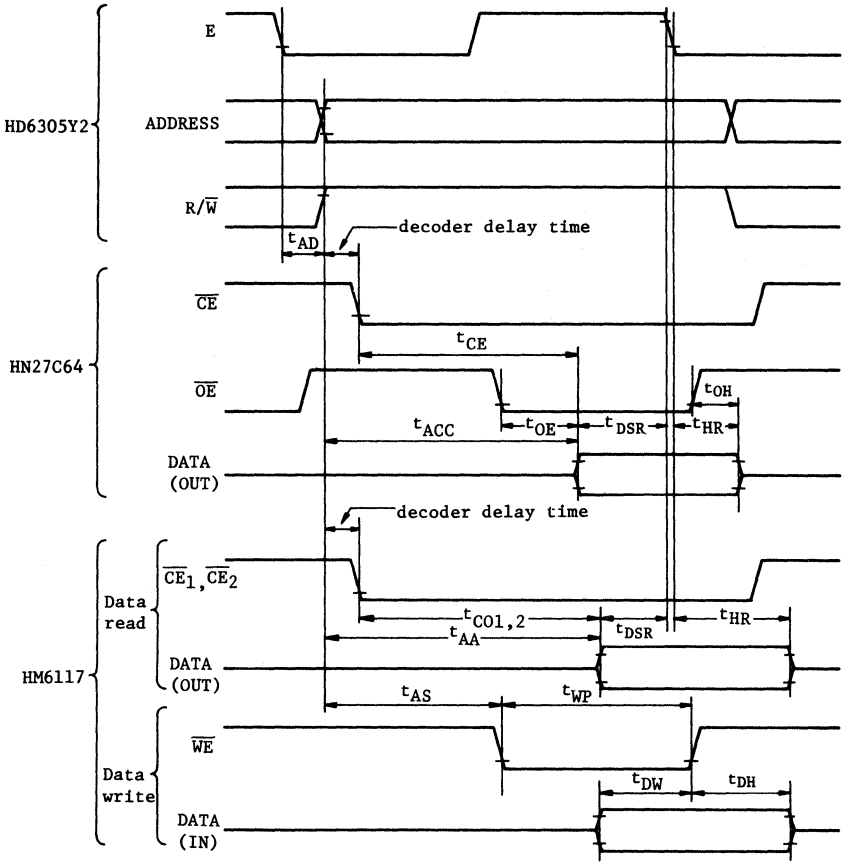


Fig. 2 System Memory Map

(5) Hardware Operation

Fig. 3 shows timing charts of the HD6305Y2 and memories (HN27C64, HM6117).



HD6305Y2	t_{AD}: Address delay time t_{DSR}: Data set-up time t_{HR}: Data hold time t_{DW}: Data delay time
HN27C64	t_{CE}: $\overline{CE}$ Output delay time t_{OE}: $\overline{OE}$ Output delay time t_{ACC}: Access time t_{OH}: Data output hold time
HM6117	t_{AA}: Address access time $t_{CO1,2}$: $\overline{CE_1}$, $\overline{CE_2}$ Output delay time t_{AS}: Address set-up time t_{WP}: Write pulse width t_{DW}: Input data set time t_{DH}: Input data hold time

Fig. 3 HD6305Y2/Memories Interface

(1) Program Module Configuration

The program module configuration to display data input from a console typewriter on liquid crystal display system in Fig. 1 is shown in Fig. 4.

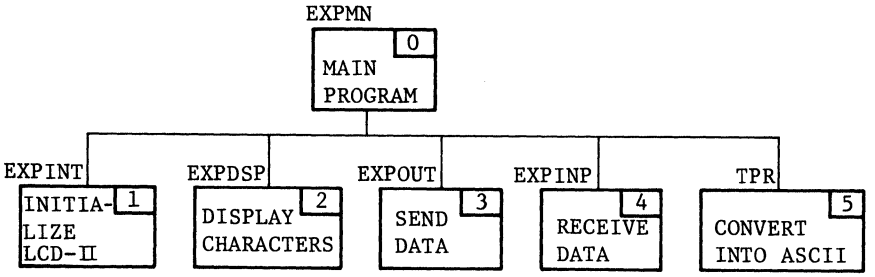


Fig. 4 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	EXPMN	Displays data input from console typewriter on console typewriter and liquid crystal display.
1	INITIALIZE LCD-II	EXPINT	Initializes LCD-II.
2	DISPLAY CHARACTERS	EXPDSP	Displays characters on liquid crystal.
3	SEND DATA	EXPOUT	Sends data to console typewriter.
4	RECEIVE DATA	EXPINP	Receives data from the console typewriter.
5	CONVERT INTO ASCII	TPR	Converts ASCII lowercase into ASCII uppercase. See subroutine TPR in HD6305 SERIES APPLICATION NOTES (SOFTWARE) for details.

(3) Program Module Sample Application (Main Program)

The flowchart in Fig. 5 is an example of displaying data input from console typewriter on liquid crystal display and console typewriter, performed by the program module in Fig. 4.

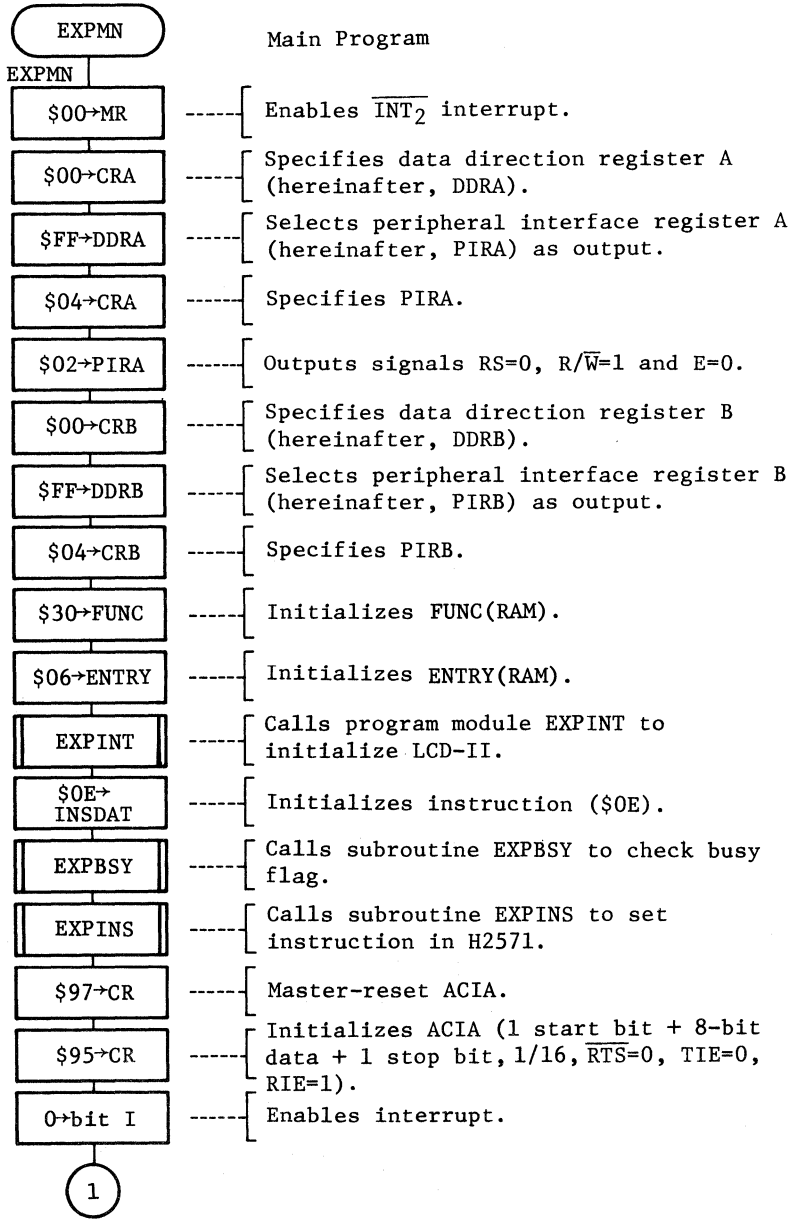


Fig. 5 Program Module Flowchart

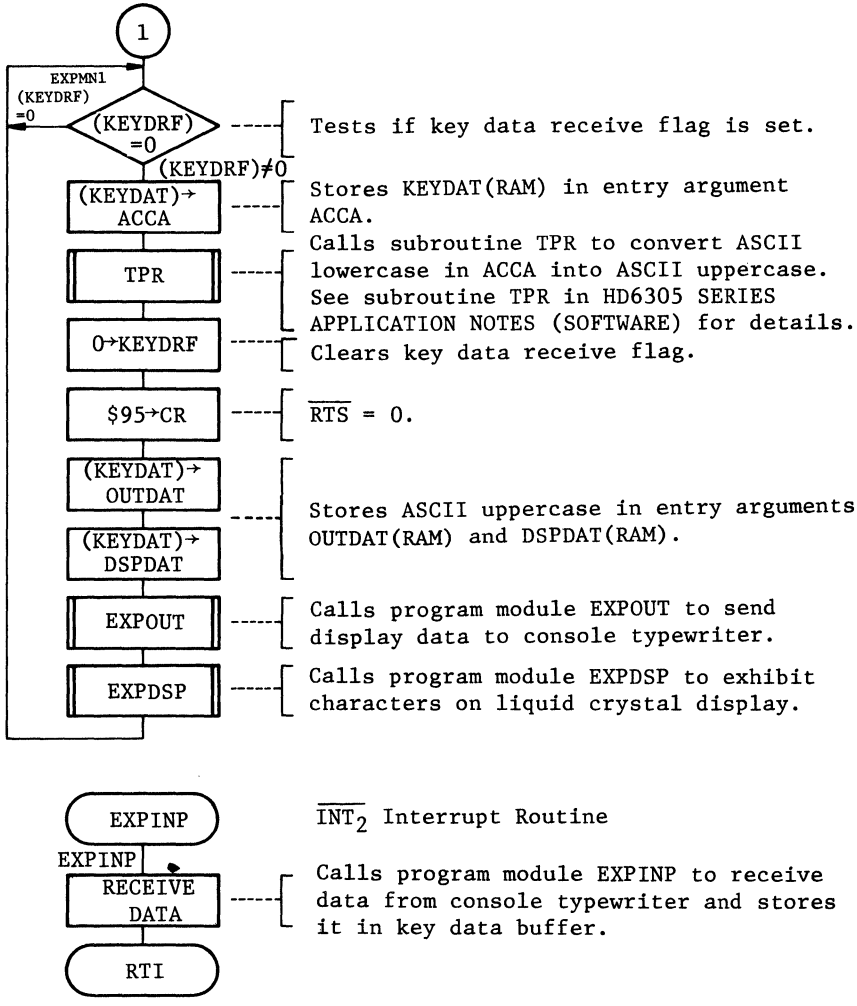


Fig. 5 (Cont.) Program Module Flowchart

PROGRAM MODULE NAME	INITIALIZE LCD-II	MCU/MPU	HD6305Y2	LABEL	EXPINT

FUNCTION

Initializes LCD-II contained in H2571.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Function initialization data($30)	FUNC (RAM)	1
		Entry mode initialization data($06)	ENTRY (RAM)	1
	Returns	-	-	-

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	×

C	Z
×	×
N	I
×	●
H	
●	

SPECIFICATIONS

ROM (Bytes)
138
RAM (Bytes)
5
Stack (Bytes)
2
No. of cycles
54435
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details
FUNC(RAM) : Holds function initialization data ($30).
ENTRY(RAM) : Holds entry mode initialization data ($06).

(b) Program module EXPINT initializes LCD-II with instructions as follows;

Display: Clear, Interface data length: 8 bits,
Display line: 1, Character font: 5 × 7 dots,
Duty ratio: 1/8, DDRAM address: Increment,
Display shift: No.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles required when subroutine EXPBSY is executed at the minimum cycles.

(c) Program module EXPINT calls other program modules or subroutines shown in Table 3.

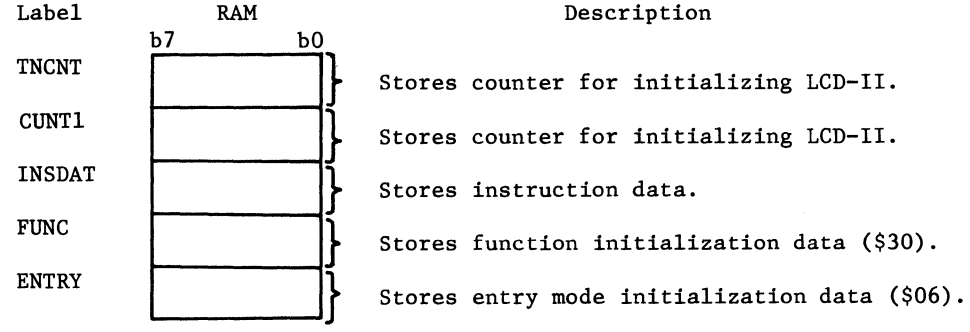
Table 3 Program Modules and Subroutines called in EXPINT

Program module/subroutine	Label	Functions
CHECK BUSY FLAG	EXPBSY	Checks LCD-II busy flag.
INSTRUCTION SET	EXPINS	Sets instructions in LCD-II.

(2) User Notes

Program module EXPINT cannot be called without initializing PIA.

(3) RAM Description



PROGRAM MODULE NAME	INITIALIZE LCD-II	MCU/MPU	HD6305Y2	LABEL	EXPINT

DESCRIPTION

(4) Sample Application

After initializing PIA and holding entry arguments, call program module EXPINT.

```
        LDA     #$00
        STA     CRA
        LDA     #$FF
        STA     DDRA
        LDA     #$04
        STA     CRA
        LDA     #$02        ----- Initializes PIA.
        STA     PIRA
        LDA     #$00
        STA     CRB
        LDA     #$FF
        STA     DDRB
        LDA     #$04
        STA     CRB
        LDA     #$30
        STA     FUNC        ----- Stores data in entry arguments FUNC(RAM) and ENTRY(RAM).
        LDA     #$06
        STA     ENTRY
        JSR     EXPINT      ----- Calls program module EXPINT.
```

(5) Basic Operation

(a) Control of PIA is shown in Fig. 6 and Fig. 7.

In Fig. 6, data ($80) is output from port A; in Fig. 7, data ($80) is input to port A.
Note that this controlling method is applied to the system in Fig. 1, and memory map in Fig. 2.

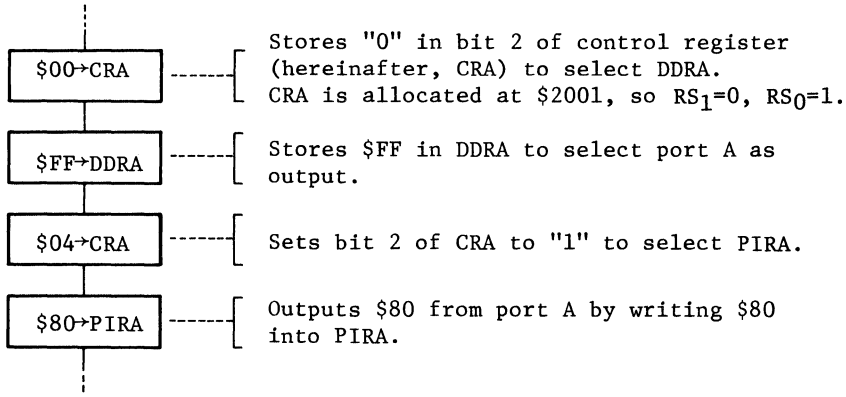


Fig. 6 Control of PIA (Port A: Output Port)

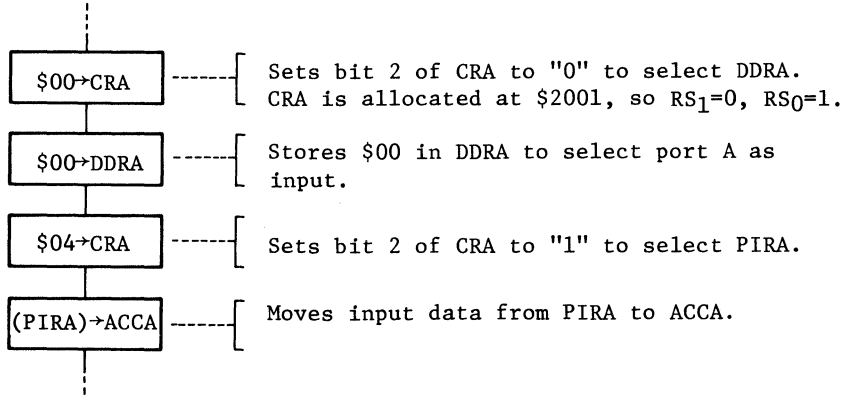
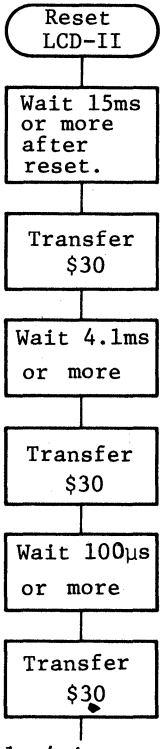


Fig. 7 Control of PIA (Port A: Input Port)

7

(b) Function initialization data ($30) must be written 3 times into LCD-II as shown below to ensure LCD-II internal RESET. Afterwards, LCD-II busy flag can be checked to select functions.



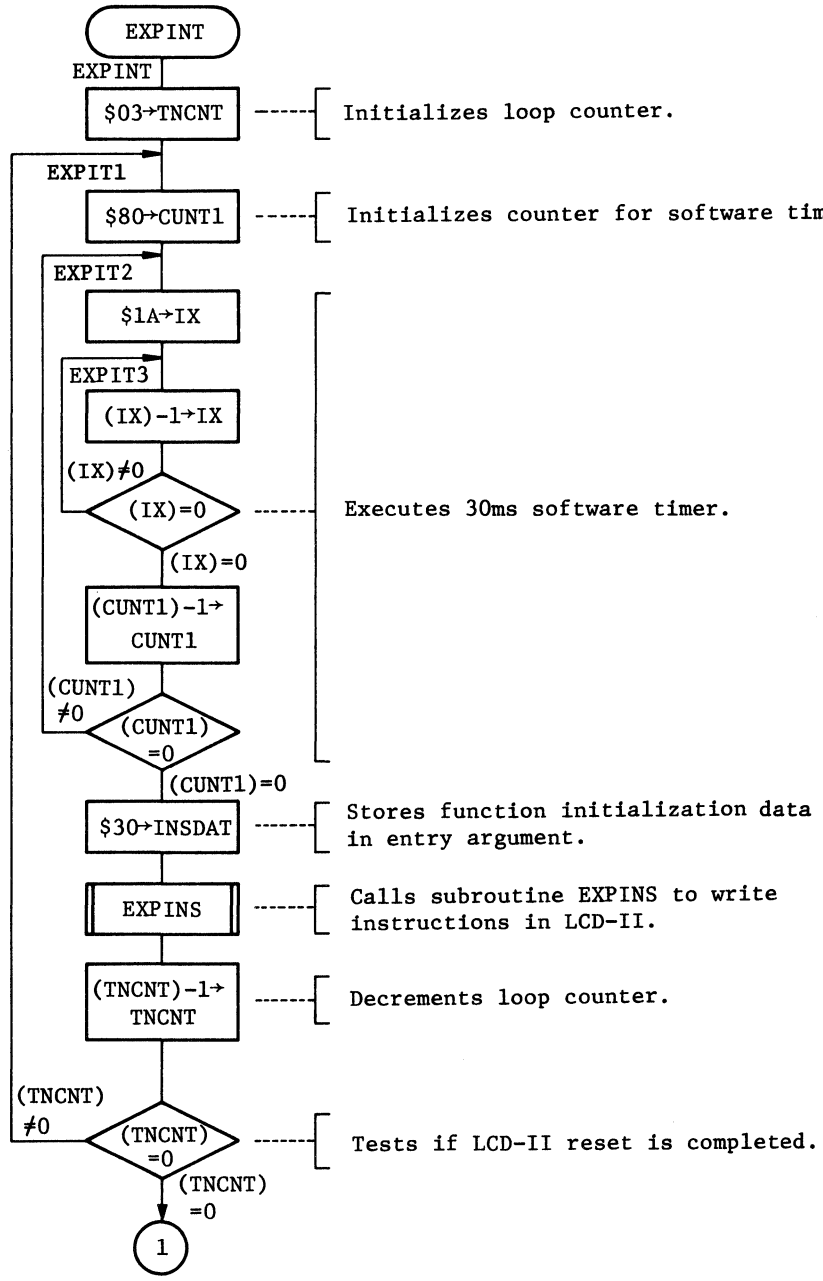
(c) Data in Table 4 is transferred to LCD-II.

Table 4 LCD-II Initialization Data

Data	Function
$30	Sets interface length to 8 bits.
$08	Turns off all display.
$01	Clears all display. Sets DDRAM address to $00
$06	Specifies cursor direction right. Does not shift display.

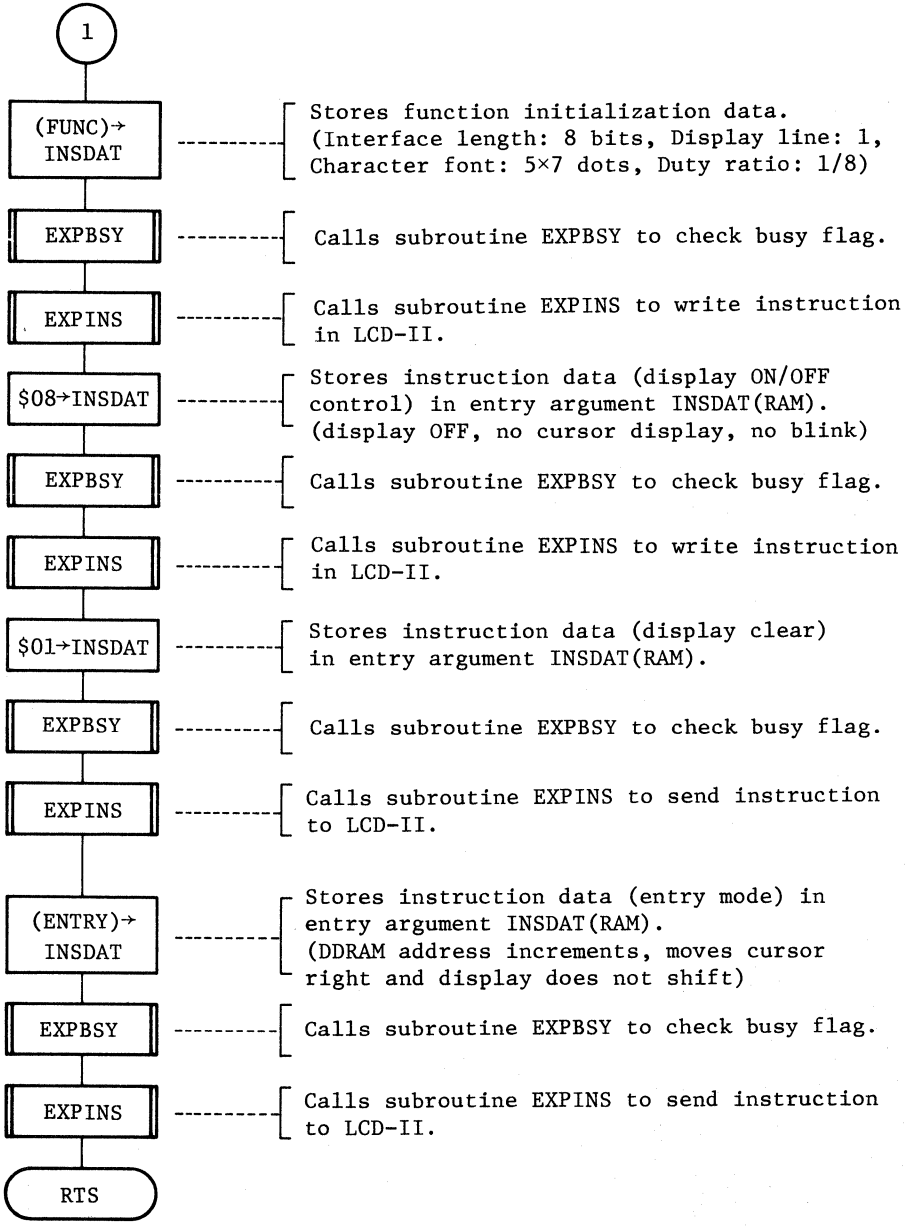
PROGRAM MODULE NAME	INITIALIZE LCD-II	MCU/MPU	HD6305Y2	LABEL	EXPINT

FLOWCHART



PROGRAM MODULE NAME	INITIALIZE LCD-II	MCU/MPU	HD6305Y2	LABEL	EXPINT

FLOWCHART



PROGRAM MODULE NAME	DISPLAY CHARACTERS	MCU/MPU	HD6305Y2	LABEL	EXPDSP

FUNCTION

Outputs display data to LCD-II and displays it on liquid crystal display.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Display data	DSPDAT (RAM)	1
	Returns	-	-	-

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
×	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
80
RAM (Bytes)
1
Stack (Bytes)
2
No. of cycles
113
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

DSPDAT(RAM): Holds data for characters to be displayed.

(b) Fig. 8 shows an example of program module EXPDSP execution.
If display data is held in DSPDAT(RAM), it is also stored in the current address in LCD-II DDRAM and display characters on liquid crystal display.

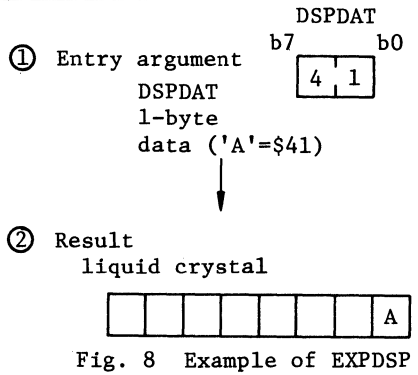


Fig. 8 Example of EXPDSP Execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles required when subroutine EXPBSY is executed by the minimum cycles.

(c) Program module EXPDSP calls other program modules or subroutines as shown in Table 5.

Table 5 Program Modules or Subroutines Called in EXPDSP.

Program modules/Subroutines	Label	Functions
CHECK BUSY FLAG	EXPBSY	Checks busy flags in LCD-II.

(2) User Notes

(a) Initializes PIA to control PIA in external expansion and to control LCD-II using PIA port.

(b) Initializes LCD-II to display characters after calling program module EXPINT.

(3) RAM Description

Label	RAM	Description
	b7 b0	
DSPDAT	[ ]	Stores display data in ASCII.

DESCRIPTION

(4) Sample Application

Program module EXPDSP is called after initializing PIA, LCD-II, and holding entry argument.

```
        LDA     #$00    ┐
        STA     CRA     │
        LDA     #$FF    │
        STA     DDRA    │
        LDA     #$04    │
        STA     CRA     │
        LDA     #$02    │
        STA     PIRA    ├──────  Initializes PIA.
        LDA     #$00    │
        STA     CRB     │
        LDA     #$FF    │
        STA     DDRB    │
        LDA     #$04    │
        STA     CRB     ┘
        LDA     #$30    ┐
        STA     FUNC    │
        LDA     #$06    │
        STA     ENTRY   │
        BSR     EXPINT  ├──────  Initializes LCD-II.
        LDA     #$0E    │
        STA     INSDAT  │
        BSR     EXPBSY  │
        JSR     EXPINS  ┘
        LDA     #$41    ┐
        STA     DSPDAT  ├──────  Stores display data in entry argument.
                        ┘
        JSR     EXPDSP  ──────  Call program module EXPDSP.
```

(5) Basic Operation

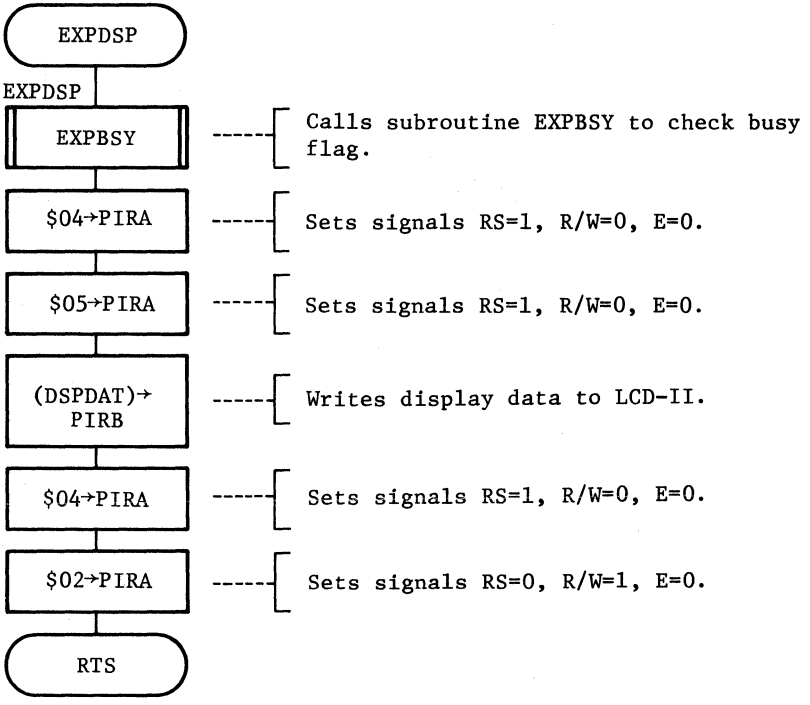
(a) Checks busy flag and waits till LCD-II can receive instructions.

(b) When LCD-II can receive instructions, makes signals RS=1, R/W=0, E=1 and stores display data into LCD-II DDRAM to display characters on liquid crystal display.

7

PROGRAM MODULE NAME	DISPLAY CHARACTERS	MCU/MPU	HD6305Y2	LABEL	EXPDSP

FLOWCHART



PROGRAM MODULE NAME	SEND DATA	MCU/MPU	HD6305Y2	LABEL	EXPOUT

FUNCTION

Sends character data to console typewriter.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	Character data	OUTDAT (RAM)	1
	Returns	-	-	-

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
●	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
13
RAM (Bytes)
1
Stack (Bytes)
0
No. of cycles
21
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Argument details

OUTDAT (RAM): Holds character data to be output to console typewriter.

(b) Fig. 9 shows an example of program module EXPOUT execution. Data in OUTDAT(RAM) is loaded into TDR in ACIA and transfer data.

① Before execution

Entry argument { OUTDAT, 1-byte data ('A'=$41)

② After execution
TDR in ACIA ('A'=$41)

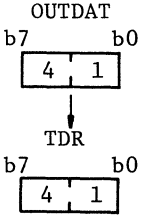


Fig. 9 Example of EXPOUT Execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles required when there is no data in TDR.

PROGRAM MODULE NAME	SEND DATA	MCU/MPU	HD6305Y2	LABEL	EXPOUT

DESCRIPTION

(c) Program module EXPOUT calls neither program modules nor subroutines.

(2) User Notes

(a) Main program does not be returned from program module EXPOUT until transmit data register (TDR) becomes empty to prevent data destruction when previous data is stored in TDR.

(b) Calls program module EXPOUT after initializing ACIA and holding entry argument in OUTDAT(RAM).

(3) RAM Description

Label	RAM	Description
	b7 b0	
OUTDAT	[ ]	Stores character data to be output to console typewriter.

(4) Sample Application

Program module EXPOUT is called after enabling $\overline{INT_2}$ interrupt, initializing ACIA, initializing key buffer and enabling interrupt.

```
        :
LDA     #$00    }
STA     MR      }  ------- Enables INT2 interrupt.
LDA     #$97    }
STA     CR      }
LDA     #$95    }  ------- Initializes ACIA.
STA     CR      }
CLI             ------- Enables interrupts.

LDA     KEYDAT  }
                }  ------- Store data in KEYDAT(RAM) in entry argument
STA     OUTDAT  }          OUTDAT(RAM).

JSR     EXPOUT  ------ Calls program module EXPOUT.
        :
```

(5) Basic Operation

(a) Control of ACIA for sending data is shown in Fig. 10.

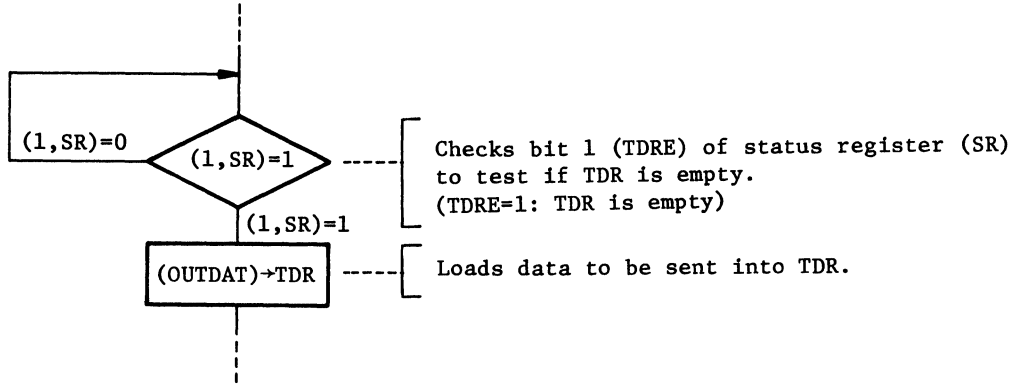


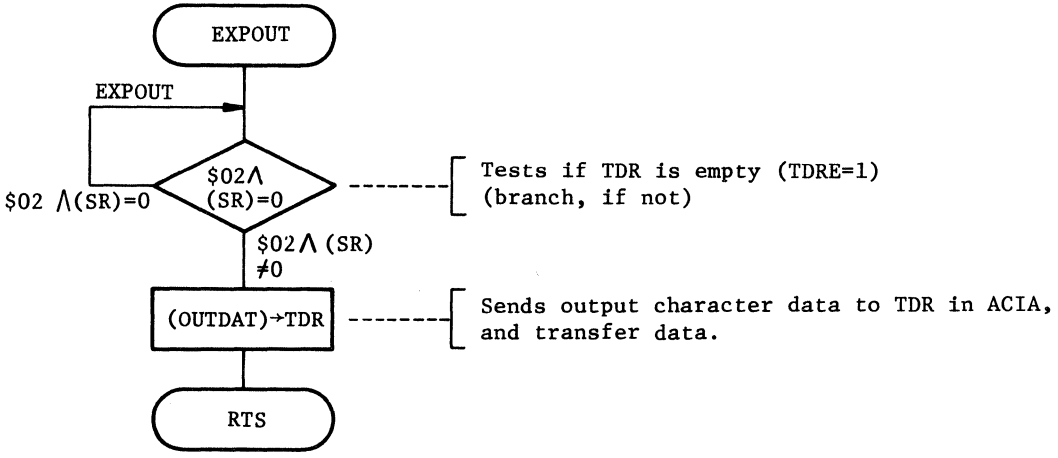
Fig. 10 Control of ACIA (sending serial data)

(b) If (TDRE)≠1, data remains in TDR in ACIA and program waits till (TDRE)=1.

(c) If (TDRE)=1, send character data in entry argument OUTDAT(RAM) to TDR in ACIA, and transfer data.

PROGRAM MODULE NAME	SEND DATA	MCU/MPU	HD6305Y2	LABEL	EXPOUT

FLOWCHART



PROGRAM MODULE NAME	RECEIVE DATA	MCU/MPU	HD6305Y2	LABEL	EXPINP

FUNCTION

Receives data from console typewriter and stores key data in key data buffer.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	-	-	-
	Returns	Received data	KEYDAT (RAM)	1
		Receive flag	KEYDRF (RAM)	1

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
●	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
29
RAM (Bytes)
1
Stack (Bytes)
0
No. of cycles
46
Reentrant
No
Relocation
No
Interrupt
No

DESCRIPTION

(1) Function Details

(a) Argument details

KEYDAT(RAM): Contains 1-byte key data from console typewriter.

KEYDRF(RAM): Indicates key data is received. Table 6 shows flag functions.

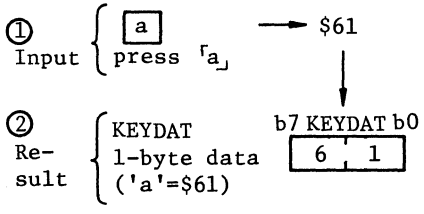


Fig. 11 Example of EXPINP Execution

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles required when data is set in RDR of ACIA.

Table 6 Flag Functions

Label	Contents	Function
KEYDRF (RAM)	$00	Indicates key data has not yet received.
	$FF	Indicates key data is received.

(b) Fig. 11 shows an example of program module EXPINP execution. If key "a" in console typewriter is pressed as shown in part ① of Fig. 11, received data is stored in key data buffer as showr in part ② of Fig. 11.

(c) Program module EXPINP calls neither program modules nor subroutines.

(2) User Notes

(a) Initializes ACIA to receive data from console typewriter by controlling ACIA in external expansion.

(b) Enables $\overline{INT_2}$ interrupt.

(c) Clears bit I to enable interrupt.

(3) RAM Description

Label	RAM	Description
	b7 b0	
KEYDAT		Used as key data buffer for received data.

(4) Sample Application

Calls program module EXPINP by $\overline{INT_2}$ interrupt after initializing ACIA, enabling $\overline{INT_2}$ interrupt and interrupt.

```
        :
        :
LDA     #$00  }
STA     MR    }  ---- Enables INT2 interrupt.
LDA     #$97  }
STA     CR    }
LDA     #$95  }  ---- Initializes ACIA.
STA     CR    }
CLI           ------ Enables interrupt.
        :
        :
```

PROGRAM MODULE NAME	RECEIVE DATA	MCU/MPU	HD6305Y2	LABEL	EXPINP

DESCRIPTION

(5) Basic Operation

(a) Control of ACIA is shown in Fig. 12 and Fig. 13. Fig. 12 shows initializing ACIA; Fig. 13 shows reading received data using interrupt. Note that this controlling method is applied to the system in Fig. 1 and memory map in Fig. 2.

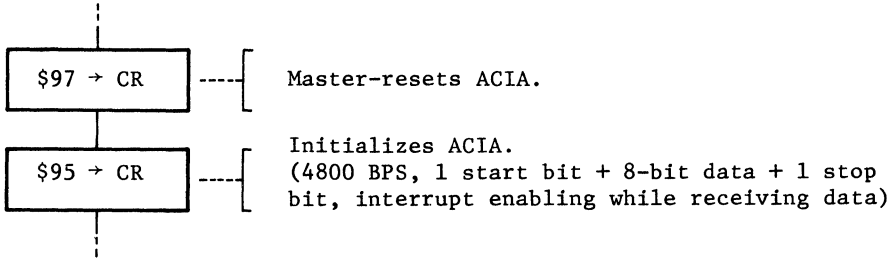


Fig. 12 Control of ACIA (Initialization)

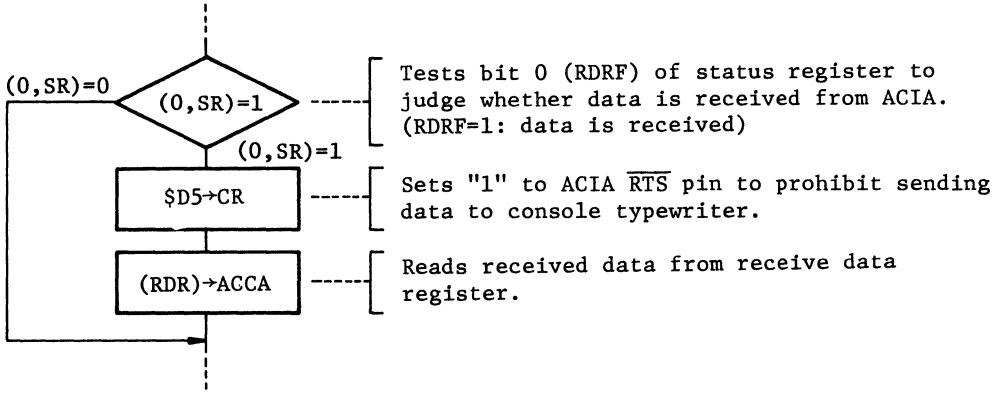


Fig. 13 Control of ACIA (receiving serial data)

(b) After data is received, set signal $\overline{RTS}$ to High to prohibit next data transfer.

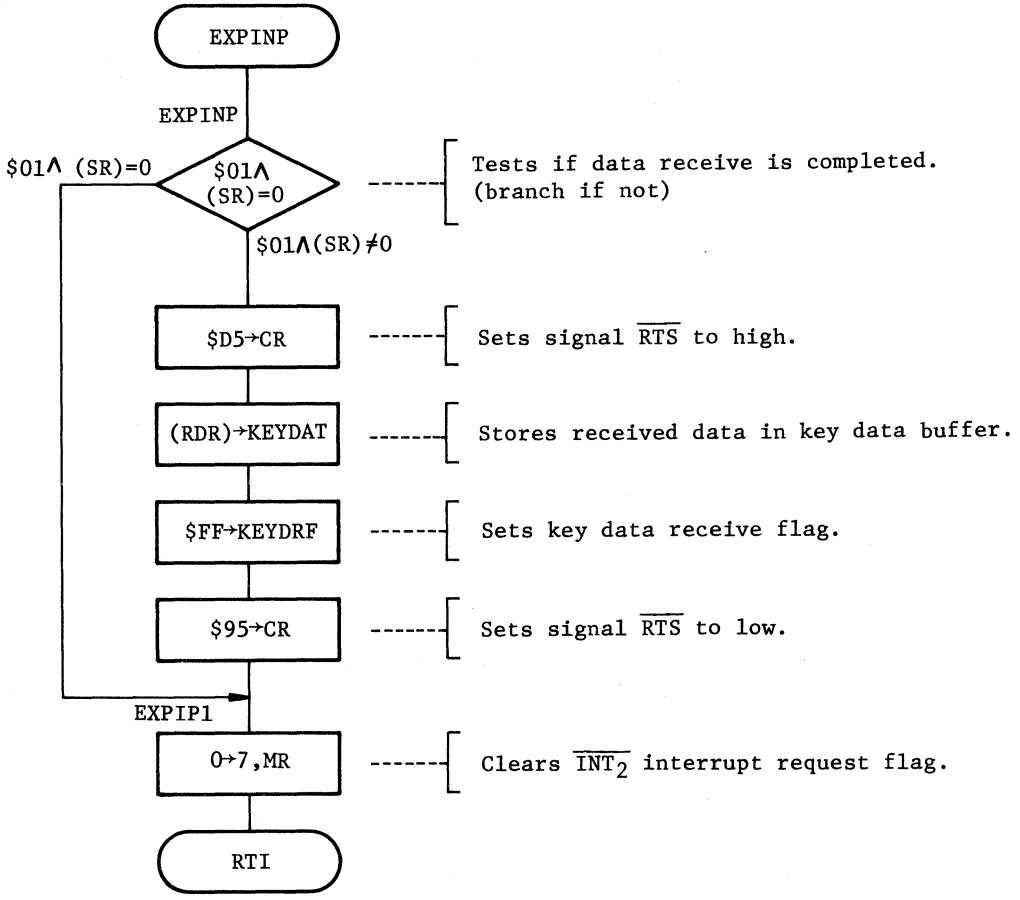
(c) Moves received data from ACIA RDR to key data buffer.

(d) Sets receive flag after data receive is completed.

7

PROGRAM MODULE NAME	RECEIVE DATA	MCU/MPU	HD6305Y2	LABEL	EXPINP

FLOWCHART



14.4 SUBROUTINE DESCRIPTION

SUBROUTINE NAME	CHECK BUSY FLAG	MCU/MPU	HD6305Y2	LABEL	EXPBSY

FUNCTION

Checks LCD-II busy flag.

BASIC OPERATION

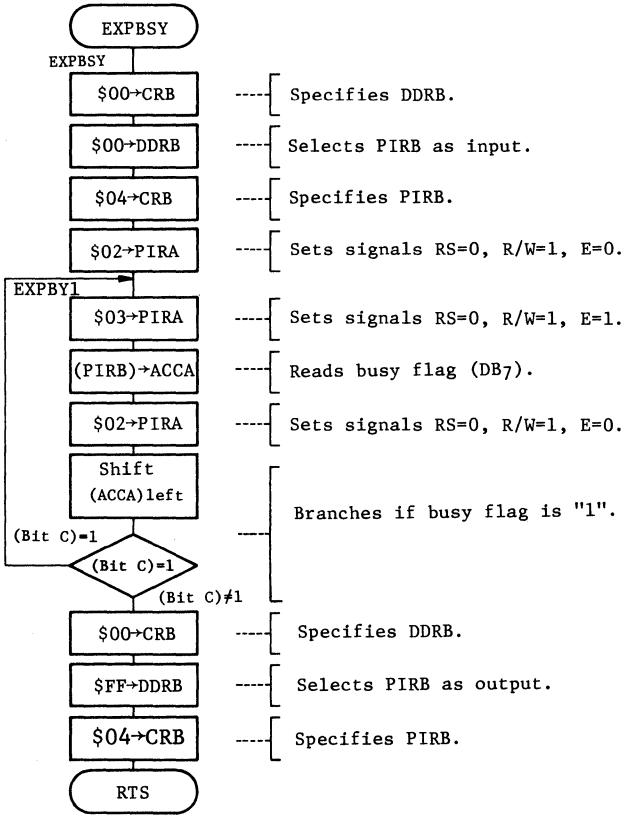
(1) When data bus of LCD-II is read during signals E=1 with RS=0, R/W=1, MSB of data bus becomes busy flag. When busy flag is "1", LCD-II does not receive any instruction.

(2) Main program does not return from subroutine EXPBSY until busy flag is "0".

FLOWCHART

PROGRAM MODULE USING THIS SUBROUTINE

EXPINT, EXPDSP



SUBROUTINE NAME	STORE INSTRUCTION	MCU/MPU	HD6305Y2	LABEL	EXPINS

FUNCTION

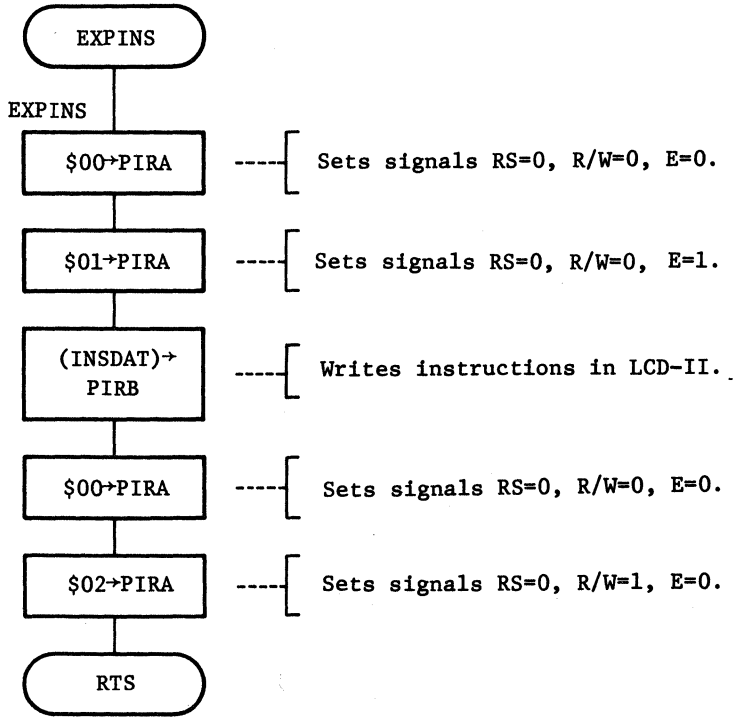
Writes instructions to LCD-II.

BASIC OPERATION

(1) Stores instructions in LCD-II with setting signals RS=0, R/W=0 and E=0.

(2) After writing instructions, set signals RS=0, R/W=1 and E=0.

FLOWCHART | **PROGRAM MODULE USING THIS SUBROUTINE** | EXPINT



```
00001                    *
00002                    *****   RAM ALLOCATION   ************************
00003                    *
00004 0080                       ORG     $80
00005                    *
00006 0080 0001          INSDAT  RMB     1       Instruction data
00007 0081 0001          OUTDAT  RMB     1       Character data to be output
00008 0082 0001          DSPDAT  RMB     1       Display data
00009 0083 0001          KEYDRF  RMB     1       Receive flag
00010 0084 0001          KEYDAT  RMB     1       Received data
00011 0085 0001          TNCNT   RMB     1       Counter for initialize LCD-II
00012 0086 0001          CUNT1   RMB     1       Counter for initialize LCD-II
00013 0087 0001          FUNC    RMB     1       Function initialize data
00014 0088 0001          ENTRY   RMB     1       Entry mode initialize data
00015                    *
00016                    *****   SYMBOL DEFINITIONS   ********************
00017                    *
00018       000A         MR      EQU     $0A     Miscellaneous register
00019       2000         DDRA    EQU     $2000   DDRA of PIA
00020       2001         CRA     EQU     $2001   CRA of PIA
00021       2002         DDRB    EQU     $2002   DDRB of PIA
00022       2003         CRB     EQU     $2003   CRB of PIA
00023       2000         PIRA    EQU     DDRA    PIRA of PIA
00024       2002         PIRB    EQU     DDRB    PIRB of PIA
00025       3000         CR      EQU     $3000   CR of ACIA
00026       3000         SR      EQU     CR      SR of ACIA
00027       3001         RDR     EQU     $3001   RDR of ACIA
00028       3001         TDR     EQU     RDR     TDR of ACIA
00029                    ************************************************
00030                    *                                              *
00031                    *       MAIN PROGRAM : EXPMN                   *
00032                    *                                              *
00033                    ************************************************
00034                    *
00035 1000                       ORG     $1000
00036                    *
00037 1000 4F            EXPMN   CLR     A
00038 1001 B7 0A                 STA     MR      Clear INT2 mask flag
00039 1003 C7 2001               STA     CRA     Select DDRA
00040 1006 A6 FF                 LDA     #$FF    Select PIRA as output
00041 1008 C7 2000               STA     DDRA
00042 100B A6 04                 LDA     #$04    Select PIRA
00043 100D C7 2001               STA     CRA
00044 1010 A6 02                 LDA     #$02    RS=0,R/W=1,E=0
00045 1012 C7 2000               STA     PIRA
00046 1015 4F                    CLR     A       Select DDRB
00047 1016 C7 2003               STA     CRB
00048 1019 A6 FF                 LDA     #$FF    Select PIRB as output
00049 101B C7 2002               STA     DDRB
00050 101E A6 04                 LDA     #$04    Select PIRB
00051 1020 C7 2003               STA     CRB
00052 1023 A6 30                 LDA     #$30    Initialize FUNC(RAM)
00053 1025 B7 87                 STA     FUNC
00054 1027 A6 06                 LDA     #$06    Initialize ENTRY(RAM)
00055 1029 B7 88                 STA     ENTRY
```

```
00056 102B CD 10AB          JSR     EXPINT    Initialize LCD-II
00057 102E AD 4B            BSR     EXPBSY    Check busy flag
00058 1030 A6 0E            LDA     #$0E      Store instruction of LCD-II
00059 1032 B7 80            STA     INSDAT
00060 1034 CD 10E7          JSR     EXPINS    Write instruction
00061 1037 A6 97            LDA     #$97      Master-reset ACIA
00062 1039 C7 3000          STA     CR
00063 103C A6 95            LDA     #$95      Initialize ACIA
00064 103E C7 3000          STA     CR
00065 1041 9A               CLI               Enable interrupts
00066 1042 B6 83    EXPMN1  LDA     KEYDRF    Test key receive flag
00067 1044 27 FC            BEQ     EXPMN1    Loop if not key on
00068 1046 B6 84            LDA     KEYDAT    Load key data
00069 1048 CD 110C          JSR     TPR       ASCII lowercase->uppercase
00070 104B 3F 83            CLR     KEYDRF    Clear key receive flag
00071 104D A6 95            LDA     #$95      Set signal RTS=0
00072 104F C7 3000          STA     CR
00073 1052 B6 84            LDA     KEYDAT
00074 1054 B7 81            STA     OUTDAT    Store output data
00075 1056 B7 82            STA     DSPDAT    Store display data
00076 1058 CD 10FF          JSR     EXPOUT    Type character
00077 105B AD 02            BSR     EXPDSP    Display character
00078 105D 20 E3            BRA     EXPMN1
00079                *************************************************
00080                *                                               *
00081                *       NAME : EXPDSP (DISPLAY CHRACTERS)       *
00082                *                                               *
00083                *************************************************
00084                *                                               *
00085                *         ENTRY : DSPDAT (DISPLAY DATA)         *
00086                *       RETURNS : NOTHING                       *
00087                *                                               *
00088                *************************************************
00089 105F AD 1A    EXPDSP  BSR     EXPBSY    Check busy flag
00090 1061 A6 04            LDA     #$04      Set RS=1,R/W=0,E=0
00091 1063 C7 2000          STA     PIRA
00092 1066 A6 05            LDA     #$05      Set E=1
00093 1068 C7 2000          STA     PIRA
00094 106B B6 82            LDA     DSPDAT    Write display data
00095 106D C7 2002          STA     PIRB
00096 1070 A6 04            LDA     #$04      Set E=0
00097 1072 C7 2000          STA     PIRA
00098 1075 A6 02            LDA     #$02      Set RS=0,R/W=1,E=0
00099 1077 C7 2000          STA     PIRA
00100 107A 81               RTS
00101                ************************************************
00102                *                                              *
00103                *       NAME : EXPBSY (CHECK BUSY FLAG)        *
00104                *                                              *
00105                ************************************************
00106 107B 4F       EXPBSY  CLR     A
00107 107C C7 2003          STA     CRB       Select DDRB
00108 107F C7 2002          STA     DDRB      Select PIRB as input
00109 1082 A6 04            LDA     #$04      Select PIRB
00110 1084 C7 2003          STA     CRB
```

```
00111 1087 A6 02           LDA    #$02     Set RS=0,R/W=1,E=0
00112 1089 C7 2000         STA    PIRA
00113 108C A6 03    EXPBY1 LDA    #$03     Set E=1
00114 108E C7 2000         STA    PIRA
00115 1091 C6 2002         LDA    PIRB     Read busy flag
00116 1094 AE 02           LDX    #$02     Set E=0
00117 1096 CF 2000         STX    PIRA
00118 1099 48              LSL    A        Shift ACCA 1 bit left
00119 109A 25 F0           BCS    EXPBY1   Branch if busy flag=1?
00120 109C 4F              CLR    A        Select DDRB
00121 109D C7 2003         STA    CRB
00122 10A0 A6 FF           LDA    #$FF     Select PIRB as output
00123 10A2 C7 2002         STA    DDRB
00124 10A5 A6 04           LDA    #$04     Select PIRB
00125 10A7 C7 2003         STA    CRB
00126 10AA 81              RTS
00127                      ************************************************
00128                      *                                              *
00129                      *      NAME : EXPINT (INITIALIZE LCD-II)       *
00130                      *                                              *
00131                      ************************************************
00132                      *                                              *
00133                      *    ENTRY : FUNC  (FUNCTION INITIALIZE DATA)  *
00134                      *            ENTRY (ENTRY MODE INITIALIZE DATA)*
00135                      *  RETURNS : NOTHING                           *
00136                      *                                              *
00137                      ************************************************
00138 10AB A6 03    EXPINT LDA    #$03     Initialize loop counter
00139 10AD B7 85           STA    TNCNT
00140 10AF A6 80    EXPIT1 LDA    #$80     Initialize timer counter
00141 10B1 B7 86           STA    CUNT1
00142 10B3 AE 1A    EXPIT2 LDX    #$1A     Execute 30ms software timer
00143 10B5 5A       EXPIT3 DEC    X
00144 10B6 26 FD           BNE    EXPIT3
00145 10B8 3A 86           DEC    CUNT1
00146 10BA 26 F7           BNE    EXPIT2
00147 10BC A6 30           LDA    #$30     Store instruction to set
00148 10BE B7 80           STA    INSDAT                          function
00149 10C0 AD 25           BSR    EXPINS   Write instruction in LCD-II
00150 10C2 3A 85           DEC    TNCNT    Decrement loop counter
00151 10C4 26 E9           BNE    EXPIT1   Test if loop counter=0
00152 10C6 B6 87           LDA    FUNC     Store initalization data
00153 10C8 B7 80           STA    INSDAT
00154 10CA AD AF           BSR    EXPBSY   Check busy flag
00155 10CC AD 19           BSR    EXPINS   Write instruction in LCD-II
00156 10CE A6 08           LDA    #$08     Store instruction to set
00157 10D0 B7 80           STA    INSDAT                          function
00158 10D2 AD A7           BSR    EXPBSY   Check busy flag
00159 10D4 AD 11           BSR    EXPINS   Write instruction in LCD-II
00160 10D6 A6 01           LDA    #$01     Store instruction to clear
00161 10D8 B7 80           STA    INSDAT                           display
00162 10DA AD 9F           BSR    EXPBSY   Check busy flag
00163 10DC AD 09           BSR    EXPINS   Write instruction in LCD-II
00164 10DE B6 88           LDA    ENTRY    Store instruction of entry mode
00165 10E0 B7 80           STA    INSDAT
```

```
00166 10E2 AD 97          BSR    EXPBSY   Check busy flag
00167 10E4 AD 01          BSR    EXPINS   Write instruction in LCD-II
00168 10E6 81             RTS
00169                  ************************************************
00170                  *                                              *
00171                  *      NAME : EXPINS (STORE INSTRUCTION)       *
00172                  *                                              *
00173                  ************************************************
00174 10E7 4F          EXPINS CLR    A        Set RS=0,R/W=0,E=0
00175 10E8 C7 2000            STA    PIRA
00176 10EB A6 01              LDA    #$01     Set E=1
00177 10ED C7 2000            STA    PIRA
00178 10F0 B6 80              LDA    INSDAT   Write instructio in LCD-II
00179 10F2 C7 2002            STA    PIRB
00180 10F5 4F                 CLR    A        Set E=0
00181 10F6 C7 2000            STA    PIRA
00182 10F9 A6 02              LDA    #$02     Set RS=0,R/W=1,E=0
00183 10FB C7 2000            STA    PIRA
00184 10FE 81                 RTS
00185                  ************************************************
00186                  *                                              *
00187                  *          NAME : EXPOUT (SEND DATA)           *
00188                  *                                              *
00189                  ************************************************
00190                  *                                              *
00191                  *      ENTRY : OUTDAT (CHARACTER DATA)         *
00192                  *    RETURNS : NOTING                          *
00193                  *                                              *
00194                  ************************************************
00195 10FF A6 02       EXPOUT LDA    #$02     Test if TDRE of ACIA=1
00196 1101 C4 3000            AND    SR
00197 1104 27 F9              BEQ    EXPOUT   Loop if TDRE=0
00198 1106 B6 81              LDA    OUTDAT   Write character data in TDR
00199 1108 C7 3001            STA    TDR
00200 110B 81                 RTS
00201                  ************************************************
00202                  *                                              *
00203                  * NAME : TPR (CONVERSION OF ASCII LOWERCASE    *
00204                  *             INTO ASCII UPPER CASE)           *
00205                  *                                              *
00206                  ************************************************
00207                  *                                              *
00208                  *        ENTRY : ACCA (ASCII LOWERCASE)        *
00209                  *      RETURNS : KEYDAT (CONVERT ASCII DATA)   *
00210                  *                                              *
00211                  ************************************************
00212 110C A1 61       TPR    CMP    #'a      ACCA-'a' ?
00213 110E 25 08              BCS    TPR1     Branch if ACCA<'a'
00214 1110 A1 7A              CMP    #'z      ACCA-'z' ?
00215 1112 22 04              BHI    TPR1     Branch if ACCA>'z'
00216 1114 A4 DF              AND    #$DF     Convert lowercase into
00217 1116 B7 84              STA    KEYDAT                          Uppercase
00218 1118 81          TPR1   RTS
00219                  ************************************************
00220                  *                                              *
```

```
00221                    *       NAME : EXPINP (RECEIVE DATA)              *
00222                    *                                                 *
00223                    ***************************************************
00224                    *                                                 *
00225                    *            ENTRY : NOTHING                      *
00226                    *          RETURNS : KEYDAT (RECEIVE DATA)        *
00227                    *                    KEYDRF (RECEIVE FLAG)        *
00228                    *                                                 *
00229                    ***************************************************
00230 1119 A6 01    EXPINP LDA     #$01      Test if RDRF of ACIA=1
00231 111B C4 3000         AND     SR
00232 111E 27 13           BEQ     EXPIP1    Loop if RDRF=0
00233 1120 A6 D5           LDA     #$D5      Set RTS of ACIA=1
00234 1122 C7 3000         STA     SR
00235 1125 C6 3001         LDA     RDR       Read received data
00236 1128 B7 84           STA     KEYDAT
00237 112A A6 FF           LDA     #$FF      Store key receive flag
00238 112C B7 83           STA     KEYDRF
00239 112E A6 95           LDA     #$95      Set RTS of ACIA=0
00240 1130 C7 3000         STA     CR
00241 1133 1F 0A    EXPIP1 BCLR    7,MR      Clear INT2 request flag
00242 1135 80              RTI
00243                    ***************************************************
00244                    *                                                 *
00245                    *              VECTOR ADDRESSES                   *
00246                    *                                                 *
00247                    ***************************************************
00248                    *
00249 1FF6                 ORG     $1FF6
00250                    *
00251 1FF6 1000            FDB     EXPMN     SCI/TIMER2
00252 1FF8 1119            FDB     EXPINP    TIMER/INT2
00253 1FFA 1000            FDB     EXPMN     INT
00254 1FFC 1000            FDB     EXPMN     SWI
00255 1FFE 1000            FDB     EXPMN     RES
00256                    *
00257                      END
```

15. LOW POWER DISSIPATION MODE AND HA1835P CONTROL

15.1 HARDWARE DESCRIPTION

(1) Function

(a) Performs low power dissipation mode, and executes fail safe mode with HA1835P using the HD6305X0. Initialization of each mode is performed by switch 2, switch 4.

(b) In low power dissipation mode, performs wait mode by using switch 1, stop mode by switch 5 standby mode by switch 3. Counts down binary counter for LED in every second.

(c) Turns on LED indicating execution of each mode.

(d) In fail-safe mode with HA1835P, Switch 1 stops outputting pulse to HA1835P and executes system runaway.

(2) Microcomputer Applications

(a) Executes wait mode, stop mode, and standby mode of the HD6305X0.

(b) Wait mode is released by timer interrupt, stop mode is by $\overline{\text{INT}}$ interrupt, and standby mode is by $\overline{\text{STBY}}$ pin.

(c) Controlling port A, MCU controls and outputs pulse to HA1835P.

(3) Circuit Diagram

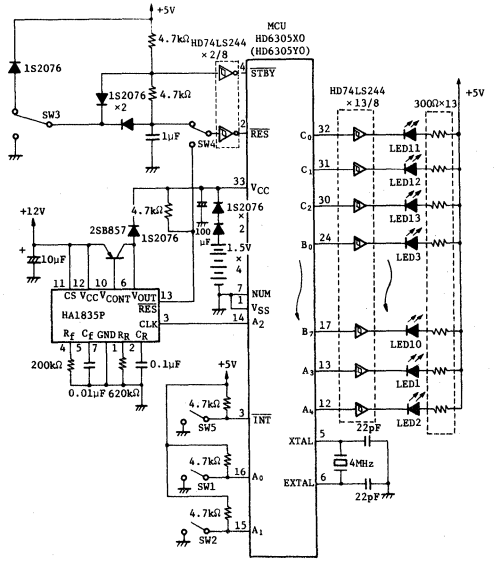


Fig. 1 Low Power Dissipation Mode and HA1835P Control

(4) Pin Functions

The HD6305X0, switch, LED name and pin functions of HA1835P are shown in Table 1.

Table 1 Pin Functions

Pin Name (HD6305X0)	Input/ Output	Active Level (High or Low)	Function		Pin Name SW, LED, HA1835P	Label
A_0	Input	—	Controls HA1835P	Low power dissipation mode	SW1	PADTR
		High	Retain Low or High	Wait mode switch input		
		Low	Pulse output	—		
A_1	Input	High	Controls HA1835P		SW2	
		Low	Low power dissipation mode			
A_2	Output	—	Controls pulse for HA1835P CLK pin.		CLK	
A_3	Output	Low	Drives LED indicating reset recovery by HA1835P.		LED1	
A_4	Output	Low	Drives LED indicating HA1835P control.		LED2	
B_0	Output	Low	Drives LED indicating 8-bit binary decrement counter.		LED3	PBDTR
B_1	Output	Low			LED4	
B_2	Output	Low			LED5	
B_3	Output	Low			LED6	
B_4	Output	Low			LED7	
B_5	Output	Low			LED8	
B_6	Output	Low			LED9	
B_7	Output	Low			LED10	
C_0	Output	Low	Drives LED during standby mode execution.		LED11	PCDTR
C_1	Output	Low	Drives LED during stop mode execution.		LED12	
C_2	Output	Low	Drives LED during wait mode execution.		LED13	
$\overline{STBY}$	Input	Low	Activates standby mode switch input.		SW3	
$\overline{RES}$	Input	Low	HA1835P control and low power dissipation mode.		SW4	
$\overline{INT}$	Input	Low	Activates stop mode.		SW5	

(5) Hardware Operation

(a) Low Power Dissipation

i) Standby Mode

Timing chart in Standby Mode is shown in Fig. 2.

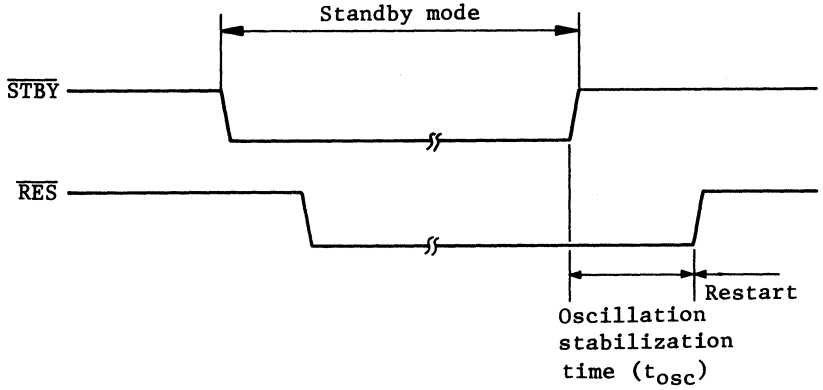


Fig. 2 Timing Chart in Standby Mode

ii) Stop Mode

Timing chart in Stop Mode is shown in Fig. 3.

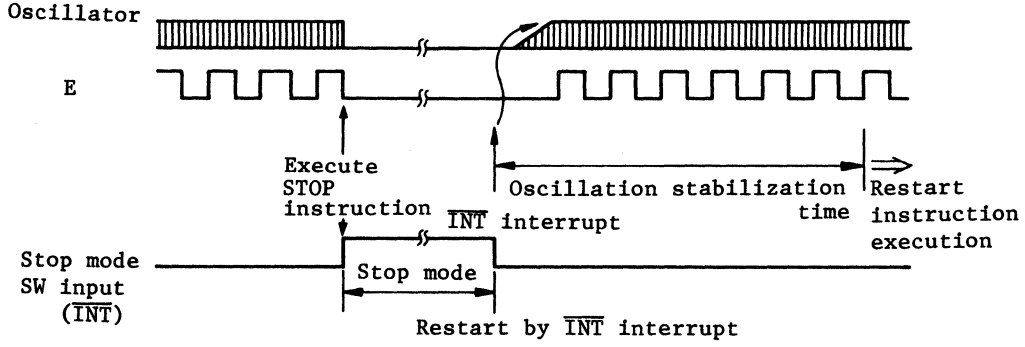


Fig. 3 Timing Chart in Stop Mode

iii) Wait Mode

Timing chart in Wait Mode is shown in Fig. 4.

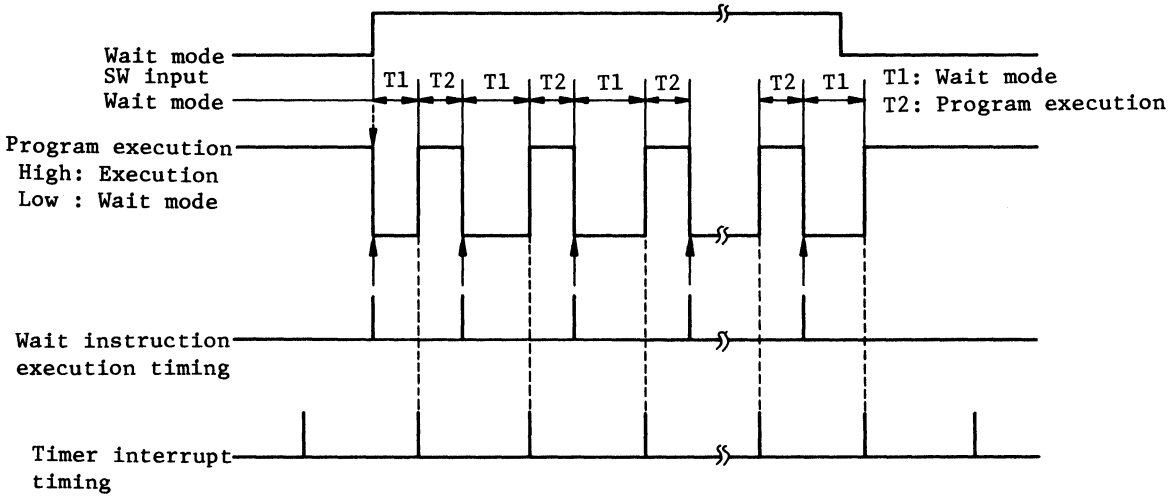


Fig. 4 Timing Chart in Wait Mode

(b) HA1835P Control

Timing chart of pulse output and reset pin for watchdog timer is shown in Fig. 5.

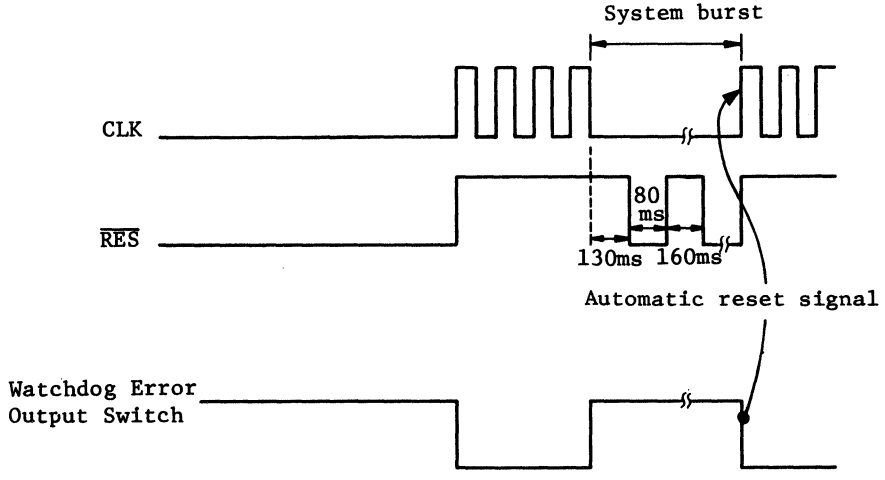


Fig. 5 Timing Chart of Pulse Output

15.2 SOFTWARE DESCRIPTION

(1) Program Module Configuration

Program module configuration for switch board controlling HA1835P and low power dissipation mode is shown in Fig. 6.

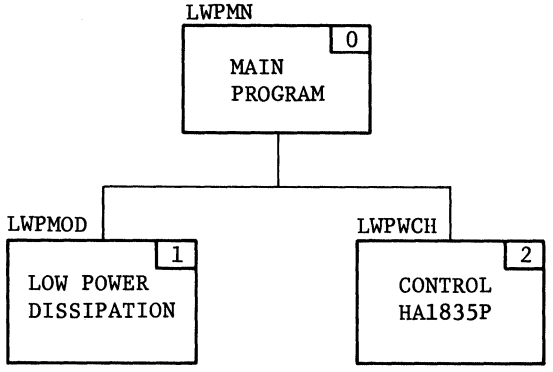


Fig. 6 Program Module Configuration

(2) Program Module Functions

Program module functions are summarized in Table 2.

Table 2 Program Module Functions

No.	Program Module Name	Label	Function
0	MAIN PROGRAM	LWPMN	Demonstrates low power dissipation and HA1835P control.
1	LOW POWER DISSIPATION	LWPMOD	Tests operation of low power dissipation mode.
2	CONTROL HA1835P	LWPWCH	Tests operation of watchdog timer using HA1835P.

(3) Program Module Sample Application (Main Program)

The flowchart in Fig. 7 is an example of low power dissipation and HA1835P control performed by the program module in Fig. 6.

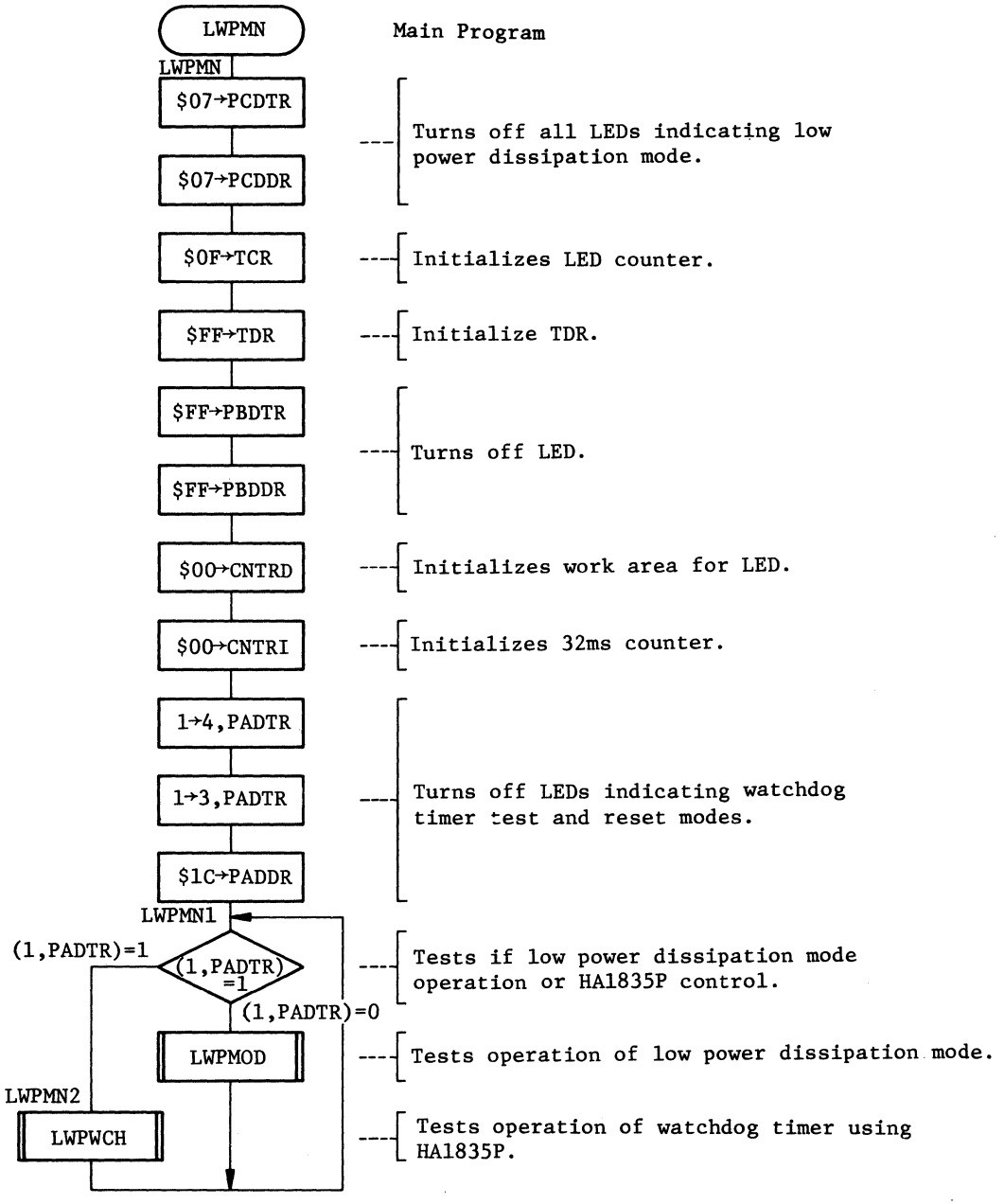


Fig. 7 Program Module Flowchart

15.3 PROGRAM MODULE DESCRIPTION

PROGRAM MODULE NAME	LOW POWER DISSIPATION	MCU/MPU	HD6305X0/Y0	LABEL	LWPMOD

FUNCTION

Tests operation of low power dissipation mode.

ARGUMENTS

		Contents	Storage Location	Byte Lgth.
Arguments	Entry	—	—	—
	Returns	—	—	—

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
×	×

N	I
×	×

H
●

SPECIFICATIONS

ROM (Bytes)
67
RAM (Bytes)
4
Stack (Bytes)
0
No. of cycles
92
Reentrant
No
Relocation
No
Interrupt
Yes

DESCRIPTION

(1) Function Details

(a) Program module LWPMOD has no arguments.

(b) Executes standby mode, stop mode, and wait mode with switch. Lights corresponding LED for each mode; LED turns off after reset.

(2) User Notes

(a) Executes only one mode at a time.

(b) Selects only one mode switch.

SPECIFICATIONS NOTES

"No. of cycles"in "SPECIFICATION" represents the number of cycles required when data is not stored in SBRAM(RAM) and each mode is executed once.

DESCRIPTION

(c) Initializes timer.

(d) Selects DDRs of bit 0-2 of port C as output.

(3) RAM Description

Label	RAM	Description
CNTRD	b7 b0	Stores 32ms counter.
CNTRI		Stores 1s counter.
SBRAM	Upper / Lower	Stores decision data to check reset data from STANDBY mode.

(4) Sample Application

Calls program module LWPMOD after selecting I/O port and clearing RAM.

```
        :
        :
LDA     #$07    }
STA     PCDTR   }---Turns off all LEDs indicating modes.
STA     PCDDR   }
CLR     CNTRD   ---Clears 32ms counter.
CLR     CNTRI   ---Clears 1s counter.
JSR     LWPMOD  ---Call program module LWPMOD.
        :
        :
```

(5) Basic Operation

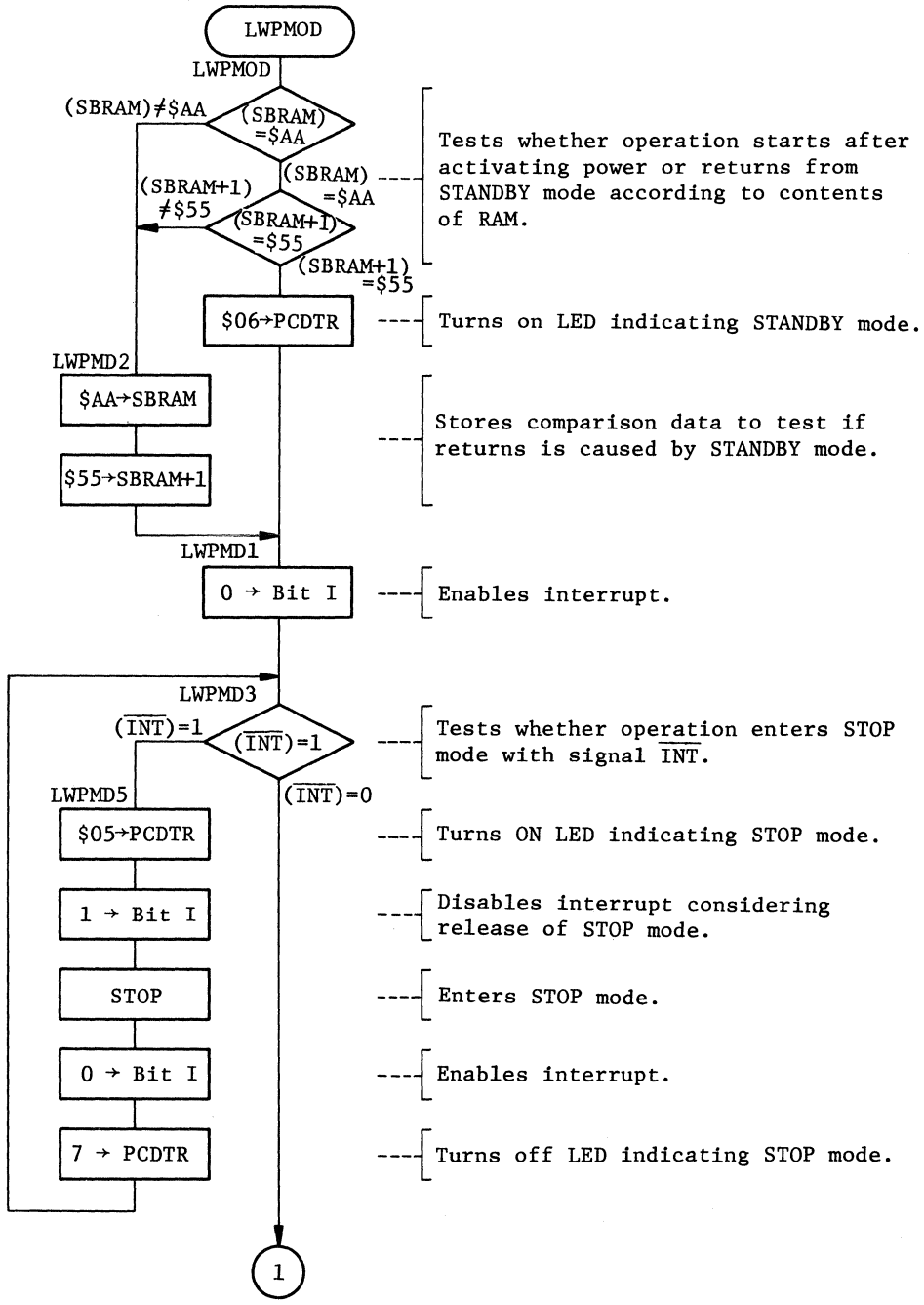
(a) Enters STANDBY mode by setting signal $\overline{\text{STBY}}$ to Low, returns with $\overline{\text{STBY}}$ to High.
Software checks whether operation starts after activating on power or returns from STANDBY mode.
When operation returns from STANDBY mode, turns on LED. if not, stores comparison data in SBRAM(RAM).

(b) Enters STOP mode by setting signal $\overline{\text{INT}}$ to High, returns with $\overline{\text{INT}}$ to Low.
When setting signal $\overline{\text{INT}}$ to High, LED indicates STOP mode and executes stop instruction. When Low, returns from STOP and turns LED off.

(c) Enters WAIT mode by setting WAIT mode switch to High, returns with Low.
Returns at every timer interrupt and executes interrupt routine.
When WAIT mode switch is Low, enters WAIT mode after interrupt.
Whether or not operation is returned at every timer interruption can be confirmed by LED counter incremented after return. When setting WAIT mode switch to High, turns on LED indicating WAIT mode and executes WAIT instruction. When Low, returns from WAIT mode and turns off LED indicating WAIT mode.

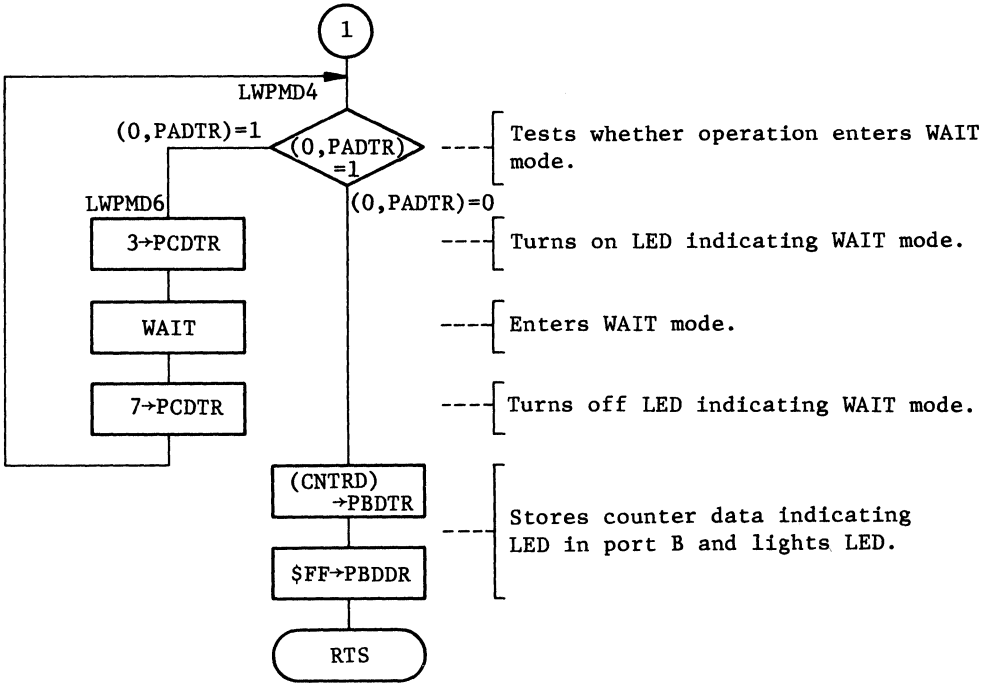
PROGRAM MODULE NAME	LOW POWER DISSIPATION	MCU/MPU	HD6305X0/Y0	LABEL	LWPMOD

FLOWCHART



PROGRAM MODULE NAME	LOW POWER DISSIPATION	MCU/MPU	HD6305X0/Y0	LABEL	LWPMOD

FLOWCHART



PROGRAM MODULE NAME	CONTROL HA1835P	MCU/MPU	HD6305X0/Y0	LABEL	LWPWCH

FUNCTION

Tests watchdog timer operation using HA1835P.

ARGUMENTS

Contents			Storage Location	Byte Lgth.
Arguments	Entry	-	-	-
	Returns	-	-	-

CHANGES IN CPU REGISTERS AND FLAGS

● : Not affected
× : Undefined
↕ : Result

ACCA	IX
×	●

C	Z
×	×

N	I
×	●

H
●

SPECIFICATIONS

ROM (Bytes)
44
RAM (Bytes)
2
Stack (Bytes)
0
No. of cycles
58
Reentrant
No
Relocation
No
Interrupt
No

DESCRIPTION

(1) Function Details

(a) Program module LWPWCH has no arguments.

(b) Controls HA1835P using switch.

(2) User Notes

(a) Selects DDRs of bits 2, 3 and 4 of port A and ports B and C as output.

(b) Selects DDRs of bits 0 and 1 of port A as input.

SPECIFICATIONS NOTES

"No. of cycles" in "SPECIFICATIONS" represents the number of cycles required when data is not stored in ERRAM(RAM) and an pulse generation is selected by switch.

(3) RAM description

Label	RAM	Description
ERRAM	b7 Upper / Lower b0	Stores check data which tests whether operation starts after activating power or returns from watchdog timer error.

(4) Sample Application

Calls program module LWPWCH after selecting I/O port and clearing RAM.

```
        :
        :
BSET    4, PADTR ---- Turns off LED indicating watchdog timer test mode.

BSET    3, PADTR ---- Turns off LED indicating watchdog timer reset.

JSR     LWPWCH   ---- Calls program module LWPWCH
        :
        :
```

(5) Basic Operation

(a) Lights LED indicating watchdog timer mode, after main switch is activated.

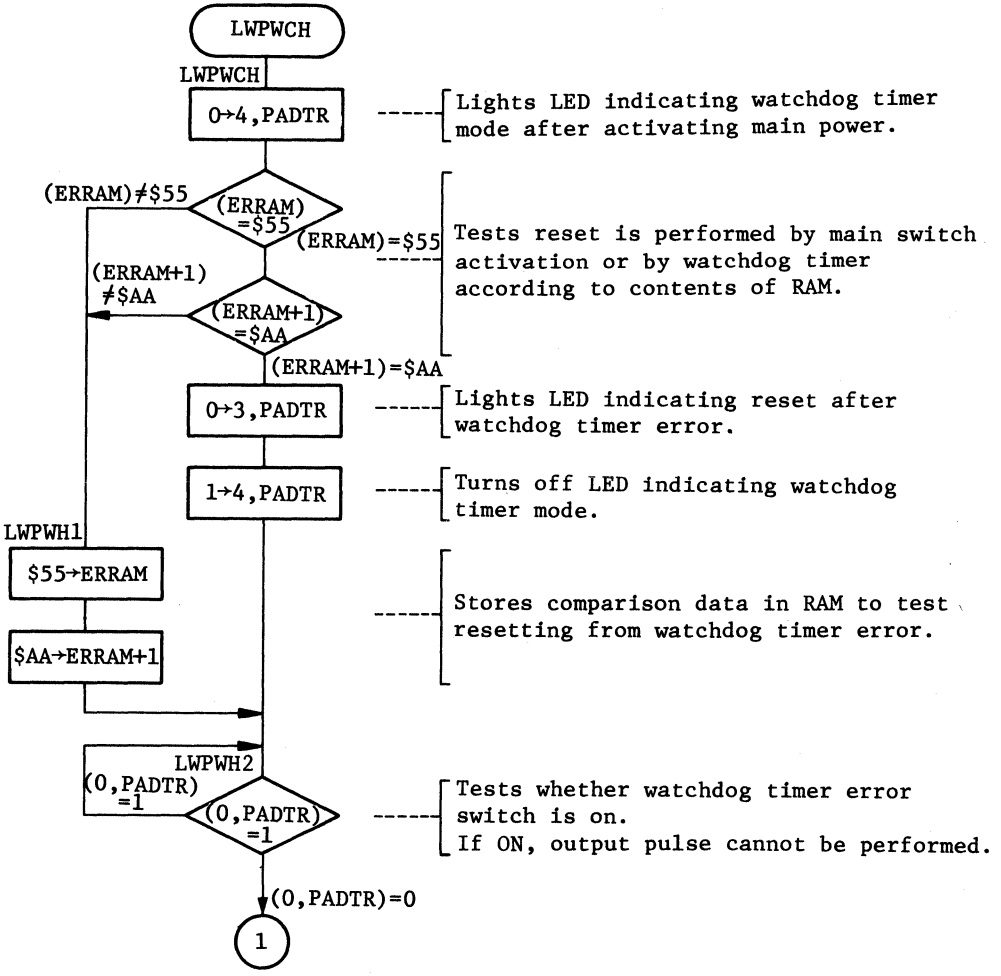
(b) When watchdog error switch is turned off, outputs pulse by 1ms software timer. When turned on, stops pulse output and enter eternity loop.

(c) When turning off watchdog timer error switch, re-calls program after reset. If data in ERRAM(RAM) and decision data are the same, turns on LED indicating reset by watchdog timer error.

(d) Stores check data if it is not stored in ERRAM(RAM).

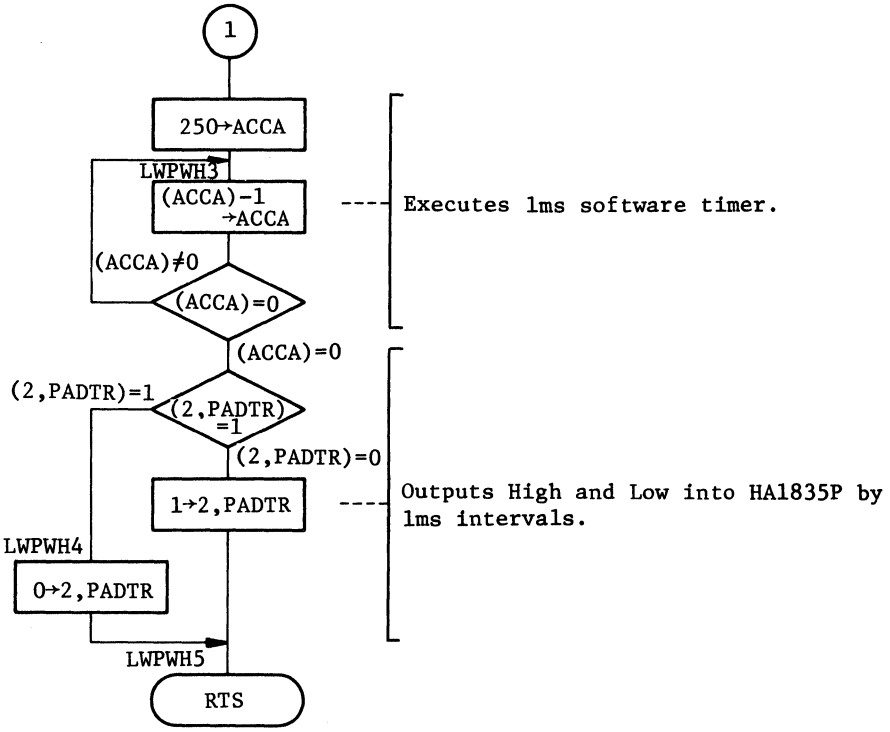
PROGRAM MODULE NAME	CONTROL HA1835P	MCU/MPU	HD6305X0/Y0	LABEL	LWPWCH

FLOWCHART



PROGRAM MODULE NAME	CONTROL HA1835P	MCU/MPU	HD6305X0/Y0	LABEL	LWPWCH

FLOWCHART



15.4 SUBROUTINE DESCRIPTION

SUBROUTINE NAME	INCREMENT COUNTER	MCU/MPU	HD6305X0/Y0	LABEL	LWPCNT

FUNCTION

Decrements LED counter CNTRI(RAM).

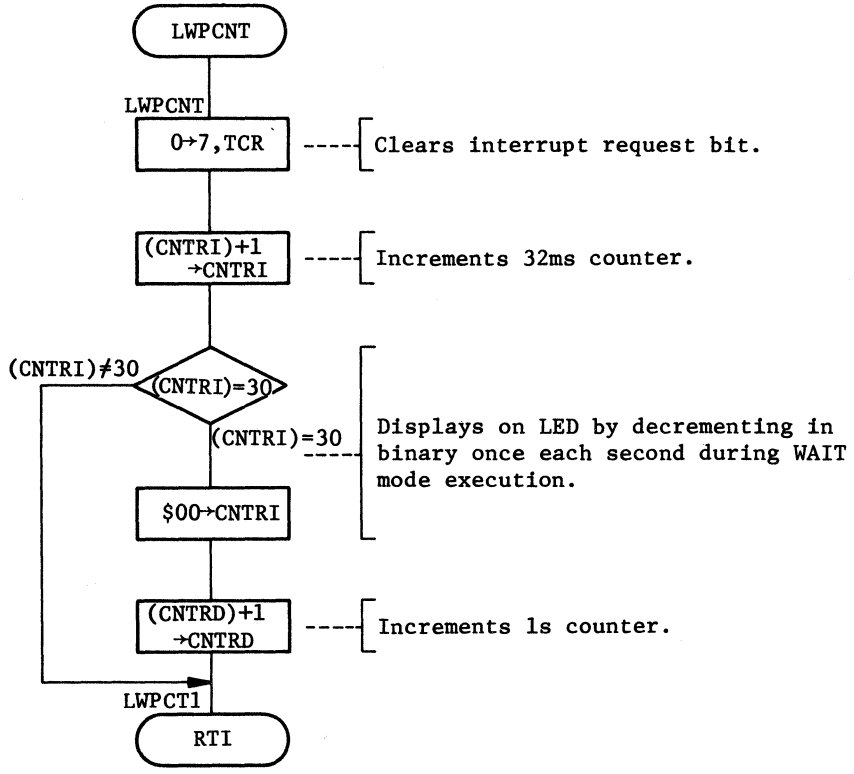
BASIC OPERATION

(1) This subroutine is called for each 32ms timer interrupt.

(2) Uses two counters to count up each second.

(3) When incrementing one counter every timer interrupt, one second interval is obtained after 30 times incrementing.

FLOWCHART | **PROGRAM MODULE USING THIS SUBROUTINE** | —



```
00001                    *
00002                    ****    RAM ALLOCATION    **************************
00003                    *
00004 0080                       ORG     $80
00005                    *
00006 0080 0001          CNTRD   RMB     1         1s counter
00007 0081 0001          CNTRI   RMB     1         32ms counter
00008 0082 0002          ERRAM   RMB     2         Watch dog data RAM area
00009 0084 0002          SBRAM   RMB     2         Standby RAM area
00010                    *
00011                    ****    SYMBOL DEFINITIONS ***********************
00012                    *
00013       0000         PADTR   EQU     $00       Port A data register
00014       0001         PBDTR   EQU     $01       Port B data register
00015       0002         PCDTR   EQU     $02       Port C data register
00016       0004         PADDR   EQU     $04       Port A data direction register
00017       0005         PBDDR   EQU     $05       Port B data direction register
00018       0006         PCDDR   EQU     $06       Port D data direction register
00019       0009         TCR     EQU     $09       Timer control register
00020       0008         TDR     EQU     $08       Timer data register
00021                    ************************************************
00022                    *                                              *
00023                    *            MAIN PROGRAM : LWPMN              *
00024                    *                                              *
00025                    ************************************************
00026                    *
00027 1000                       ORG     $1000
00028                    *
00029 1000 A6 07         LWPMN   LDA     #$07
00030 1002 B7 02                 STA     PCDTR     Turn off LED
00031 1004 B7 06                 STA     PCDDR     Select port C bit0-2 as output
00032 1006 A6 0F                 LDA     #$F       Initialize TCR
00033 1008 B7 09                 STA     TCR
00034 100A A6 FF                 LDA     #$FF      Initialize TDR
00035 100C B7 08                 STA     TDR
00036 100E B7 01                 STA     PBDTR     Turn off LED
00037 1010 B7 05                 STA     PBDDR     Select portB as output
00038 1012 3F 80                 CLR     CNTRD     Clear 1s counter
00039 1014 3F 81                 CLR     CNTRI     Clear 32ms counter
00040 1016 18 00                 BSET    4,PADTR   Turn off LED
00041 1018 16 00                 BSET    3,PADTR   Turn off LED
00042 101A A6 1C                 LDA     #$1C      Select port A bit2-4 as output
00043 101C B7 04                 STA     PADDR
00044 101E 02 00 05 LWPMN1 BRSET  1,PADTR,LWPMN2  Branch if HA1835P
                                                                  test mode
00045 1021 CD 102B               JSR     LWPMOD    Execute low power mode
00046 1024 20 F8                 BRA     LWPMN1
00047 1026 CD 106E  LWPMN2 JSR    LWPWCH    Execute HA1835P mode
00048 1029 20 F3                 BRA     LWPMN1
00049                    ************************************************
00050                    *                                              *
00051                    *       NAME : LWPMOD (LOW POWER DISSIPATION)  *
00052                    *                                              *
00053                    ************************************************
00054                    *                                              *
00055                    *       INPUT : NOTHING                        *
```

```
00056                    *      OUTPUT : NOTHING                          *
00057                    *                                                *
00058                    **************************************************
00059 102B A6 AA     LWPMOD LDA    #$AA      Test if SBRAM=$AA?
00060 102D B1 84            CMP    SBRAM
00061 102F 26 0D            BNE    LWPMD2    Branch if not equal
00062 1031 A6 55            LDA    #$55      Test if SBRAM+1=$55?
00063 1033 B1 85            CMP    SBRAM+1
00064 1035 26 07            BNE    LWPMD2    Branch if not equal
00065 1037 A6 06            LDA    #$06      Stand by LED on
00066 1039 B7 02            STA    PCDTR
00067 103B 9A        LWPMD1 CLI               Enable interrupt
00068 103C 20 0A            BRA    LWPMD3
00069 103E A6 AA     LWPMD2 LDA    #$AA      Store $AA in SBRAM
00070 1040 B7 84            STA    SBRAM
00071 1042 A6 55            LDA    #$55      Store $55 in SBRAM+1
00072 1044 B7 85            STA    SBRAM+1
00073 1046 20 F3            BRA    LWPMD1
00074 1048 2F 0C     LWPMD3 BIH    LWPMD5    Test if STOP mode?
00075 104A 00 00 16  LWPMD4 BRSET  0,PADTR,LWPMD6  Branch if WAIT mode?
00076 104D B6 80            LDA    CNTRD     Counter LED on
00077 104F B7 01            STA    PBDTR
00078 1051 A6 FF            LDA    #$FF      Select port B as output
00079 1053 B7 05            STA    PBDDR
00080 1055 81               RTS
00081 1056 A6 05     LWPMD5 LDA    #$05      Stop mode LED on
00082 1058 B7 02            STA    PCDTR
00083 105A 9B               SEI              Disable intrrupt
00084 105B 8E               STOP             STOP mode
00085 105C 9A               CLI              Enable interrupt
00086 105D A6 07            LDA    #07
00087 105F B7 02            STA    PCDTR     All LED off
00088 1061 20 E5            BRA    LWPMD3    (INT) = 1 -> WAIT mode loop
00089 1063 A6 03     LWPMD6 LDA    #03       Wait mode LED on
00090 1065 B7 02            STA    PCDTR
00091 1067 8F               WAIT             WAIT mode
00092 1068 A6 07            LDA    #07       All LED off
00093 106A B7 02            STA    PCDTR
00094 106C 20 DC            BRA    LWPMD4
00095                    **************************************************
00096                    *                                                *
00097                    *       NAME : LWPWCH (CONTROL HA1835P)          *
00098                    *                                                *
00099                    **************************************************
00100                    *                                                *
00101                    *       INPUT : NOTHING                          *
00102                    *      OUTPUT : NOTHING                          *
00103                    *                                                *
00104                    **************************************************
00105 106E 19 00     LWPWCH BCLR   4,PADTR   Watch dog LED on
00106 1070 A6 55            LDA    #$55      Test if ERRAM+1=$55
00107 1072 B1 82            CMP    ERRAM
00108 1074 26 0C            BNE    LWPWH1    Branch not equal
00109 1076 A6 AA            LDA    #$AA      Test if ERRAM=$AA?
00110 1078 B1 83            CMP    ERRAM+1
```

```
00111 107A 26 06          BNE    LWPWH1   Branch if not equal
00112 107C 17 00          BCLR   3,PADTR  Watch dog errar LED on
00113 107E 18 00          BSET   4,PADTR  Watch dog LED off
00114 1080 20 08          BRA    LWPWH2
00115 1082 A6 55    LWPWH1 LDA    #$55     Store $55 in ERRAM
00116 1084 B7 82          STA    ERRAM
00117 1086 A6 AA          LDA    #$AA     Store $AA in ERRAM+1
00118 1088 B7 83          STA    ERRAM+1
00119 108A 00 00 FD LWPWH2 BRSET  0,PADTR,LWPWH2  Loop if W.D sw on
00120 108D A6 FA          LDA    #250     Execute 1ms software timer
00121 108F 4A       LWPWH3 DEC    A
00122 1090 26 FD          BNE    LWPWH3
00123 1092 04 00 04       BRSET  2,PADTR,LWPWH4  Branch if CLK=1
00124 1095 14 00          BSET   2,PADTR  Set CLK=1
00125 1097 20 02          BRA    LWPWH5
00126 1099 15 00    LWPWH4 BCLR   2,PADTR  Set CLK=0
00127 109B 81       LWPWH5 RTS
00129                      ************************************************
00130                      *                                              *
00131                      *       NAME : LWPCNT (INCREMENT COUNTER)      *
00132                      *                                              *
00133                      ************************************************
00134 109C 1F 09    LWPCNT BCLR   7,TCR    Clear interrupt request bit
00135 109E 3C 81          INC    CNTRI    Increment 32ms counter
00136 10A0 A6 30          LDA    #$30     Test if CNTRI=30
00137 10A2 B1 81          CMP    CNTRI
00138 10A4 26 04          BNE    LWPCT1   Brabch not equal
00139 10A6 3F 81          CLR    CNTRI    Clear 32ms counter
00140 10A8 3C 80          INC    CNTRD    Increment 1s counter
00141 10AA 80       LWPCT1 RTI
00142                      ************************************************
00143                      *                                              *
00144                      *              VECTOR ADDRESSES                *
00145                      *                                              *
00146                      ************************************************
00147                      *
00148 1FF6                 ORG    $1FF6
00149                      *
00150 1FF6 1000            FDB    LWPMN    SCI/TIMER2
00151 1FF8 109C            FDB    LWPCNT   TIMER/INT2
00152 1FFA 1000            FDB    LWPMN    INT
00153 1FFC 1000            FDB    LWPMN    SWI
00154 1FFE 1000            FDB    LWPMN    RES
00155                      *
00156                      END
```

7

Section Eight

APPENDIX:

Technical Q and A (Part I)

8-Bit Single-Chip Microcomputer

HD6305X0, HD6305X1, HD6305X2
HD6305Y0, HD6305Y1, HD6305Y2

The HD6305X and HD6305Y are microprogram-controlled 8-bit single-chip microcomputers that use CMOS 2.5 µm technology. They have a very high compatibility with the HD6805 in terms of instruction set. The CPU, ROM, RAM, I/O, timer, and serial communications interface (SCI) are all integrated into one chip.

Each HD6305X-series microcomputer consists of: A HD6305X0 with a 128-byte RAM and a 4-kbyte ROM on a single chip; a HD6305X1 connectable to external memory; and a HD6305X2 without ROM.

Each HD6305Y-series microcomputer consists of: A HD6305Y0 with a 256-byte RAM and a 8-kbyte ROM on a single chip; a HD6305Y1 with optional external memory; and a HD6305Y2 without ROM.

The CMOS technology enables these models to operate over a wide range of supply voltages (Vcc operates in the 3 V - 6 V range at operating frequencies of 0.1 MHz - 0.5 MHz) with less power consumption. The low-power-consumption modes (STOP, WAIT, and STANDBY) available with these models permit further power savings.

The instruction set includes DAA (decimal adjust instruction), and STOP and WAIT (used to enter the corresponding low-power-consumption modes), in addition to the HD6805 instruction set. The minimum instruction execution time is 0.5 µsec/cycle (f=2 MHz), permitting high-speed operation.

HOW TO USE THIS MANUAL

This TECHNICAL QUESTIONS AND ANSWERS is a user reference manual that has been compiled in question-and-answer form. It is based on technical inquiries from HITACHI microcomputer users.

This manual should be used in conjunction with the appropriate data book (which you should already have). You can make best use of this manual either by: reading all of it before you start to design any microcomputer-applied products, in order to strengthen your technical background; or referring to it during the actual design process, using it to help you solve specific problems that may arise.

Although some of the items supplement the data book, most of them are inquiries from the users. In order to meet future needs, we are prepared to increase the number of Q & A items and to update the data book.

Contents

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Outputting Data from Ports after a Reset

Question

Which operation should be performed first to output data through input ports A, B, C, D and G after a reset?; storing data in the Data Register of the ports or designating the corresponding DDR (Data Direction Register)?

Classification

*	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Store the data to be output in the Data Register first, and then designate the corresponding DDR to data output (DDR=1).
Since a reset causes the Data Register to become 0, if the DDR is designated to data output before data is stored in the Data register, 0 is output to the ports.

Applicable Manual

Title: 8-bit Single-chip Microcomputer Data Book

Other Data Document

Title:

Reference Q & A Sheet

No.: QA635-015B

Supplement

The HD6305X1/X2 and HD6305Y1/Y2 do not have port G.

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Serial I/O Pin Status		

Question

After ports C_5 to C_7 are respectively used as $\overline{CK}$, Rx and Tx pins of the SCI (serial I/O), these ports are specified as I/O pins using the SCR (SCI Control Register)(SCR7=SCR6=SCR5=0). What is the value of the DDR (Data Direction Register) when the ports are used as I/O pins?

Classification	
*	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

The DDR retains the value for the SCI, as shown below.

Pin	DDR value for SCI	Notes
C7/Tx	1 (output port)	When Tx is used, SCR7=1.
C6/Rx	0 (input port)	When Rx is used, SCR6=1.
C5/$\overline{CK}$	0 (input port)	In the case of external clock source mode.
	1 (output port)	In the case of internal clock source mode.

Even when the SCR value is changed after the SCI is used, the DDR value shown in the above table is maintained. When using C_5 to C_7 as I/O ports, rewrite the DDR value using software.

Applicable Manual

Title

8-bit Single-chip Microcomputer Data Book

Other Data Document

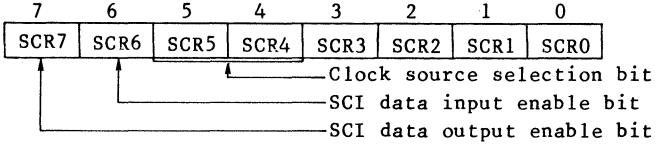
Title

Reference Q & A Sheet

No.

Supplement

SCR (SCI Control Register: $0010)



Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme	Using Port C when Serial I/O is Used

Question

When ports C_5 to C_7 are used as the SCI (serial I/O), is it possible to use ports C_0 to C_4 as usual I/O pins?

Classification	
*	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Yes. These ports can be used as usual I/O pins because they are independent of the SCI. The ports affected by the SCR (SCI Control Register) when the SCI is used are ports C_5 to C_7 only. Input and output functions of ports C_0 to C_4 are selected using bits 0 to 4 of port C's DDR (Data Direction Register).

Applicable Manual

Title

8-bit Single-chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Designating Input or Output Operation of Serial I/O Clock Pin

Question

Does setting 0 or 1 in the corresponding DDR (Data Direction Register) enable the clock pin $\overline{CK}$ to be designated as an input or output pin when the SCI (serial I/O) is used?

Classification

	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

No. Bits 4 and 5 (SCR4, 5) of the SCR (SCI Control Register) designate the $\overline{CK}$ pin as input or output when the SCI is used, not the DDR.
As shown below, the combination of bits 4 and 5 enables the SCI clock source selection, determining whether the $\overline{CK}$ pin is specified as an input or output pin.

SCR5	SCR4	Clock source	$\overline{CK}$ pin (port C5)
0	0	—	Used as I/O pin (according to the DDR)
0	1		
1	0	Internal	Clock output (DDR output)
1	1	External	Clock input (DDR input)

Note: The selection of input or output is made by the DDR when SCR5 is 0 and SCR4 is 0 or 1.

Applicable Manual

Title: 8-bit Single-chip Microcomputer Data Book

Other Data Document

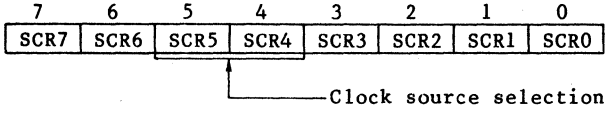
Title:

Reference Q & A Sheet

No.:

Supplement

SCR (SCI Control Register: $0010)



Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	SCI Prescaler Initialize Timing and Clock Output Timing		

Question

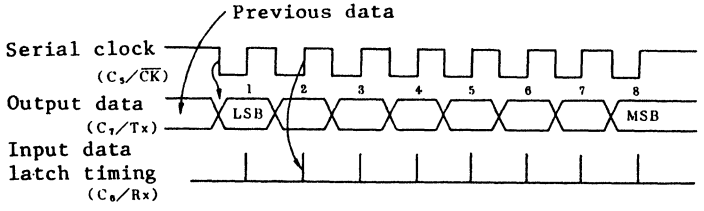
At what timing is the SCI (serial I/O) prescaler initialized?
In addition, when the internal clock is used (SCR4=0, SCR5=1), at what timing is the CK (serial clock) output?

Classification

	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

The prescaler is initialized when data is read from or written into the SDR (SCI Data Register).
When the internal clock is used, the CK is output at the baud rate specified by bits 0 to 3 of the SCR (SCI Control Register), immediately after the data reading/writing.
The SCI timing chart is shown below.



SCI Timing Chart

Applicable Manual

Title: 8-bit Single-chip Microcomputer Data Book

Other Data Document

Title:

Reference Q & A Sheet

No.

Supplement

8

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: SSR7 (SCI Interrupt Request Bit) Set Timing

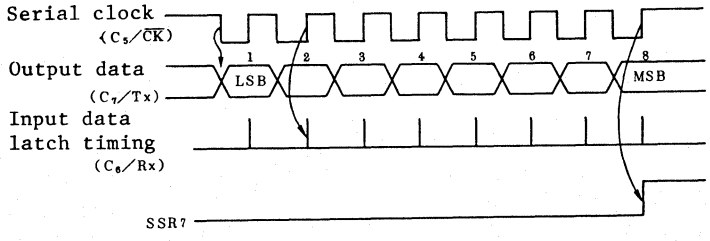
Question

After 8-bit data transmitting or receiving through the SCI (serial I/O) is completed, SSR7 (SCI interrupt request bit) of the SSR (SCI Status Register) is set. At what timing is this setting performed?

Classification	
	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

SSR7 is set at the rising edge of the 8th $\overline{CK}$ (serial clock) as shown below.



Applicable Manual

Title

Other Data Document

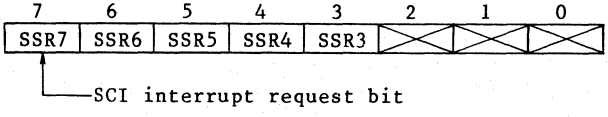
Title

Reference Q & A Sheet

No.

Supplement

SSR (SCI Status Register: $0011)



Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Using SDR (SCI Data Register) when Serial I/O is not Used

Question

Is it possible to use the SDR (SCR Data Register: $0012) as a general-purpose register when the SCI (serial I/O) is not used?

Classification

	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Yes. The SDR can be used as a general-purpose register when the SCI is not used i.e., when the SCI clock is stopped with SCR4 to SCR7 being 0.

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

SCR (SCI Control Register: $0010)

7	6	5	4	3	2	1	0
SCR7	SCR6	SCR5	SCR4	SCR3	SCR2	SCR1	SCR0

SCR5, SCR4: Clock source selection bit
SCR6: SCI data input pin enable bit
SCR7: SCI data output pin enable bit

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Accessing SDR (SCI Data Register)

Question

What will happen if data is read from or written into the SDR (SCI Data Register) while another data is being transmitted or received through the SCI (serial I/O)?

Classification

	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Normal operation can not be guaranteed for either transmission or reception. Do not access the SDR during data transmitting or receiving since the SCI becomes disabled after the access (the MCU must be reset to use the SCI again).

Applicable Manual

Title:

Other Data Document

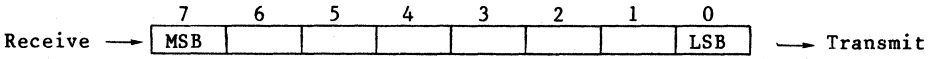
Title:

Reference Q & A Sheet

No.: QA635-033A

Supplement

SDR (SCI Data Register: $0012)



Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Clearing SSR (SCI Interrupt Request Bit)

Question

After data is transmitted or received through the SCI (serial I/O), SSR7 (SCI interrupt request bit) is set to 1. At this time, if 0 is set in SSR7 by software to clear the bit, is it possible to transmit or receive the next data?

Classification

	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

No. When 0 is set in SSR7 by software, this bit is cleared, and the next data transmitting and receiving cannot be performed. SSR7 is cleared under the following conditions.

① When data is read from or written into the SDR (SCI Data Register).
(After SSR7 is cleared and the octal counter is reset, the next data transmitting/receiving is executed.)

② When 0 is set in SSR7 by software.
(The SCI's octal counter is not reset, enabling no data transmitting/receiving.)

Therefore, repeat data reading or writing from/into the SDR to transmit/receive data repeatedly.
Dummy-read the SDR before receiving the first data.

Applicable Manual

Title:

Other Data Document

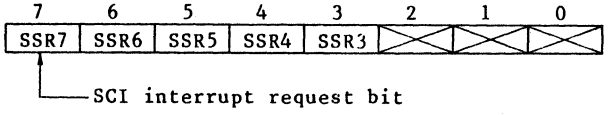
Title:

Reference Q & A Sheet

No.:

Supplement

SSR (SCI Status Register: $0011)



8

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Transmitting and Receiving Data Simultaneously through Serial I/O		

Question

Is is possible to transmit and receive data simultaneously through the SCI (serial I/O) by using the internal clock for transmitting and the external clock for receiving, or vice versa?

Classification	
	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

No. Only one clock source can be selected as the SCI transfer clock. Simultaneous data transmission and reception using two transfer clocks are impossible.
When a single clock source is used, data can be transmitted and received at the same time.

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Notes on Receiving Data through SCI in External Clock Mode		

Question

What should we pay attention to when using the external clock to receive data through the SCI (serial I/O)?

Classification	
	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

The external transfer clock source does not enable the receiving side to check when data is transmitted. Therefore, it is necessary to read out data in the SDR (SCI Data Register) as soon as the data is received, to prepare for receiving the next data.
Whether or not receiving has been completed can be checked by an SCI interrupt or by testing bit 7 of the SSR (SCI Status Register) by software.

Applicable Manual

Title

Other Data Document

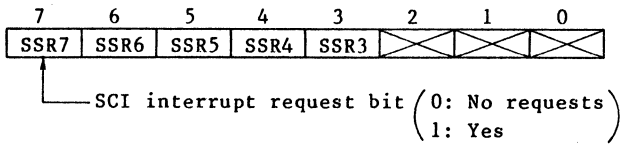
Title

Reference Q & A Sheet

No.

Supplement

SSR (SCI Status Register: $0011)



8

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: SCI Operation in External Clock Mode

Question

The SCI (serial I/O) is in external clock mode;
If the external clock is applied to the CK pin before the CPU writes/reads data into/from the SDR (Serial Data Register) after the one-byte data transmitting/receiving is completed, will the SCI start the next data transmitting/receiving?

Classification

	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

No. The SCI does not start transmitting or receiving the next data until the CPU writes/reads data into/from the SDR. That is, any clock signal applied to the CK pin before the CPU accesses the SDR will be ignored.

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

The $\overline{CK}$ pin can also used as C_5 pin. It is controlled by bits 4 and 5 of the SCR (SCI Control Register).

SCR5	SCR4	Clock source	C_5 pin
0	0	-	Used as I/O pin (according to the DDR)
0	1	-	
1	0	Internal	Clock output (DDR output)
1	1	External	Clock input (DDR input)

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Initializing the Transfer Clock Generator Prescaler

Question

The prescaler of the SCI transfer clock generator is initialized by reading/writing data from/into the SDR (SCI Status Register) or by setting 1 in SSR3 (SCI Status Register bit 3). What is the difference of the initialization performed by these two methods?

Answer

There are differences in the following points.

(1) When data is read/written from/into the SDR, the SCI octal counter is initialized at the same time the prescaler is initialized. This causes the SCI to start transmitting or receiving the next data.
(2) When 1 is set in SSR3, only the prescaler is initialized. The SCI does not start data trasmitting/receiving.

SSR3 is the bit to be used to initialize the prescaler when the transfer clock generator is utilized as Timer 2.

Classification

	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title:

Other Data Document

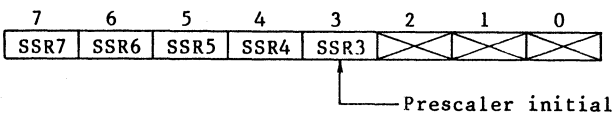
Title:

Reference Q & A Sheet

No.: QA635-025B

Supplement

SSR (SCI Status Register: $0011)



8

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Timer Count-down Timing when External Clock is Input		

Question

When an event input from the TIMER pin is used as the clock input source for the built-in timer, at which edge (rising or falling) of the input signal is the count-down performed?

Answer

It is performed at the rising edge of the input signal.

Classification	
	Parallel Port
	Serial Port
*	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-035A

Supplement

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Timer 2 Interrupt Cycles

Question

The Timer 2 interrupt cycle is selected by bits 0 to 3 of the SCR (SCI Control Register). Are there any limits to the selection?

Classification

	Parallel Port
	Serial Port
*	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

The selectable Timer 2 interrupt cycles are as shown below.

	SCR3	SCR2	SCR1	SCR0	Timer 2 interrupt cycle f_{osc} = 4 MHz	Timer 2 interrupt cycle f_{osc} = 4.194 MHz
Not selectable	0	0	0	0	1 μs	0.95 μs
	0	0	0	1	2 μs	1.91 μs
	0	0	1	0	4 μs	3.82 μs
Selectable	〃	〃	〃	〃	〃	〃
	1	1	1	1	32768 μs	1/32 s

Applicable Manual

Title: 8-bit Single-chip Microcomputer Data Book

Other Data Document

Title:

Reference Q & A Sheet

No.:

Supplement

SCR (SCI Control Register: $0010)

7	6	5	4	3	2	1	0
SCR7	SCR6	SCR5	SCR4	SCR3	SCR2	SCR1	SCR0

- SCR5, SCR4, SCR3: Transfer clock rate selection bit
- SCR4: Clock source selection bit
- SCR6: SCI data input enable bit
- SCR7: SCI data output enable bit

8

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Reading/Writing Data from/into the TDR during Timer Operation

Question

What will happen if data is written into or read from the TDR (Timer Data Register) during count-down?

Classification

	Parallel Port
	Serial Port
*	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

If data is written into the TDR during count-down, the count-down restarts from this data.
However, data reading during count-down does not affect the TDR value, and the count-down continues.

Applicable Manual

Title

8-bit Single-chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Timer Clock Input Source

Question

The clock input source to the timer can be selected by software from the following three input sources.

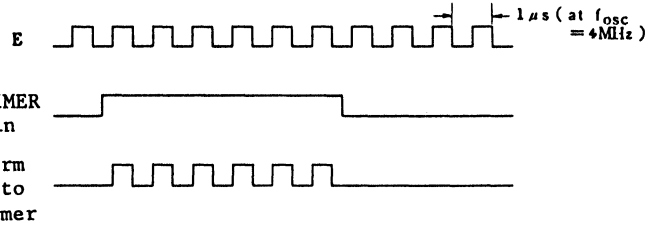
① Internal clock E
② TIMER-pin-controlled E
③ Event input from the TIMER pin

What is the ② TIMER-pin-controlled E?

Classification	
	Parallel Port
	Serial Port
*	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

This is the AND of the TIMER pin status and the internal clock E (1/4 of the frequency of the crystal oscillator).



When this input source is selected, the duration of the High state of the pulse applied to the TIMER pin can be measured.

Applicable Manual
Title

Other Data Document
Title

Reference Q & A Sheet
No.

QA635-006B
QA635-014B

Supplement

The timer input source selection is controlled by bits 4 and 5 of the TCR (Timer Control Register: $0009).

TCR		Clock input source
Bit 5	Bit 4	
0	0	Internal clock E
0	1	TIMER-pin-controlled E
1	0	No clock input (count stopped)
1	1	Event input from TIMER pin

8

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Schmitt Trigger Circuit of Interrupt Pin

Question

Do interrupt pins $\overline{RES}$, $\overline{STBY}$, $\overline{INT}$ and $\overline{INT_2}$ contain a Schmitt trigger circuit?

Answer

The $\overline{RES}$ and $\overline{STBY}$ contain a Schmitt trigger circuit, but the $\overline{INT}$ and $\overline{INT_2}$ do not.

Classification	
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Servicing Timer Interrupt while Masked

Question

While the timer interrupt is masked (TCR6=1 or CCR I bit=1), if a timer interrupt occurs and then the interrupt is enabled (TCR6=0 and CCR I bit=0), will the timer interrupt be serviced?

Answer

Yes. Even while the timer interrupt is masked, a timer interrupt occurs and TCR7 (timer interrupt request flag) is set to 1. TCR7 retains this value until 0 is entered in TCR7 by software.
Therefore, the timer interrupt is serviced when TCR6 and CCR I bit are set to 0 after TCR7 becomes 1.
To prevent the servicing of any timer interrupts, set TCR6 and the CCR I bit to 0 after entering 0 in TCR7.

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title: 8-bit Single-chip Microcomputer Data Book

Other Data Document

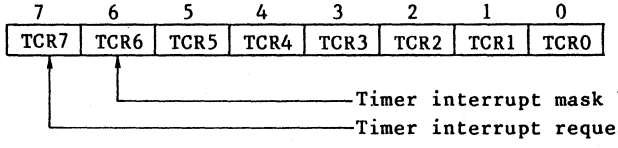
Title:

Reference Q & A Sheet

No.:

Supplement

TCR (Timer Control Register: $0009)



Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Servicing $\overline{\text{INT}}$ External Interrupt while Masked		

Question

If an $\overline{\text{INT}}$ interrupt occurs while it is masked (CCR I bit=1), and then the interrupt is enabled (CCR I bit=0), will the $\overline{\text{INT}}$ interrupt be serviced?

Answer

Yes. The CPU will service the $\overline{\text{INT}}$ interrupt immediately after the CCR I bit becomes 0. Even while the $\overline{\text{INT}}$ interrupt is masked (CCR I bit=1), an interrupt request occurs at the falling edge of the $\overline{\text{INT}}$ signal, and is latched until the $\overline{\text{INT}}$ interrupt servicing routine is executed. When the interrupt is enabled (CCR I bit=0), the CPU services the interrupt.

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title

8-bit Single-chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-010B

Supplement

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Servicing $\overline{INT2}$ External Interrupt while Masked

Question

If an $\overline{INT_2}$ interrupt occurs while masked (MR6=1 or CCR I bit =1), and then the interrupt is enabled (MR6=0 and CCR I bit=0), will the $\overline{INT_2}$ interrupt be serviced?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Yes. The CPU will service the $\overline{INT_2}$ interrupt immediately after MR6 and CCR I bit become 0.
Even while the interrupt is masked (MR6=1 or CCR I bit=1), an $\overline{INT_2}$ interrupt request occurs at the falling edge of the $\overline{INT_2}$ signal and MR7 is set to 1. Since MR7 retains the value until 0 is entered in MR7 by software, the interrupt request will be serviced when the interrupt is enabled.
To prevent the servicing of any $\overline{INT_2}$ interrupts that occur while masked, enter 0 in the MR7 before enabling the interrupts.

Applicable Manual

Title: 8-bit Single-chip Microcomputer Data Book

Other Data Document

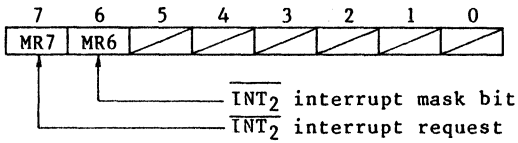
Title:

Reference Q & A Sheet

No.: QA635-009

Supplement

MR (Miscellaneous Register: $000A)



8

No. QA635-011B

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Servicing an Interrupt after a Reset (CCR I bit initializing)

Question

What is the status of the I (Interrupt) bit of the CCR (Condition Code Register) after a reset?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

After a reset, the CPU sets the I bit to 1, servicing no maskable interrupts until the I bit is cleared.

Applicable Manual

Title: 8-bit Single-chip Microcomputer Data Book

Other Data Document

Title:

Reference Q & A Sheet

No.:

Supplement

Maskable interrupts: External interrupts ($\overline{INT}$, $\overline{INT_2}$)
Internal timer interrupts (TIMER, TIMER2)
Serial interrupt (SCI)

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Servicing External Interrupt after Returning from Standby Mode		

Question

If an external interrupt ($\overline{INT}$ or $\overline{INT_2}$) occurs in standby mode, and then the MCU returns to active mode with INT or INT_2 being Low, will the interrupt be serviced?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

No. As you know, no external interrupts is serviced in standby mode. $\overline{INT}$ and $\overline{INT_2}$ interrupt requests are serviced at the falling edges of the signals. However, in this case, no falling edges occurs after the MCU returns to active mode from standby mode, causing no interrupts to be serviced.

Applicable Manual

Title

8-bit Single-chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

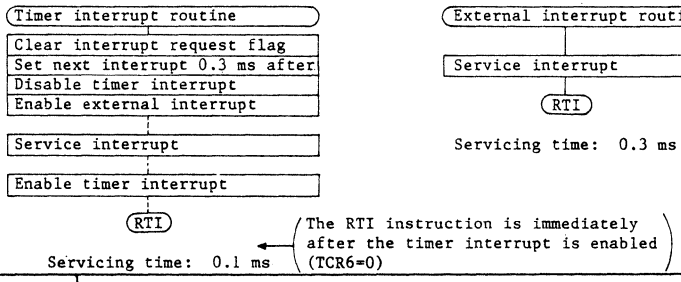
No.

Supplement

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Servicing Multiple Interrupts		

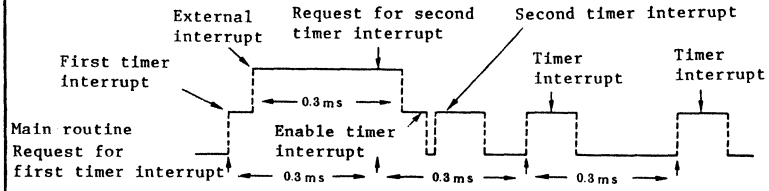
Question

In a program having the interrupt servicing routines shown below, an external interrupt occurs during the execution of the servicing routine for the first timer interrupt. Then the second timer interrupt occurs during the execution of the external interrupt servicing routine. When will the second timer interrupt be serviced?

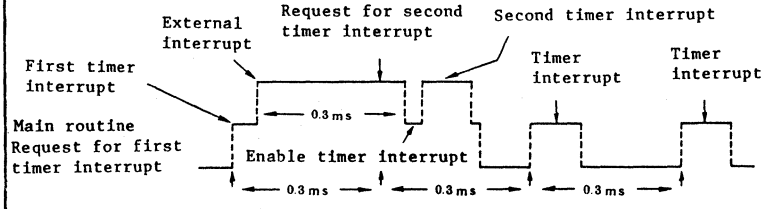


Answer

The second timer interrupt is serviced in the main routine immediately after the RTI instruction is executed.



If there are 2 or more cycles between the timer interrupt enable instruction and the RTI instruction in the timer interrupt routine, the second timer interrupt is serviced in the same timer interrupt routine.



Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Time from Interrupt Occurrence to Interrupt Servicing Routine Execution		

Question

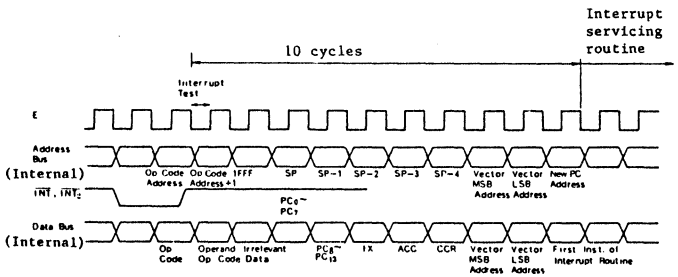
How many cycles are required between the occurrence of an interrupt request and the execution of an interrupt servicing routine?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

10 cycles are needed.
At the last cycle of the instruction being executed, the CPU checks for the existence of an interrupt request. If there is one, control is passed to the corresponding interrupt servicing routine 10 cycles after the interrupt request occurrence.
Examples INT and INT_2 are shown below.



Interrupt Sequence

Applicable Manual

Title

8-bit Single-chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

8

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Servicing SCI (Serial I/O) Interrupt while Masked		

Question

While the SCI (serial I/O) interrupt is masked (SSR5=1, CCR I bit=1), an SCI interrupt request (SSR7=1) occurs after the SCI completes data transmitting or receiving. If the SCI interrupt is then enabled (SSR=0, CCR I bit=0), will this SCI interrupt be serviced?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Yes. Even while the SCI interrupt is masked, an SCI interrupt occurs and SSR7 (SCI interrupt request bit) is set to 1. SSR7 retains this value until 0 is entered in SSR7 by software, or data is read/written from/into the SDR (Serial Data Register).
Therefore, the SCI interrupt is serviced when SSR5 and CCR I bit are set to 0 after SSR7 becomes 1.
To prevent the servicing of any SCI interrupts, enter 0 in SSR7 by software, or set SSR5 and CCR I bit to 0 after reading/writing data from/into the SDR.

Applicable Manual

Title: 8-bit Single-chip Microcomputer Data Book

Other Data Document

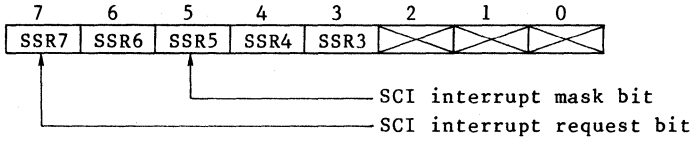
Title:

Reference Q & A Sheet

No.:

Supplement

SSR (SCI Status Register: $0011)



Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Timing of External Clock Input to the Oscillator and E Clock Timing		

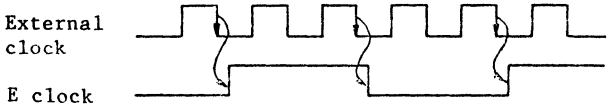
Question

If an external clock is input to the oscillator's EXTAL pin, with which edge of the external clock (rising or falling) does the E clock synchronize?

Classification	
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
*	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

The E clock synchronizes with the falling edge of the external clock.



Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-035A

Supplement

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Port Status at a Reset		

Question

What are the statuses of the ports at a reset?

Answer

The ports are initialized as shown below.

Port	DDR	Data register	Status at a reset
A, B, C (I/O pin)	0	0	Designated as an input ports (high impedance)
D (Input pin)	-	-	High impedance
E, F (Output pin)	-	0	0 output
G (I/O pin)	0	0	Designated as an input port (high impedance)

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
*	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title

8-bit Single-chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-001B
QA635-016B

Supplement

The HD6305X1/X2 and HD6305Y1/Y2 do not have ports E, F and G.

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Bus Status at a Reset

Question

What are the statuses of the address bus, data bus and R/$\overline{W}$ signal at a reset?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
*	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Their statuses are as shown below.

	Status at a reset
Address bus	Outputs $1FFF
Data bus	High impedance
R/$\overline{W}$ signal	High level (read)

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-015B
QA635-017B

Supplement

The above answer does not apply to the HD6305X0 and HD6305Y0 since these units cannot be expanded externally.

8

No. QA635-017B

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Bus Status in Low-Power-Consumption Modes		

Question

What are the statuses of the address bus, data bus and R/$\overline{W}$ signal in low-power-consumption modes (wait, stop and standby)?

Classification	
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Their statuses are as shown below.

Mode	Address bus	Data bus	R/W signal	E pin
Wait	$1FFF	High impedance	High level	E clock output
Stop	$1FFF	High impedance	High level	Low level
Standby	High impedance	High impedance	High impedance	High impedance

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

The above answer does not apply to the HD6305X0 and HD6305Y0 since these units cannot be expanded externally.

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

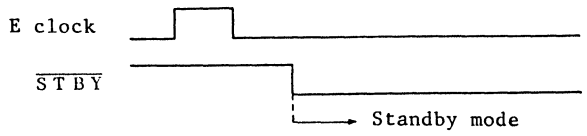
Theme: Executing an Instruction when Entering Standby Mode

Question

The MCU enters standby mode by setting the STBY pin in Low. What will happen to the instruction that was being executed at that time?

Answer

The instruction execution is stopped since the MCU enters standby mode whether or not an instruction execution sequence is executed. The MCU enters standby mode immediately after the STBY pin becomes Low.
In standby mode, the internal oscillator is stopped and the contents of the MCU internal registers are destroyed. So, be sure to perform a reset-start to return from standby mode (the built-in RAM values are retained).



Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title: 8-bit Single-chip Microcomputer Data Book

Other Data Document

Title:

Reference Q & A Sheet

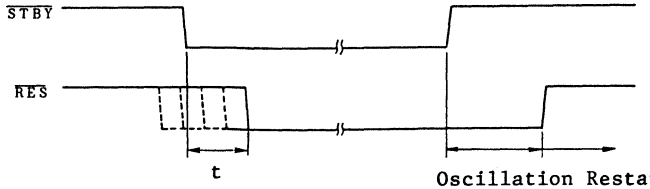
No.: QA635-019B

Supplement

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Standby Mode Timing		

Question

In the following chart showing the timing at which the MCU enters standby mode, is there any limit to the value of t? In addition, at what timing should Vcc be dropped?



Timing Chart of Returning from Standby Mode

Answer

There is no limit to the value of t.
Either the STBY pin or RES pin can be Low first. However, the RES must be raised 20 msec or more behind the rising edge of the STBY to restart operation.
The Vcc must be dropped after the STBY becomes Low. It should be returned to the specified level before the STBY returns to High.

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title

8-bit Single-chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-018B
QA635-020B

Supplement

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Returning from Standby Mode

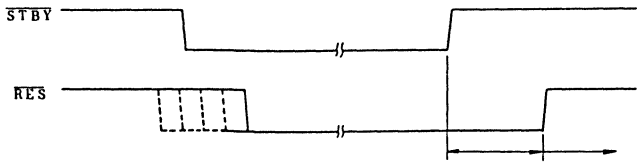
Question

Is there any method of returning from standby mode other than reset-start?

Classification	
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

No. In standby mode, the internal oscillator is stopped and the MCU internal register values are destroyed. (The contents of the Program Counter are also destroyed.) Therefore, only setting the STBY pin in High caused the MCU to burst.
Be sure to perform a reset-start to return from standby mode.



Timing Chart of Returning from Standby Mode

Oscillation Restart Stabilizing Time (t_{OSC}=20 ms)

Applicable Manual

Title: 8-bit Single-chip Microcomputer Data Book

Other Data Document

Title:

Reference Q & A Sheet

No.: QA635-019B

Supplement

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Entering Low-Power-Consumption Modes

Question

Does executing the WAIT or STOP instruction always cause the MCU to enter wait or stop mode?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

No. The MCU enters wait or stop mode only in the following conditions.

① When there is no interrupt requests.
(The $\overline{RES}$, $\overline{STBY}$ and $\overline{INT}$ pins are all High and the $\overline{INT}$ interrupt latch and all interrupt bits are cleared.)

② When the $\overline{INT_2}$ and internal interrupts are disabled by the mask bits, and there is no other interrupts.
(The $\overline{RES}$, $\overline{STBY}$ and $\overline{INT}$ pins are all High, the $\overline{INT}$ interrupt latch is cleared, and each interrupt mask bit is set.)

If the above conditions are not met, the MCU executes the next instruction after the execution of the WAIT and STOP instructions which requires 4 cycles.

Applicable Manual

Title:

Other Data Document

Title:

Reference Q & A Sheet

No.:

Supplement

In case ② above, the absence or presence of the $\overline{INT_2}$ and internal interrupt requests has no effect.

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Entering Wait Mode

Question

Are there any precautions about entering wait mode by the WAIT instruction execution?

Classification	
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Note that the method of entering wait mode differs depending on the returning method from wait mode.

Operation after a return \ Returning method from wait mode	$\overline{\text{INT}}$ interrupt	$\overline{\text{INT}_2}$, TIMER, TIMER2, SCI
Executing interrupt routine	Clear the CCR I bit. Set all interrupt mask bits.	Clear the CCR I bit. Set all mask bits except the one for the interrupt to be used. Do not cause any $\overline{\text{INT}}$ interrupts in wait mode.
Executing an operation after the WAIT instruction*	Set the CCR I bit. Set all interrupt mask bits.	Set the CCR I bit. Set all mask bits except the one for the interrupt to be used. Do not cause any $\overline{\text{INT}}$ interrupts in wait mode.

*; The $\overline{\text{INT}}$ interrupt requested (by detecting a falling edge in the $\overline{\text{INT}}$ pin) before the WAIT instruction execution has already been serviced.

Applicable Manual
Title

Other Data Document
Title

Reference Q & A Sheet
No.

Supplement

CCR (Condition Code Register) I bit: Interrupt mask bit (except software interrupts)

Interrupt mask bits:
- $\overline{\text{INT}_2}$ ——— Miscellaneous Register (MR: $0A) bit 6
- TIMER ——— Timer control Register (TCR: $09) bit 6
- TIMER2 ——— SCI Status Register (SSR: $11) bit 4
- SCI ——— SCI Status Register (SSR: $11) bit 5

8

No. QA635-029B

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Entering Stop Mode		

Question

Are there any precautions about entering stop mode by the STOP instruction execution?

Classification	
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Note that the method of entering stop mode differs depending on the returning method from stop mode.

Operation after a return \ Returning method from stop mode	$\overline{INT}$ interrupt	$\overline{INT_2}$ interrupt
Executing interrupt routine	Clear the CCR I bit. Set the INT_2 interrupt mask bit (MR6).	Clear the CCR I bit. Set all mask bits except the one for the interrupt to be used. Do not cause any $\overline{INT}$ interrupts in stop mode.
Executing an operation after the STOP instruction*	Set the CCR I bit. Set the INT_2 interrupt mask bit (MR6).	Set the CCR I bit. Set all mask bits except the one for the interrupt to be used. Do not cause any $\overline{INT}$ interrupts in stop mode.

*; The $\overline{INT}$ interrupt requested (by detecting a falling edge in the $\overline{INT}$ pin) before the STOP instruction execution has already been serviced.

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

CCR (Condition Code Register) I bit: Interrupt mask bit (except software interrupts)

$\overline{INT_2}$ interrupt mask bit ----- Miscellaneous Register (MR: $0A) bit 6

No. QA635-038A

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Returning Time from Stop Mode

Question

How long is the internal delay when the MCU returns from stop mode to operating mode by the interrupt INT or INT_2.

Answer

It is approximately 14 ms when a 4-MHz crystal oscillator is used.
The internal delay time indicates the period while the CPU execution is stopped and until the oscillating circuit which was stopped in stop mode performs stabilized oscillation. This delay is automatically generated in the MCU.

Classification	
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

8

No. QA635-039A

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Current Consumption in Low-Power-Consumption Mode

Question

Is the Icc (current consumption) in low-power-consumption modes which is specified in the data sheets the value when no load is applied to the I/O ports?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Yes. Since the I/O ports maintain their output levels even in stop and wait modes, the Icc will be increased by I_{OH} or I_{OL} when a load is applied to the ports.
In standby mode, however, the I/O ports are in the high impedance state, and the Icc is the same with or without load.

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Accessing Not Used Areas on Memory Map		

Question

What will happen if data is read from or written into the area designated as Not Used on the memory map?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
*	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Nothing will happen as long as the areas are not addresses $13 to $1F.
Never access the $13 to $1F areas since they are used for IC testing. Accessing (reading/writing) these areas causes the MCU to burst.

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

8

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Using Bit Manipulating Instruction for Output Ports

Question

Is it possible to use the bit manipulating instruction for the Data Registers of the output ports (ports A, B, C and G used for output and output ports E and F)?

Answer

Yes.
The bit manipulating instruction automatically reads out an address, manipulates data in it and re-enters a result in the address. This instruction can be used for any read/write registers.
However, when data is read from the Port Data Register, data output to port G is determined by a pin voltage. That is, a voltage of less than 2.0 V due to the external circuit causes incorrect data to be output to port G. Design the external circuit appropriately to prevent incorrect data output.
To ports A, B, C, E and F, the logical levels stored in the Port Data Register are output.

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
*	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title: 8-bit Single-chip Microcomputer Data Book

Other Data Document

Title:

Reference Q & A Sheet

No.:

Supplement

The HD6305X1/X2 and HD6305Y1/Y2 do not have port E, F and G.

Type	HD6305X0/X1/X2 HD6305Y0/Y1/Y2	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Statuses of Address Bus, Data Bus and Control Line when the Internal Address Space is Accessed		

Question

What values are output to the address bus, data buses and control line when the CPU access the internal address space?

Answer

The same value as the accessed data is output to the address bus and control line, whether the CPU accesses the internal address space or external address space and whether the CPU reads or writes data.
However, the data buses are in the high impedance state when the CPU read data from the internal address space, though the same value as the data written into the internal space by the CPU is output to the data buses.

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
*	Data Buffer
	Others

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

8

Section Eight

APPENDIX:

Technical Q and A (Part II)

8-Bit Single-Chip Microcomputer

HD6305U0, HD6305V0

The HD6305U0 and HD6305V0 are microprogram-controlled 8-bit single-chip microcomputers that use CMOS 2.5-µm technology. These models are 40-pin versions of the HD6305X0. The CPU, memory, timer, serial communications interface (SCI), and I/O are all integrated into one chip. The only difference between the HD6305U0 and the HD6305V0 lies in their memory capacities. The former has a 2-kbyte ROM and a 128-byte RAM; the latter has a 4-kbyte ROM and a 192-byte RAM.

The CMOS technology enables these models to operate over a wide range of supply voltages (Vcc operates in the 3 V - 6 V range at operating frequencies of 0.1 MHz - 0.5 MHz) with less power consumption. The low-power-consumption modes (STOP, WAIT, and STANDBY) available with these models permit further power savings.

The instruction set includes DAA (decimal adjust instruction), and STOP and WAIT (used to enter the corresponding low-power-consumption modes), in addition to the HD6805 instruction set. The minimum instruction execution time is 0.5 µsec/cycle (f=2 MHz), permitting high-speed operation.

This TECHNICAL QUESTIONS AND ANSWERS is a user reference manual that has been compiled in question-and-answer form. It is based on technical inquiries from HITACHI microcomputer users.

This manual should be used in conjunction with the appropriate data book (which you should already have). You can make best use of this manual either by: reading all of it before you start to design any microcomputer-applied products, in order to strengthen your technical background; or referring to it during the actual design process, using it to help you solve specific problems that may arise.

Although some of the items supplement the data book, most of them are inquiries from the users. In order to meet future needs, we are prepared to increase the number of Q & A items and to update the data book.

Contents

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Outputting Data from Ports after a Reset		

Question

Which operation should be performed first to output data through input ports A, B, C and D after a reset?; storing data in the Data Register of the ports or designating the corresponding DDR (Data Direction Register)?

Classification

*	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Store the data to be output in the Data Register first, and then designate the corresponding DDR to data output (DDR=1).
Since a reset causes the Data Register to become 0, if the DDR is designated to data output before data is stored in the Data register, 0 is output to the ports.

Applicable Manual

Title

HD6305U0, HD6305V0
Data Sheets
8-bit Single-Chip Micro-computer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-328A

Supplement

8

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Serial I/O Pin Status

Question

After ports D_3 to D_5 are respectively used as $\overline{CK}$, Rx and Tx pins of the SCI (serial I/O), these ports are specified as I/O pins using the SCR (SCI Control Register) (SCR7=SCR6=SCR5=0). What is the value of the DDR (Data Direction Register) when the ports are used as I/O pins?

Classification

*	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

The DDR retains the value for the SCI, as shown below.

Pin	DDR value for SCI	Notes
D_3/Tx	1 (output port)	When Tx is used, SCR7=1.
D_4/Rx	0 (input port)	When Rx is used, SCR6=1.
$D_5/\overline{CK}$	0 (input port)	In the case of external clock source mode.
	1 (output port)	In the case of internal clock source mode.

Even when the SCR value is changed after the SCI is used, the DDR value shown in the above table is maintained. When using D_3 to D_5 as I/O ports, rewrite the DDR value using software.

Applicable Manual

Title: HD6305U0, HD6305V0 Data Sheets 8-bit Single-Chip Microcomputer Data Book

Other Data Document

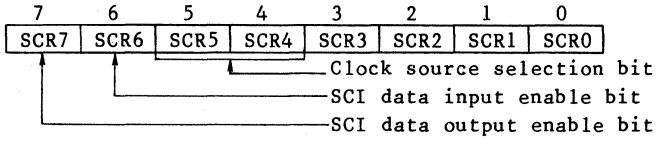
Title:

Reference Q & A Sheet

No.:

Supplement

SCR (SCI Control Register: $0010)



Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Using Port D when Serial I/O is Used

Question

When ports D_3 to D_5 are used as the SCI (serial I/O), is it possible to use ports D_0 to D_2 and D_6 as usual I/O pins?

Classification

*	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Yes. These ports can be used as usual I/O pins because they are independent of the SCI. The ports affected by the SCR (SCI Control Register) when the SCI is used are ports D_3 to D_5 only. Input and output functions of ports D_0 to D_2 and D_6 are selected using bits 0 to 2 and 6 of port D's DDR (Data Direction Register).

Applicable Manual

Title: HD6305U0, HD6305V0 Data Sheets 8-bit Single-Chip Microcomputer Data Book

Other Data Document

Title:

Reference Q & A Sheet

No.:

Supplement

8

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Designating Input or Output Operation of Serial I/O Clock Pin

Question

Does setting 0 or 1 in the corresponding DDR (Data Direction Register) enable the clock pin $\overline{CK}$ to be designated as an input or output pin when the SCI (serial I/O) is used?

Classification

	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

No. Bits 4 and 5 (SCR4, 5) of the SCR (SCI Control Register) designate the $\overline{CK}$ pin as input or output when the SCI is used, not the DDR.

As shown below, the combination of bits 4 and 5 enables the SCI clock source selection, determining whether the $\overline{CK}$ pin is specified as an input or output pin.

SCR5	SCR4	Clock source	$\overline{CK}$ pin (port D_5)
0	0	—	Used as I/O pin (according to the DDR)
0	1		
1	0	Internal	Clock output (DDR output)
1	1	External	Clock input (DDR input)

Note: The selection of input or output is made by the DDR when SCR5 is 0 and SCR4 is 0 or 1.

Applicable Manual

Title: HD6305U0, HD6305V0 Data Sheets 8-bit Single-Chip Microcomputer Data Book

Other Data Document

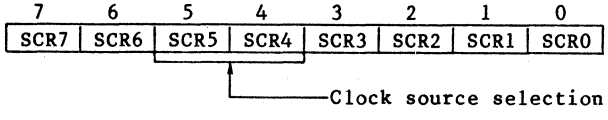
Title:

Reference Q & A Sheet

No.:

Supplement

SCR (SCI Control Register: $0010)



Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme	Using SDR (SCI Data Register) when Serial I/O is not Used

Question

Is it possible to use the SDR (SCR Data Register: $0012) as a general-purpose register when the SCI (serial I/O) is not used?

Classification

	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Yes. The SDR can be used as a general-purpose register when the SCI is not used i.e., when the SCI clock is stopped with SCR4 to SCR7 being 0.

Applicable Manual

Title

Other Data Document

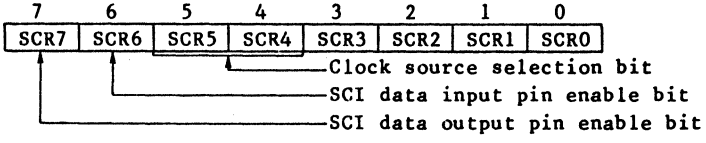
Title

Reference Q & A Sheet

No.

Supplement

SCR (SCI Control Register: $0010)



8

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: SSR7 (SCI Interrupt Request Bit) Set Timing

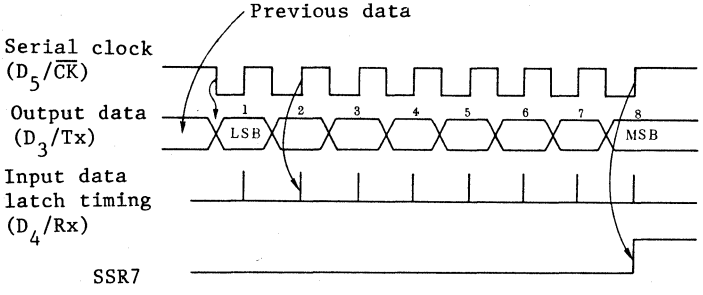
Question

After 8-bit data transmitting or receiving through the SCI (serial I/O) is completed, SSR7 (SCI interrupt request bit) of the SSR (SCI Status Register) is set. At what timing is this setting performed?

Classification	
	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

SSR7 is set at the rising edge of the 8th $\overline{CK}$ (serial clock) as shown below.



Applicable Manual

Title

Other Data Document

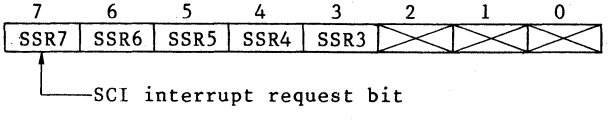
Title

Reference Q & A Sheet

No.

Supplement

SSR (SCI Status Register: $0011)



Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Clearing SSR (SCI Interrupt Request Bit)

Question

After data is transmitted or received through the SCI (serial I/O), SSR7 (SCI interrupt request bit) is set to 1. At this time, if 0 is set in SSR7 by software to clear the bit, is it possible to transmit or receive the next data?

Classification

	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

No. When 0 is set in SSR7 by software, this bit is cleared, and the next data transmitting and receiving cannot be performed. SSR7 is cleared under the following conditions.

① When data is read from or written into the SDR (SCI Data Register).
(After SSR7 is cleared and the octal counter is reset, the next data transmitting/receiving is executed.)

② When 0 is set in SSR7 by software.
(The SCI's octal counter is not reset, enabling no data transmitting/receiving.)

Therefore, repeat data reading or writing from/into the SDR to transmit/receive data repeatedly.
Dummy-read the SDR before receiving the first data.

Applicable Manual

Title:

Other Data Document

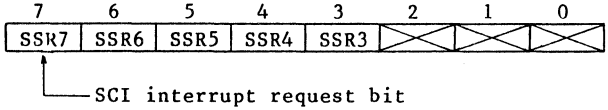
Title:

Reference Q & A Sheet

No.:

Supplement

SSR (SCI Status Register: $0011)



8

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Accessing SDR (SCI Data Register)

Question

What will happen if data is read from or written into the SDR (SCI Data Register) while another data is being transmitted or received through the SCI (serial I/O)?

Classification

	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Normal operation can not be guaranteed for either transmission or reception. Do not access the SDR during data transmitting or receiving since the SCI becomes disabled after the access (the MCU must be reset to use the SCI again).

Applicable Manual

Title:

Other Data Document

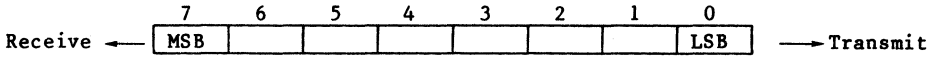
Title:

Reference Q & A Sheet

No.: QA635-312A

Supplement

SDR (SCI Data Register: $0012)



Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme	Transmitting and Receiving Data Simultaneously through Serial I/O

Question

Is is possible to transmit and receive data simultaneously through the SCI (serial I/O) by using the internal clock for transmitting and the external clock for receiving, or vice versa?

Classification

	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

No. Only one clock source can be selected as the SCI transfer clock. Simultaneous data transmission and reception using two transfer clocks are impossible.
When a single clock source is used, data can be transmitted and received at the same time.

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

8

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme	Notes on Receiving Data through SCI in External Clock Mode

Question

What should we pay attention to when using the external clock to receive data through the SCI (serial I/O)?

Classification

	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

The external transfer clock source does not enable the receiving side to check when data is transmitted.
Therefore, it is necessary to read out data in the SDR (SCI Data Register) as soon as the data is received, to prepare for receiving the next data.
Whether or not receiving has been completed can be checked by an SCI interrupt or by testing bit 7 of the SSR (SCI Status Register) by software.

Applicable Manual

Title

Other Data Document

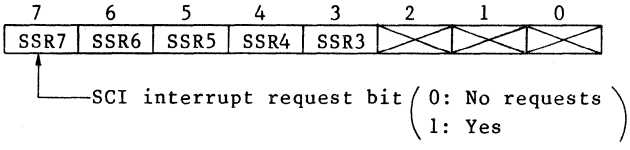
Title

Reference Q & A Sheet

No.

Supplement

SSR (SCI Status Register: $0011)



Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme	SCI Operation in External Clock Mode

Question

The SCI (serial I/O) is in external clock mode;
If the external clock is applied to the CK pin before the CPU writes/reads data into/from the SDR (Serial Data Register) after the one-byte data transmitting/receiving is completed, will the SCI start the next data transmitting/receiving?

Classification	
	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

No. The SCI does not start transmitting or receiving the next data until the CPU writes/reads data into/from the SDR. That is, any clock signal applied to the CK pin before the CPU accesses the SDR will be ignored.

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

The CK pin can also used as D_5 pin. It is controlled by bits 4 and 5 of the SCR (SCI Control Register).

SCR5	SCR4	Clock source	D_5 pin
0	0	-	Used as I/O pin (according to the DDR)
0	1	-	
1	0	Internal	Clock output (DDR output)
1	1	External	Clock input (DDR input)

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Initializing the Transfer Clock Generator Prescaler

Question

The prescaler of the SCI transfer clock generator is initialized by reading/writing data from/into the SDR (SCI Status Register) or by setting 1 in SSR3 (SCI Status Register bit 3). What is the difference of the initialization performed by these two methods?

Classification	
	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

There are differences in the following points.

(1) When data is read/written from/into the SDR, the SCI octal counter is initialized at the same time the prescaler is initialized. This causes the SCI to start transmitting or receiving the next data.
(2) When 1 is set in SSR3, only the prescaler is initialized. The SCI does not start data trasmitting/receiving.

SSR3 is the bit to be used to initialize the prescaler when the transfer clock generator is utilized as Timer 2.

Applicable Manual

Title:

Other Data Document

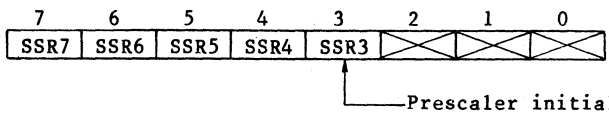
Title:

Reference Q & A Sheet

No.: QA635-308A

Supplement

SSR (SCI Status Register: $0011)



Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: SCI Prescaler Initialize Timing and Clock Output Timing

Question

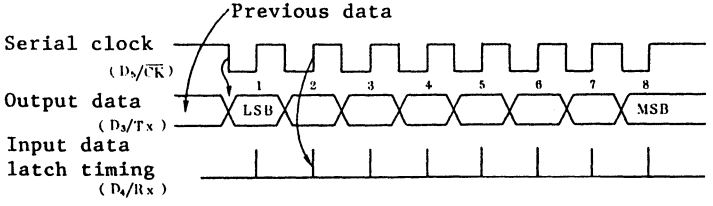
At what timing is the SCI (serial I/O) prescaler initialized?
In addition, when the internal clock is used (SCR4=0, SCR5=1), at what timing is the $\overline{CK}$ (serial clock) output?

Classification

	Parallel Port
*	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

The prescaler is initialized when data is read from or written into the SDR (SCI Data Register).
When the internal clock is used, the $\overline{CK}$ is output at the baud rate specified by bits 0 to 3 of the SCR (SCI Control Register), immediately after the data reading/writing.
The SCI timing chart is shown below.



SCI Timing Chart

Applicable Manual

Title: HD6305U0, HD6305V0 Data Sheets 8-bit Single-Chip Microcomputer Data Book

Other Data Document

Title:

Reference Q & A Sheet

No.:

Supplement

8

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Reading/Writing Data from/into the TDR during Timer Operation		

Question

What will happen if data is written into or read from the TDR (Timer Data Register) during count-down?

Classification

	Parallel Port
	Serial Port
*	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

If data is written into the TDR during count-down, the count-down restarts from this data.
However, data reading during count-down does not affect the TDR value, and the count-down continues.

Applicable Manual

Title

HD6305U0, HD6305V0
Data Sheets
8-bit Single-Chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme	Timer Count-down Timing when External Clock is Input

Question

When an event input from the TIMER pin is used as the clock input source for the built-in timer, at which edge (rising or falling) of the input signal is the count-down performed?

Answer

It is performed at the rising edge of the input signal.

Classification	
	Parallel Port
	Serial Port
*	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-316A

Supplement

8

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Timer Clock Input Source

Question

The clock input source to the timer can be selected by software from the following three input sources.

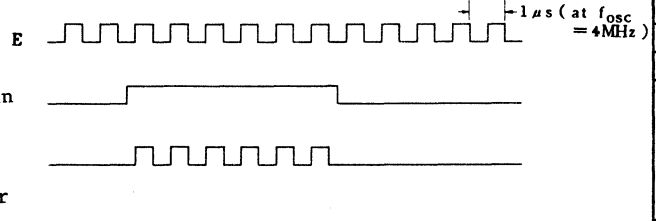
① Internal clock E
② TIMER-pin-controlled E
③ Event input from the TIMER pin

What is the ② TIMER-pin-controlled E?

Classification	
	Parallel Port
	Serial Port
*	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

This is the AND of the TIMER pin status and the internal clock E (1/4 of the frequency of the crystal oscillator).



When this input source is selected, the duration of the High state of the pulse applied to the TIMER pin can be measured.

Applicable Manual

Title:

Other Data Document

Title:

Reference Q & A Sheet

No.:

QA635-315A
QA635-327A

Supplement

The timer input source selection is controlled by bits 4 and 5 of the TCR (Timer Control Register: $0009).

TCR		Clock input source
Bit 5	Bit 4	
0	0	Internal clock E
0	1	TIMER-pin-controlled E
1	0	No clock input (count stopped)
1	1	Event input from TIMER pin

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Timer 2 Interrupt Cycles

Question

The Timer 2 interrupt cycle is selected by bits 0 to 3 of the SCR (SCI Control Register). Are there any limits to the selection?

Classification	
	Parallel Port
	Serial Port
*	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

The selectable Timer 2 interrupt cycles are as shown below.

	SCR3	SCR2	SCR1	SCR0	Timer 2 interrupt cycle	
					f_{osc} = 4 MHz	f_{osc} = 4.194 MHz
Not selectable	0	0	0	0	1 μs	0.95 μs
	0	0	0	1	2 μs	1.91 μs
Selectable	0	0	1	0	4 μs	3.82 μs
	0	0	1	1	8 μs	7.64 μs
	/	/	/	/	/	/
	1	1	1	1	32768 μs	1/32 s

Applicable Manual

Title: HD6305U0, HD6305V0 Data Sheets 8-bit Single-Chip Microcomputer Data Book

Other Data Document

Title:

Reference Q & A Sheet

No.:

Supplement

SCR (SCI Control Register: $0010)

7	6	5	4	3	2	1	0
SCR7	SCR6	SCR5	SCR4	SCR3	SCR2	SCR1	SCR0

SCR3: Transfer clock rate selection bit
SCR4: Clock source selection bit
SCR6: SCI data input enable bit
SCR7: SCI data output enable bit

8

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Time from Interrupt Occurrence to Interrupt Servicing Routine Execution		

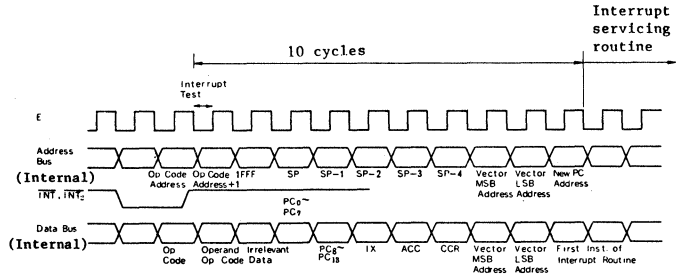
Question

How many cycles are required between the occurrence of an interrupt request and the execution of an interrupt servicing routine?

Classification	
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

10 cycles are needed.
At the last cycle of the instruction being executed, the CPU checks for the existence of an interrupt request. If there is one, control is passed to the corresponding interrupt servicing routine 10 cycles after the interrupt request occurrence.
Examples INT and INT_2 are shown below.



Interrupt Sequence

Applicable Manual
Title

Other Data Document
Title

Reference Q & A Sheet
No.

Supplement

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme	Schmitt Trigger Circuit of Interrupt Pin

Question

Do interrupt pins $\overline{RES}$, $\overline{STBY}$, $\overline{INT}$ and $\overline{INT_2}$ contain a Schmitt trigger circuit?

Classification	
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

The $\overline{RES}$ and $\overline{STBY}$ contain a Schmitt trigger circuit, but the INT and $\overline{INT_2}$ do not.

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

8

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme	Servicing an Interrupt after a Reset (CCR I bit initializing)

Question

What is the status of the I (Interrupt) bit of the CCR (Condition Code Register) after a reset?

Classification	
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

After a reset, the CPU sets the I bit to 1, servicing no maskable interrupts until the I bit is cleared.

Applicable Manual

Title

HD6305U0, HD6305V0
Data Sheets
8-bit Single-Chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

Maskable interrupts: External interrupts ($\overline{INT}$, $\overline{INT_2}$)
Internal timer interrupts (TIMER, TIMER2)
Serial interrupt (SCI)

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Servicing $\overline{\text{INT}}$ External Interrupt while Masked

Question

If an $\overline{\text{INT}}$ interrupt occurs while it is masked (CCR I bit=1), and then the interrupt is enabled (CCR I bit=0), will the $\overline{\text{INT}}$ interrupt be serviced?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Yes. The CPU will service the $\overline{\text{INT}}$ interrupt immediately after the CCR I bit becomes 0. Even while the $\overline{\text{INT}}$ interrupt is masked (CCR I bit = 1), an interrupt request occurs at the falling edge of the $\overline{\text{INT}}$ signal, and is latched until the $\overline{\text{INT}}$ interrupt servicing routine is executed. When the interrupt is enabled (CCR I bit=0), the CPU services the interrupt.

Applicable Manual

Title: HD6305U0, HD6305V0 Data Sheets 8-bit Single-Chip Microcomputer Data Book

Other Data Document

Title:

Reference Q & A Sheet

No.: QA635-322A

Supplement

8

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Servicing $\overline{INT2}$ External Interrupt while Masked

Question

If an $\overline{INT_2}$ interrupt occurs while masked (MR6=1 or CCR I bit =1), and then the interrupt is enabled (MR6=0 and CCR I bit=0), will the $\overline{INT_2}$ interrupt be serviced?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Yes. The CPU will service the $\overline{INT_2}$ interrupt immediately after MR6 and CCR I bit become 0.
Even while the interrupt is masked (MR6=1 or CCR I bit=1), an $\overline{INT_2}$ interrupt request occurs at the falling edge of the $\overline{INT_2}$ signal and MR7 is set to 1. Since MR7 retains the value until 0 is entered in MR7 by software, the interrupt request will be serviced when the interrupt is enabled.
To prevent the servicing of any $\overline{INT_2}$ interrupts that occur while masked, enter 0 in the MR7 before enabling the interrupts.

Applicable Manual

Title: HD6305U0, HD6305V0 Data Sheets 8-bit Single-Chip Microcomputer Data Book

Other Data Document

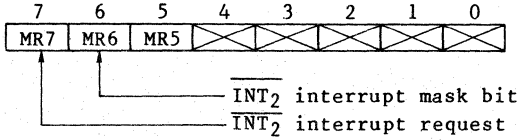
Title:

Reference Q & A Sheet

No.: QA635-321A

Supplement

MR (Miscellaneous Register: $000A)



Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme	Servicing SCI (Serial I/O) Interrupt while Masked

Question

While the SCI (serial I/O) interrupt is masked (SSR5=1 or CCR I bit=1), an SCI interrupt request (SSR7=1) occurs after the SCI completes data transmitting or receiving. If the SCI interrupt is then enabled (SSR5=0 and CCR I bit=0), will this SCI interrupt be serviced?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Yes. Even while the SCI interrupt is masked, an SCI interrupt occurs and SSR7 (SCI interrupt request bit) is set to 1. SSR7 retains this value until 0 is entered in SSR7 by software, or data is read/written from/into the SDR (Serial Data Register).
Therefore, the SCI interrupt is serviced when SSR5 and CCR I bit are set to 0 after SSR7 becomes 1.
To prevent the servicing of any SCI interrupts, enter 0 in SSR7 by software, or set SSR5 and CCR I bit to 0 after reading/writing data from/into the SDR.

Applicable Manual

Title: HD6305U0, HD6305V0 Data Sheets 8-bit Single-Chip Microcomputer Data Book

Other Data Document

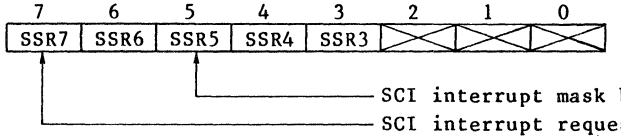
Title:

Reference Q & A Sheet

No.:

Supplement

SSR (SCI Status Register: $0011)



8

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Servicing Timer Interrupt while Masked

Question

While the timer interrupt is masked (TCR6=1 or CCR I bit=1), if a timer interrupt occurs and then the interrupt is enabled (TCR6=0 and CCR I bit=0), will the timer interrupt be serviced?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Yes. Even while the timer interrupt is masked, a timer interrupt occurs and TCR7 (timer interrupt request flag) is set to 1. TCR7 retains this value until 0 is entered in TCR7 by software.
Therefore, the timer interrupt is serviced when TCR6 and CCR I bit are set to 0 after TCR7 becomes 1.
To prevent the servicing of any timer interrupts, set TCR6 and the CCR I bit to 0 after entering 0 in TCR7.

Applicable Manual

Title: HD6305U0, HD6305V0 Data Sheets 8-bit Single-Chip Microcomputer Data Book

Other Data Document

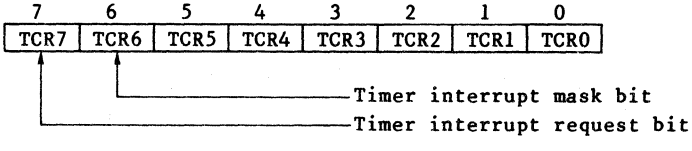
Title:

Reference Q & A Sheet

No.:

Supplement

TCR (Timer Control Register: $0009)



Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme	Servicing External Interrupt after Returning from Standby Mode

Question

If an external interrupt ($\overline{INT}$ or $\overline{INT_2}$) occurs in standby mode, and then the MCU returns to active mode with $\overline{INT}$ or $\overline{INT_2}$ being Low, will the interrupt be serviced?

Answer

No. As you know, no external interrupts is serviced in standby mode. $\overline{INT}$ (when MR5=0) and $\overline{INT_2}$ interrupt requests are serviced at the falling edges of the signals. However, in this case, no falling edges occurs after the MCU returns to active mode from standby mode, causing no interrupts to be serviced.
For $\overline{INT}$, however, Low-level sense is possible depending on the MR5 (Miscellaneous Register bit 5) value. When MR5 is 1, the $\overline{INT}$ external interrupt is serviced after the MCU returns from standby mode.

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title

HD6305U0, HD6305V0
Data Sheets
8-bit Single-Chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

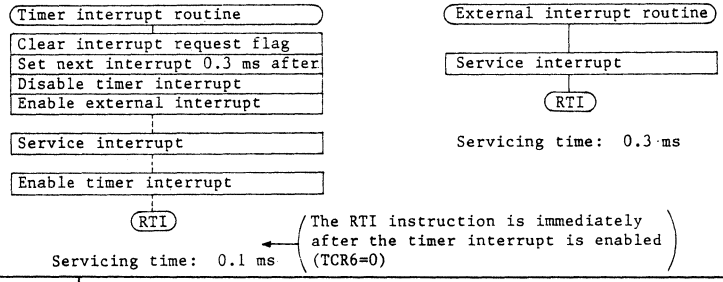
8

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Servicing Multiple Interrupts

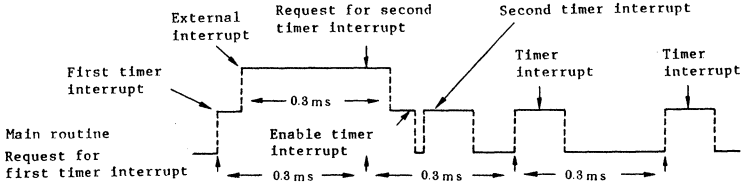
Question

In a program having the interrupt servicing routines shown below, an external interrupt occurs during the execution of the servicing routine for the first timer interrupt. Then the second timer interrupt occurs during the execution of the external interrupt servicing routine. When will the second timer interrupt be serviced?

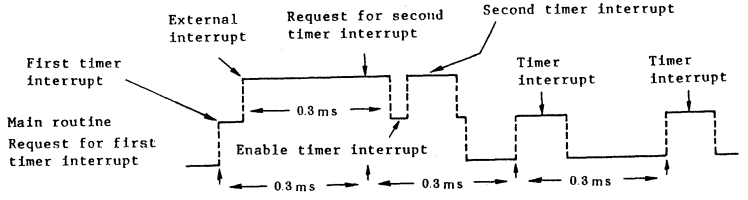


Answer

The second timer interrupt is serviced in the main routine immediately after the RTI instruction is executed.



If there are 2 or more cycles between the timer interrupt enable instruction and the RTI instruction in the timer interrupt routine, the second timer interrupt is serviced in the same timer interrupt routine.



Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
*	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

No. QA635-327A

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Timing of External Clock Input to the Oscillator and E Clock Timing

Question

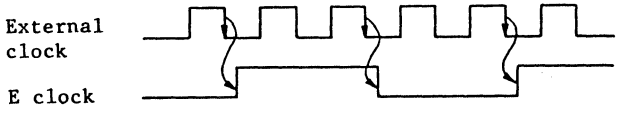
If an external clock is input to the oscillator's EXTAL pin, with which edge of the external clock (rising or falling) does the E clock synchronize?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
*	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

The E clock synchronizes with the falling edge of the external clock.



Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-316A

Supplement

8

No. QA635-328A

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Port Status at a Reset		

Question

What are the statuses of the ports at a reset?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
*	Reset
	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

The ports are initialized as shown below.

	Ports A to D (I/O)
DDR (Data Direction Register)	0
Data Register	0
Port status at a reset	Designated as input ports (high impedance)

Applicable Manual

Title

HD6305U0, HD6305V0
Data Sheets
8-bit Single-Chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-301A

Supplement

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Entering Low-Power-Consumption Modes

Question

Does executing the WAIT or STOP instruction always cause the MCU to enter wait or stop mode?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

No. The MCU enters wait or stop mode only in the following conditions.

① When there is no interrupt requests.
(The RES, STBY and $\overline{INT}$ pins are all High and the $\overline{INT}$ interrupt latch and all interrupt bits are cleared.)

② When the $\overline{INT_2}$ and internal interrupts are disabled by the mask bits, and there is no other interrupts.
(The RES, STBY and $\overline{INT}$ pins are all High, the $\overline{INT}$ interrupt latch is cleared, and each interrupt mask bit is set.)

If the above conditions are not met, the MCU executes the next instruction after the execution of the WAIT and STOP instructions which requires 4 cycles.

Applicable Manual

Title:

Other Data Document

Title:

Reference Q & A Sheet

No.: QA635-330A

Supplement

In case ② above, the absence or presence of the $\overline{INT_2}$ and internal interrupt requests has no effect.

8

No. QA635-330A

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Entering Wait Mode

Question

Are there any precautions about entering wait mode by the WAIT instruction execution?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Note that the method of entering wait mode differs depending on the returning method from wait mode.

Returning method from wait mode	$\overline{\text{INT}}$ interrupt	$\overline{\text{INT}_2}$, TIMER, TIMER2, SCI
Notes on entering wait mode	Set all interrupt mask bits.	Set all mask bits except the one for the interrupt to be used. Do not cause any $\overline{\text{INT}}$ interrupts in wait mode.

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-329A

Supplement

Interrupt mask bits:

- $\overline{\text{INT}_2}$ ——— Miscellaneous Register (MR: $0A) bit 6
- TIMER ——— Timer control Register (TCR: $09) bit 6
- TIMER2 ——— SCI Status Register (SSR: $11) bit 4
- SCI ——— SCI Status Register (SSR: $11) bit 5

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme	Entering Stop Mode

Question

Are there any precautions about entering stop mode by the STOP instruction execution?

Classification	
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Note that the method of entering stop mode differs depending on the returning method from stop mode.

Returning method from stop mode	$\overline{\text{INT}}$ interrupt	$\overline{\text{INT}_2}$ interrupt
Notes on entering stop mode	Set $\overline{\text{INT}_2}$ interrupt mask bit (MR6)	Do not cause any $\overline{\text{INT}}$ interrupts in stop mode.

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

$\overline{\text{INT}_2}$ Interrupt mask bits; Miscellaneous Register (MR: $0A) bit 6

8

No. QA635-332A

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme	Returning Time from Stop Mode

Question

How long is the internal delay when the MCU returns from stop mode to operating mode by the interrupt INT or INT_2.

Answer

It is approximately 14 ms when a 4-MHz crystal oscillator is used.
The internal delay time indicates the period while the CPU execution is stopped and until the oscillating circuit which was stopped in stop mode performs stabilized oscillation. This delay is automatically generated in the MCU.

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

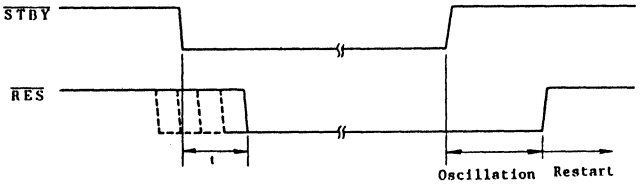
Supplement

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme: Standby Mode Timing

Question

In the following chart showing the timing at which the MCU enters standby mode, is there any limit to the value of t? In addition, at what timing should Vcc be dropped?



Timing Chart of Returning from Standby Mode

Answer

There is no limit to the value of t.
Either the STBY pin or RES pin can be Low first. However, the RES must be raised 20 msec or more behind the rising edge of the STBY to restart operation.
The Vcc must be dropped after the STBY becomes Low. It should be returned to the specified level before the STBY returns to High.

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title

HD6305U0, HD6305V0 Data Sheets
8-bit Single-Chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-334A
QA635-335A

Supplement

8

No. QA635-334A

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Returning from Standby Mode		

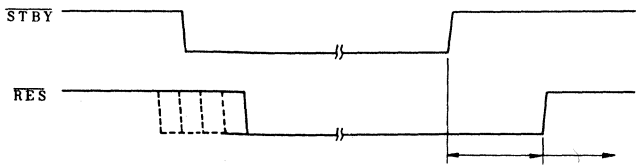
Question

Is there any method of returning from standby mode other than reset-start?

	Classification
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

No. In standby mode, the internal oscillator is stopped and the MCU internal register values are destroyed. (The contents of the Program Counter are also destroyed.) Therefore, only setting the STBY pin in High caused the MCU to burst.
Be sure to perform a reset-start to return from standby mode.



Timing Chart of Returning from Standby Mode

Applicable Manual

Title

HD6305U0, HD6305V0 Data Sheets
8-bit Single-Chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-333A

Supplement

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme	Executing an Instruction when Entering Standby Mode

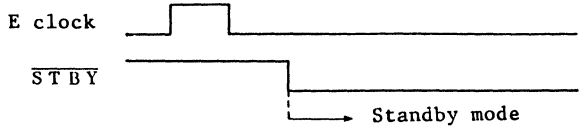
Question

The MCU enters standby mode by setting the $\overline{\text{STBY}}$ pin in Low. What will happen to the instruction that was being executed at that time?

Classification	
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

The instruction execution is stopped since the MCU enters standby mode whether or not an instruction execution sequence is executed. The MCU enters standby mode immediately after the $\overline{\text{STBY}}$ pin becomes Low.
In standby mode, the internal oscillator is stopped and the contents of the MCU internal registers are destroyed. So, be sure to perform a reset-start to return from standby mode (the built-in RAM values are retained).



Applicable Manual

Title

HD6305U0, HD6305V0
Data Sheets
8-bit Single-Chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-333A

Supplement

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Current Consumption in Low-Power-Consumption Mode		

Question

Is the Icc (current consumption) in low-power-consumption modes which is specified in the data sheets the value when no load is applied to the I/O ports?

Answer

Yes. Since the I/O ports maintain their output levels even in stop and wait modes, the Icc will be increased by I_{OH} or I_{OL} when a load is applied to the ports.
In standby mode, however, the I/O ports are in the high impedance state, and the Icc is the same with or without load.

Classification	
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
*	Low Power Consumption
	EPROM-on-chip
	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

QA635-333A

Supplement

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC
Theme	Using Bit Manipulating Instruction for Output Ports		

Question

Is it possible to use the bit manipulating instruction for the Data Registers of the output ports (ports A, B, C and D used for output)?

Classification

	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
*	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Yes.
The bit manipulating instruction automatically reads out an address, manipulates data in it and re-enters a result in the address. This instruction can be used for any read/write registers.

Applicable Manual

Title

HD6305U0, HD6305V0
Data Sheets
8-bit Single-Chip Microcomputer Data Book

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

No. QA635-338A

Type	HD6305U0 HD6305V0	Device	☐ 4S ■ 8S ☐ 8M ☐ 16M ☐ Software ☐ Evaluation kit, Emulator ☐ SD ☐ SBC

Theme	Accessing Not Used Areas on Memory Map

Question

What will happen if data is read from or written into the area designated as Not Used on the memory map?

Classification	
	Parallel Port
	Serial Port
	Timer/Counter
	BUS Interface
	Interrupt
	A/D Converter
	Oscillator
	Reset
	Low Power Consumption
	EPROM-on-chip
*	Software
	Evaluation Kit
	Emulator
	SD
	Data Buffer
	Others

Answer

Nothing will happen as long as the areas are not addresses $13 to $1F.
Never access the $13 to $1F areas since they are used for IC testing. Accessing (reading/writing) these areas causes the MCU to burst.

Applicable Manual

Title

Other Data Document

Title

Reference Q & A Sheet

No.

Supplement

Section Nine

System Application Notes

Introduction

HD6305Y0 is a series of 8-bit single chip microcomputers characterized by micro-programming control. The HD6305Y0 has on chip a CPU, a clock generator, ROM, RAM, I/O, 2 timers, and a serial communication interface (SCI).

This APPLICATION NOTE describes the system operation of an intelligent telephone using the HD6305Y0 to help users better understand the instruction set and to provide them with a useful reference for developing other systems.

The intelligent telephone described in APPLICATION NOTE makes the best use of HD6305Y0 features as follows:

HARDWARE

- 7872-byte ROM ..... For storing large programs
- 256-byte RAM ..... For reserving large work areas
- I/O terminals ..... For interfacing with external devices
- 2 timers ..... For processing various interrupts

SOFTWARE

- Byte efficient instruction set
  ..... For improving program efficiency
- Powerful bit manipulation instructions
  ..... Bit set, Bit clear, and Bit test and Branch for manipulating RAM and all I/O terminals

Although Programs described in APPLICATION NOTE have already been debugged, please be sure to check their operation in actual use.

Section 9
HD6305Y0 System Application Notes
Table of Contents

Section 9
HD6305Y0 System Application Notes
Contents of Module

Section 9
HD6305Y0 System Application Notes
Contents of Figures

Section 9
HD6305Y0 System Application Notes
Contents of Tables

This chapter describes the general construction and functions of the Intelligent Telephone, the system model for the APPLICATION NOTE.

1.1 System Configuration

1.1.1 External view

The external view of the Intelligent Telephone is shown in Fig. 1.1.

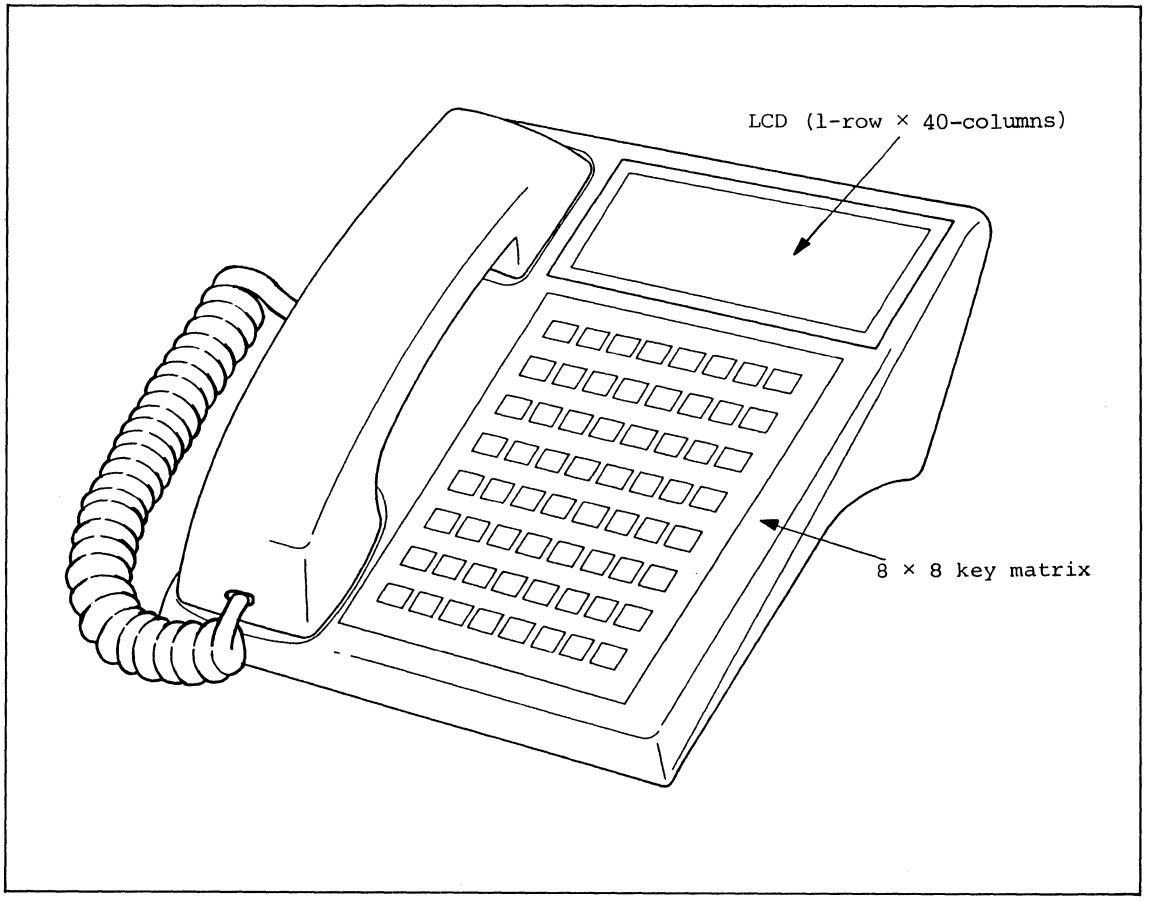
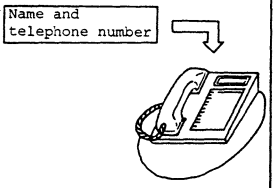


Fig. 1.1 External View

1.1.2 System specification outline

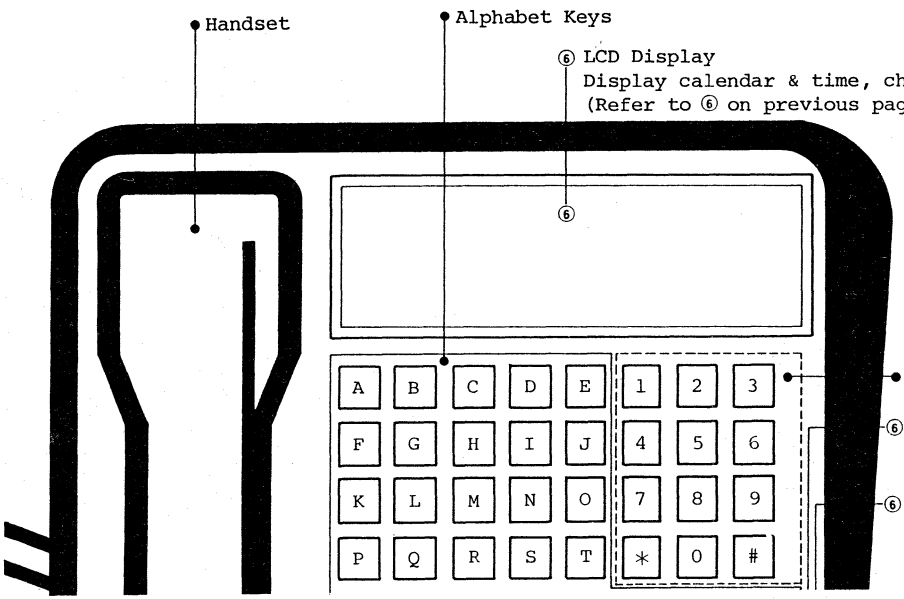
Individual components and functions of the Intelligent Telephone are described in Fig. 1.2. Numbers in the figure correspond to numbers on previous page.

① Speed dialing	② Redialing	③ Retrying	④ On-hook dialing	⑤ Storing telephone numbers
A Max. of 10 speed dialing numbers can be entered. Speed dialing can be performed by pushing SPEED DIAL key and entering the speed dial code.	Pushing REDIAL key will automatically dial the previously called telephone number once.	Pushing RETRY key will automatically redial the previously called telephone number up to a max. of 10 times.	Dialing can be performed with handset down. 	A max. of 200 names and corresponding telephone numbers can be stored. 

⑥ Displaying information on LCD

(i) Calendar & time (normal display)	(ii) Telephone number	(iii) Elapsed time	(iv) Charge	(v) Accumulated charge
*11/30 09:17:46	032121111	TIME 00:03:23	RATE 00000010 SEC180 CHARGE 00000010	CHARGE 00001500
Example: Indicates Nov. 30, 09:17:46 AM.	Example: Indicates dialing "03-212-1111".	Example: Indicates talking for 3 minutes and 23 seconds.	rate / charge Example: Indicates a rate of 10 cents/180 sec and a current charge of 10 cents.	Example: Indicates a current accumulated charge of 1500 cents ($15.00).

HITACHI


Handset
Alphabet Keys
⑥ LCD Display
Display calendar & time, charge, elapsed time and so on.
(Refer to ⑥ on previous page)
⑥
A B C D E
F G H I J
K L M N O
P Q R S T
1 2 3
4 5 6
7 8 9
* 0 #
Digit Keys
⑥Display Charge Key
Push for displaying charge.
(Refer to ⑥ (iv) on previous page)
⑥Display Accumulated Charge Key
Push for displaying
accumulated charge.
(Refer to ⑥ (v) on previous page)

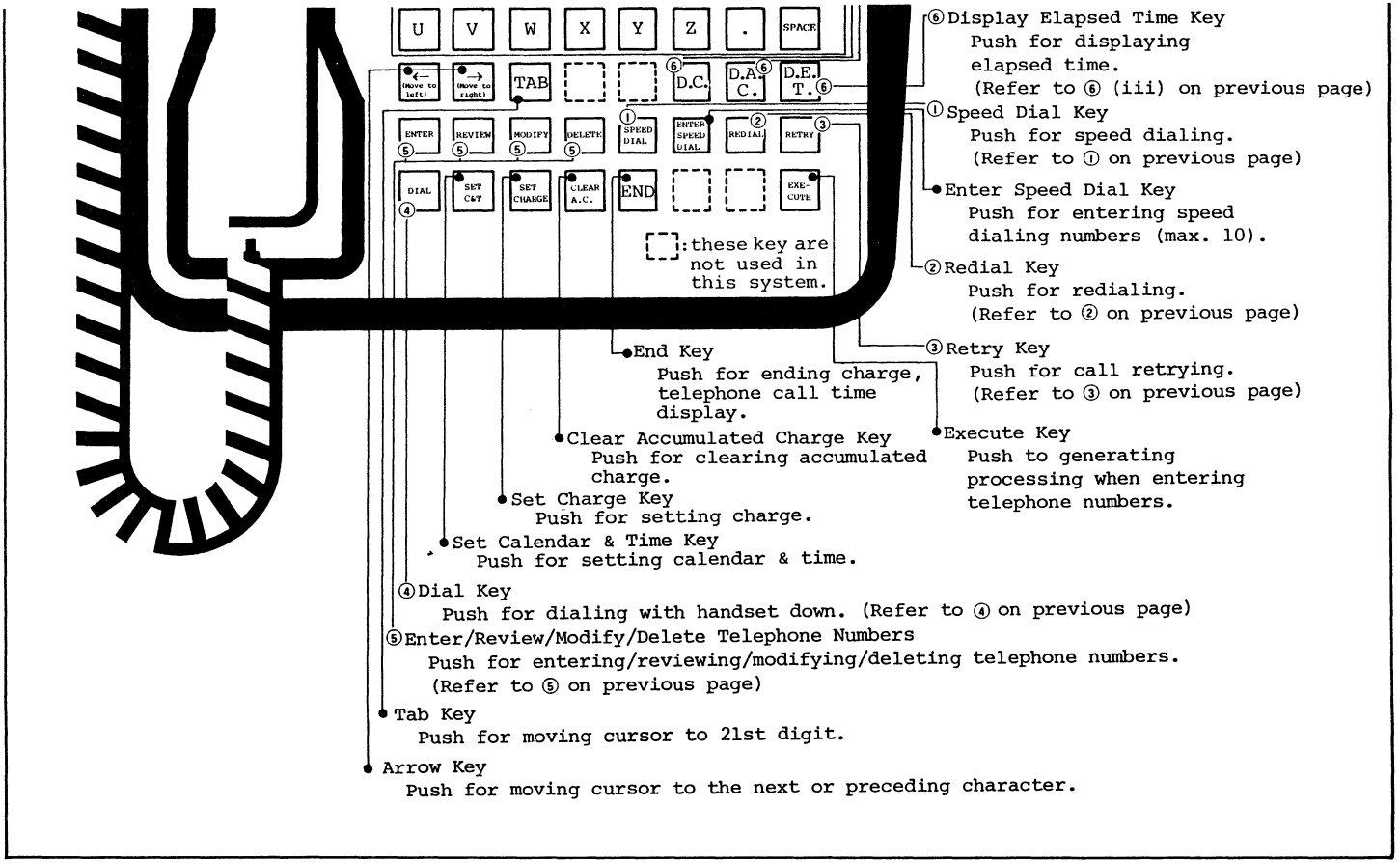
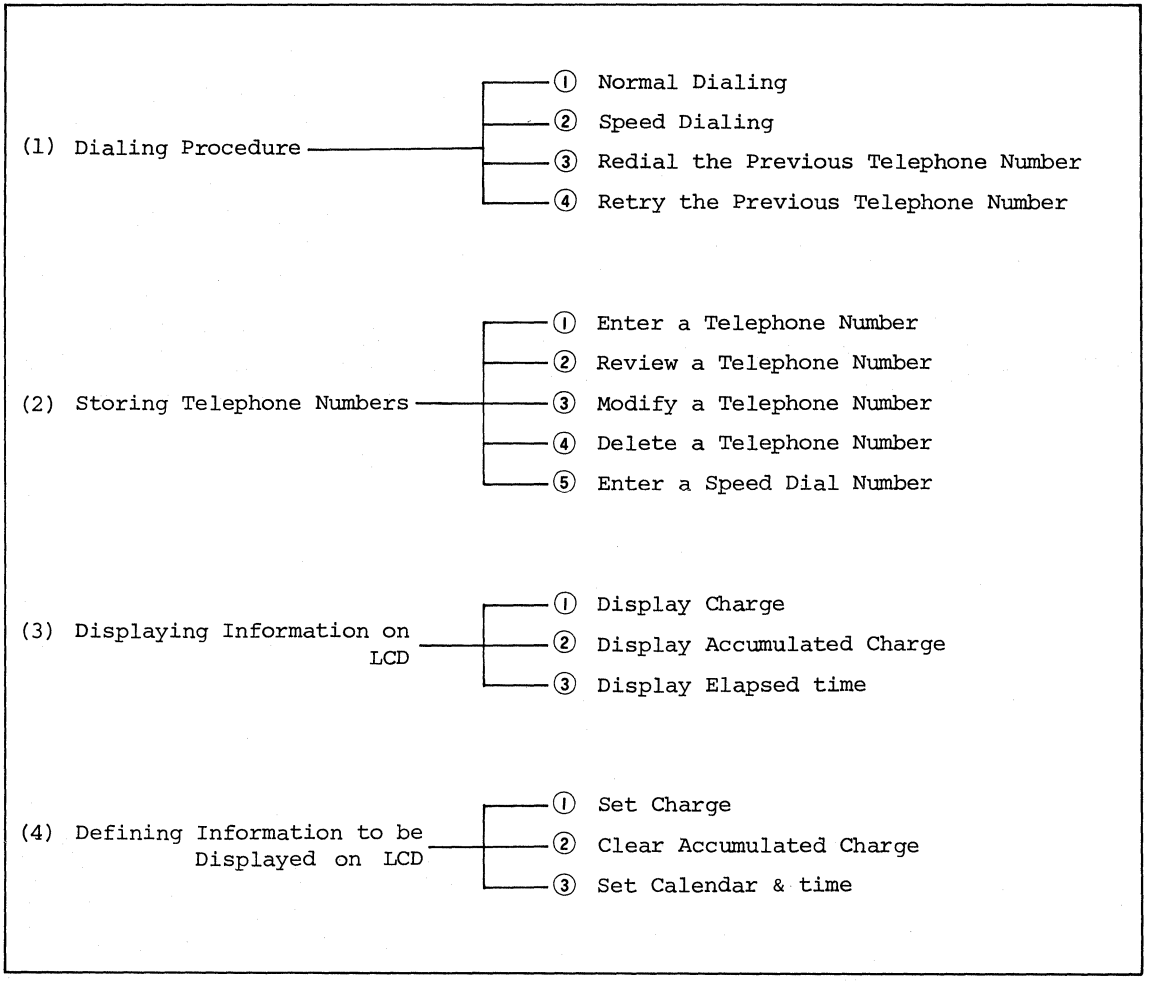


Fig. 1.2 Individual Components and Functions of the Intelligent Telephone

1.2 Operation

This section describes how to operate the Intelligent Telephone.

Each item of operation is shown below.

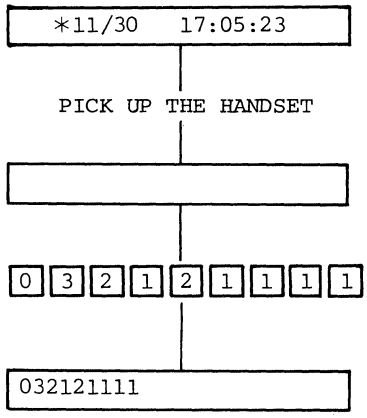


① Normal Dialing

Dialing a normal telephone number with handset up or down.

Example: How to dial "032121111"

(1) Dialing with handset up



(Calendar & time is displayed.)

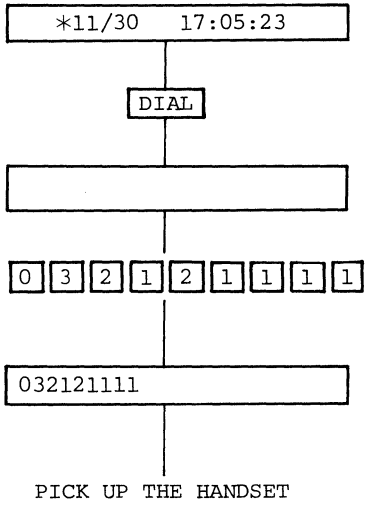
1. Pick up the handset.

(Calendar & time is deleted.)

2. Listen for the dial tone and then dial the number.

(Telephone number is displayed and call is made.)

(2) Dialing with handset down



(Calendar & time is displayed.)

1. Push DIAL key.

(Calendar & time is deleted.)

2. Listen for the dial tone and then dial the number.

(Telephone number is displayed and call is made.)

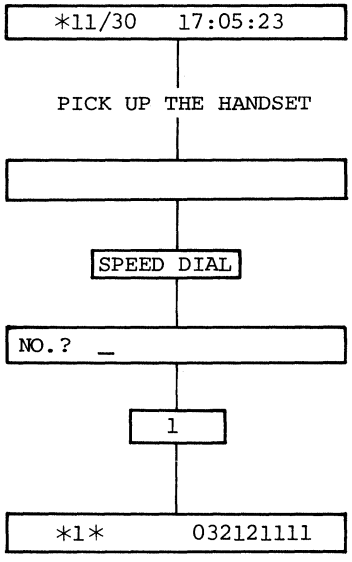
3. If call is answered, pick up the handset.

② Speed Dialing

Dialing a telephone number using a preassigned abbreviated number.

Example: How to dial "032121111" using speed dialing

(1) Dialing with handset up



(Calendar & time is displayed.)

1. Pick up the handset.

(Calendar & time is deleted.)

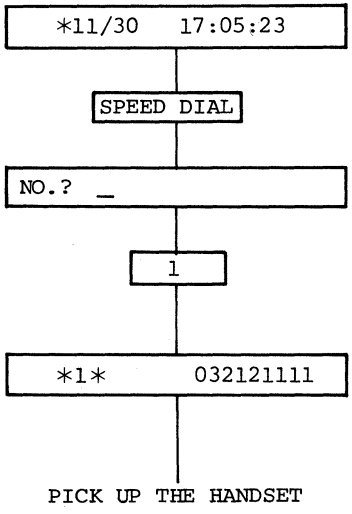
2. Push [SPEED DIAL] Key.

(Prompt is displayed.)

3. Enter speed dial code.

(Speed dial code and telephone number are displayed and call is made.)

(2) Dialing with handset down



(Calendar & time is displayed.)

1. Push [SPEED DIAL] key.

(Prompt is displayed.)

2. Enter speed dial code.

(Speed dial code and telephone number are displayed and call is made.)

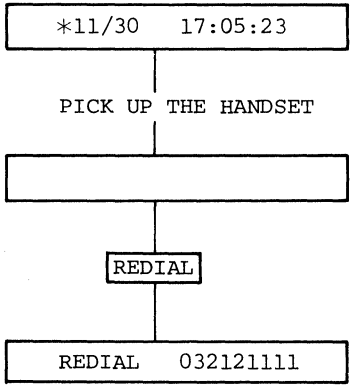
3. If call is answered, pick up the handset.

③ Redial the Previous Telephone Number

Redial the previously called telephone number once.

Example: How to redial the previous telephone number in the case of "032121111"

(1) Dialing with handset up



(Calendar & time is displayed.)

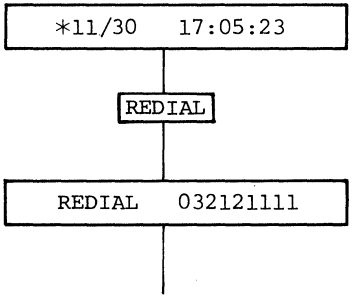
1. Pick up the handset.

(Calendar & time is deleted.)

2. Push [REDIAL] key.

(Telephone number is displayed and call is made.)

(2) Dialing with handset down



(Calendar & time is displayed.)

1. Push [REDIAL] key.

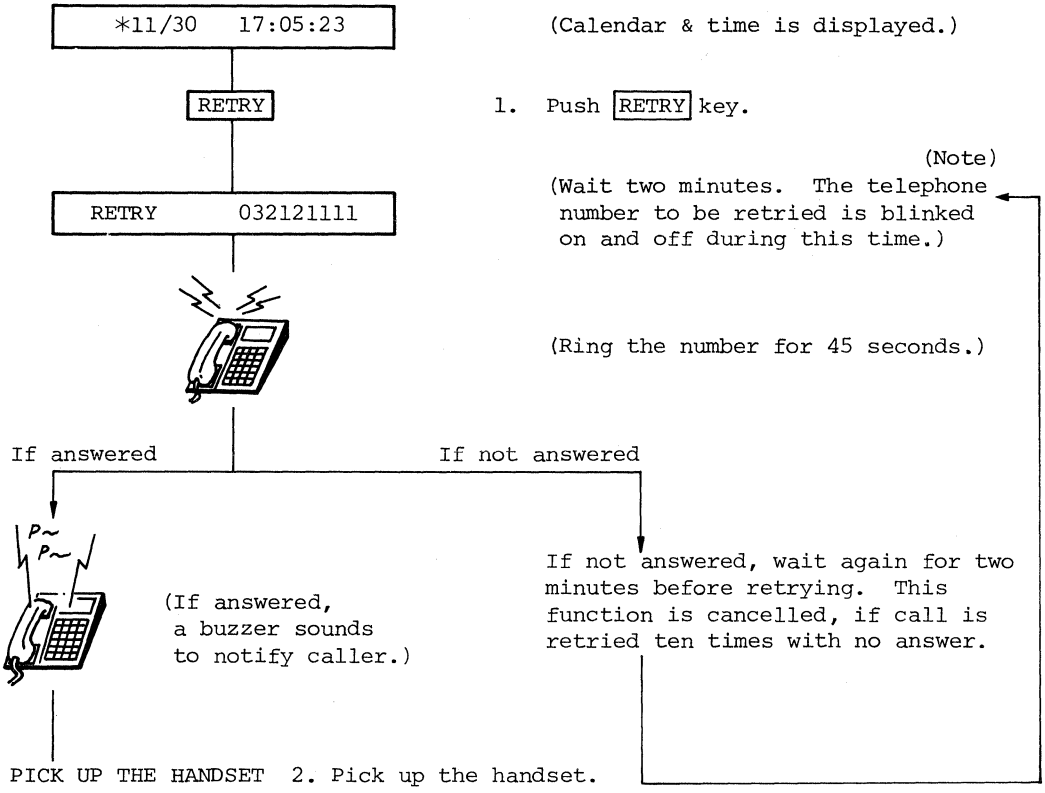
(Telephone number is displayed and call is made.)

2. If call is answered, pick up the handset.

④ Retry the Previous Telephone Number

Automatically retry the previously called telephone number up to a maximum of ten times.

Example: How to retry the previous telephone number in the case of "032121111"

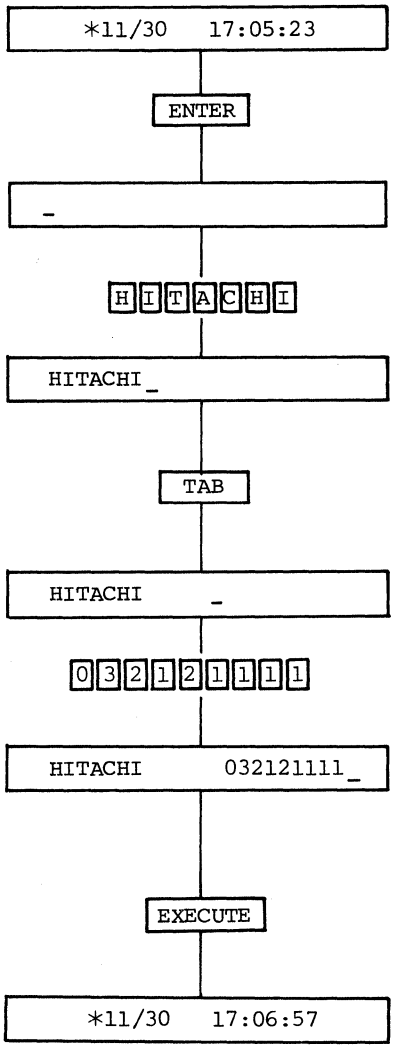


Note: To cancel retrying during this time, the user can push the END key. Moreover, if a telephone call is received, retrying is cancelled automatically.

① Enter a Telephone Number

Enter a name (max. 20 figures) and corresponding telephone number (max. 18 figures) for a maximum of 200 names/telephone numbers.

Example: How to enter "HITACHI 032121111"



(Calendar & time is displayed.)

1. Push [ENTER] key.

(Calendar & time is deleted and cursor is displayed.)

2. Enter name.

(Name is displayed.)

3. Push [TAB] key to move cursor to 21st digit on LCD (Telephone number must be input from 21st digit).

(Cursor is displayed on 21st digit.)

4. Enter telephone number.

(Both name and telephone number are displayed.)

5. Confirm whether correct name and telephone number is input or not. If correct, push [EXECUTE] key. If incorrect, modify data and push [EXECUTE] key.

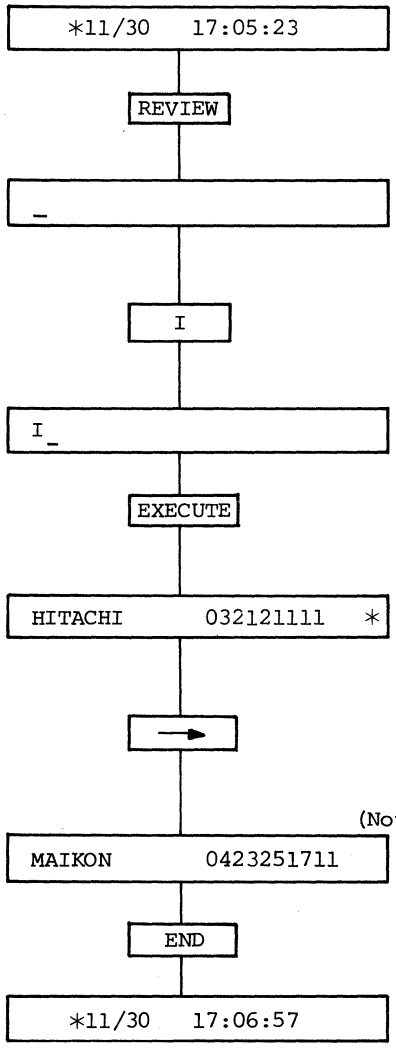
(Calendar & time is displayed again.)

Note: If a user tries to enter more than 201 names/telephone numbers, "FULL" is displayed and data cannot be entered; push [END] key to return to calendar & time display.

② Review a Telephone Number

Review a previously entered name and telephone number.

Example: How to review "MAIKON" and corresponding telephone number



(Calendar & time is displayed.)

1. Push [REVIEW] key.

(Calendar & time is deleted and cursor is displayed.)

2. Enter a character string matching some part of the name sought for (1 character OK).

(Data to review is displayed.)

3. Push [EXECUTE] key to review.

(Data matching string is displayed. If "*" is displayed on the last digit, there are more data.)

4. Push [→] key or [←] key to find desired data. ([→] key displays next data. [←] key displays previous data.)

(Note)

(Desired data is found.)

5. Push [END] key, if review is completed.

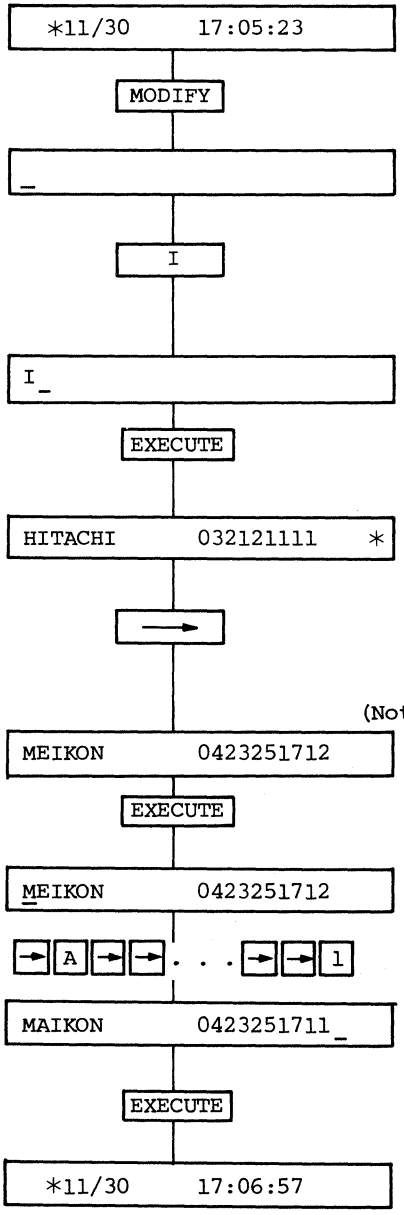
(Calendar & time is displayed again.)

Note: After [→] key is pushed, "*" may be deleted. This means that no data follows. If needed, push [←] key and review previous data. In the same way, if [←] key is pushed and "*" deleted, there are no more previous data. If needed, push [→] key and review next data.

③ Modify a Telephone Number

Modify a previously entered name and telephone number.

Example: How to modify "MEIKON 0423251712" to "MAIKON 0423251711"



(Calendar & time is displayed.)

1. Push [MODIFY] key.

(Calendar & time is deleted and cursor is displayed.)

2. Enter a character string matching some part of the name sought for (1 character OK).

(Data to review is displayed.)

3. Push [EXECUTE] key to review.

(Data matching string is displayed. If "*" is displayed on the last digit, there are more data.)

4. Push [→] key or [←] key to find desired data. ([→] key displays next data. [←] key displays previous data.)

(Data to be modified is displayed.)

5. If data to be modified is found, push [EXECUTE] key to stop reviewing data.

(Cursor is displayed.)

6. Modify data using [→] key and [←] key.

(Modified data is displayed.)

7. If modification is completed, push [EXECUTE] key.

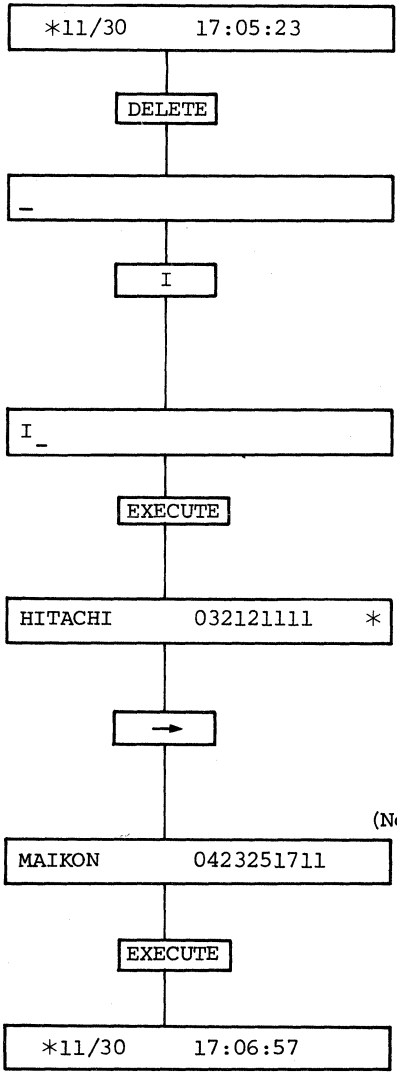
(Calendar & time is displayed again.)

Note: After [→] key is pushed, "*" may be deleted. This means that no data follows. If needed, push [←] key and review previous data. In the same way, if [←] key is pushed and "*" deleted, there are no more previous data. If needed, push [→] key and review next data.

④ Delete a Telephone Number

Delete a previously entered name and telephone number.

Example: How to delete "MAIKON 0423251711"



(Calendar & time is displayed.)

1. Push [DELETE] key.

(Calendar & time is deleted and cursor is displayed.)

2. Enter a character string matching some part of the name sought for (1 character OK).

(Data to review is displayed.)

3. Push [EXECUTE] key to review.

(Data matching string is displayed. If "*" is displayed on the last digit, there are more data.)

4. Push [→] key or [←] key to find desired data. ([→] key displays next data. [←] key displays previous data.)

(Desired data is found.)

5. If data to be deleted is found, push [EXECUTE] key to delete data.

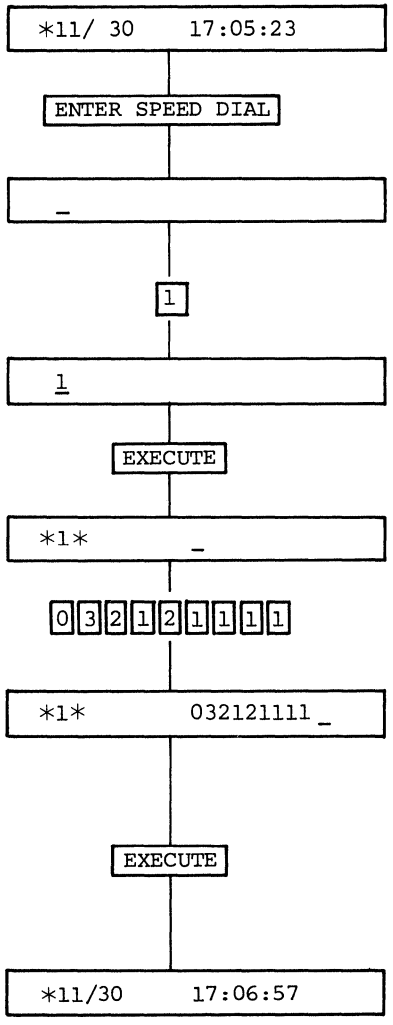
(Calendar & time is displayed again.)

Note: After [→] key is pushed, "*" may be deleted.
This means that no data follows.
If needed, push [←] key and review previous data.
In the same way, if [←] key is pushed and "*" deleted, there are no more previous data.
If needed, push [→] key and review next data.

⑤ Enter a Speed Dial Number

Enter an abbreviated speed dial code up to a maximum of ten telephone numbers.

Example: How to enter "1 = 032121111"



(Calendar & time is displayed.)

1. Push [ENTER SPEED DIAL] key.

(Calendar & time is deleted and cursor is displayed.)

(Note 1)

2. Enter speed dial code.

(Speed dial code is displayed.)

3. Push [EXECUTE] key.

(Cursor is moved to 21st digit.)

4. Enter telephone number.

(Both speed dial code and telephone number are displayed.)

(Note 2)

5. Confirm whether correct speed dial code and telephone number is input or not. If correct, push [EXECUTE] key. If incorrect, modify data and push [EXECUTE] key.

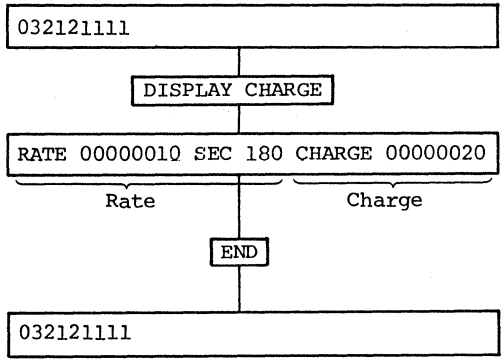
(Calendar & time is displayed again.)

Notes: 1. Only a digit numeral (0∿9) can be used for a speed dial code.
2. If a number previously entered is reentered, previous data is deleted.

① Display Charge

(Note 1)
Display current charge during a telephone conversation.

Example: How to display charge in the case of RATE=10 cents/180 sec, elapsed time = 200 seconds.



(Telephone number is displayed.)

1. Push DISPLAY CHARGE key.

(Both rate and charge are displayed.)

2. After confirming charge, push END key.

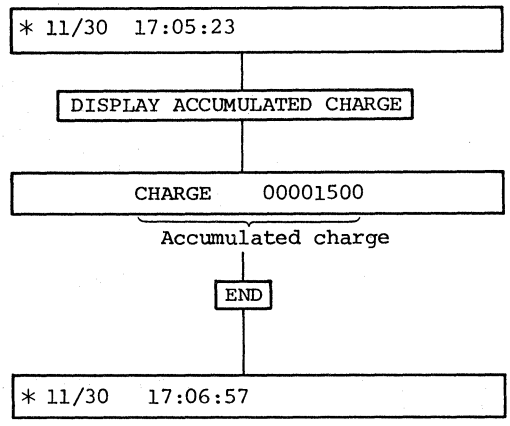
(Note 2)
(Telephone number is displayed again.)

Notes: 1. While not in a telephone conversation, "RATE 00000010 SEC 180 CHARGE 00000000" is displayed. In this case, push END key to return to calendar & time display.
2. During speed dialing, redialing, or retrying, only the telephone number is displayed.

② Display Accumulated Charge

(Note)
Display accumulated charge up to the present for one telephone number.

Example: How to display accumulated charge in case of the 1500 cents ($15.00).



(Calendar & time is displayed.)

1. Push DISPLAY ACCUMULATED CHARGE key.

(The accumulated charge is displayed.)

2. After confirming accumulated charge, push END key.

(Calendar & time is displayed again.)

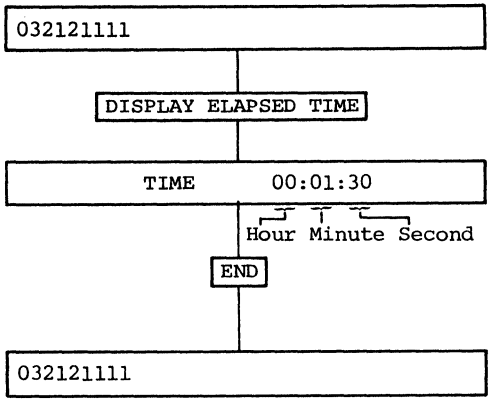
Note: If DISPLAY ACCUMULATED CHARGE key is pushed during a telephone conversation, accumulated charge is displayed. In this case, the accumulated charge is incrementing during the conversation, based on the rate for charge. If the END key is pushed, the telephone number is displayed again.

③ Display Elapsed Time

(Note 1)

Display elapsed time of a telephone conversation.

Example: How to display elapsed time in the case of 1 minute and 30 seconds.



(Telephone number is displayed.)

1. Push [DISPLAY ELAPSED TIME] key.

(Elapsed time is displayed.)

2. After confirming elapsed time, push [END] key.

(Note 2)

(Telephone number is displayed again.)

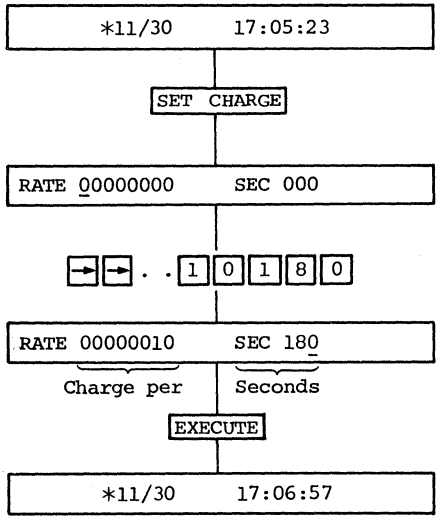
Notes: 1. While not in a telephone conversation, "TIME 00:00:00" is displayed. In this case, push [END] key to return to calendar & time display.
2. During speed dialing, redialing, or retrying, only the telephone number is displayed.

1.2.4 Defining information to be displayed on LCD

① Set Charge

Enter the standard rate for telephone call charges.

Example: How to set the rate for charges in the case of rate = 10 cents/180 sec.



(Calendar & time is displayed.)

1. Push [SET CHARGE] key.

(Present rate for charge is displayed.)

(Note)
2. Enter new rate.

(New rate is displayed.)

3. After confirming rate, push [EXECUTE] key.

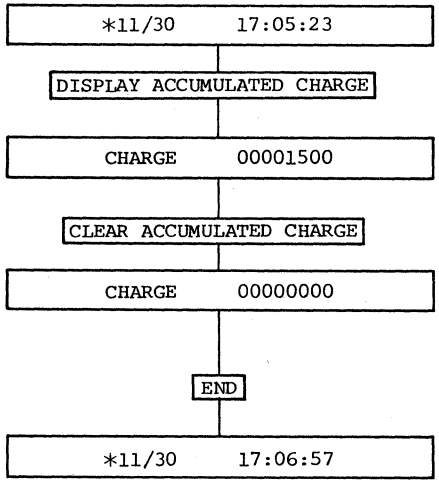
(Calendar & time is displayed again.)

Note: In this case, the cursor moves only where numerals are displayed.

② Clear Accumulated Charge

Clear the present accumulated charge.

Example: How to clear accumulated charge in the case of 1500 cents ($15.00).



(Calendar & time is displayed.)

1. Push [DISPLAY ACCUMULATED CHARGE] key.

(Present accumulated charge is displayed.)

2. Push [CLEAR ACCUMULATED CHARGE] key.

(Present accumulated charge is cleared.)

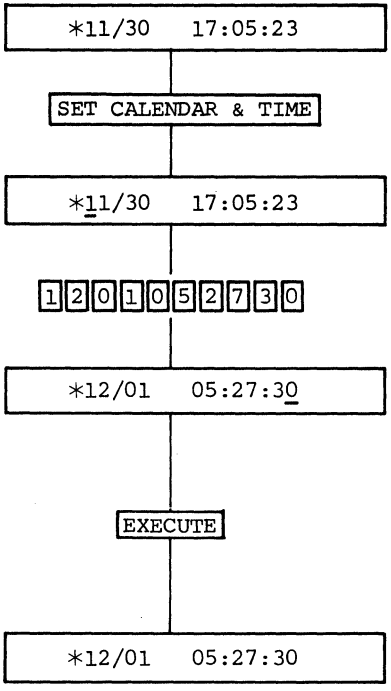
3. After confirming accumulated charge, push [END] key.

(Calendar & time is displayed again.)

③ Set Calendar & Time

(Note 1)
Modify calendar & time.

Example: How to modify "11/30 17:05:23" to "12/01 05:27:30".



(Calendar & time is displayed.)

1. Push [SET CALENDAR & TIME] key.

(Present calendar & time is displayed.)

(Note 2)

2. Modify calendar & time.

(Modified calendar & time is displayed.)

3. Confirm whether correct calendar & time is input or not. If correct, push [EXECUTE] key. If incorrect, modify data and push [EXECUTE] key.

(Modified calendar & time is displayed.)

Notes: 1. Time is displayed in military time.
2. In this case, the cursor moves only where numerals are displayed.

Section 2. Hardware Description

This chapter describes the hardware configuration of the Intelligent Telephone.

The devices and circuits making up this hardware can be devided into four major functional blocks, as shown in Fig. 2.1.

(1) Transmitting and Receiving Control Circuit
(2) Control Circuit for Storing and Retrieving Telephone Numbers in External RAM
(3) Driving the Liquid Crystal Display Module (H2572)
(4) 8×8 Key Matrix

The hardware is explained in terms of each block. The function of each block is listed in Table 2.1.

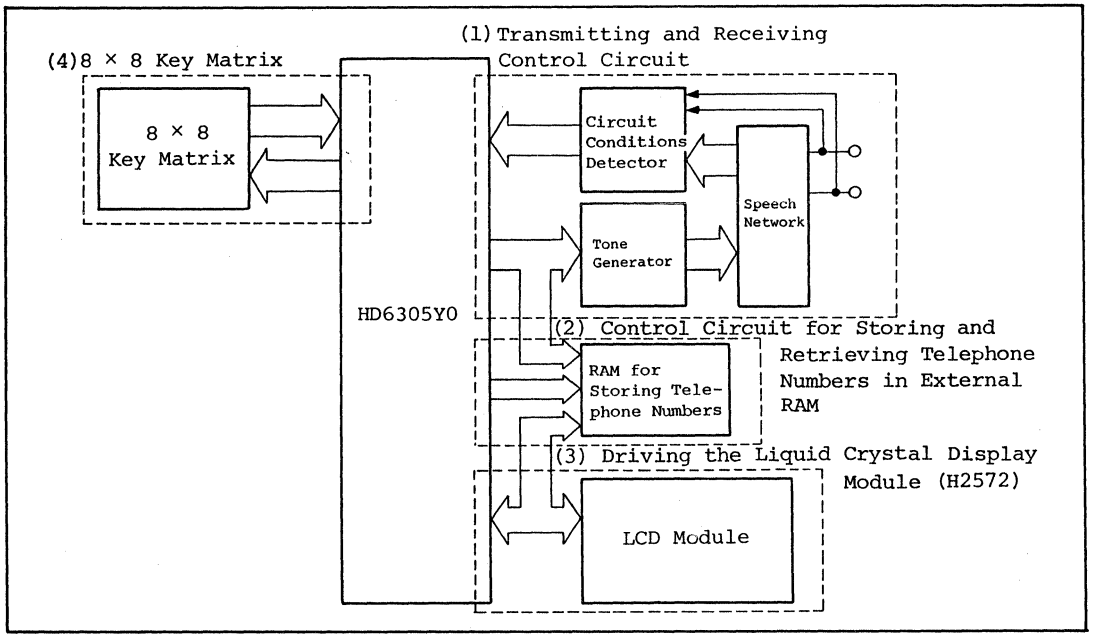


Fig. 2.1 Hardware Configuration

Table 2.1 Hardware Blocks Function

No.	Block	Function
1	Transmitting and Receiving Control Circuit	Control transmitting and receiving, and detect circuit conditions.
2	Control Circuit for Storing and Retrieving Telephone Numbers in External RAM	Store names and telephone numbers for 200 persons and 10 speed dial numbers.
3	Driving the Liquid Crystal Display Module (H2572)	Display ASCII character, stored in display RAM, on LCD (1-row × 40-columns).
4	8 × 8 Key Matrix	Perform key scan for 8 × 8 key matrix.

2.1 Transmitting and Receiving Control Circuit

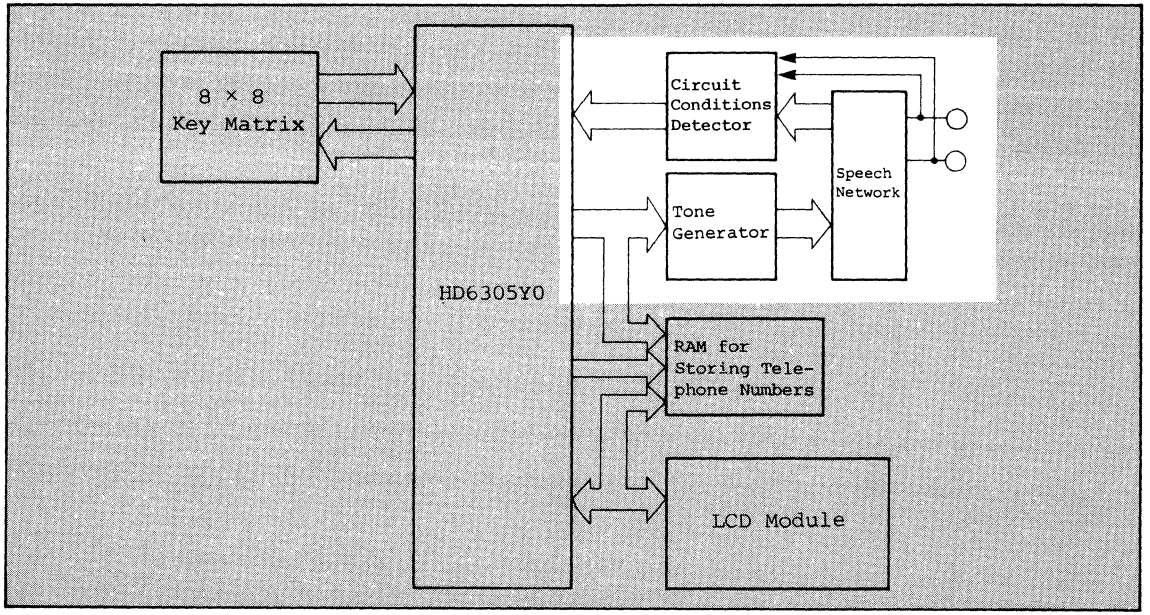


Fig. 2.2 "Transmitting and Receiving Control Circuit" Part

(1) Hardware and Operation Description

(a) For transmitting, the HD6305Y0 outputs a telephone number to the HD61826 tone generator, which generates tones and outputs them to the speech network, consisting mainly of the HA16808NT tone ringer. Finally, the HA16808NT outputs the tones onto the subscriber line.

(b) For receiving, the HA16808NT receives a ring signal, from the subscriber line which triggers buzzing. The ring signal, output from the HA16808NT, is input to the HD6305Y0 through a photo-coupler to indicate receiving.

(c) The HD6305Y0 is connected with the subscriber line through 2 photo-couplers. Based on data from these 2 photo-coupler, circuit conditions can be detected, e.g., handset is up or down, call answered or not.

(2) Microcomputer Applications

(a) Port E of the HD6305Y0 outputs data corresponding to a telephone number to the HD61826. Tone output from the HD61826 is performed by 16 ms timer 2.

(b) The ring signal is input to bit 1 of port D.

(c) Output from 2 photo-coupler, detecting circuit conditions, is input to bit 2, bit 3 of port D.

(3) Circuit Diagram

The interface circuit between the HD6305Y0 and "Transmitting and Receiving Control Circuit" is shown in Fig. 2.3.

The "Transmitting and Receiving Control Circuit" consists of four sections.

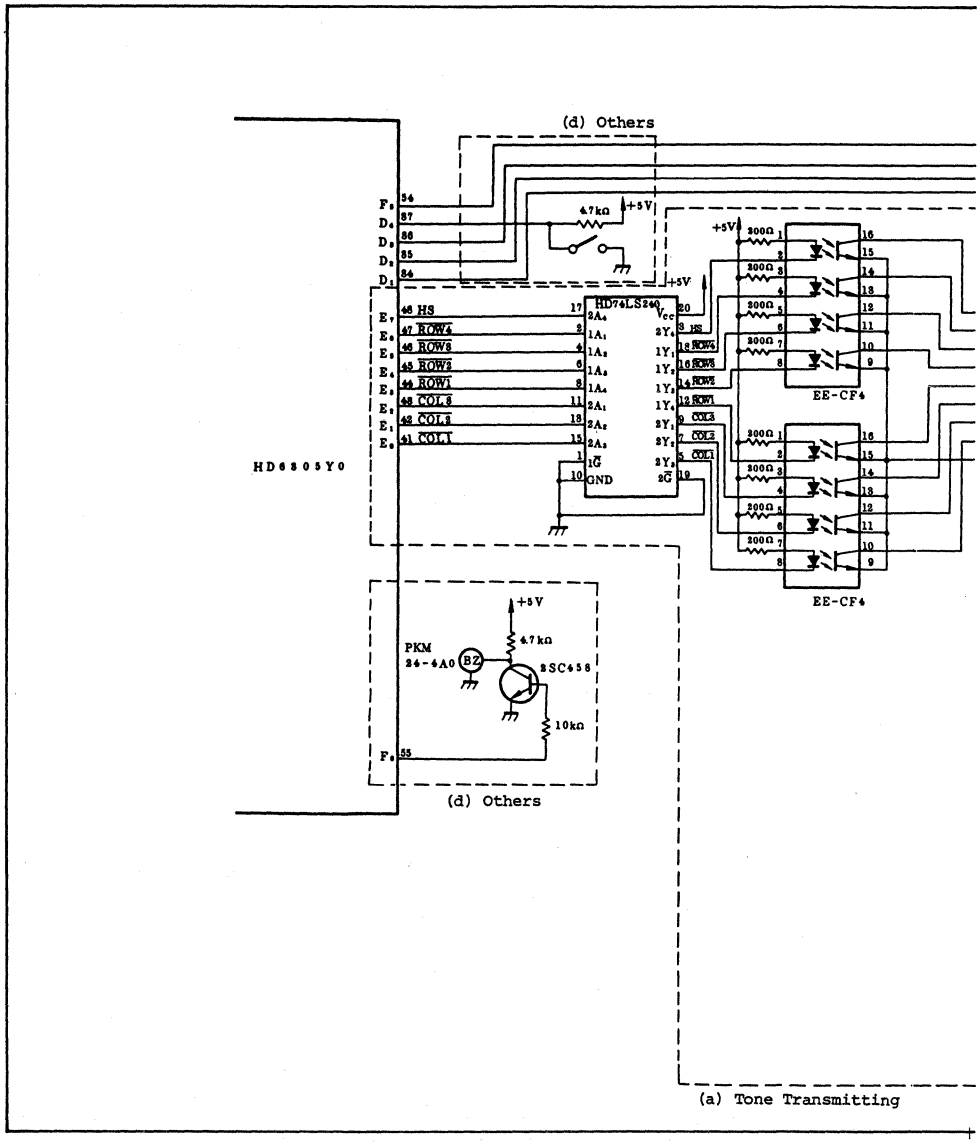


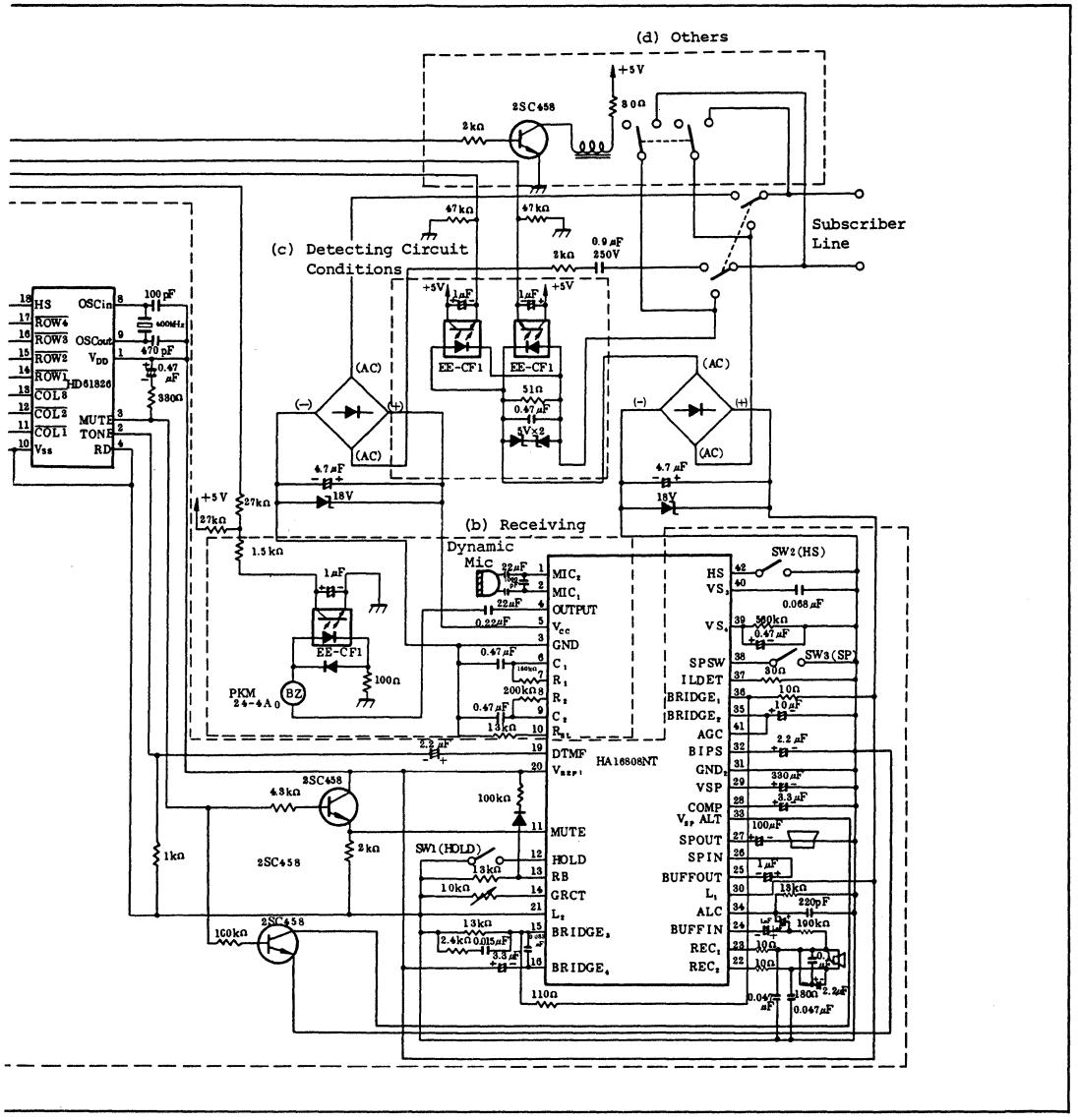
Fig. 2.3 Interface Circuit Between the HD6305Y0 and

(a) Tone transmitting

(b) Receiving

(c) Detecting circuit conditions

(d) Others



"Transmitting and Receiving Control Circuit"

(4) Pin Functions

Pin functions at the interface between the HD6305Y0 and "Transmitting and Receiving Control Circuit" are shown in Table 2.2.

Table 2.2 Pin Functions at the Interface Between the HD6305Y0 and "Transmitting and Receiving Control Circuit"

Section Name	Pin Name (HD6305Y0)	Input/ Output	Active Level (High or Low)	Function	Device and Pin Name		Program Label
(a) Tone transmitting	E_0	Output	—	Data for tone output	HD61826	$\overline{COL1}$	PEDTR
	E_1	Output	—			$\overline{COL2}$	
	E_2	Output	—			$\overline{COL3}$	
	E_3	Output	—			$\overline{ROW1}$	
	E_4	Output	—			$\overline{ROW2}$	
	E_5	Output	—			$\overline{ROW3}$	
	E_6	Output	—			$\overline{ROW4}$	
	E_7	Output	Low	ON/OFF switch for HS pin		HS	
(b) Receiving	D_1	Input	High	Detect receiving	Photo-coupler	Collecter	PDDTR
(c) Detecting circuit conditions	D_2	Input	—	Detect circuit conditions	Photo-coupler	Emitter	
	D_3	Input	—		Photo-coupler	Emitter	
(d) Others	D_4	Input	High	Control hook switch	Switch		
	F_5	Output	High	Control relay	Relay		PFDTR
	F_6	Output	—	Buzz	Buzzer		

Note: "Active Level" in Table 2.2 indicates the following;

High: logical 1

Low : logical 0

— : logical 1 or logical 0

(5) Hardware Operation

(a) Tone transmitting

In this application system, port E of the HD6305Y0 outputs data corresponding to a telephone number to the HD61826, which in turn outputs the corresponding tones.

The HD61826 outputs tone (0∿9, ∗, #) corresponding to 7-bit input signal ($\overline{ROW1}$∿$\overline{ROW4}$ and $\overline{COL1}$∿$\overline{COL3}$). The relation between tone output and row/column input is shown in Fig. 2.4. For example, to output "2", set both $\overline{ROW1}$ and $\overline{COL2}$ to High.

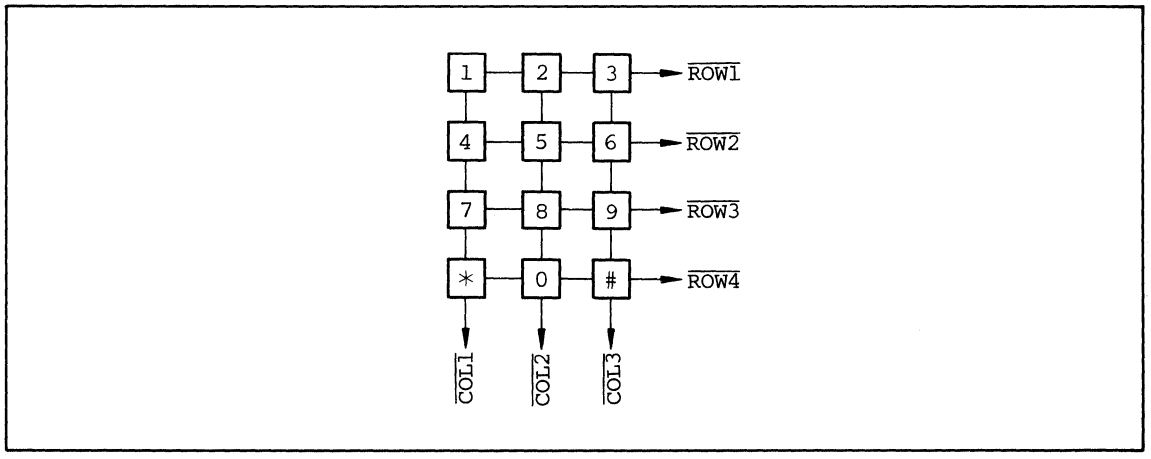


Fig. 2.4 Relation Between Tone Output and Row/Column Input

In this system, the HD61826 is connected to the HD6305Y0 through photo-couplers. Thus, to output "2", set bits 1 and 3 of port E to High. Bit 7 of port E is also used to control the HS pin of the HD61826. The interface circuit between port E of the HD6305Y0 and Row/Column/HS of the HD61826 is shown in Fig. 2.5.

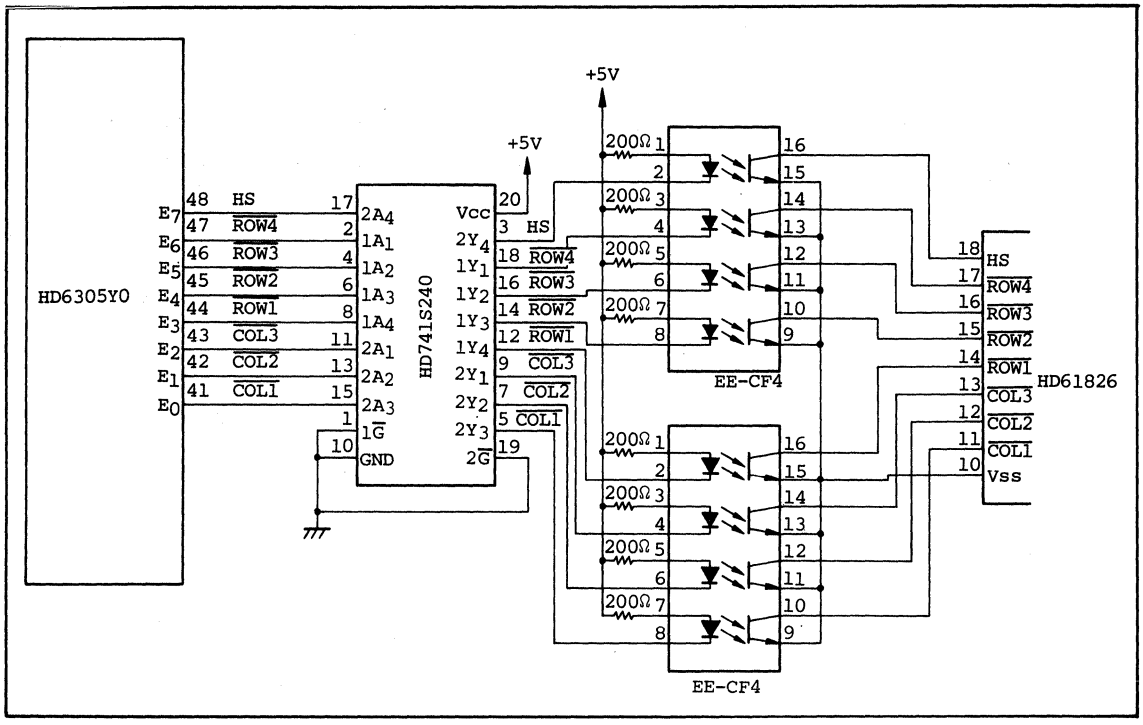


Fig. 2.5 Interface Circuit Between Port E of the HD6305Y0 and Row/Column/ HS pin of the HD61826

Interface timing between the HD6305Y0 (port E) and the HD61826 (Row, Column, HS, OSC, MUTE, TONE) is shown in Fig. 2.6.

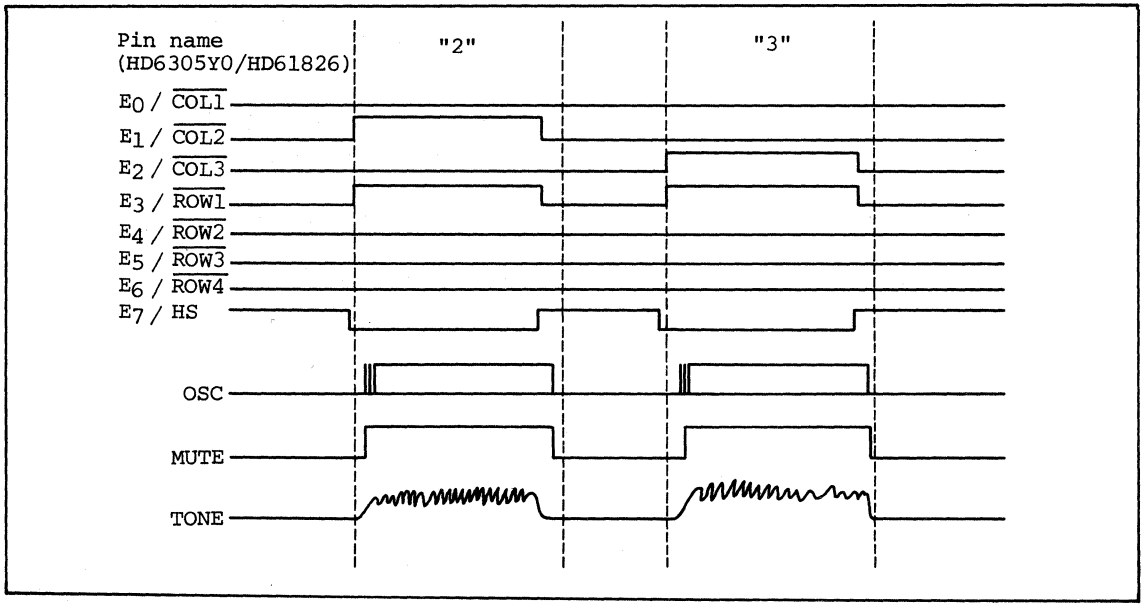


Fig. 2.6 Tone Transmitting Interface Timing

(b) Receiving

When this telephone is called from another telephone, a ring signal is input to the HA16808NT from the subscriber line. Ring signal output from the HA16808NT is input to bit 1 of port D through a photo-coupler. When bit 1 of port D is "1", the HD6305Y0 determines a receiving condition.

Interface timing for the ring signal, photo-coupler, and port is shown in Fig. 2.7.

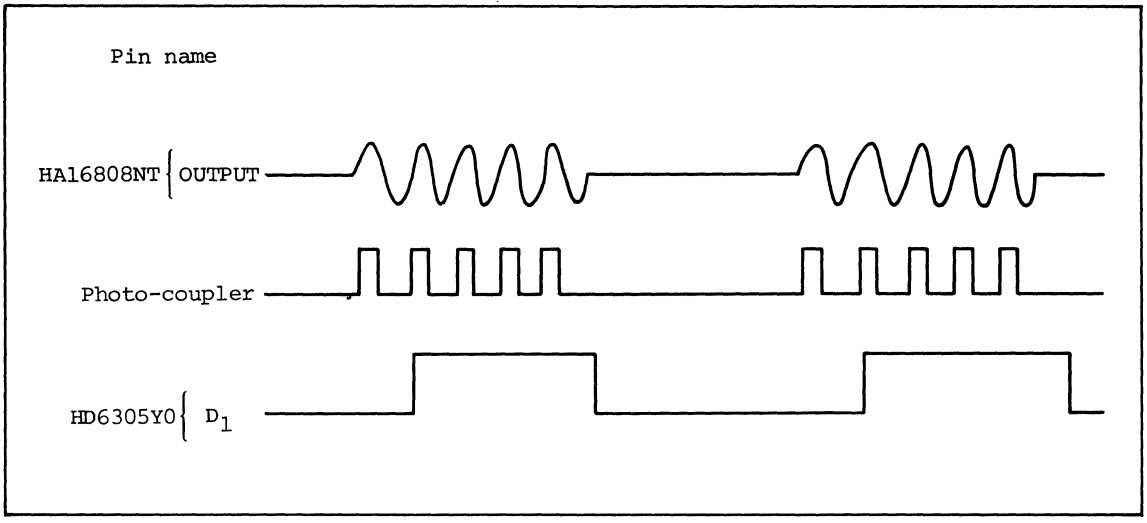


Fig. 2.7 Receiving Interface Timing

(c) Detect circuit conditions

In this application system, circuit conditions are detected using features of the subscriber line.

Two subscriber lines are connected to this system as shown in Fig. 2.8.

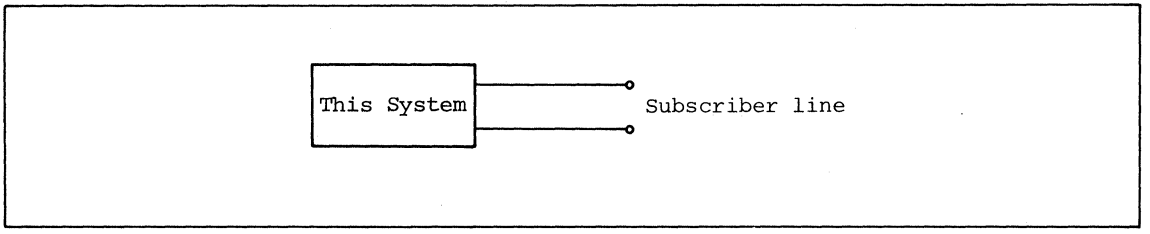


Fig. 2.8 This System and Subscriber Line

- When the handset is down, both subscriber lines are Low.
- When the handset is up to call, either subscriber line is High. After the tones are output and the call answered, the subscriber line which was previously High goes Low, and that which was Low goes High. When the conversation is over and the handset is put down, both subscriber lines go Low again.

In this system, the changing High and Low states of the subscriber lines are input to bits 2 and 3 of port D of the HD6305Y0 through photocouplers. As mentioned above, using these data, various circuit condition can be detected.

(i) When handset is up to call

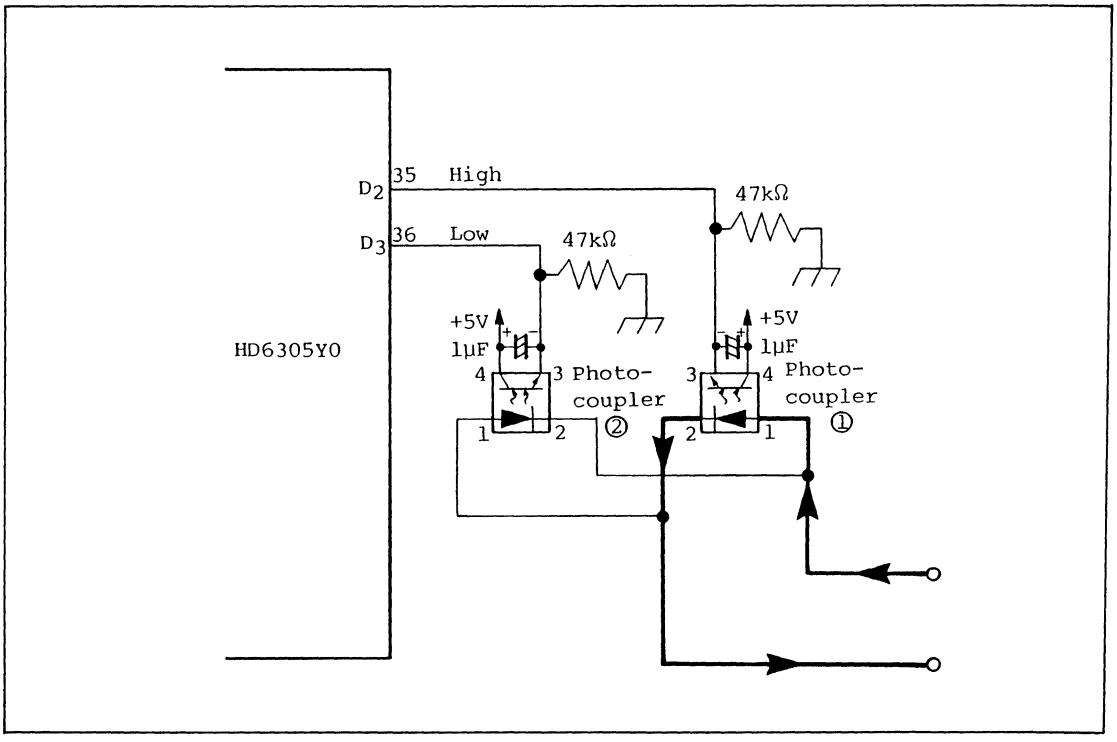


Fig. 2.9 Circuit Condition When Handset is Up to Call

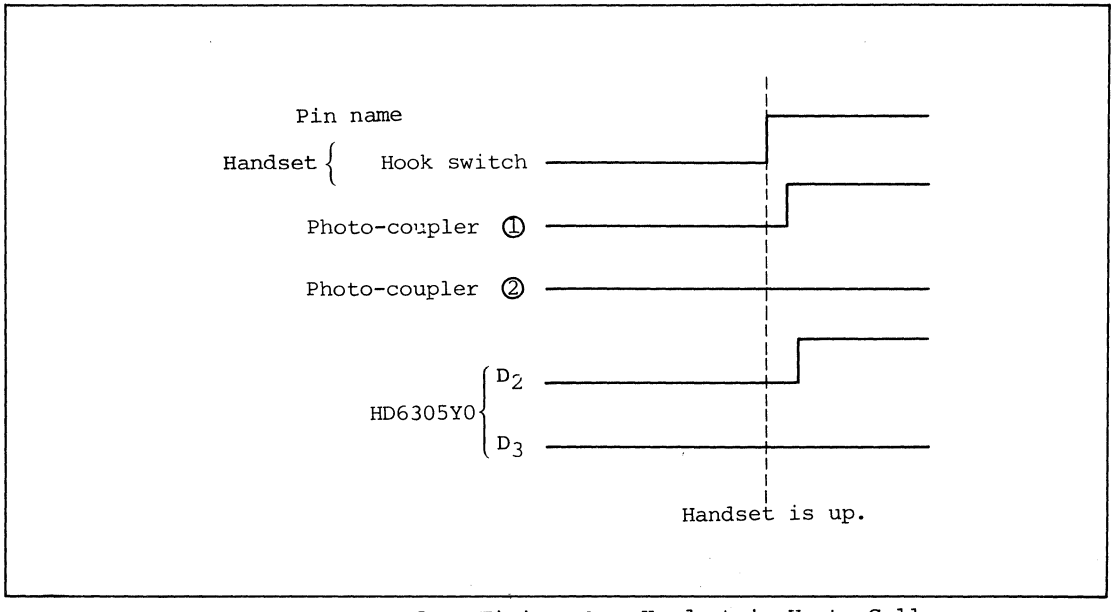


Fig. 2.10 Interface Timing When Handset is Up to Call

(ii) When call is answered (conversation in progress)

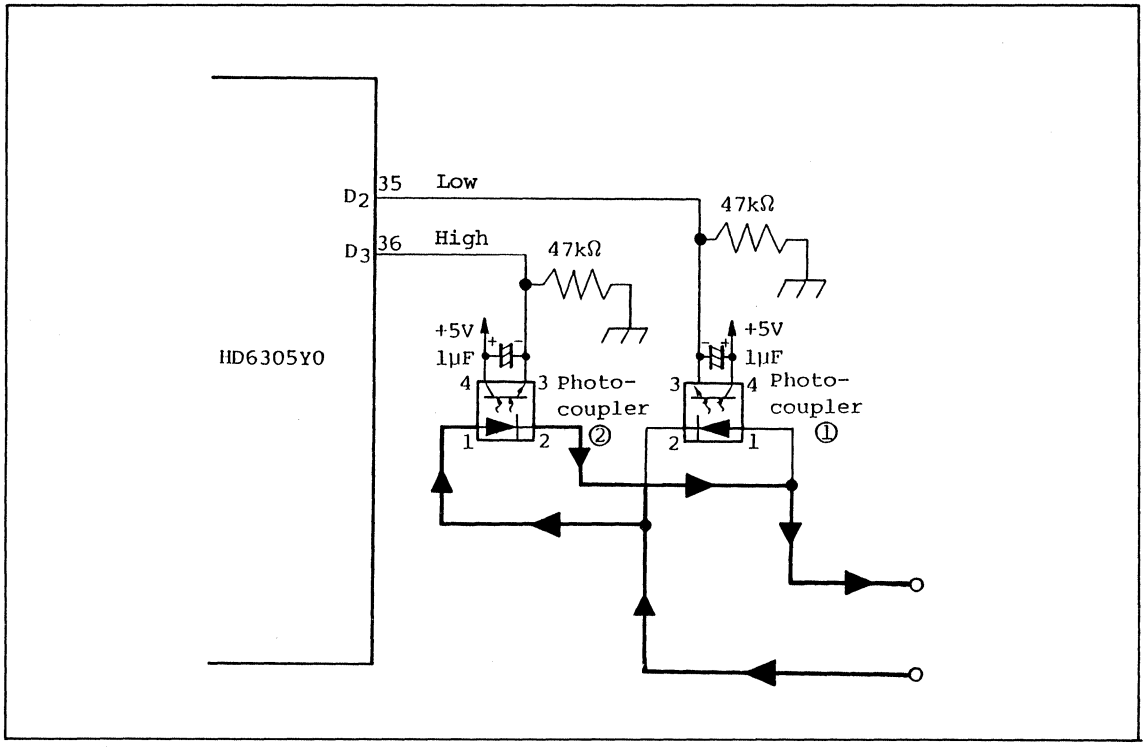


Fig. 2.11 Circuit Condition When Call is Answered

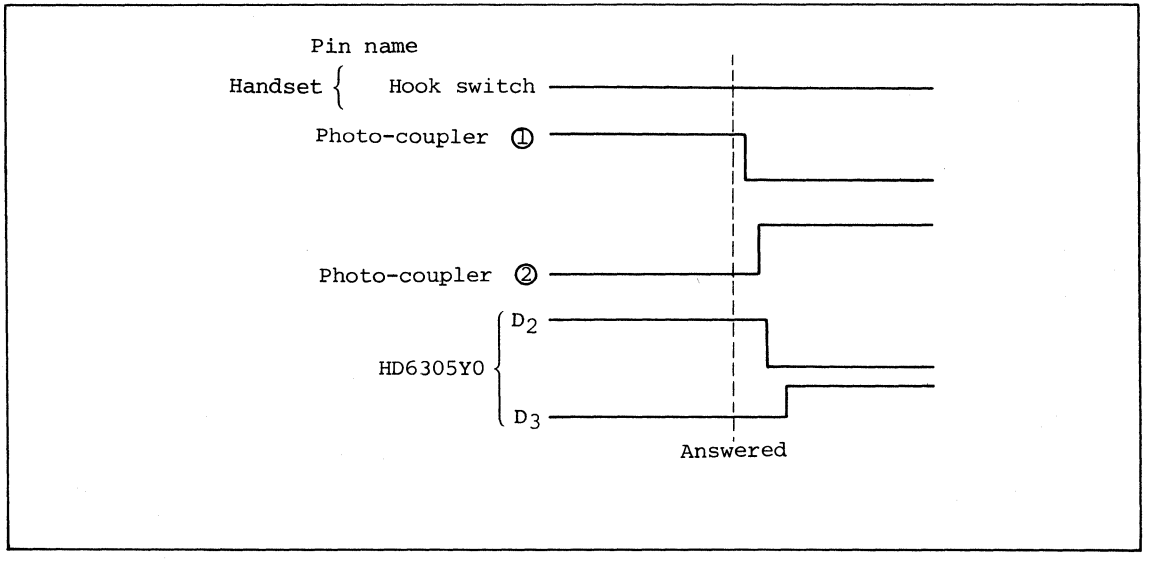


Fig. 2.12 Interface Timing When Call is Answered

(d) Others

(i) Control relay

When tone is output using on-hook dialing, bit 5 of port F is set to High and relay ON.

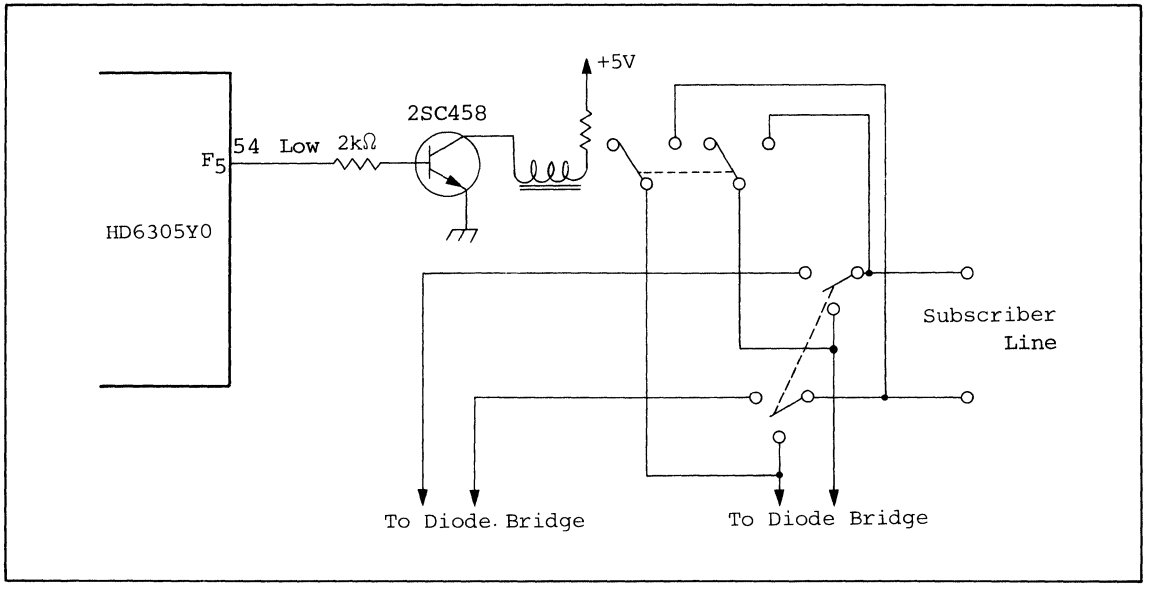


Fig. 2.13 Relay OFF

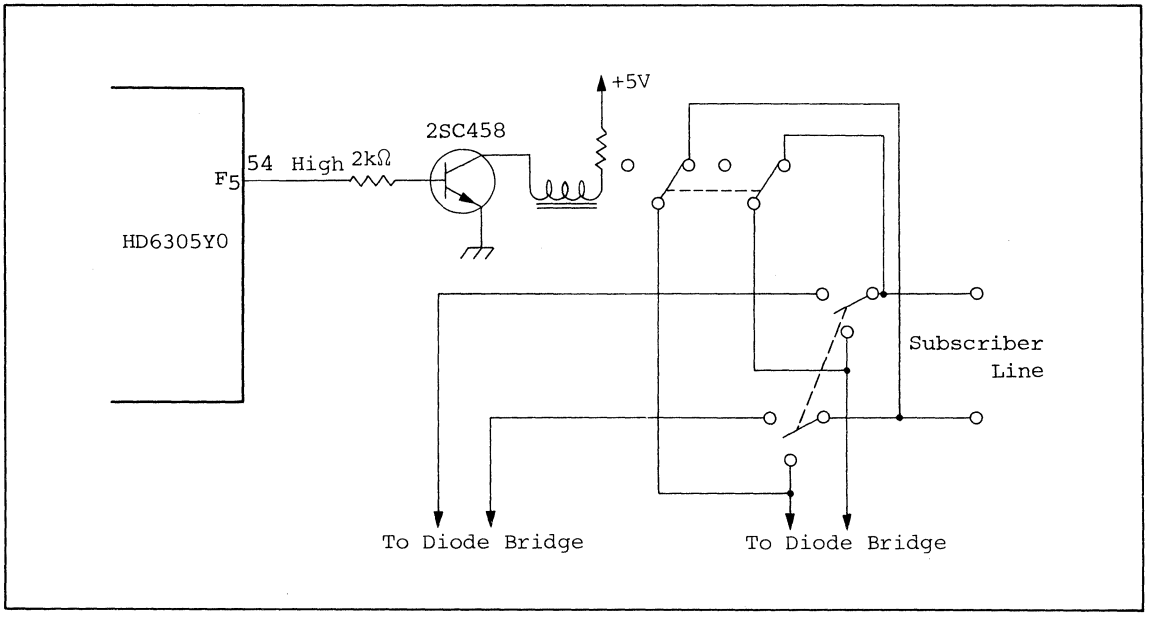


Fig. 2.14 Relay ON

(ii) Control hook-switch

When tone is output using on-hook dialing (handset is down), this switch is ON. If High is input to bit 4 of port D, software forces bit 5 of port F to Low to turn relay OFF.

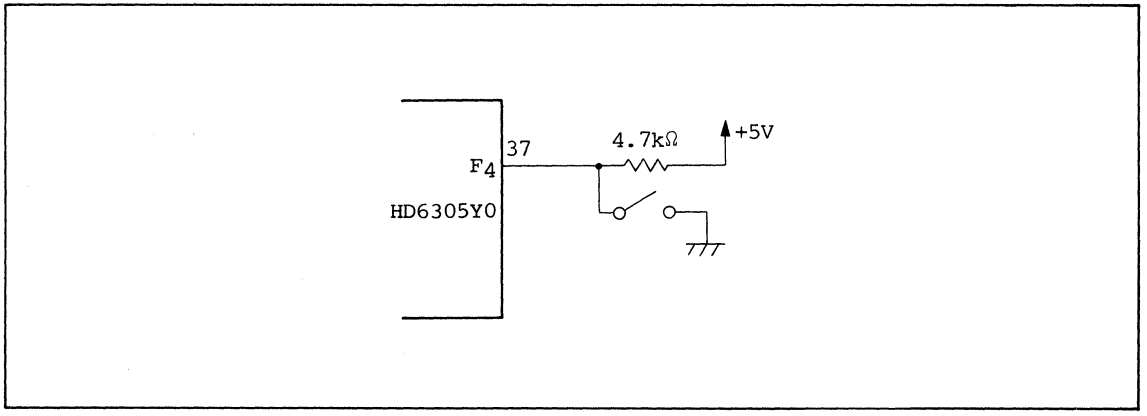


Fig. 2.15 Control Hook Switch

(iii) Buzz

If a call is answered during automatic retrying, software outputs a pulse from bit 6 of port F to generate buzzing.

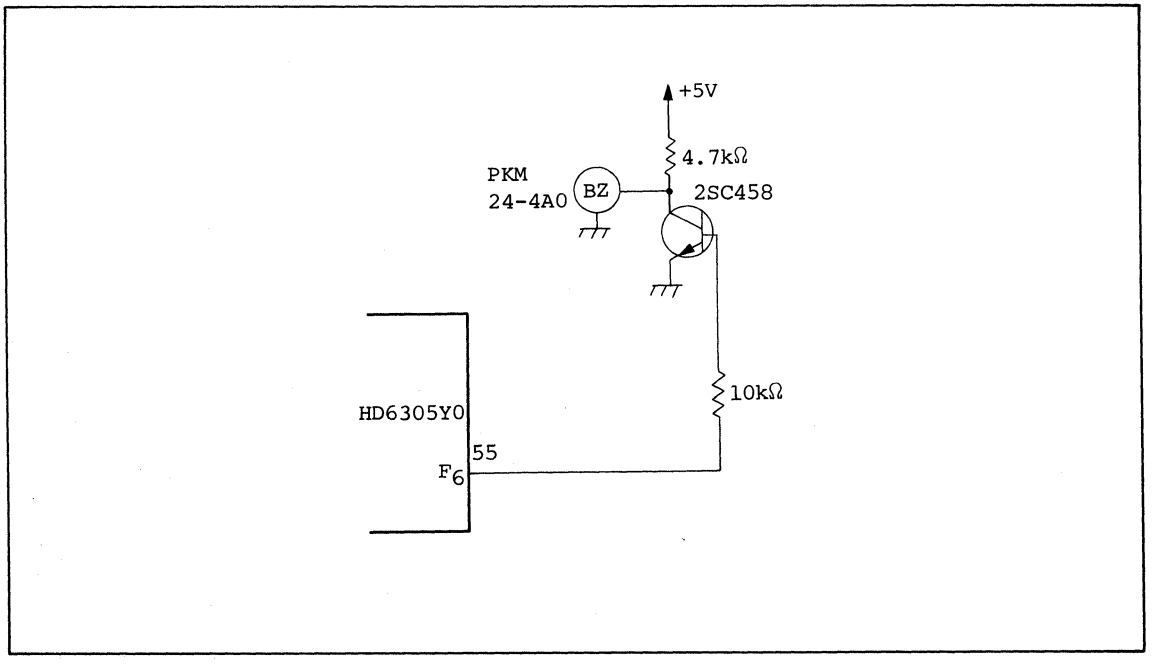


Fig. 2.16 Buzz

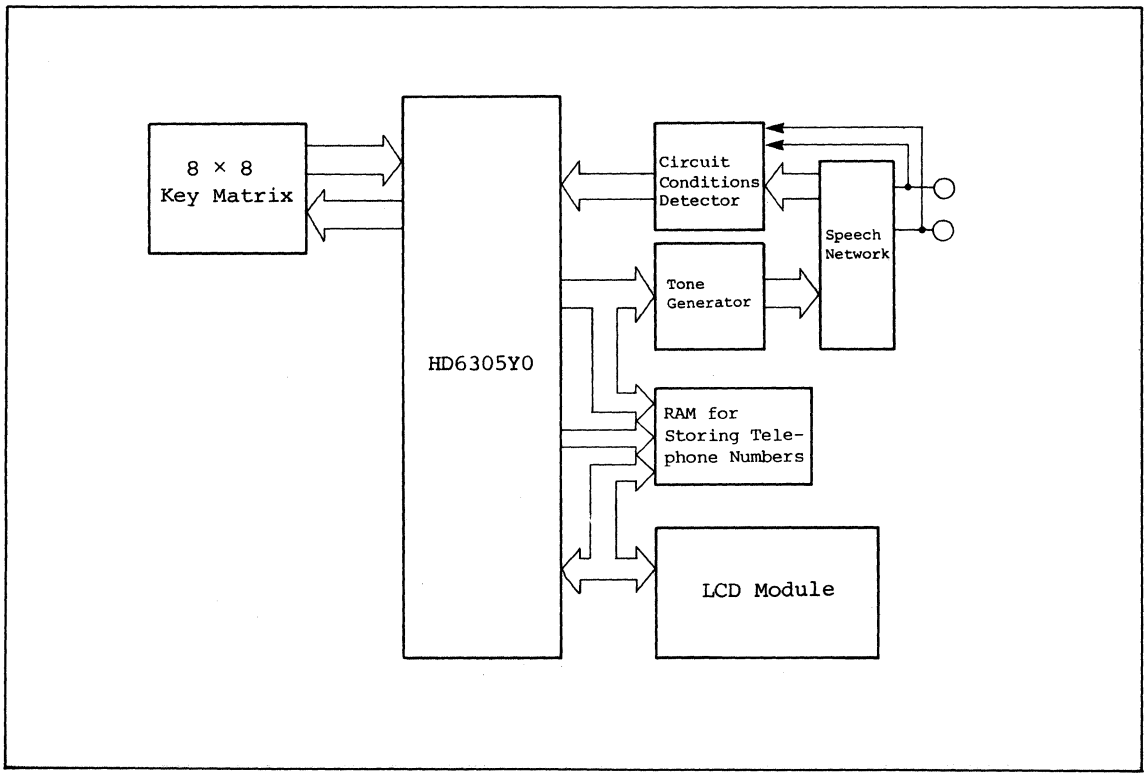


Fig. 2.17 "Control Circuit for Storing and Retrieving Telephone Numbers in External RAM" Part

(1) Hardware and Operation Description

(a) The HD6305Y0 controls the HM6264AP external RAM for storing and retrieving telephone numbers.

(b) The HM6264AP is an 8192-word × 8-bit static RAM.

(2) Microcomputer Applications

(a) I/O ports of the HD6305Y0 control the HM6264AP.

(b) The data bus is controlled by port A, lower 8 bits of the address bus by port B, and upper 5 bits of the address bus by port E, bits 0∿4.

(c) Control pins of the HM6264AP are controlled by port E, bits 5∿7.

(3) Circuit Diagram

The interface circuit between the HD6305Y0 and HM6264AP is shown in Fig. 2.18.

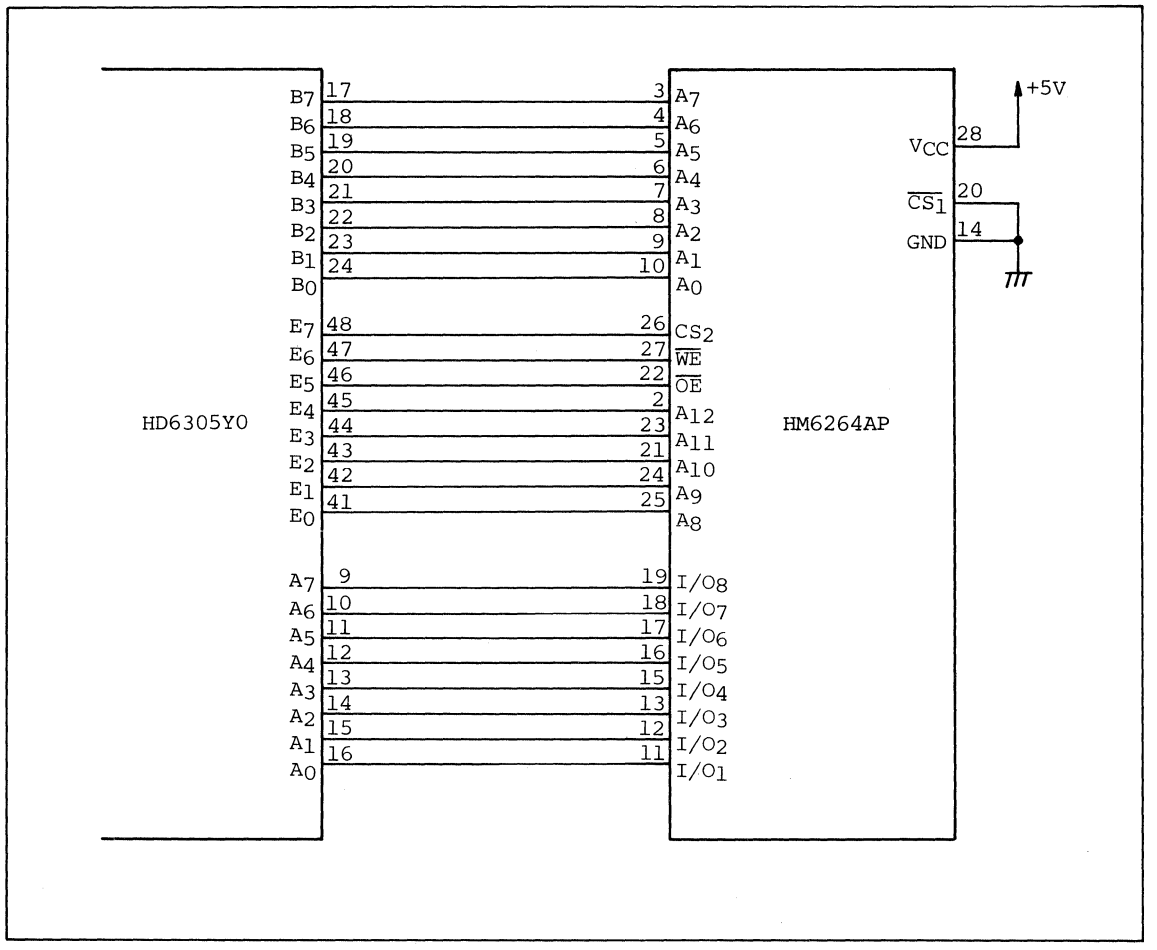


Fig. 2.18 Interface Circuit Between the HD6305Y0 and HM6264AP

(4) Pin Functions

Pin functions at the interface between the HD6305Y0 and HM6264AP are shown in Table 2.3.

Table 2.3 Pin Functions at the Interface Between the HD6305Y0 and HM6264AP

Pin Name (HD6305Y0)	Input/ Output	Active Level (High or Low)	Function	HM6264AP	Program Label
A_0	Input/ Output	—	Data lines	I/O_1	PADTR
A_1	Input/ Output	—		I/O_2	
A_2	Input/ Output	—		I/O_3	
A_3	Input/ Output	—		I/O_4	
A_4	Input/ Output	—		I/O_5	
A_5	Input/ Output	—		I/O_6	
A_6	Input/ Output	—		I/O_7	
A_7	Input/ Output	—		I/O_8	
B_0	Output	—	Lower 8 bits of address lines	A0	PBDTR
B_1	Output	—		A_1	
B_2	Output	—		A_2	
B_3	Output	—		A_3	
B_4	Output	—		A_4	
B_5	Output	—		A_5	
B_6	Output	—		A_6	
B_7	Output	—		A_7	
E_0	Output	—	Upper 5 bits of address lines	A_8	PEDTR
E_1	Output	—		A_9	
E_2	Output	—		A_{10}	
E_3	Output	—		A_{11}	
E_4	Output	—		A_{12}	
E_5	Output	Low	Output enable signal	$\overline{OE}$	
E_6	Output	Low	Write enable signal	$\overline{WE}$	
E_7	Output	High	Chip select	CS_2	

Note: "Active Level" in Table 2.3 indicates the following;
High: logical 1
Low : logical 0
— : logical 1 or logical 0

(5) Hardware Operation

Upper 3 bits of port E control CS_2, $\overline{OE}$, and $\overline{WE}$ of HM6264AP. The interface timing is shown in Fig. 2.19.

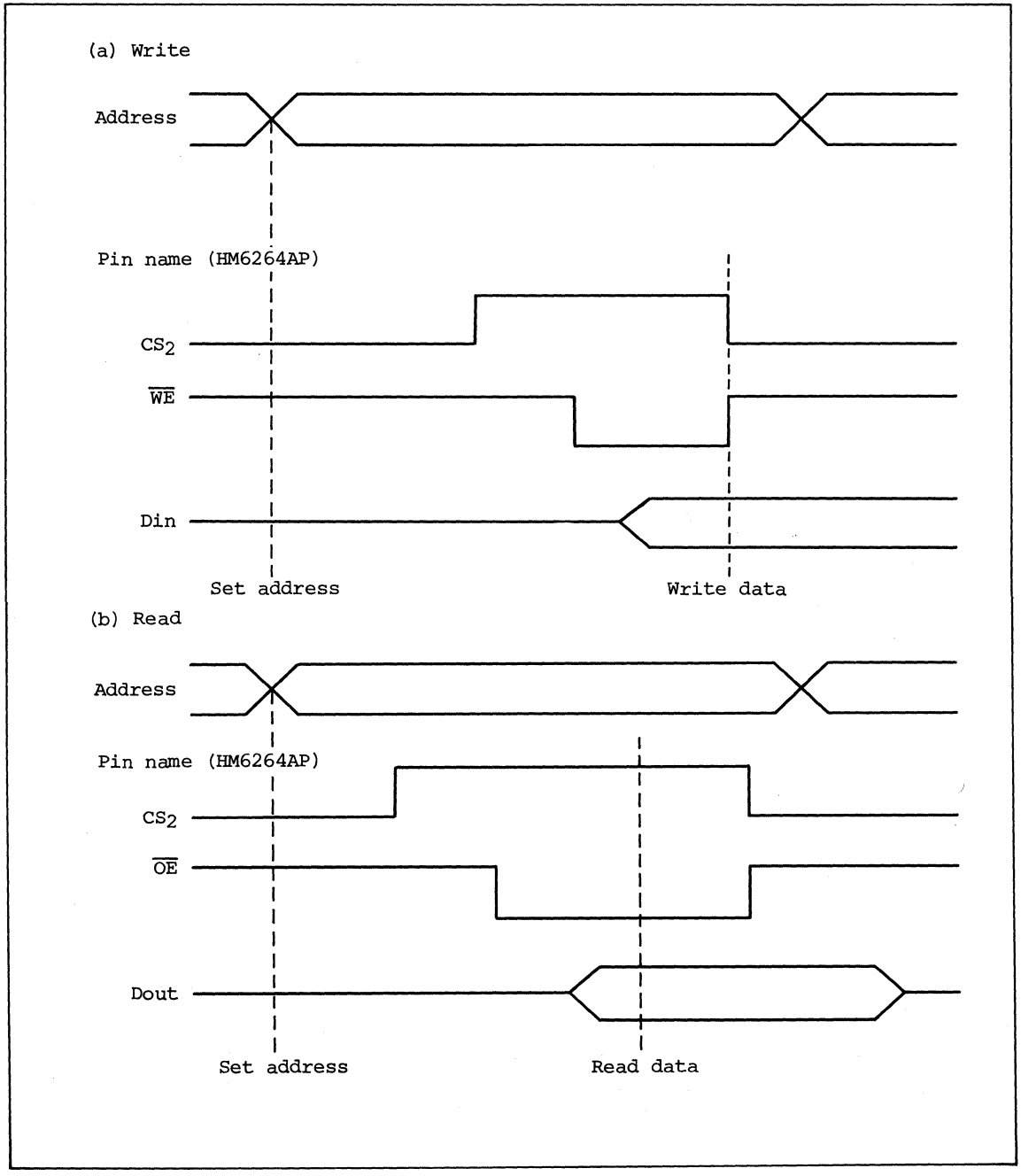


Fig. 2.19 HD6305Y0←→HM6264AP Interface Timing

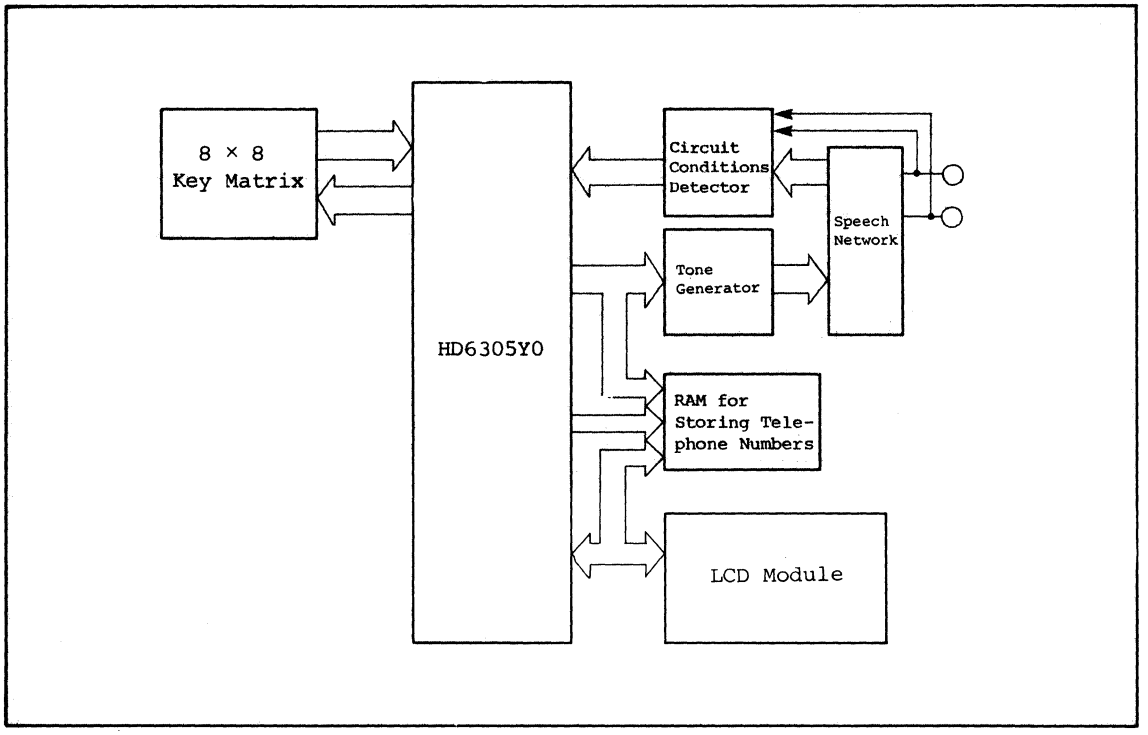


Fig. 2.20 "Driving the Liquid Crystal Display Module (H2572)" Part

(1) Hardware and Operation Description

(a) Information is displayed on the H2572 LCD module in character mode under control of the HD44780 liquid crystal driver controller (LCD-II), which in turn is controlled by the HD6305Y0.

(b) The H2572 supports 5 × 10 dot characters with cursor on 1-row × 40-column display.

(c) After receiving display data from the HD6305Y0 in ASCII code, the LCD-II drives the H2572 via the HD44100 liquid crystal drivers, as shown in Fig. 2.21.

(2) Microcomputer Applications

The HD6305Y0 displays characters on the H2572 by controlling the LCD-II data bus ($DB_0 \sim DB_7$) and control signals (E, RS and R/W) through ports A and B, as shown in Fig. 2.21.

(3) Circuit Diagram

The interface circuit between the HD6305Y0 and LCD-II is shown in Fig. 2.21.

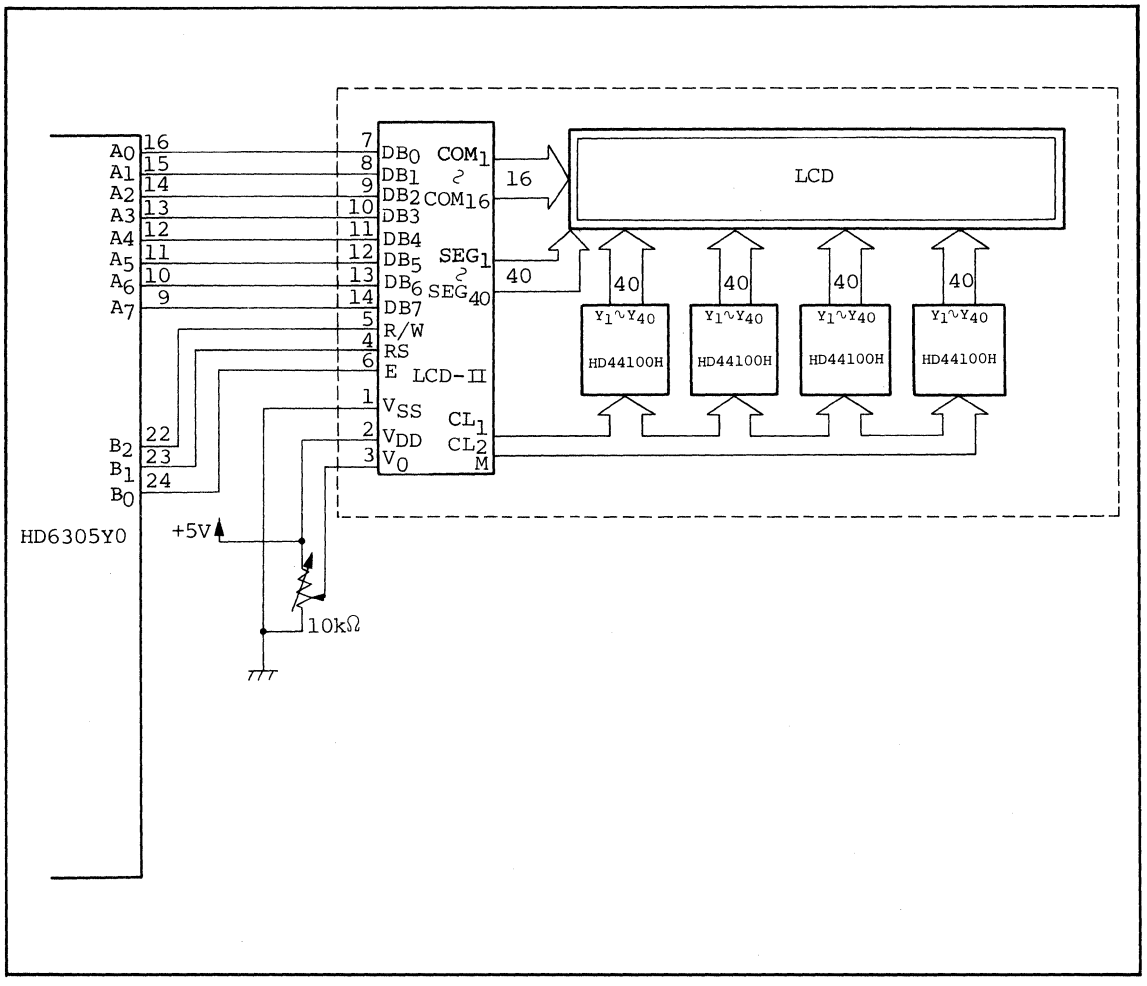


Fig. 2.21 Interface Circuit Between the HD6305Y0 and LCD-II

(4) Pin Functions

Pin functions at the interface between the HD6305Y0 and LCD-II are shown in Table 2.4.

Table 2.4 Pin Functions at the Interface Between the HD6305Y0 and LCD-II

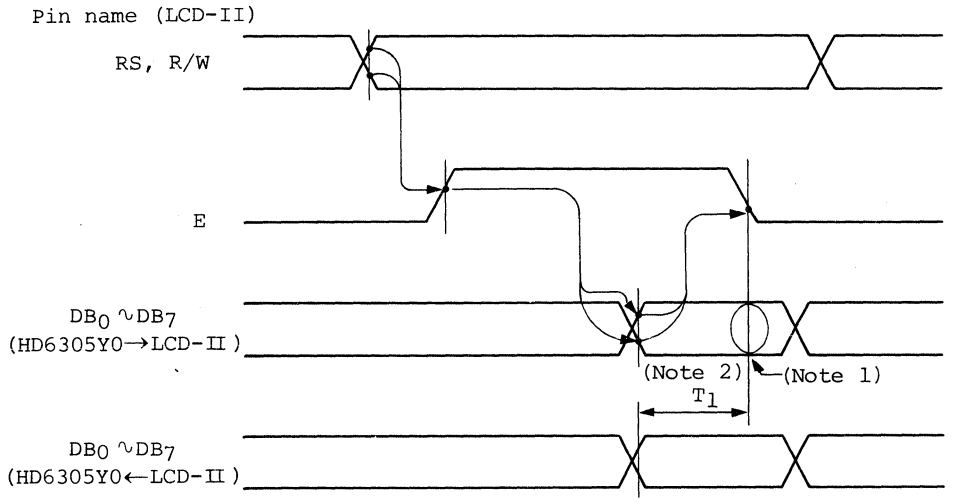
Pin Name (HD6305Y0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (LCD-II)	Program Label
F_0	Output	High	Enable signal	E	PBDTR
F_1	Output	Low	Select instruction register	RS	
		High	Select data register		
F_2	Output	Low	Write data (Microcomputer → LCD-II)	R/W	
		High	Read data (Microcomputer ← LCD-II)		
A_0	Input/ Output	—	Data lines	DB_0	PADTR
A_1	Input/ Output	—		DB_1	
A_2	Input/ Output	—		DB_2	
A_3	Input/ Output	—		DB_3	
A_4	Input/ Output	—		DB_4	
A_5	Input/ Output	—		DB_5	
A_6	Input/ Output	—		DB_6	
A_7	Input/ Output	—		DB_7	

Note: "Active Level" in Table 2.4 indicates the following;
High: logical 1
Low : logical 0
— : logical 1 or logical 0

(5) Hardware Operation

LCD-II control signals (RS, R/W, E) are controlled by software. LCD-II control signal timing is shown in Fig. 2.22.

The HD6305Y0 utilizes I/O ports rather than buses to control LCD-II to eliminate any timing restrictions.



Notes: 1. Data is written to LCTC at the falling edge of E.

2. Data from LCD-II can be read during period T_1.

Fig. 2.22 HD6305Y0←→LCD-II Interface Timing

2.4 8 × 8 Key Matrix

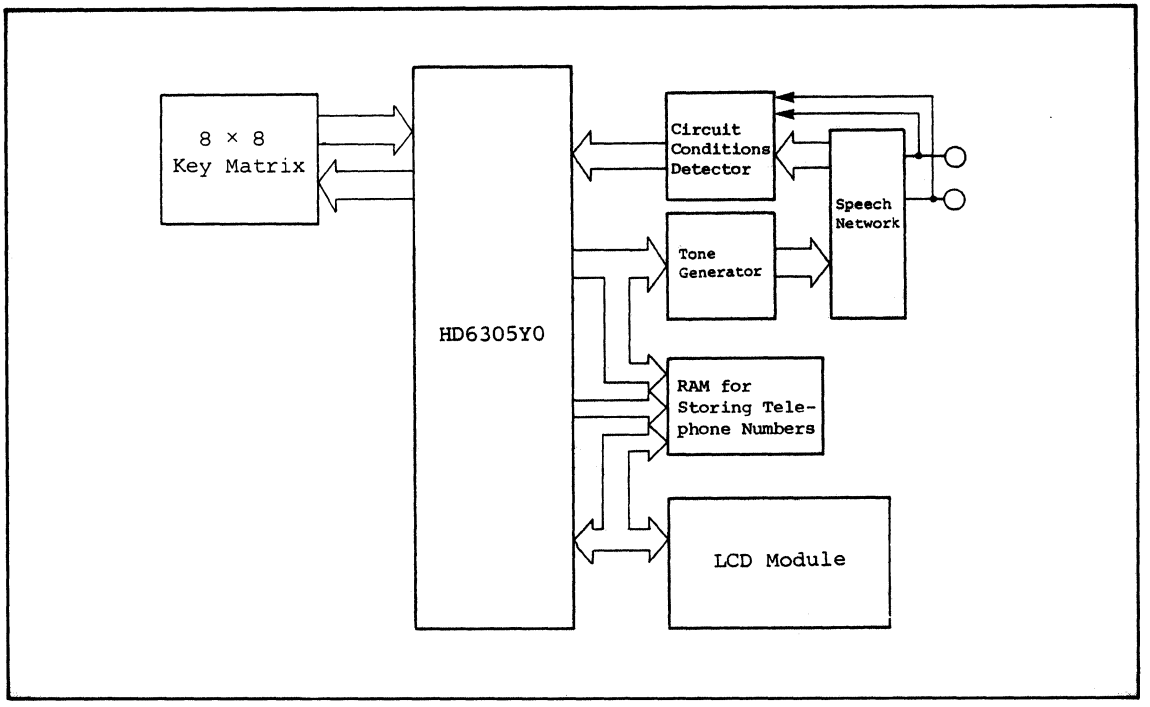


Fig. 2.23 "8 × 8 Key Matrix" Part

(1) Hardware and Operation Description

(a) The HD6305Y0 executes key scan of an 8×8 key matrix.

(b) Key data is sent to main program to distinguish which key is depressed. (In this case, key data is not ASCII code.)

(c) If two keys are depressed at the same time, data is invalid.

(2) Microcomputer Applications

(a) The HD6305Y0 timer 2 interrupt executes an interrupt routine every 16 ms.

(b) Key scan is executed by outputting a strobe signal from port C in the interrupt routine.

(c) The interrupt routine cancels key chatter.

(d) Strobe signal for key scan is controlled by switching input/output directions using data direction register (DDR) of port C. In this method, ports which output no signals are input ports (high-impedance); therefore, a diode for preventing collision between input and output signals is unnecessary.

(3) Circuit Diagram

The interface circuit between the HD6305Y0 and 8×8 key matrix is shown in Fig. 2.24.

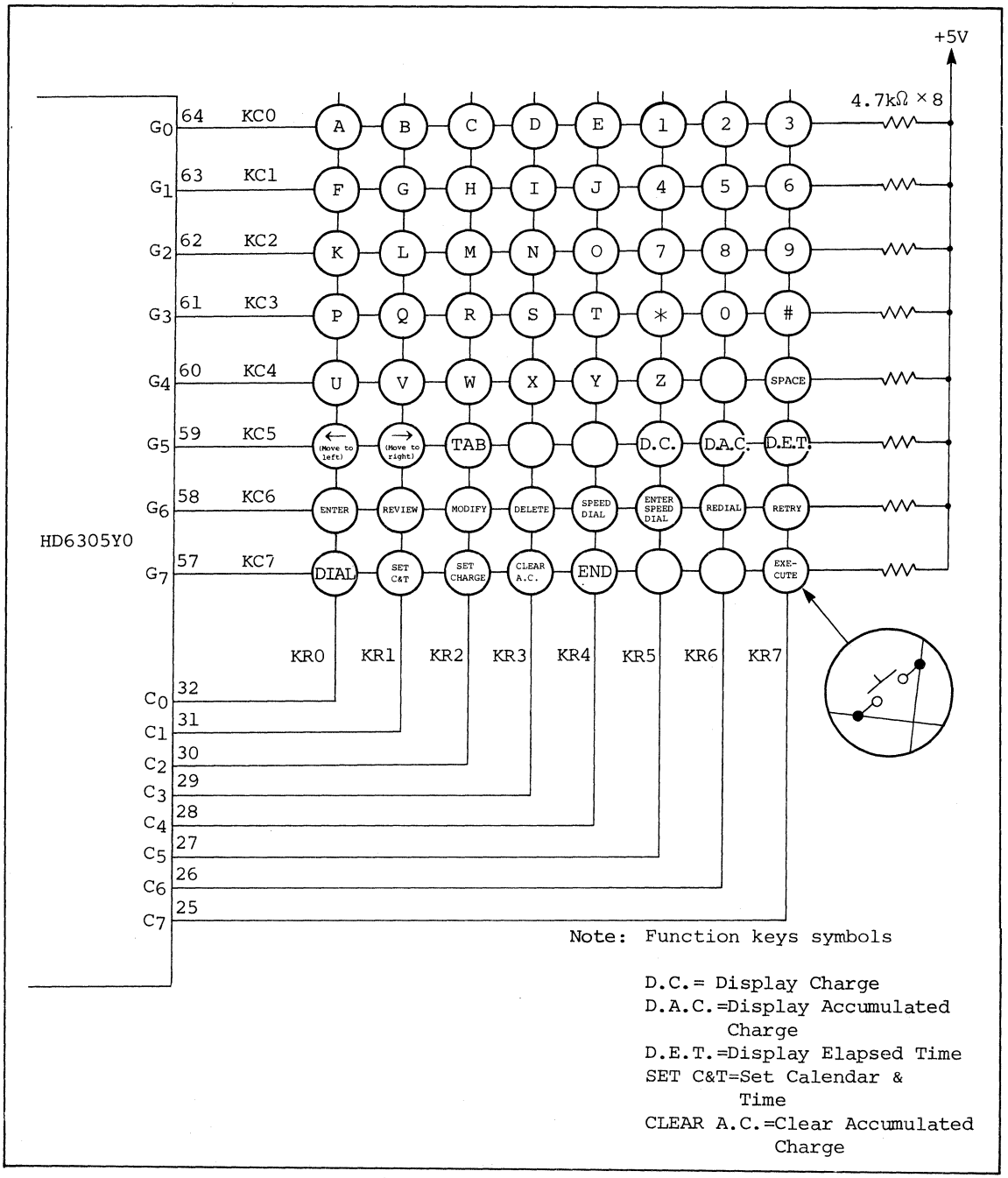


Fig. 2.24 Interface Circuit Between the HD6305Y0 and 8×8 Key Matrix

(4) Pin Functions

Pin functions at the interface between the HD6305Y0 and 8×8 key matrix are shown in Table 2.5.

Table 2.5 Pin Functions at the Interface Between the HD6305Y0 and 8×8 Key Matrix

Pin Name (HD6305Y0)	Input/ Output	Active Level (High or Low)	Function	Pin Name (Key Matrix)	Program Label
C_0	Output	Low	Output strobe signal for 8×8 key matrix. (Define ports that do not output strobe as input)	KR0	PCDTR
C_1	Output	Low		KR1	
C_2	Output	Low		KR2	
C_3	Output	Low		KR3	
C_4	Output	Low		KR4	
C_5	Output	Low		KR5	
C_6	Output	Low		KR6	
C_7	Output	Low		KR7	
G_0	Input	—	Input key data of 8 × 8 key matrix.	KC0	PGDTR
G_1	Input	—		KC1	
G_2	Input	—		KC2	
G_3	Input	—		KC3	
G_4	Input	—		KC4	
G_5	Input	—		KC5	
G_6	Input	—		KC6	
G_7	Input	—		KC7	

Note: "Active Level" in Table 2.5 indicates the following;
High: logical 1
Low : logical 0
— : logical 1 or logical 0

(5) Hardware Operation

Timing chart for preventing chatter is shown in Fig. 2.25.

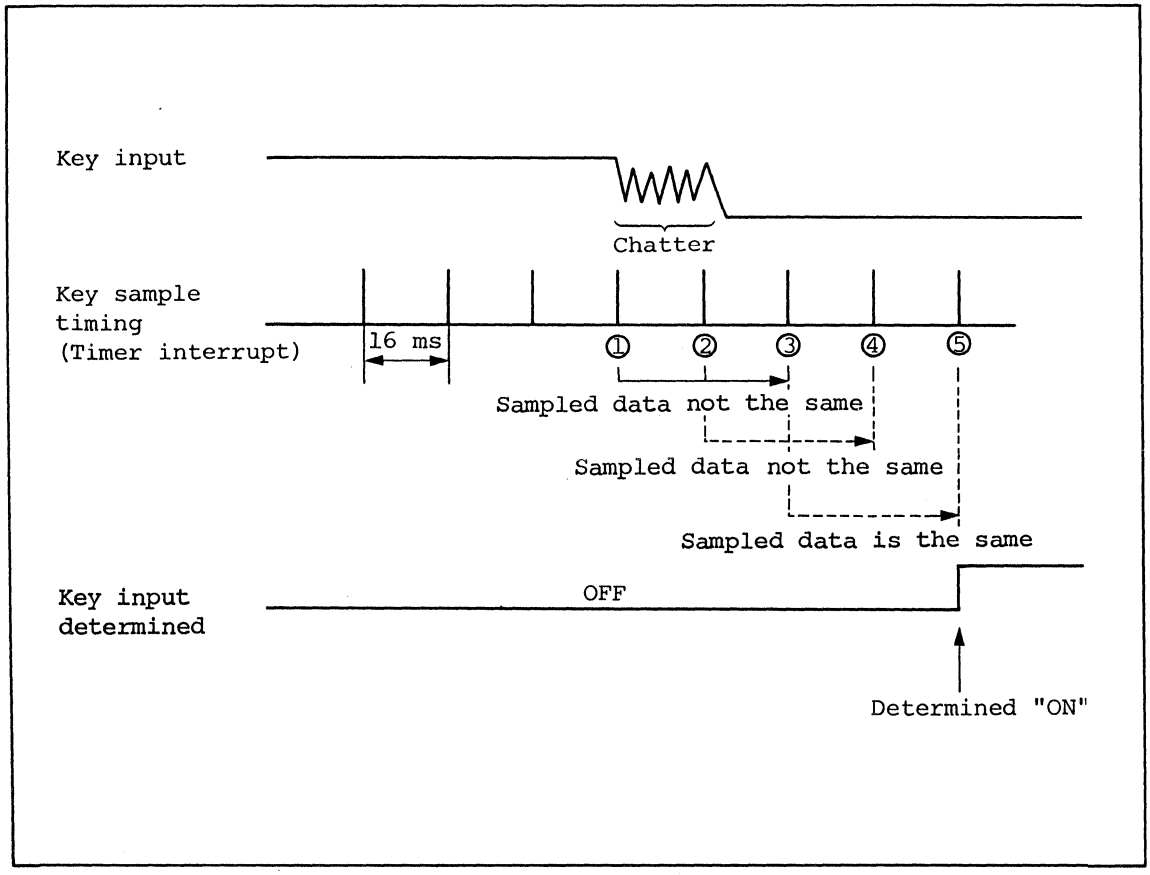


Fig. 2.25 Chatter Prevention Timing

In chatter prevention program, key input signals are sampled every 16 ms. If the same key input signal is sampled three times consecutively, key input data is established.

In Fig. 2.25, since data sampled at ① ∿ ③, ② ∿ ④ are not the same three times consecutively, the program regards them as chatter and ignores them. Data sampled at ③ ∿ ⑤, however, are the same and is thus determined as valid key input.

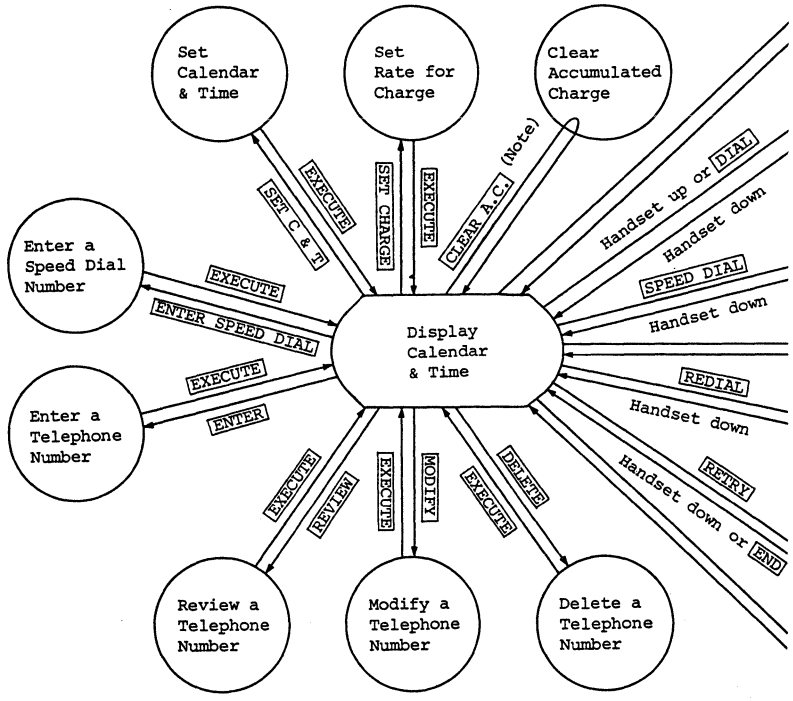
Section 3. Software Description

This chapter describes the software configuration of the Intelligent Telephone, the system model for this APPLICATION NOTE.

3.1 Transition Diagram

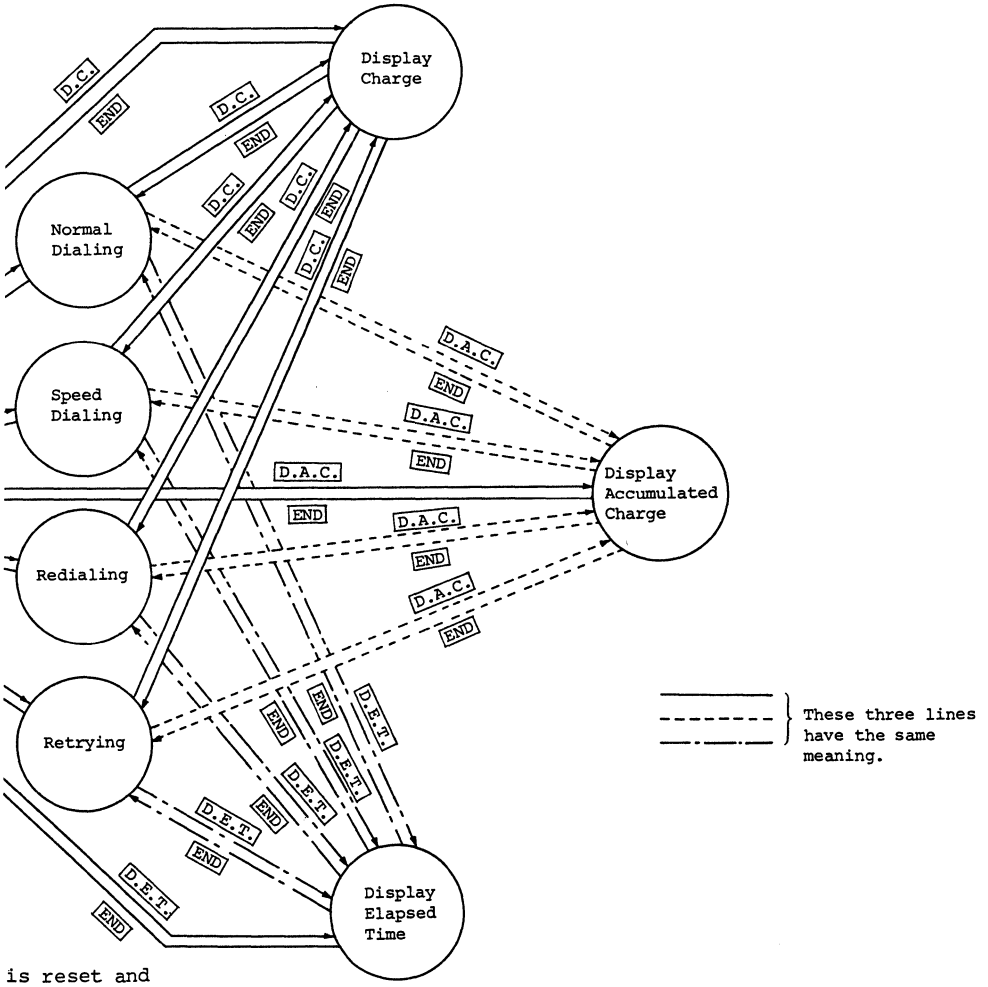
The transition diagram is shown in Fig. 3.1.

The symbol [ ] indicates pushing a key. For example, [EXECUTE] simply indicates pushing [EXECUTE] key.



Note: In case of "Clear Accumulated Charge", if [CLEAR A.C.] key is pushed, accumulated charge display remains unchanged during this process.

Fig. 3.1



is reset and

Transition Diagram

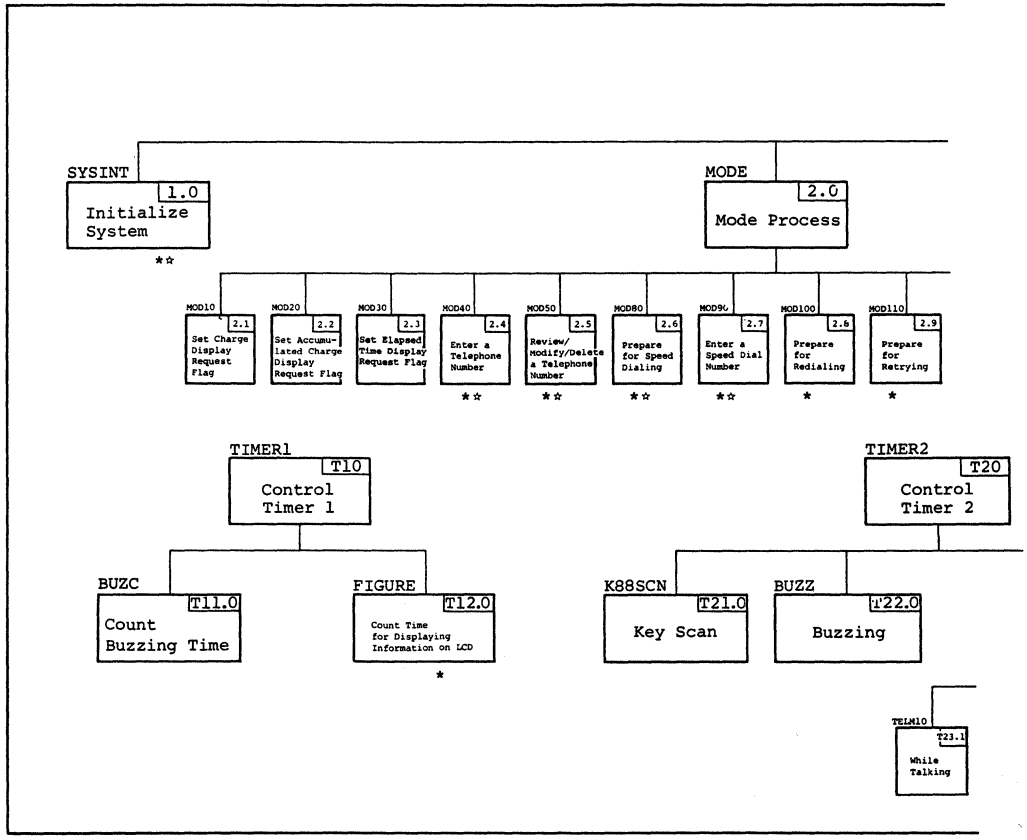
3.2 Program Module Configuration

The programs for the Intelligent Telephone consists of the following three functional groups.

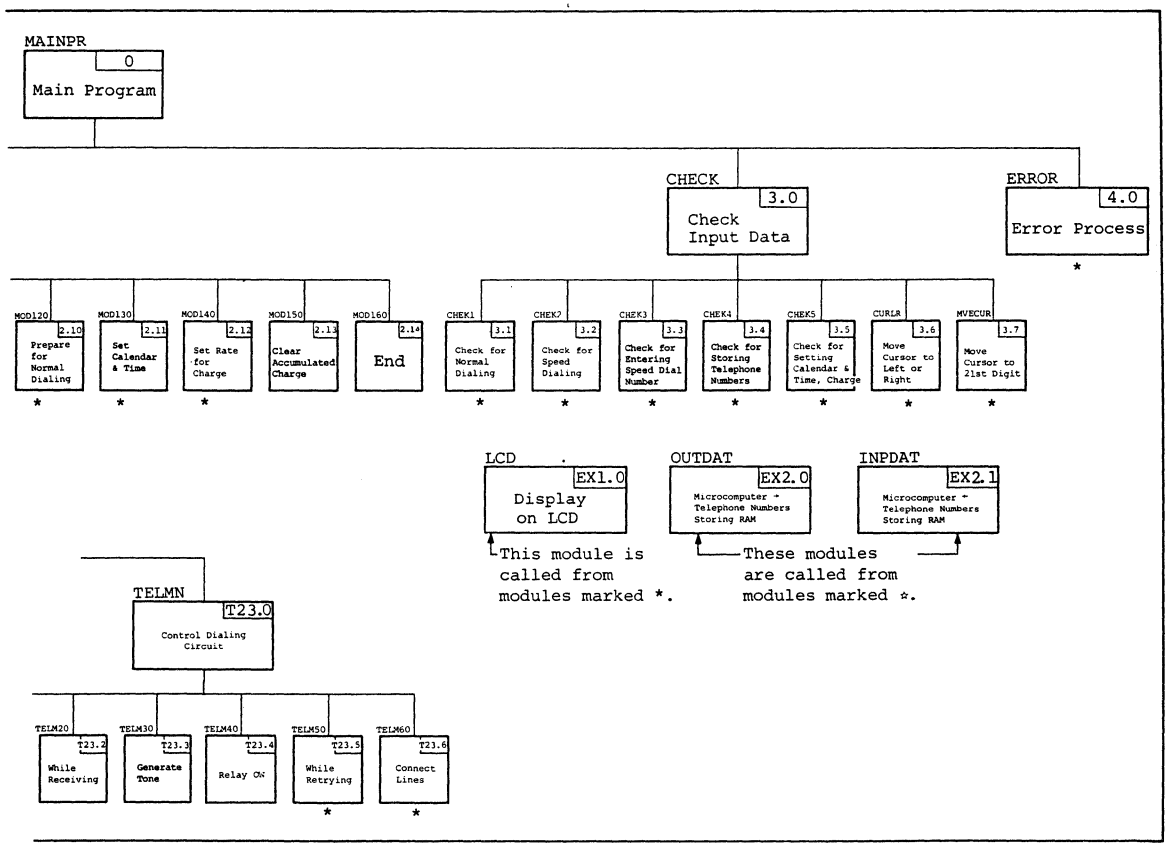
(1) Processing programs controlled by the main program. (Initialize System Mode Process · Check Input Data · Error Process).

(2) Interrupt Programs Controlled by Timer 1. (Count Buzzing Time · Count Time for Displaying Information on LCD).

(3) Interrupt Programs Controlled by Timer 2. (Key Scan · Buzzing · Control Dialing Circuit).



These three functional groups make up a triple hierarohy of program modules. Fig. 3.2 shows program module configuration and program module names (e.g.:Main Program), labels (e.g.:MAINPR), and module numbers (e.g.:0), which are used in Program Module Description.

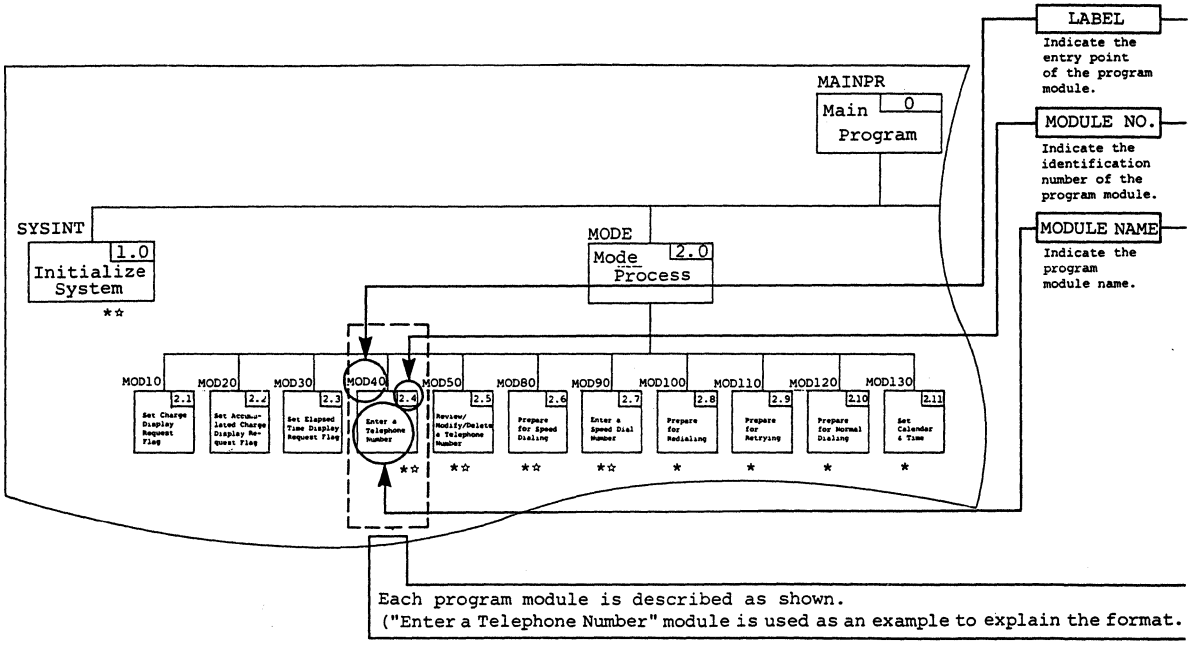


3.3 "SOFTWARE DESCRIPTION" Format

(1) Program module configuration and program module description.

The program modules are explained in "3.2 Program Module Configuration" and "3.4 Program Module Description". This part explains how these two parts are described and how the program modules are referred to.

(a) Program module configuration



(3) Program details

(a) The internal RAM used in the program is listed in "3.5 RAM Table". Some special RAM used as flags are listed in "3.6 Flag Table".

(b) The subroutines used in the program are listed in "3.7 Subroutine Table".

(c) Information on the RAM for storing telephone numbers is described in "3.8 RAM Memory Map for Storing Telephone Numbers".

(d) Port labels used in program routines are listed in "3.9 Ports Label Table".

(2) RAM description format

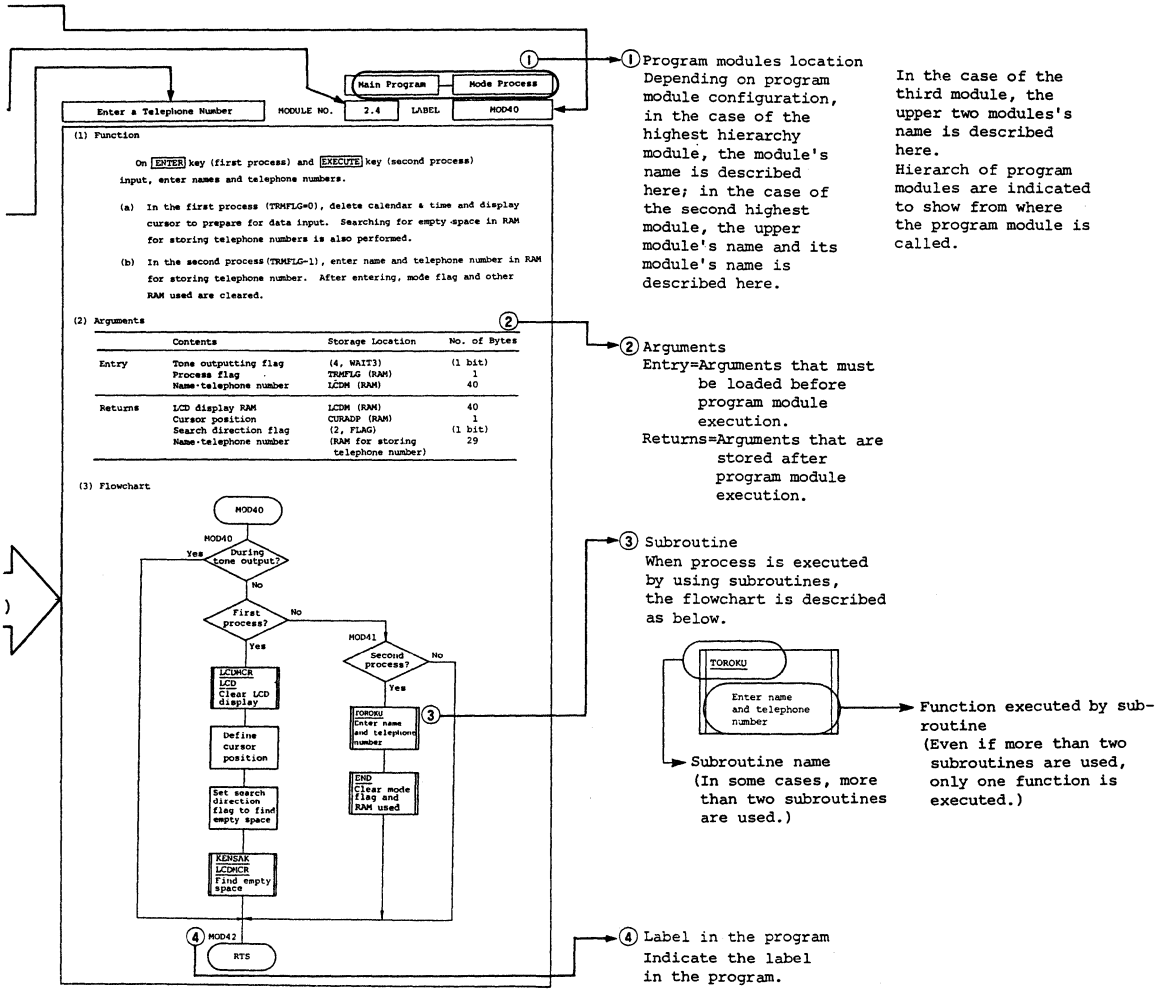
(a) RAM used by the programs is described in the following format.

Example : KEYDAT (RAM) ----- Indicates "KEYDAT" is a label in RAM.

(b) RAM used as flags is described in the following format.

Example : (0, FLAG) -------- Indicates bit position 0 of "FLAG".

(b) Program module description



① Program modules location
Depending on program module configuration, in the case of the highest hierarchy module, the module's name is described here; in the case of the second highest module, the upper module's name and its module's name is described here.

In the case of the third module, the upper two modules's name is described here.
Hierarch of program modules are indicated to show from where the program module is called.

② Arguments
Entry=Arguments that must be loaded before program module execution.
Returns=Arguments that are stored after program module execution.

③ Subroutine
When process is executed by using subroutines, the flowchart is described as below.

Function executed by subroutine
(Even if more than two subroutines are used, only one function is executed.)

Subroutine name
(In some cases, more than two subroutines are used.)

④ Label in the program
Indicate the label in the program.

Each program module in "3.2 Program Module Configuration" is explained in this section. All the module's functions are shown in Table 3.1.

Table 3.1 Program Module Function

Program Module	Label	Module No.	Function
Main Program	MAINPR	0	Controls the Intelligent Telephone
Initialize System	SYSINT	1.0	Initializes the system
Mode Process	MODE	2.0	Executes the mode process
Set Charge Display Request Flag	MOD10	2.1	Sets charge display request flag
Set Accumulated Charge Display Request Flag	MOD20	2.2	Sets accumulated charge display request flag
Set Elapsed Time Display Request Flag	MOD30	2.3	Sets elapsed time display request flag
Enter a Telephone Number	MOD40	2.4	Enters names and telephone numbers
Review/Modify/Delete a Telephone Number	MOD50	2.5	Reviews/Modifies/Deletes names and telephone numbers
Prepare for Speed Dialing	MOD80	2.6	Prepares for speed dialing
Enter a Speed Dial Number	MOD90	2.7	Enter speed dialing numbers
Prepare for Redialing	MOD100	2.8	Prepares for redialing
Prepare for Retrying	MOD110	2.9	Prepares for retrying
Prepare for Normal Dialing	MOD120	2.10	Prepares for normal dialing
Set Calendar & Time	MOD130	2.11	Sets correct calendar & time
Set Rate for Charge	MOD140	2.12	Sets charge per time
Clear Accumulated Charge	MOD150	2.13	Clears current accumulated charge
End	MOD160	2.14	Clears mode flag and RAM used
Check Input Data	CHECK	3.0	Checks input data and moves cursor
Check for Normal Dialing	CHEK1	3.1	Checks if correct data for normal dialing
Check for Speed Dialing	CHEK2	3.2	Checks if correct data for speed dialing
Check for Entering Speed Dial Number	CHEK3	3.3	Checks if correct data for entering speed dial
Check for Storing Telephone Numbers	CHEK4	3.4	Checks if correct data for storing telephone numbers
Check for Setting Calendar & Time, Charge	CHEK5	3.5	Checks if correct data for setting calendar & time, charge
Move Cursor to Left or Right	CURLR	3.6	Moves cursor to left or right
Move Cursor to 21st Digit	MVECUR	3.7	Moves cursor to 21st digit
Error Process	ERROR	4.0	Executes error process
Control Timer 1	TIMER1	T10	Controls timer 1
Count Buzzing Time	BUZC	T11.0	Counts buzzing time
Count Time for Displaying Information on LCD	FIGURE	T12.0	Counts time or charge for LCD display
Control Timer 2	TIMER2	T20	Controls timer 2
Key Scan	K88SCN	T21.0	Executes 8×8 key scan
Buzzing	BUZZ	T22.0	Outputs buzzing
Control Dialing Circuit	TELMN	T23.0	Controls dialing
While Talking	TELM10	T23.1	Tests if talking
While Receiving	TELM20	T23.2	Tests if receiving
Generate Tone	TELM30	T23.3	Performs tone output

Table 3.1 Program Module Function (cont.)

Program Module	Label	Module No.	Function
Relay ON	TELM40	T23.4	Connects line using relay
While Retrying	TELM50	T23.5	Executes retrying
Connect Lines	TELM60	T23.6	Prepares for dialing
Display on LCD	LCD	EX1.0	Displays figure on LCD
Microcomputer → Telephone Numbers Storing RAM	OUTDAT	EX2.0	Moves data from microcomputer to RAM
Microcomputer ← Telephone Numbers Storing RAM	INPDAT	EX2.1	Moves data from RAM to microcomputer

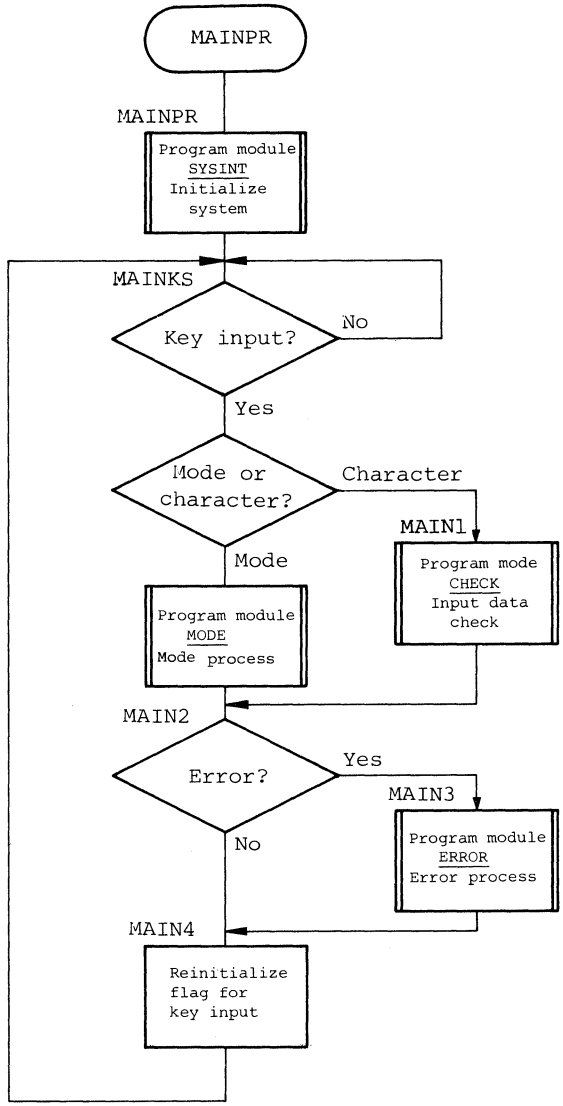
Main Program	MODULE NO.	0	LABEL	MAINPR

(1) Function

"Main program" controls the major processes of the Intelligent Telephone. It first initializes the system, and after enabling timer interrupts, waits for key input. If any key is depressed, data is input identifying the key. The main program divides this data into mode and character types. If any error occurs during mode and character processing, the error flag is set and the error process routine is executed. (Relation between the main program and key data is described in "Note".)

Main Program	MODULE NO.	0	LABEL	MAINPR

(2) Flowchart



Note: Main program and key data

The keyboard configuration for this system is shown in Fig. 3.3. The data shown above each key identifies which key is depressed. (These are not ASCII.) This data, called "key data", is input to the main program which divides them into mode and character types for respective processing.

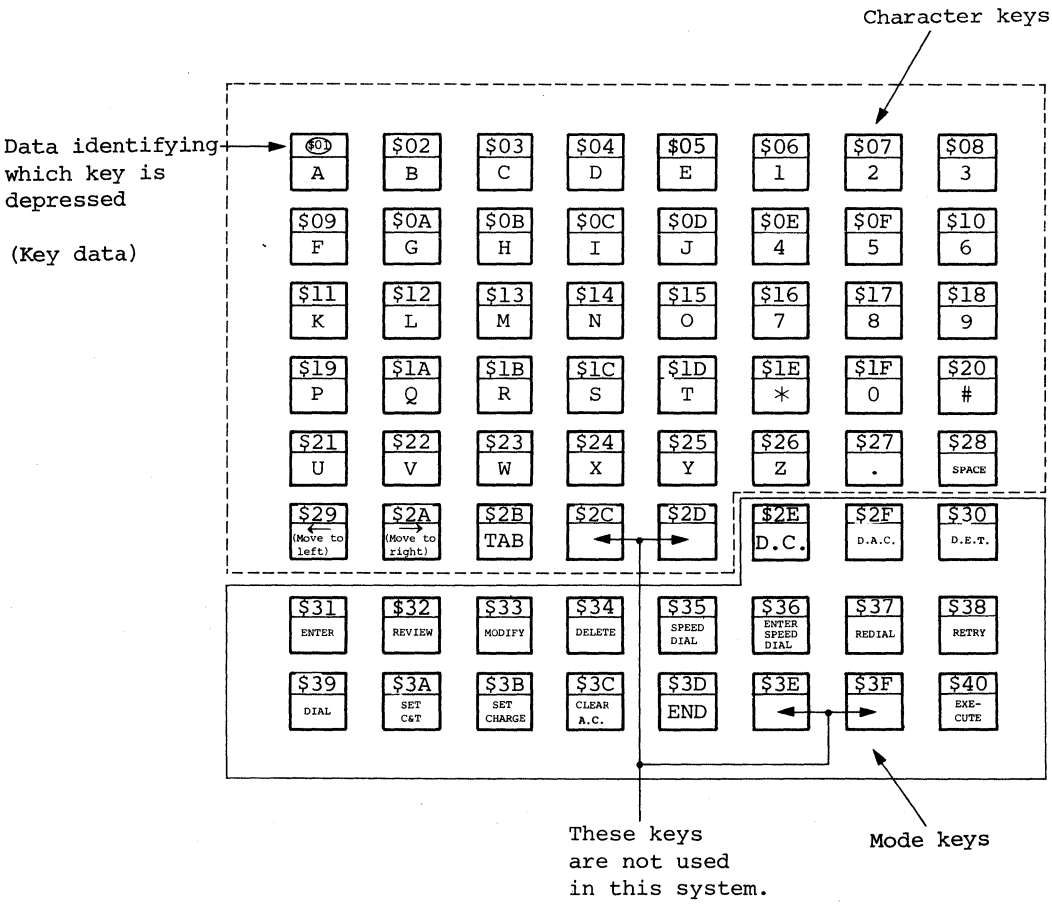


Fig. 3.3 Keys

Initialize System	MODULE NO.	1.0	LABEL	SYSINT

(1) Function

"Initialize System" initializes timers, ports, and internal RAM. If necessary, it initializes RAM for storing telephone numbers.

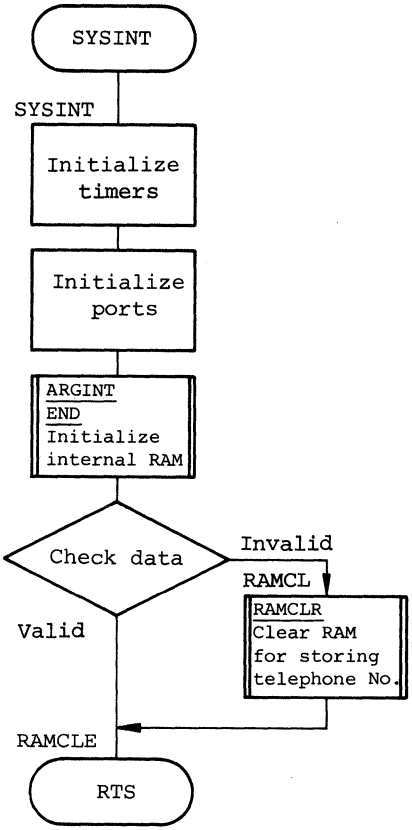
Before initializing RAM for storing telephone numbers, test its check data fields so as not to clear stored telephone numbers unnecessarily. After checking data, even if only one check data field is invalid, RAM is cleared and new check data is entered again for reinifialization. If all the check data are valid, data in RAM is determined to be valid and RAM is not cleared.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	———	———	———
Returns	———	———	———

Initialize System	MODULE NO.	1.0	LABEL	SYSINT

(3) Flowchart



Mode Process	MODULE NO.	2.0	LABEL	MODE

(1) Function

"Mode Process" is a driver routine for executing the various mode processes corresponding to key data. Before branching to a mode, the mode flag (Note 1) is defined first. A particular mode is executed depending on this mode flag.

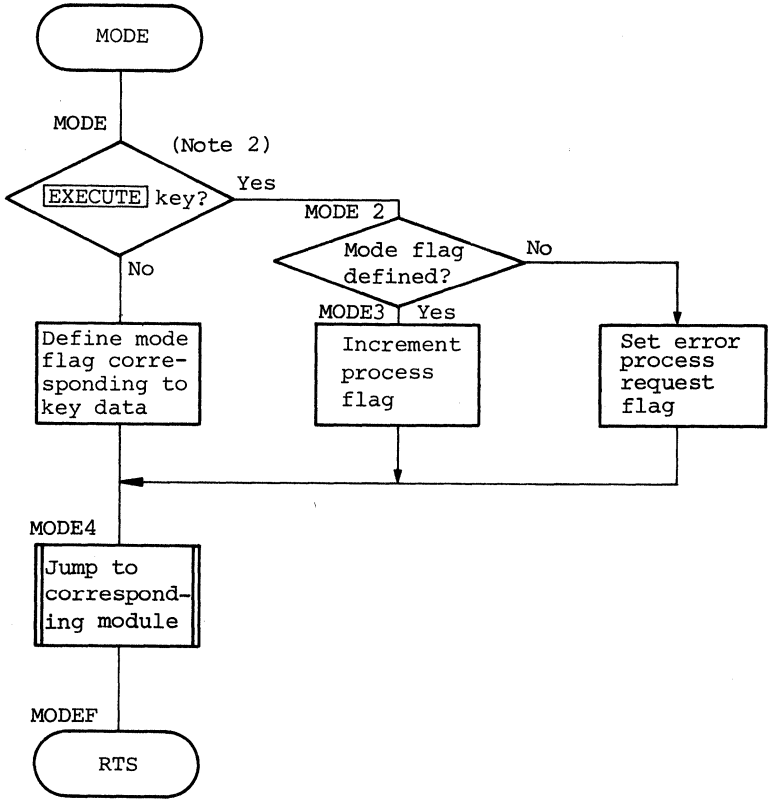
(2) Arguments

Common entry and return arguments for all the mode processes are shown below. Particular entry and return arguments for each mode are explained in each mode's description in detail.

	Contents	Storage Location	No. of Bytes
Entry	Key data	KEYDAT (RAM)	1
	Mode flag	MODFLG (RAM)	1
	Process flag	TRMFLG (RAM)	1
Returns	Mode flag	MODFLG (RAM)	1
	Process flag	TRMFLG (RAM)	1
	Error process request flag	(0, ERFLG)	(1 bit)

Mode Process	MODULE NO.	2.0	LABEL	MODE

(3) Flowchart



Notes: 1. Key data input to "Mode Process"

Key data (KEYDAT) is checked as to whether it is the EXECUTE key or another key. If another key is input, key data is changed to mode flag (MODFLG) (If the EXECUTE key, refer to the next page). Jump to the corresponding module is executed depending on this mode flag.

The mode flag and corresponding modules are shown in Table 3.2.

Table 3.2 Mode Flag and Corresponding Modules of "Mode Process"

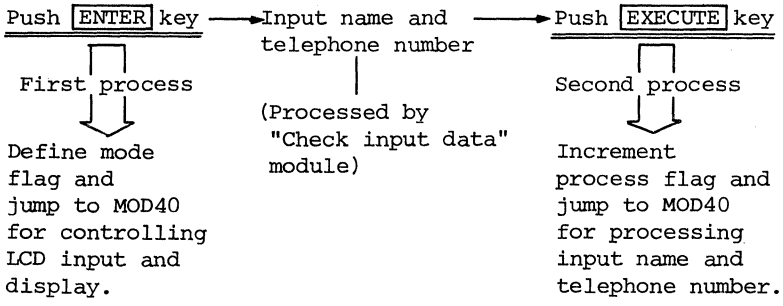
Key data (KEYDAT)		Mode flag (MODFLG)		Module	
				Label	Function
$2E		$1		MOD10	Set charge display request flag
$2F		$2		MOD20	Set accumulated charge display request flag
$30		$3		MOD30	Set elapsed time display request flag
$31		$4		MOD40	Enter a telephone number
$32		$5		MOD50	Review/Modify/Delete a telephone number
$33		$6		MOD50	
$34	Change	$7	Call	MOD50	
$35		$8		MOD80	Prepare for speed dialing
$36		$9		MOD90	Enter a speed dial number
$37		$A		MOD100	Prepare for redialing
$38		$B		MOD110	Prepare for retrying
$39		$C		MOD120	Prepare for normal dialing
$3A		$D		MOD130	Set calendar & time
$3B		$E		MOD140	Set rate for charge
$3C		$F		MOD150	Clear accumulated charge
$3D		$10		MOD160	End

2. How to process [EXECUTE] key input:

For example, when entering a telephone number, the following procedure is performed as below.

Push [ENTER] key ⟶ Input name and telephone number ⟶ Push [EXECUTE] key

This procedure corresponds to "First" and "Second" processes performed by the module corresponding to the defined mode flag.



The [EXECUTE] key may never be depressed at the beginning of a module. If it is depressed, the program processes it as an error.)

- When a mode flag except for the [EXECUTE] key is depressed, the program regards it as "First process".
- When the [EXECUTE] key is depressed, the program regards it as "Second process".

Main Program	Mode Process

Set Charge Display Request Flag	MODULE NO.	2.1	LABEL	MOD10

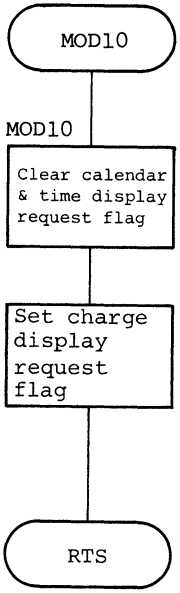
(1) Function

On DISPLAY CHARGE key input, clear calendar & time display request flag and set charge display request flag to display charge as counted by timer 1.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	————	————	————
Returns	Calender & time display request flag	(1, FLAG)	(1 bit)
	Charge display request flag	(7, FLAG)	(1 bit)

(3) Flowchart



Main Program	Mode Process

Set Accumulated Charge Display Request Flag	MODULE NO.	2.2	LABEL	MOD20

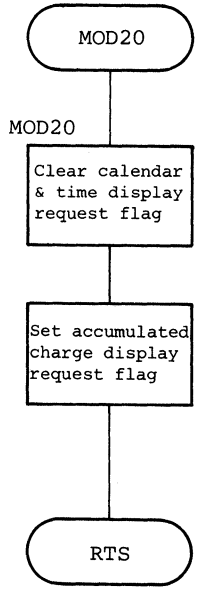
(1) Function

On DISPLAY ACCUMULATED CHARGE key input, clear calendar & time display request flag and set accumulated charge display request flag to display accumulated charge as counted by timer 1.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	———	———	———
Returns	Calender & time display request flag	(1, FLAG)	(1 bit)
	Accumulated charge display request flag	(6, FLAG)	(1 bit)

(3) Flowchart



Set Elapsed Time Display Request Flag	MODULE NO.	2.3	LABEL	MOD30

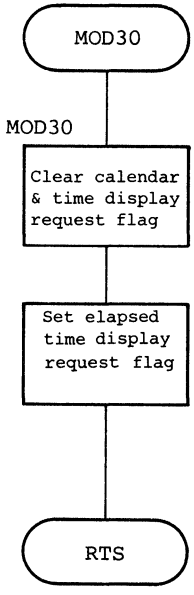
(1) Function

On DISPLAY ELAPSED TIME key input, clear calendar & time display request flag and set elapsed time display request flag to display elapsed time as counted by timer 1.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	———	———	———
Returns	Calender & time display request flag	(1, FLAG)	(1 bit)
	Elapsed time display request flag	(3, FLAG)	(1 bit)

(3) Flowchart



Main Program	Mode Process

Enter a Telephone Number	MODULE NO.	2.4	LABEL	MOD40

(1) Function

On [ENTER] key (first process) and [EXECUTE] key (second process) input, enter names and telephone numbers.

(a) In the first process (TRMFLG=0), delete calendar & time and display cursor to prepare for data input. Searching for empty space in RAM for storing telephone numbers is also performed.

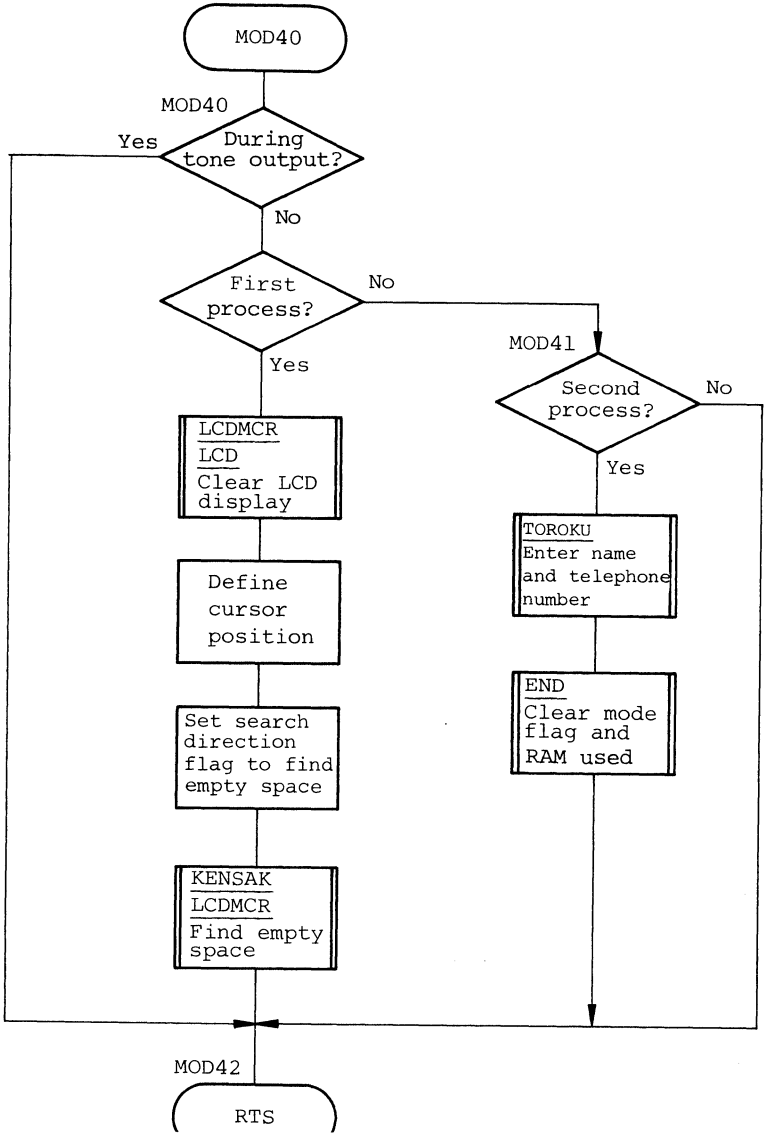
(b) In the second process (TRMFLG=1), enter name and telephone number in RAM for storing telephone number. After entering, mode flag and other RAM used are cleared.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Tone outputting flag	(4, WAIT3)	(1 bit)
	Process flag	TRMFLG (RAM)	1
	Name·telephone number	LCDM (RAM)	40
Returns	LCD display RAM	LCDM (RAM)	40
	Cursor position	CURADP (RAM)	1
	Search direction flag	(2, FLAG)	(1 bit)
	Name·telephone number	(RAM for storing telephone number)	29

Enter a Telephone Number	MODULE NO.	2.4	LABEL	MOD40

(3) Flowchart



(1) Function

On [ENTER] / [MODIFY] / [DELETE] key input, review/modify/delete a telephone number previously entered and stored in RAM. To modify or delete a telephone number, the desired data must first be reviewed, after which the previously specified process is executed.

(a) In the first process (TRMFLG=0), clear the display and define cursor position.

(b) In the second process (TRMFLG=1), display the first telephone number to be reviewed.

(c) In the third process (TRMFLG=2), corresponding to [→] key or [←] key input, locate the desired data. In this process, [EXECUTE] key is not depressed, however sub-processes for reviewing data are instead determined by the "Input Data Check" routine. Reviewing of data is continued until the desired data is found.

(d) In the fourth process (TRMFLG=3), determine whether reviewing is completed, or modifying or deleting data.

(i) Review .... In the fourth process (TRMFLG=3), complete reviewing and clear mode flag, process flag, and RAM used.

(ii) Modify .... In the fourth process (TRMFLG=3), display data to be modified and enable cursor movement. In the fifth process (TRMFLG=4), store modified telephone number in RAM for storing telephone number and clear mode flag, process flag, and RAM used.

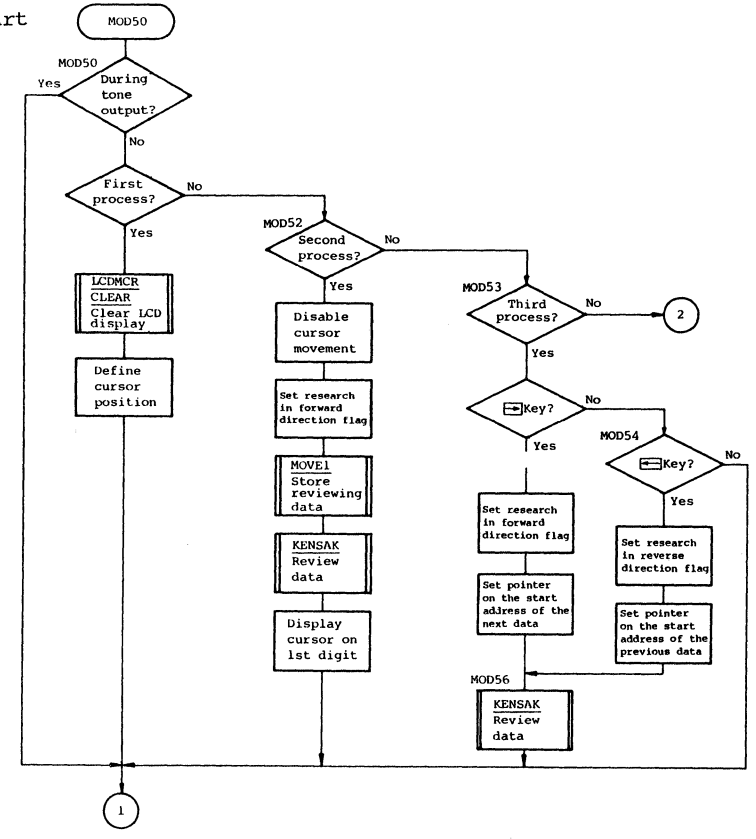
(iii) Delete .... In the fourth process (TRMFLG=3), delete desired data and clear mode flag, process flag, and RAM used.

Review/Modify/Delete a Telephone Number	MODULE NO.	2.5	LABEL	MOD50

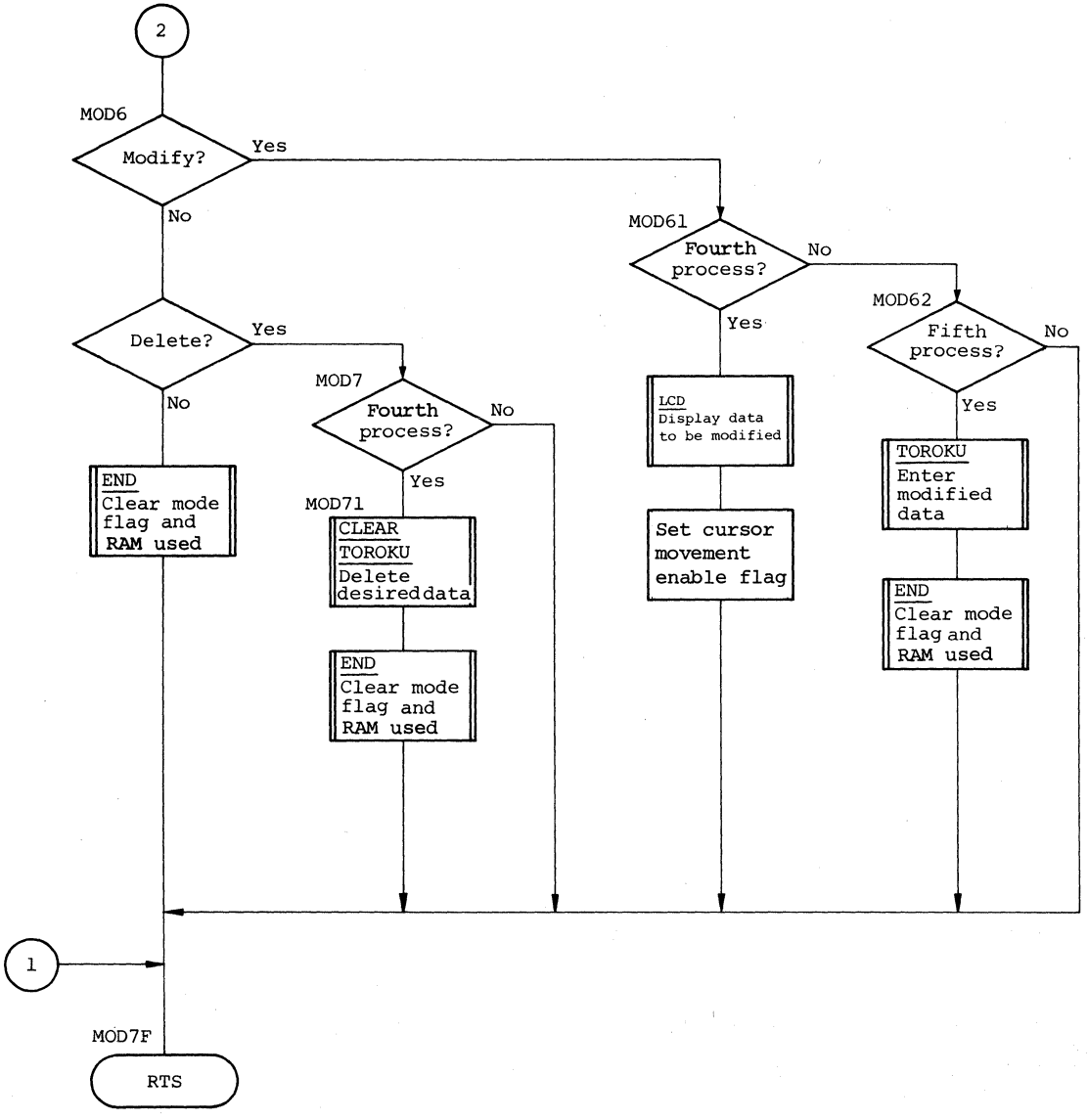
(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Tone outputting flag	(4, WAIT3)	(1 bit)
	Process flag	TRMFLG (RAM)	1
	Reviewing data	LCDM (RAM)	40
	Key data	KEYDAT (RAM)	1
	Modified data	LCDM (RAM)	1
Returns	LCD display RAM	LCDM (RAM)	40
	Cursor position	CURADP (RAM)	1
	LCD display pointer	LCDMP (RAM)	1
	Cursor movement disable/enable flag	(7, BZRFLG)	(1 bit)
	Search direction flag	(2, FLAG)	(1 bit)
	Modified/deleted data	(RAM for storing telephone number)	29

(3) Flowchart



Review/Modify/Delete a Telephone Number | MODULE NO. 2.5 | LABEL MOD50



Prepare for Speed Dialing	MODULE NO.	2.6	LABEL	MOD80

(1) Function

(Note)
On [SPEED DIAL] key (first process) and [SPEED DIAL CODE] (second process) input, load telephone number corresponding to desired speed dial number from RAM for storing telephone number. Actual tone output, however, is executed by timer 2.

Note: In this case, [EXECUTE] key is not depressed, however sub-processes for reviewing data are instead determined by the "Input Data Check" routine.

(a) In the first process (TRMFLG=0), clear display and display "NO.?_".

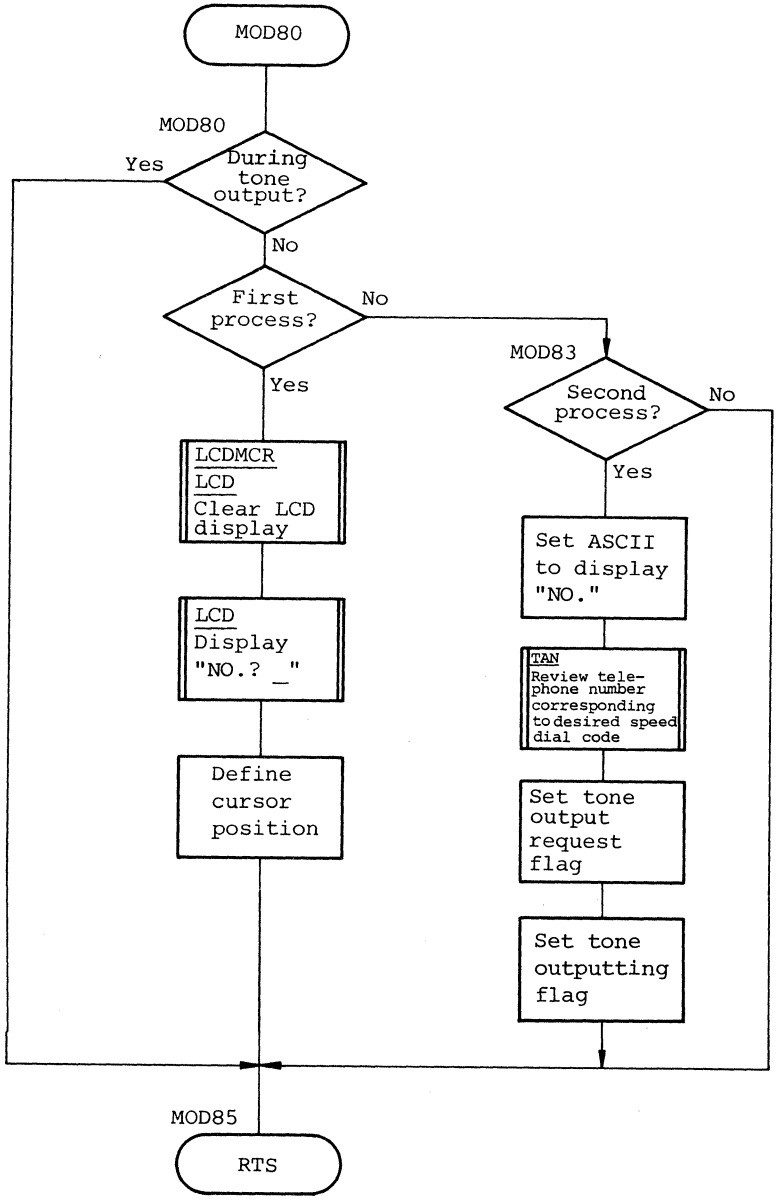
(b) In the second process (TRMFLG=1), display telephone number corresponding to desired speed dial code and set tone output request flag.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Tone outputting flag	(4, WAIT3)	(1 bit)
	Process flag	TRMFLG (RAM)	1
	Speed dial code	LCDM (RAM)	40
Returns	Telephone number corresponding to desired speed dial code	LCDM (RAM)	40
	Cursor position	CURADP (RAM)	1
	LCD display pointer	LCDMP (RAM)	1
	Tone output request flag	(1, BZR)	(1 bit)
	Tone outputting flag	(4, WAIT3)	(1 bit)

Prepare for Speed Dialing	MODULE NO.	2.6	LABEL	MOD80

(3) Flowchart



Enter a Speed Dial Number	MODULE NO.	2.7	LABEL	MOD90

(1) Function

On [ENTER SPEED DIAL] key (first process) and [EXECUTE] key (second and third process) input, enter speed dial to RAM for storing telephone number.

(a) In the first process (TRMFLG=0), clear calendar & time and display cursor.

(b) In the second process (TRMFLG=1), display speed dial code and display cursor for telephone number input.

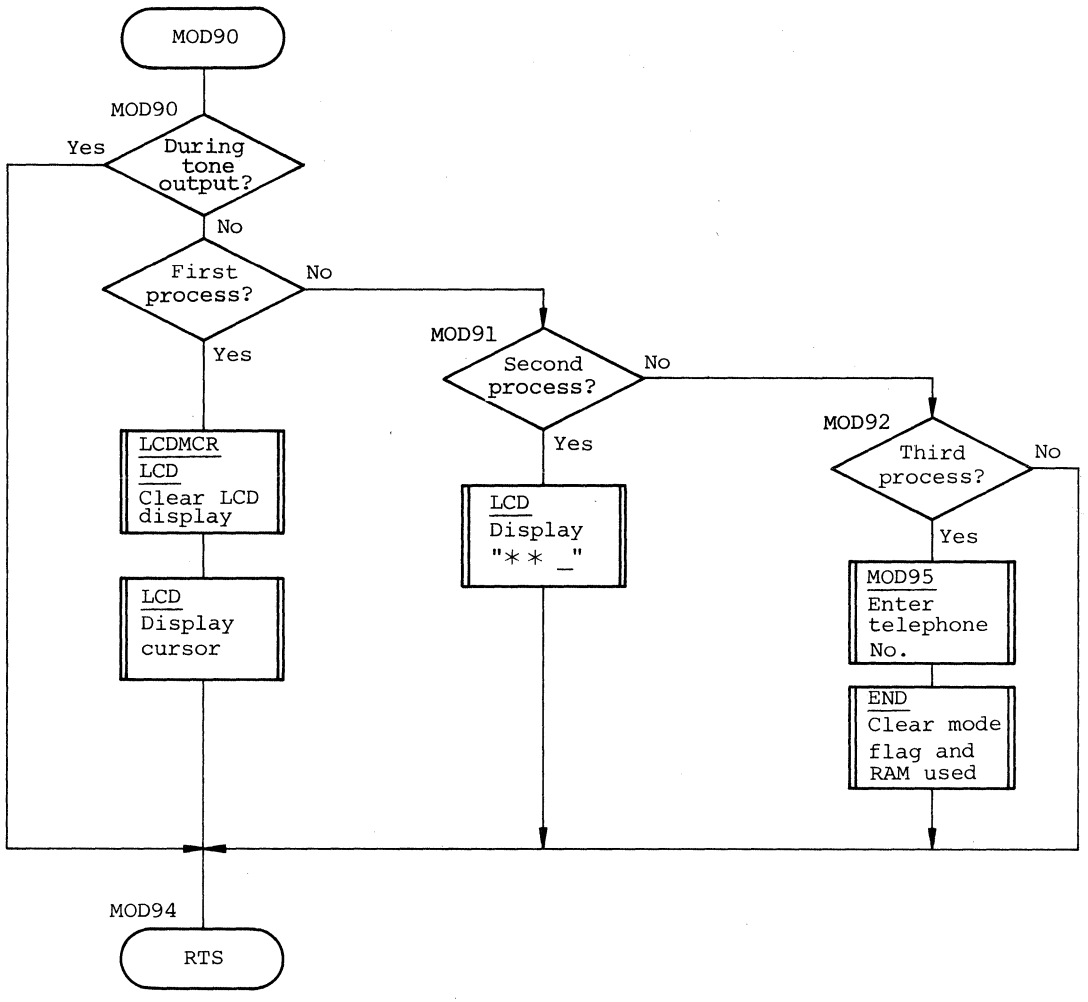
(c) In the third process (TRMFLG=2), enter speed dial code and corresponding telephone number. Then clear mode flag, process flag and RAM used.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Tone outputting flag	(4, WAIT3)	(1 bit)
	Process flag	TRMFLG (RAM)	1
	Speed dial code · telephone number	LCDM (RAM)	40
Returns	LCD display RAM	LCDM (RAM)	40
	Cursor position	CURADP (RAM)	1
	LCD display pointer	LCDMP (RAM)	1
	Telephone number	(RAM for storing telephone number)	9

Enter a Speed Dial Number | MODULE NO. 2.7 | LABEL MOD90

(3) Flowchart



Prepare for Redialing	MODULE NO.	2.8	LABEL	MOD100

(1) Function

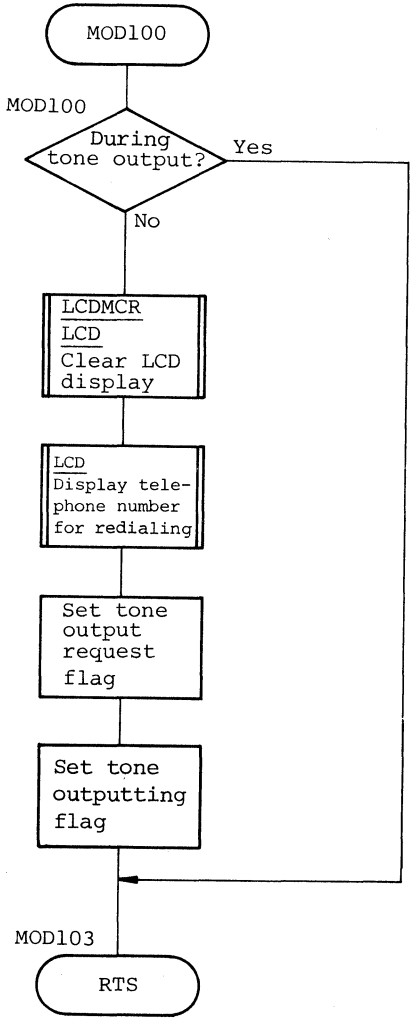
On [REDIAL] key input, load the telephone number, previously output, from RAM for redialing. Actual tone output, however, is executed by timer 2.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Tone outputting flag	(4, WAIT3)	(1 bit)
	Telephone number previously output	MEMO (RAM)	21
Returns	Telephone number for redialing	LCDM (RAM)	40
	Tone output request flag	(1, BZR)	(1 bit)
	Tone outputting flag	(4, WAIT3)	(1 bit)

Prepare for Redialing	MODULE NO.	2.8	LABEL	MOD100

(3) Flowchart



Main Program	Mode Process

Prepare for Retrying	MODULE NO.	2.9	LABEL	MOD110

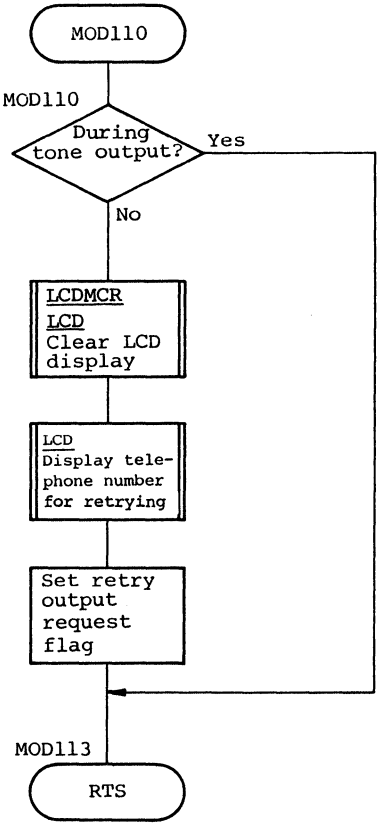
(1) Function

On [RETRY] key input, load the telephone number, previously output, from RAM for retrying. Actual tone output, however, is executed by timer 2.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Tone outputting flag	(4, WAIT3)	(1 bit)
	Telephone number previously output	MEMO (RAM)	21
Returns	Telephone number for retrying	LCDM (RAM)	40
	Retry output request flag	(0, BZRFLG)	(1 bit)

(3) Flowchart



Main Program	Mode Process

Prepare for Normal Dialing	MODULE NO.	2.10	LABEL	MOD120

(1) Function

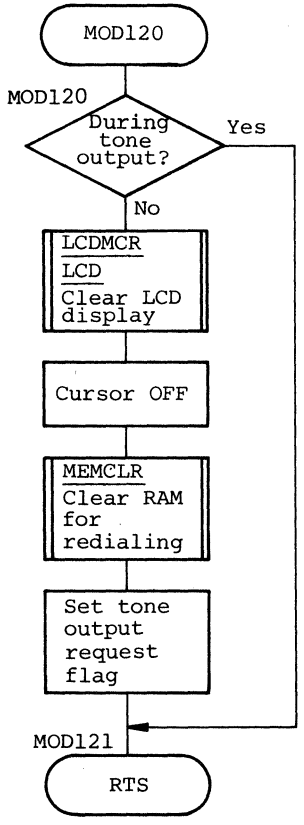
On [DIAL] key input, prepare for normal dialing.

Actual tone output, however, is executed by timer 2.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Tone outputting flag	(4, WAIT3)	(1 bit)
Returns	LCD display RAM	LCDM (RAM)	40
	Cursor OFF	CURADP (RAM)	1
	RAM for redialing	MEMO (RAM)	21
	Tone output request flag	(1, BZR)	(1 bit)

(3) Flowchart



Main Program	Mode Process

Set Calendar & Time	MODULE NO.	2.11	LABEL	MOD130

(1) Function

On [SET CALENDAR & TIME] key (first process) and [EXECUTE] key (second process) input, set calendar & time.

(a) In the first process (TRMFLG=0), clear calendar & time display request flag and display the current calendar & time.

(b) In the second process (TRMFLG=1), determine whether or not the modified data is correct. If data is correct, set modified calendar & time and clear mode flag, process flag, and RAM used. If incorrect, wait for modified data input again.

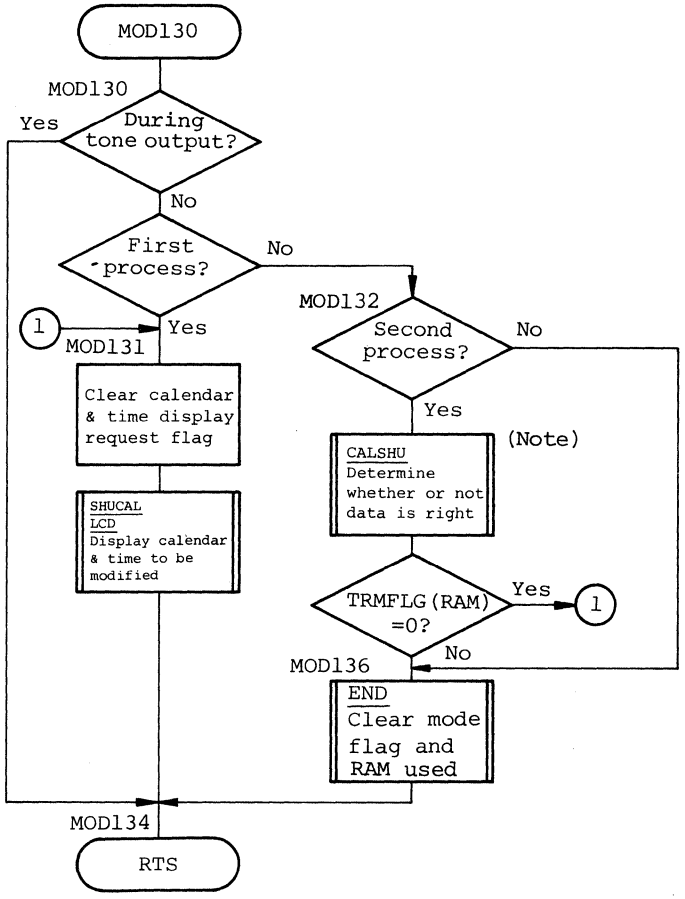
(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Tone outputting flag	(4, WAIT3)	(1 bit)
	Process flag	TRMFLG (RAM)	1
	Modified calendar & time	LCDM (RAM)	40
Returns	Calendar & time display request flag	(1, FLAG)	(1 bit)
	Calendar & time to be modified	LCDM (RAM)	40
	Calendar & time modified and determined to be correct	CALNDR (RAM)	5

Main Program	Mode Process

Set Calendar & Time	MODULE NO.	2.11	LABEL	MOD130

(3) Flowchart



Note: Determine whether or not modified data is correct format as calendar & time. For example, 30:65:01 is not correct format for time. If data is correct, set modified calendar & time. If incorrect, store "0" in TRMFLG (RAM).

Set Rate for Charge	MODULE NO.	2.12	LABEL	MOD140

(1) Function

On [SET CHARGE] key (first process) and [EXECUTE] key (second process) input, set charge per unit time.

(a) In the first process (TRMFLG=0), clear calendar & time display request flag and display the current charge per unit time.

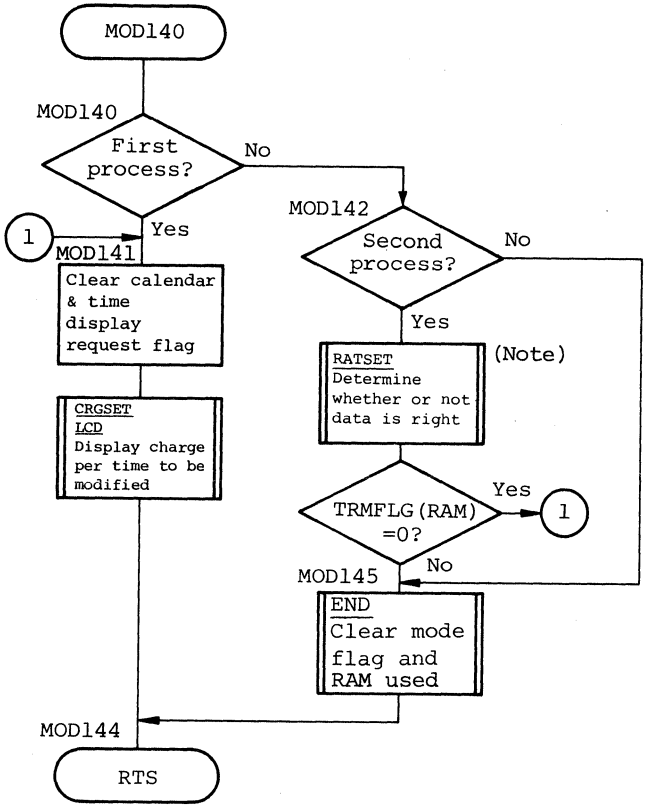
(b) In the second process (TRMFLG=1), determine whether or not the modified data is correct. If data is correct, set modified charge per unit time and clear mode flag, process flag, and RAM used. If incorrect, wait for modified data input again.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Process flag	TRMFLG (RAM)	1
	Modified charge per time	LCDM (RAM)	40
Returns	Calendar & time display request flag	(1, FLAG)	(1 bit)
	Charge per time to be modified	LCDM (RAM)	40
	Charge per time modified and determined to be correct	ADDRAT (RAM) SEC (RAM)	4 2

Set Rate for Charge	MODULE NO.	2.12	LABEL	MOD140

(3) Flowchart



Note:

Determine whether or not modified data is correct format as charge per time. If data is correct, set modified charge per time. If incorrect, store "0" in TRMFLG (RAM).

Clear Accumulated Charge	MODULE NO.	2.13	LABEL	MOD150

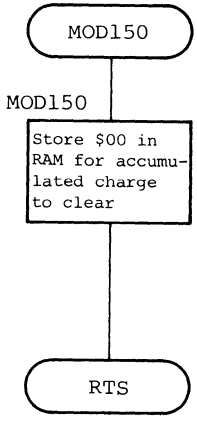
(1) Function

On CLEAR ACCUMULATED CHARGE key input, clear the current accumulated charge.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	————	————	————
Returns	Accumulated charge	TOTAL (RAM)	4

(3) Flowchart



End	MODULE NO.	2.14	LABEL	MOD160

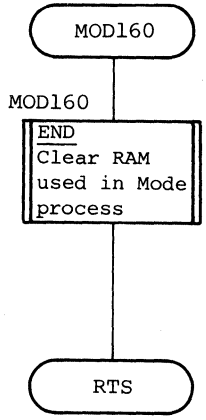
(1) Function

On [END] key input, clear mode flag and the internal RAM used in Mode Process.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	———	———	———
Returns	RAM used in Mode Process	The internal RAM	———

(3) Flowchart



Check Input Data	MODULE NO.	3.0	LABEL	CHECK

(1) Function

"Check Input Data" determines whether key data is character or cursor movement at the beginning. In the case of characters, a check process is executed corresponding to the previously defined mode flag. In the case of cursor movement, a move process is executed corresponding to the key data (Refer to "Note").

(2) Arguments

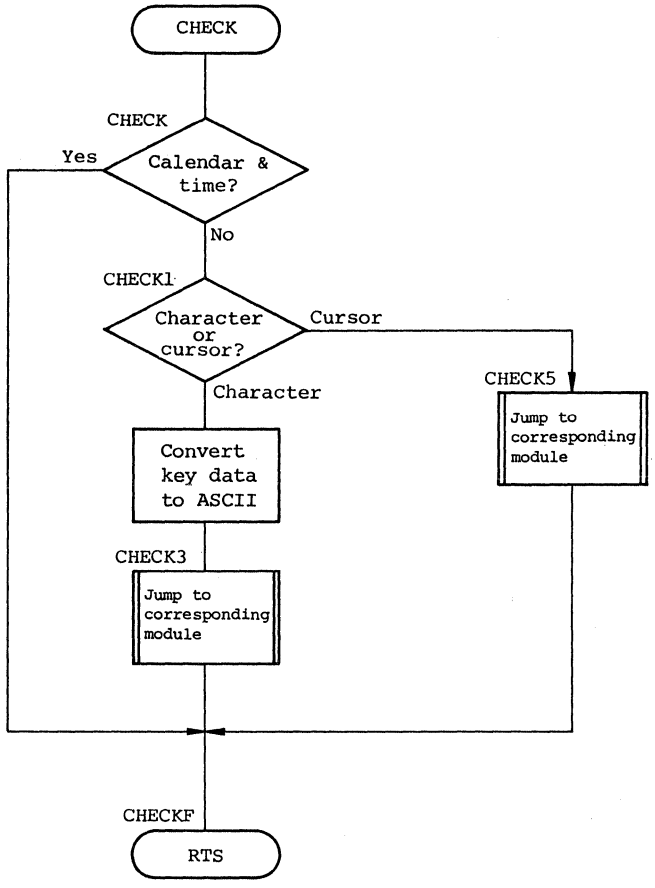
Common entry and return arguments for all the check input data processes are shown below. Particular entry and return arguments for each checking data are explained in each checking data's description in detail.

	Contents	Storage Location	No. of Bytes
Entry	Key data	KEYDAT (RAM)	1
	Mode flag	MODFLG (RAM)	1
	Calendar & time display request flag	(1, FLAG)	(1 bit)
Returns	ASCII code	KEYSET (RAM)	1

Main Program	Check Input Data

Check Input Data	MODULE NO.	3.0	LABEL	CHECK

(3) Flowchart



Note: Key data input to "Check Input Data"

Key data is checked as to whether it represents characters or cursor movement. After checking, the corresponding process is executed.

(i) Character

Key data (KEYDAT) is converted to ASCII character codes, after which a particular module is called depending on the mode flag (MODFLG). Mode flag and corresponding modules are shown in Table 3.3.

Table 3.3 Mode Flag and Corresponding Modules of "Check Input Data"

Mode flag (MODFLG)		Module Label	Module Function
$0,$12		CHEK1	For normal dialing; valid data (0∿9, ∗, #)
$8		CHEK2	For speed dialing; valid data (0∿9)
$9	Call	CHEK3	For entering speed dial; valid data (① : 0∿9, ② : 0∿9, ∗, #)
$4,$5,$6,$7		CHEK4	For storing telephone number; valid data (① : all ② : 0∿9, ∗, #)
$D,$E		CHEK5	For setting calendar & time, charge; valid data (0∿9)

(ii) Cursor

A particular module is called depending on key data stored in KEYDAT. Key data and corresponding modules is shown in Table 3.4.

Table 3.4 Key Data and Corresponding Modules

Key data (KEYDAT)		Module Label	Module Function
$29		CURLR	Move cursor to left or right
$2A	Call		
$2B		MVECUR	Move cursor to 21st digit

Check for Normal Dialing	MODULE NO.	3.1	LABEL	CHEK1

(1) Function

For normal dialing, valid data is (0 ∿ 9, ∗, #).

(a) Valid data is stored in DIALNO (RAM) up until the 20th digit. From the 21st digit, it is stored in DIADAT (RAM).

(b) Valid data is displayed on LCD up until the 38th digit.

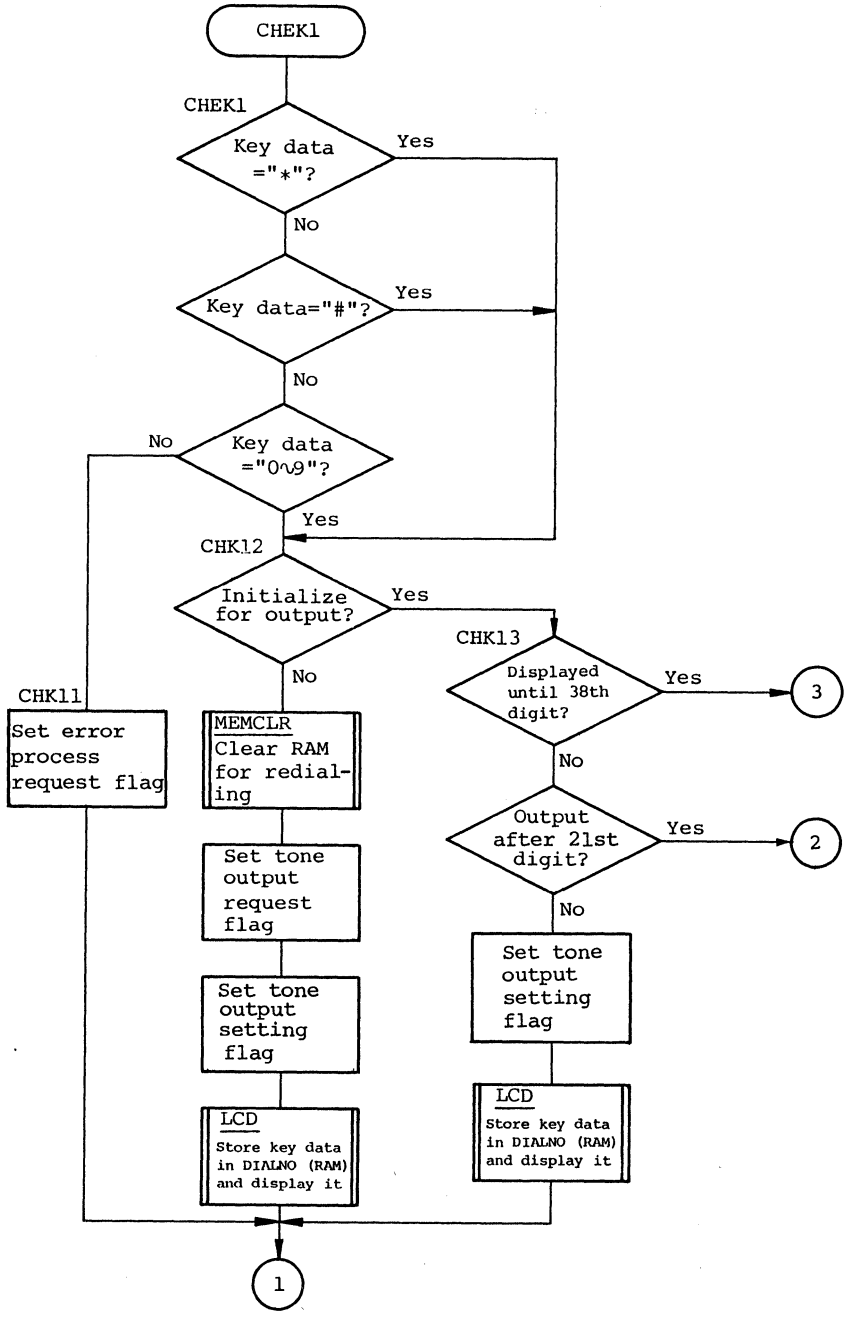
(c) If invalid data is input, set error process request flag.

(2) Arguments

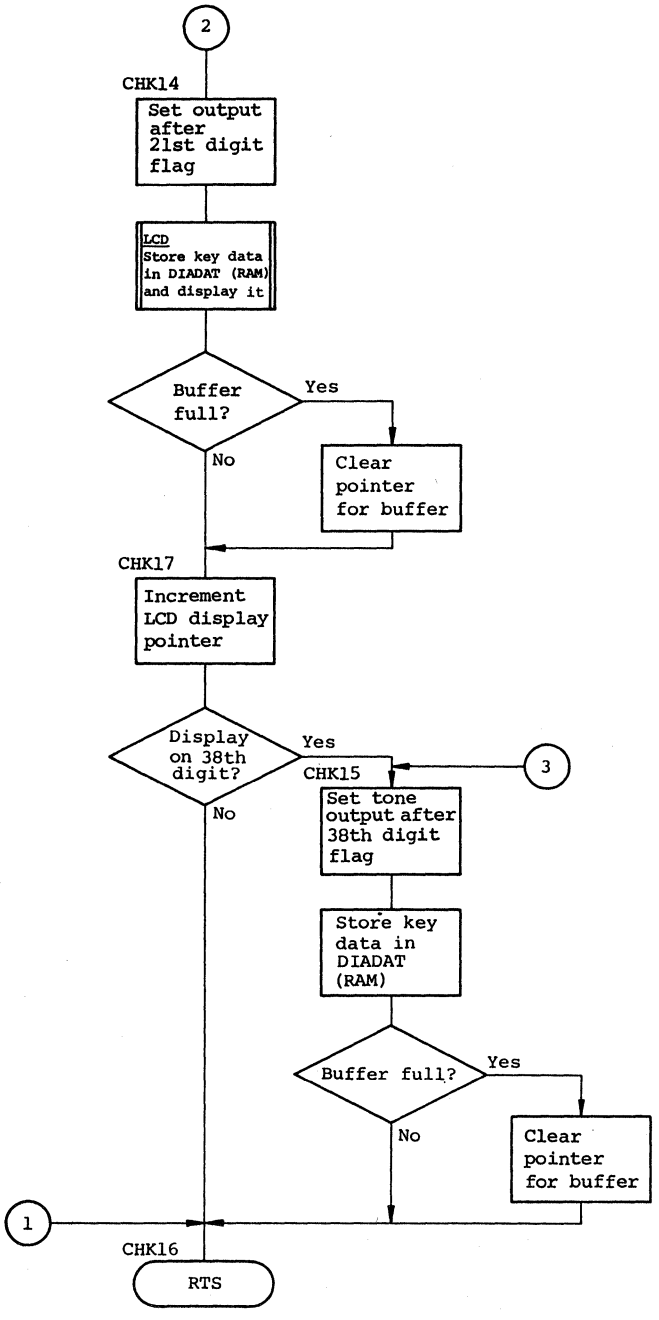
	Contents	Storage Location	No. of Bytes
Entry	Key data	KEYSET (RAM)	1
	Tone output setting flag	(5, WAIT3)	(1 bit)
	Display on 38th digit flag	(4, WAIT5)	(1 bit)
	Tone after 21st digit flag	(3, WAIT5)	(1 bit)
	Tone on 21st digit	LCDMP (RAM)	1
	Pointer for buffer	DIAP1 (RAM)	1
Returns	Valid data (until 20th digit)	DIALNO (RAM)	21
	Valid data (after 21st digit)	DIADAT (RAM)	5
	RAM for redialing	MEMO (RAM)	21
	Tone output request flag	(1, RZR)	(1 bit)
	Tone outputting flag	(4, WAIT3)	(1 bit)
	Tone output setting flag	(5, WAIT3)	(1 bit)
	LCD display RAM	LCDM (RAM)	40
	Pointer for buffer	DIAP1 (RAM)	1
	LCD display pointer	LCDMP (RAM)	1
	Cursor OFF	CURADP (RAM)	1
	LCD display after 38th digit flag	(4, WAIT5)	(1 bit)
	Error process request flag	(0, ERFLG)	(1 bit)

Check for Normal Dialing	MODULE NO.	3.1	LABEL	CHEK1

(3) Flowchart



Check for Normal Dialing | MODULE NO. 3.1 | LABEL CHEK1



Main Program	Check Input Data

Check for Speed Dialing	MODULE NO.	3.2	LABEL	CHEK2

(1) Function

For speed dialing, valid data is (0∿9).

(a) Valid data is stored in LCDM (RAM) and displayed on the 10th digit.

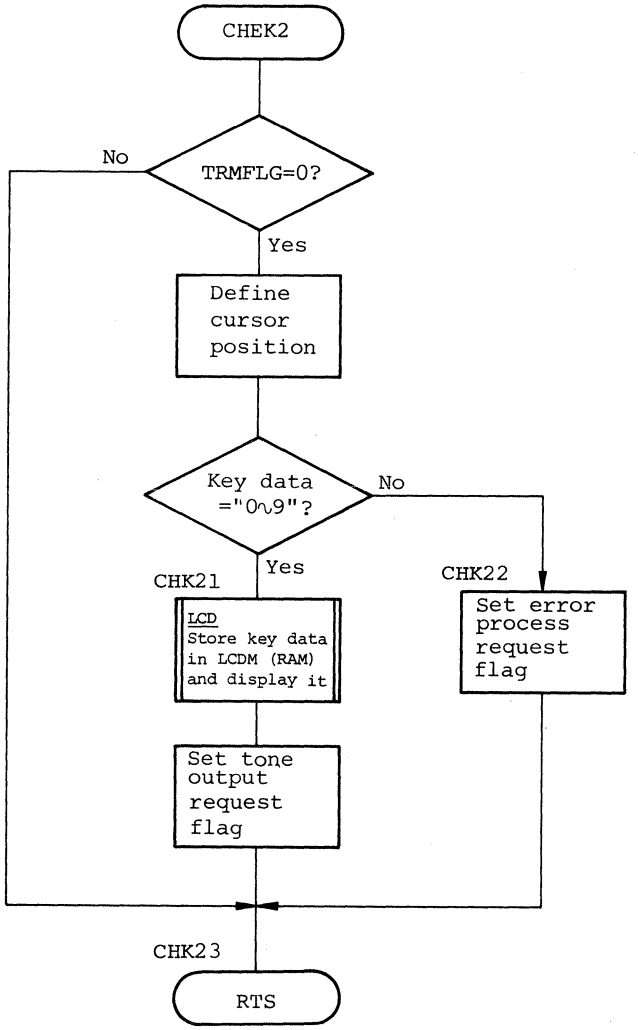
(b) If invalid data is input, set error process request flag.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Key data	KEYSET (RAM)	1
	Process flag	TRMFLG (RAM)	1
Returns	Cursor position	CURADP (RAM)	1
	LCD display pointer	LCDMP (RAM)	1
	Valid data	LCDM (RAM)	40
	Tone output request flag	(1, BZR)	(1 bit)
	Error process request flag	(0, ERFLG)	(1 bit)

Check for Speed Dialing	MODULE NO.	3.2	LABEL	CHEK2

(3) Flowchart



Check for Entering Speed Dial Number	MODULE NO.	3.3	LABEL	CHEK3

(1) Function

For entering speed dial number, valid data is (in the first process : 0∿9; in the second process : 0∿9, *, #).

(a) Valid data is stored in LCDM (RAM) and displayed. (In the first process, valid data is displayed on the 10th digit. In the second process, valid data is displayed on 21∿38th digit.)

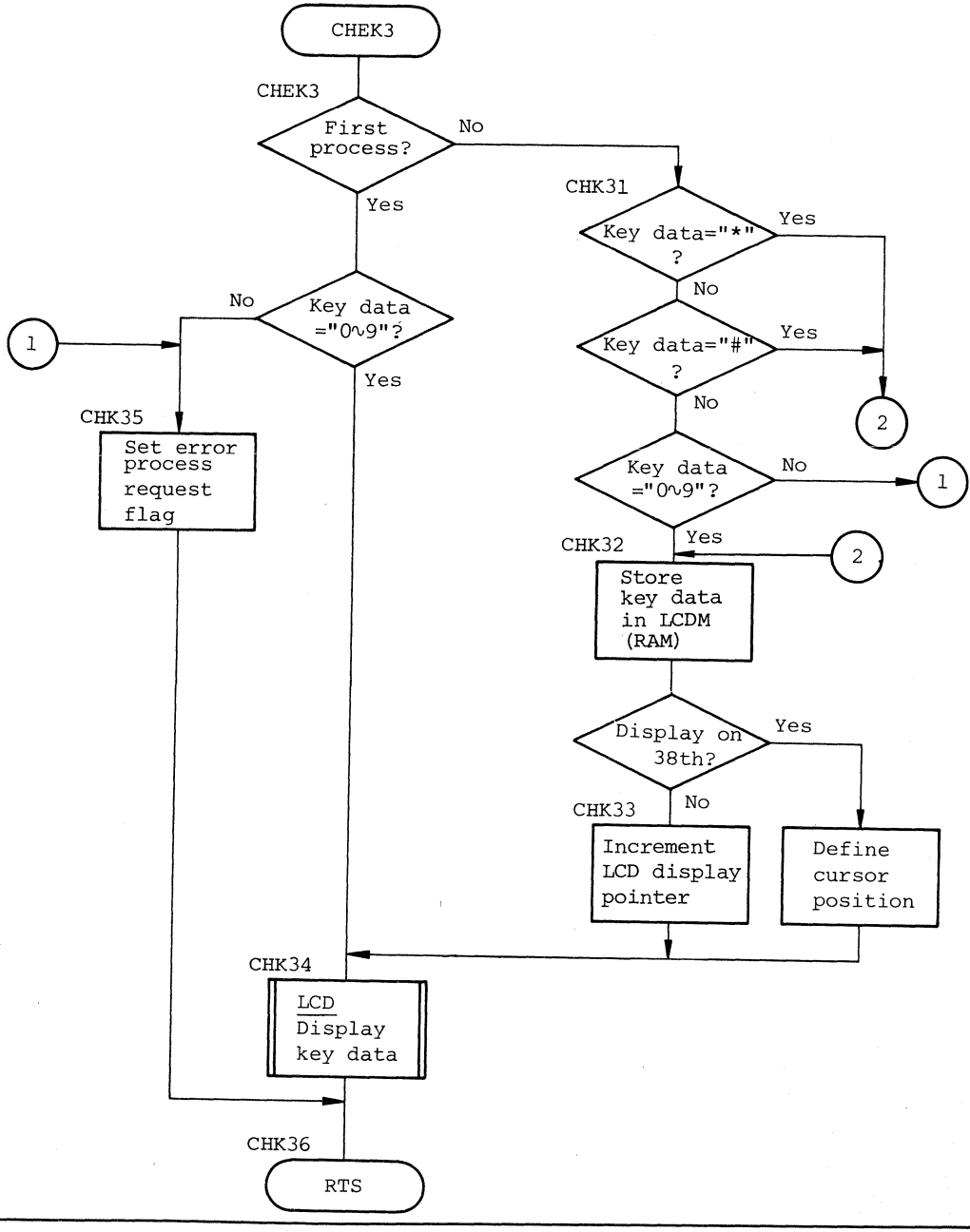
(b) If invalid data is input, set error process request flag.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Key data	KEYSET (RAM)	1
	Process flag	TRMFLG (RAM)	1
	LCD display pointer	LCDMP (RAM)	1
	Cursor position	CURADP (RAM)	1
Returns	Valid data	LCDM (RAM)	40
	Cursor position	CURADP (RAM)	1
	LCD display pointer	LCDMP (RAM)	1
	Error process request flag	(0, ERFLG)	(1 bit)

Check for Entering Speed Dial Number	MODULE NO.	3.3	LABEL	CHEK3

(3) Flowchart



Main Program	Check Input Data

Check for Storing Telephone Numbers	MODULE NO.	3.4	LABEL	CHEK4

(1) Function

For storing telephone numbers, valid data is (1∿20th digit : all key; 21∿38th digit : 0∿9, ∗, #).

(a) Valid data is stored in LCDM (RAM) and displayed.

(b) If invalid data is input, set error process request flag.

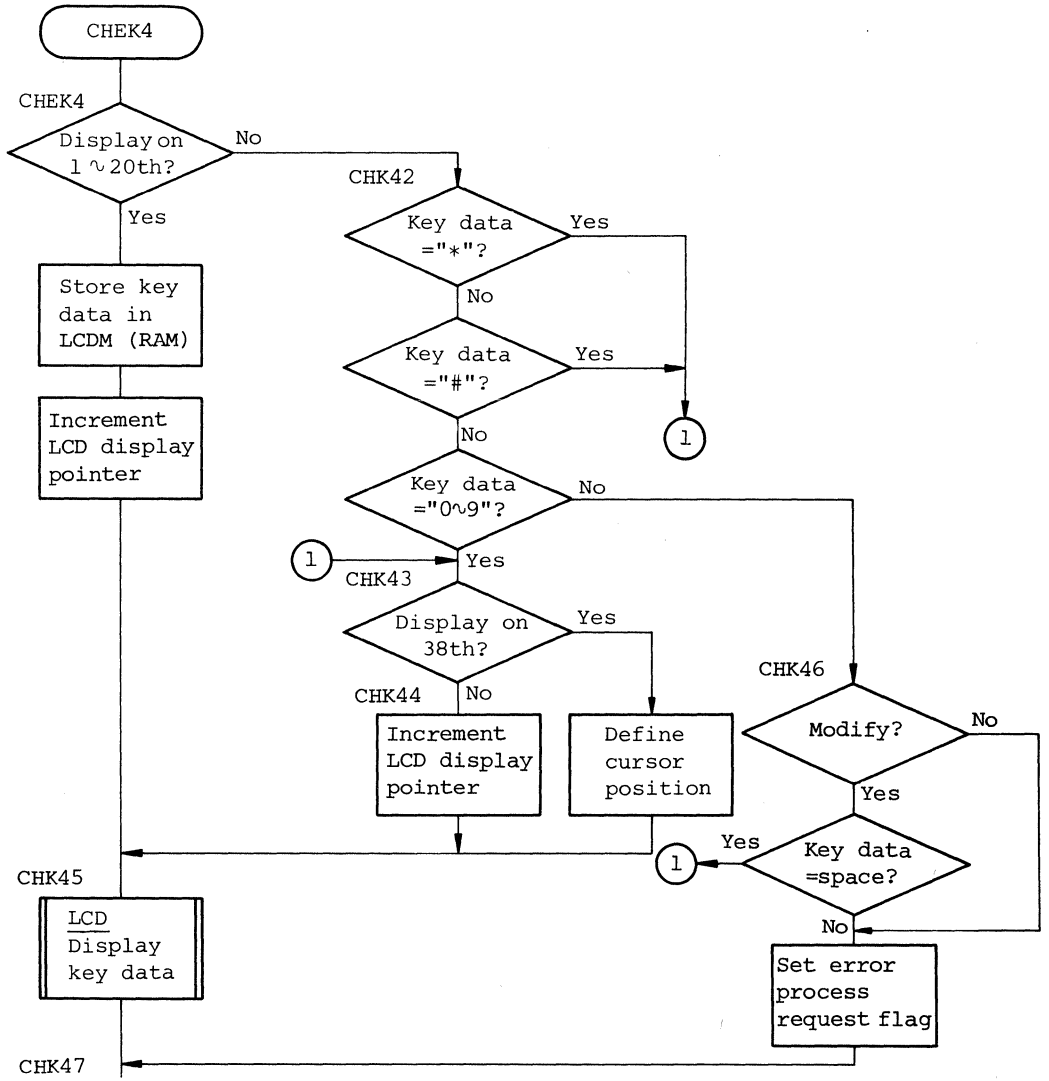
(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Key data	KEYSET (RAM)	1
	LCD display pointer	LCDMP (RAM)	1
	Mode flag	MODFLG (RAM)	1
Returns	Valid data	LCDM (RAM)	40
	LCD display pointer	LCDMP (RAM)	1
	Cursor position	CURADP (RAM)	1
	Error process request flag	(0, ERFLG)	(1 bit)

Main Program	Check Input Data

Check for Storing Telephone Numbers	MODULE NO.	3.4	LABEL	CHEK4

(3) Flowchart



Main Program	Check Input Data

Check for Setting Calendar & Time, Charge	MODULE NO.	3.5	LABEL	CHEK5

(1) Function

For setting calendar & time or charge, valid data is (0∿9).

(a) Valid data is stored in LCDM (RAM) and displayed.

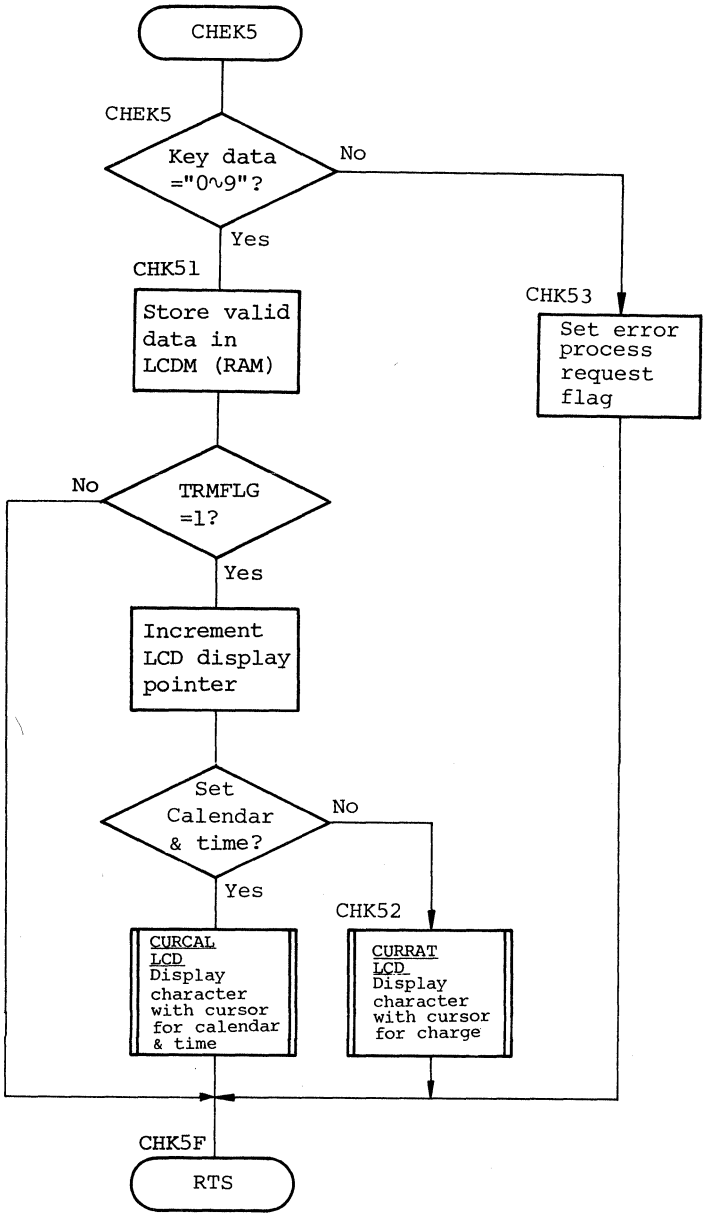
(b) If invalid data is input, set error process request flag.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Key data	KEYSET (RAM)	1
	Process flag	TRMFLG (RAM)	1
	Mode flag	MODFLG (RAM)	1
Returns	LCD display pointer	LCDMP (RAM)	1
	Cursor position	CURADP (RAM)	1
	Valid data	LCDM (RAM)	40
	Error process request flag	(0, ERFLG)	(1 bit)

Check for Setting Calendar & Time, Charge	MODULE NO.	3.5	LABEL	CHEK5

(3) Flowchart



Move Cursor to Left or Right	MODULE NO.	3.6	LABEL	CURLR

(1) Function

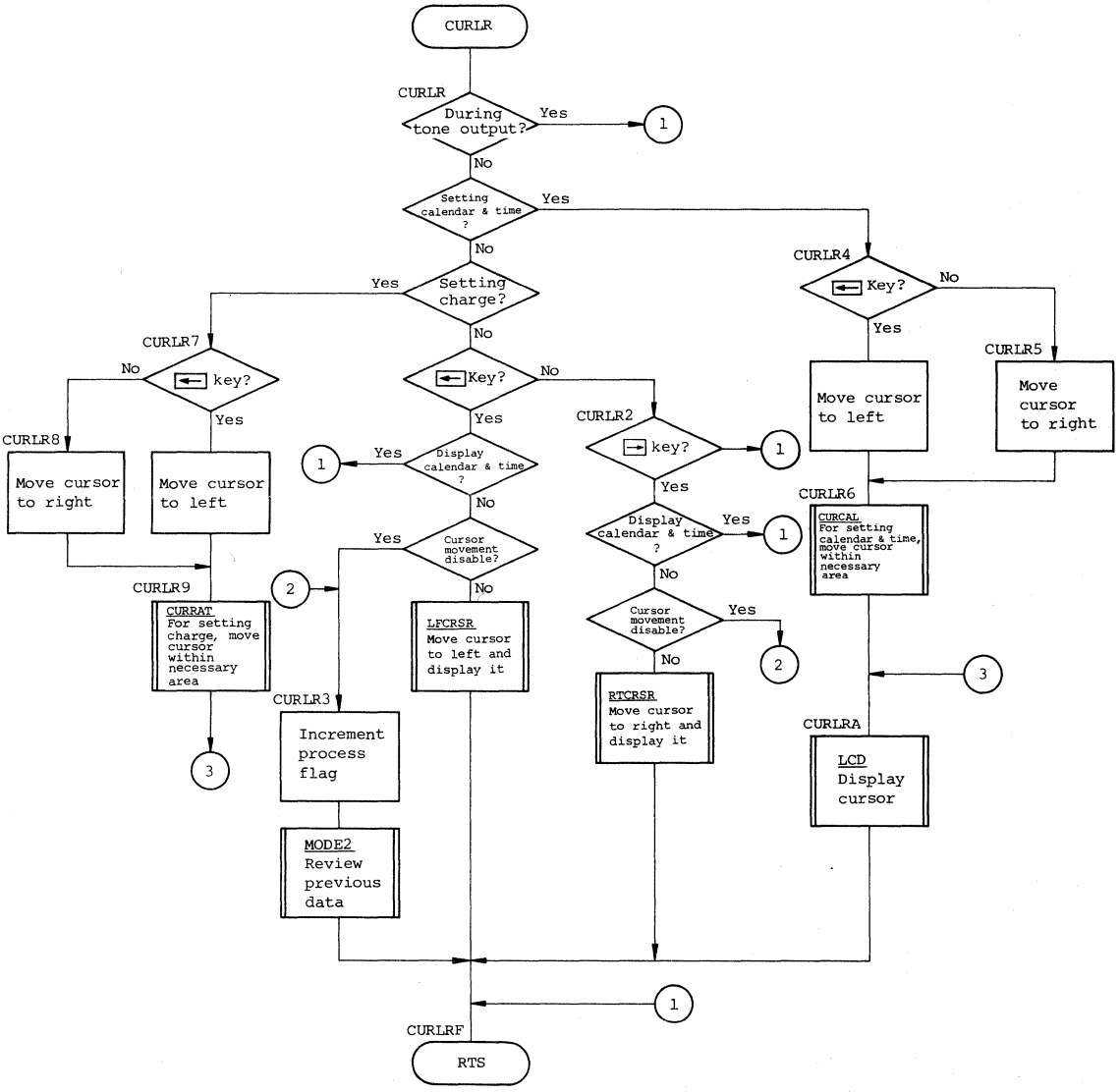
Moves cursor to left or right. If cursor movement disable flag is set, review the next or previous data. While setting calendar & time or charge, cursor cannot be moved to unnecessary area.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Key data	KEYDAT (RAM)	1
	Calendar & time display request flag	(1, FLAG)	(1 bit)
	Cursor movement disable flag	(7, BZRFLG)	(1 bit)
	Mode flag	MODFLG (RAM)	1
	Process flag	TRMFLG (RAM)	1
	Tone outputting flag	(4, WAIT3)	(1 bit)
	LCD display pointer	LCDMP (RAM)	1
	Cursor position	CURADP (RAM)	1
Returns	Reviewing data	LCDM (RAM)	40
	LCD display pointer	LCDMP (RAM)	1
	Cursor position	CURADP (RAM)	1

Move Cursor to Left or Right	MODULE NO.	3.6	LABEL	CURLR

(3) Flowchart



Move cursor to 21st Digit	MODULE NO.	3.7	LABEL	MVECUR

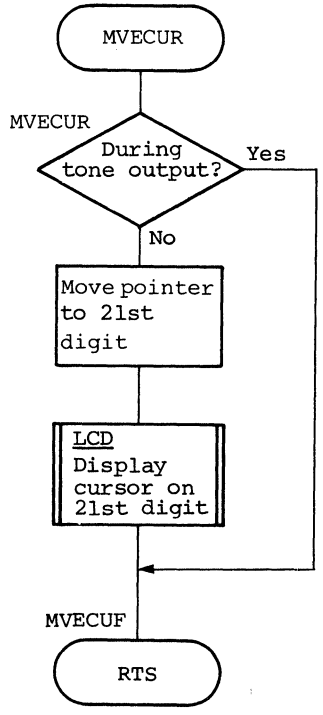
(1) Function

Moves cursor to 21st digit.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Tone outputting flag	(4, WAIT3)	(1 bit)
Returns	LCD display pointer	LCDMP (RAM)	1
	Cursor position	CURADP (RAM)	1

(3) Flowchart



Error Process	MODULE NO.	4.0	LABEL	ERROR

(1) Function

"Error Process" performs error processing corresponding to error process request flag. The actual generating of the buzzer, however, is executed by timer 2. Counting for "ERROR" or "FULL" display time is executed by timer 1. Details of the error process request flag are shown in Table 3.5.

Table 3.5 Error Process Request Flag

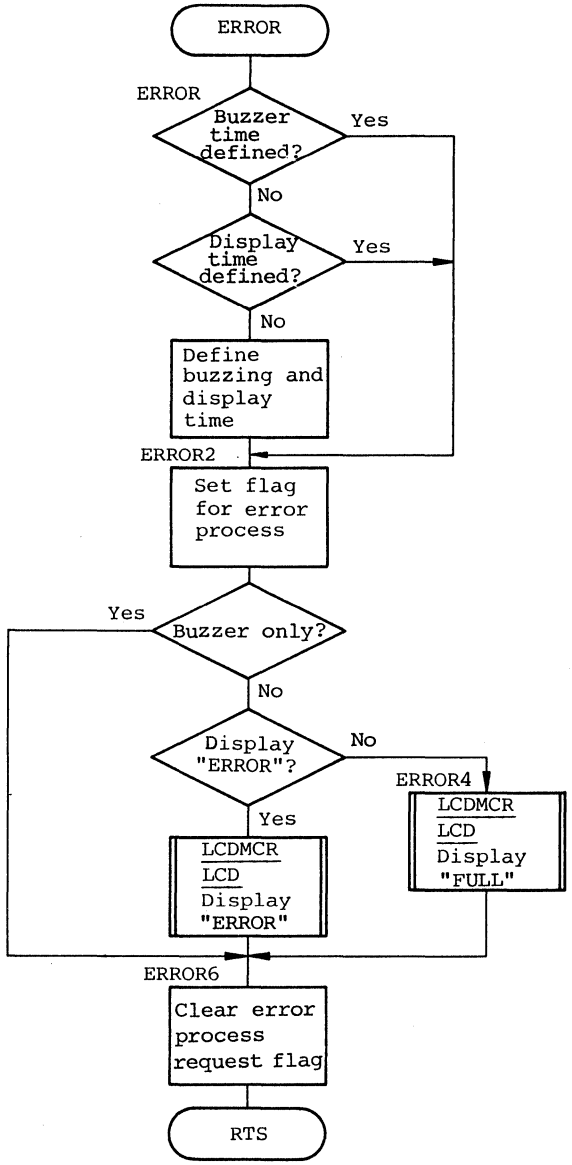
ERFLG (RAM) Bit			
2	1	0	Function
0	0	0	No error
0	0	1	Generate buzzing
0	1	1	Generate buzzing and display "ERROR"
1	0	1	Generate buzzing and display "FULL"

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Error process request flag	ERFLG (RAM)	1
	Buzzer output time	BZRCNT (RAM)	2
	Error message display time	ERR1 (RAM), ERR2 (RAM)	2
Returns	Error process complete flag	ERFLG (RAM)	1

Error Process	MODULE NO.	4.0	LABEL	ERROR

(3) Flowchart



Control Timer 1	MODULE NO.	T10	LABEL	TIMER1

(1) Function

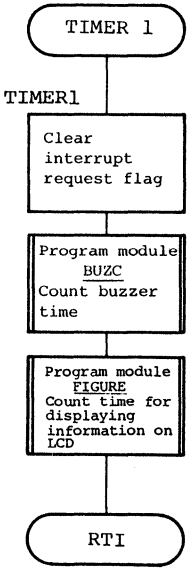
"Control Timer 1" controls counting buzzing time and displaying information on LCD.

(2) Arguments

There are no common entry and return arguments for controlling timer 1. Particular entry and return arguments for each module are explained in each module's description in detail.

	Contents	Storage Location	No. of Bytes
Entry	———	———	———
Returns	———	———	———

(3) Flowchart



Count Buzzing Time	MODULE NO.	T11.0	LABEL	BUZC

(1) Function

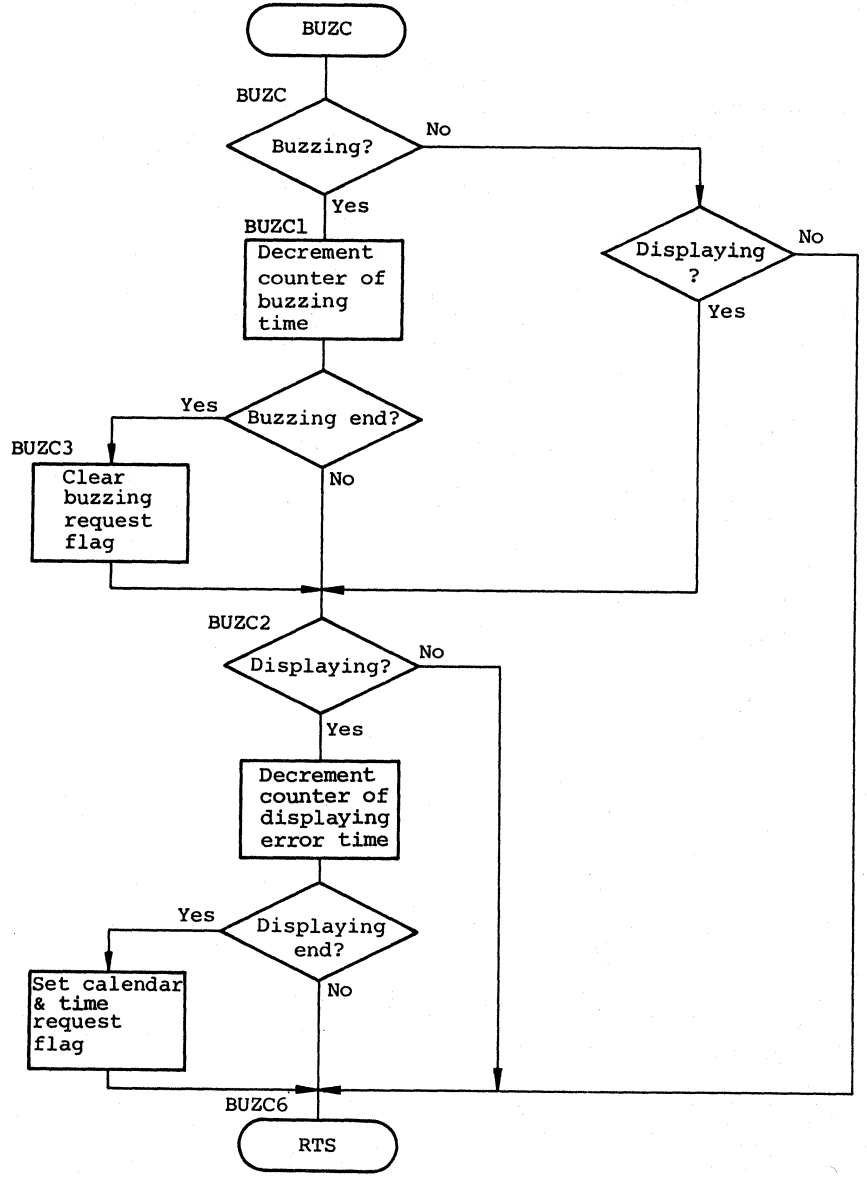
Counts displaying error message time and buzzing time.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Buzzing output request flag	(3, BZRFLG)	(1 bit)
	Counter of buzzing time	BZRCNT (RAM)	2
	Counter of displaying error time	ERR1 (RAM), ERR2 (RAM)	2
	Error display request flag	(7, WAIT5)	(1 bit)
Returns	Buzzing output complete flag	(3, BZRFLG)	(1 bit)
	Counter of buzzing time	BZRCNT (RAM)	2
	Counter of displaying error time	ERR1 (RAM), ERR2 (RAM)	2
	Calendar & time display request flag	(1, FLAG)	(1 bit)
	Error display complete flag	(7, WAIT5)	(1 bit)

Count Buzzing Time	MODULE NO.	T11.0	LABEL	BUZC

(3) Flowchart



Count Time for Displaying Information on LCD	MODULE NO.	T12.0	LABEL	FIGURE

(1) Function

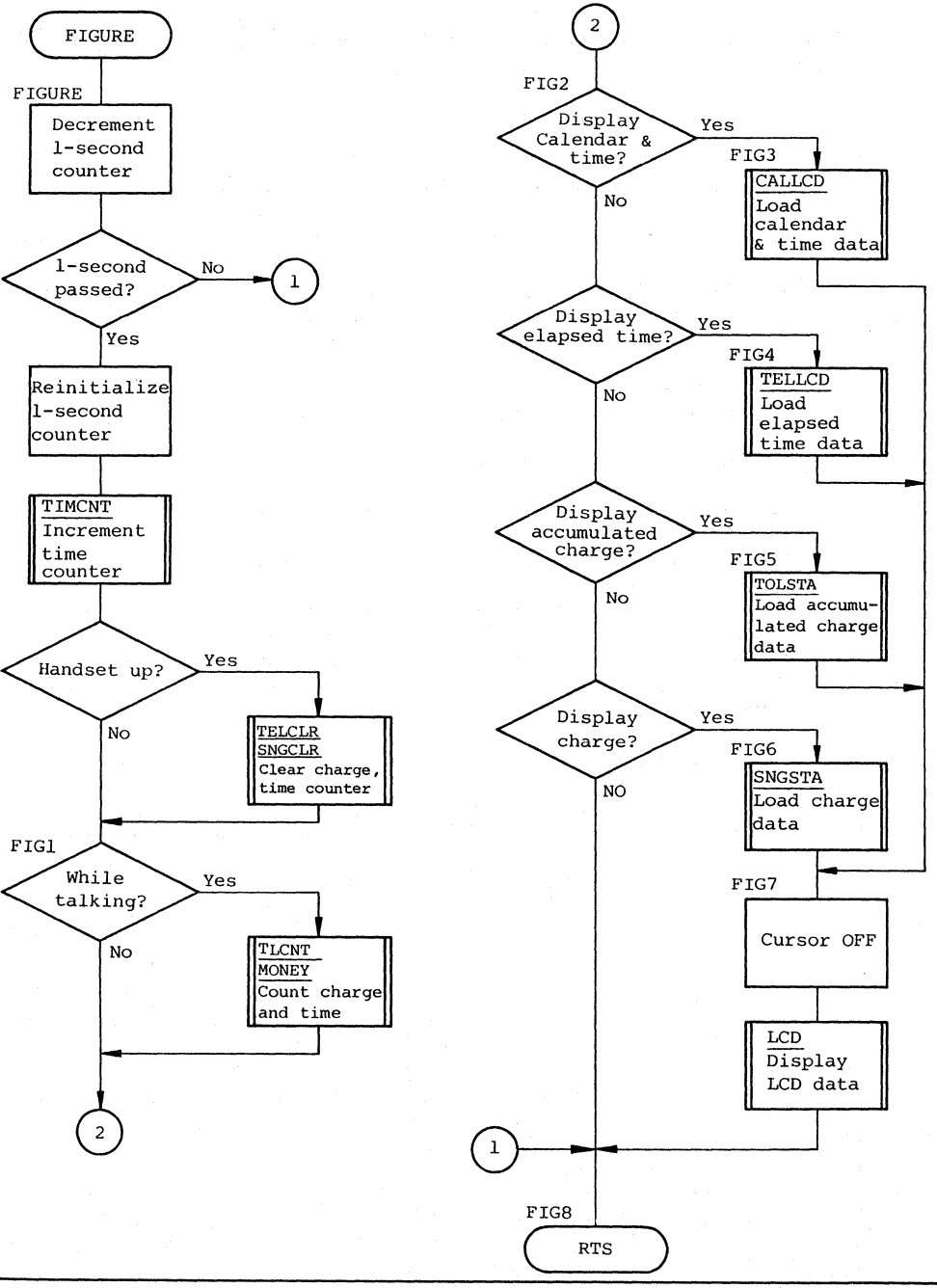
Counts calendar & time, elapsed time, charge, and accumulated charge.
Displaying information on LCD is selected by flags.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	1-second counter	TCNTR (RAM)	1
	Handset up flag	(0, TELFLG)	(1 bit)
	While talking flag	(1, TELFLG)	(1 bit)
	Calendar & time display request flag	(1, FLAG)	(1 bit)
	Charge display request flag	(7, FLAG)	(1 bit)
	Accumulated charge display request flag	(6, FLAG)	(1 bit)
	Elapsed time display request flag	(3, FLAG)	(1 bit)
	Calendar & time counter	CALNDR (RAM)	5
	Elapsed time counter	TEL (RAM)	3
	Charge counter	SINGL (RAM)	4
	Accumulated charge counter	TOTAL (RAM)	4
Returns	1-second counter	TCNTR (RAM)	1
	Calendar & time counter	CALNDR (RAM)	5
	Elapsed time counter	TEL (RAM)	3
	Charge per time counter	SECCNT (RAM)	2
	Charge counter	SINGL (RAM)	4
	Accumulated charge counter	TOTAL (RAM)	4
	LCD display RAM	LCDM (RAM)	40
	Cursor OFF	CURADP (RAM)	1

Count Time for Displaying Information on LCD	MODULE NO.	T12.0	LABEL	FIGURE

(3) Flowchart



Control Timer 2	MODULE NO.	T20	LABEL	TIMER2

(1) Function

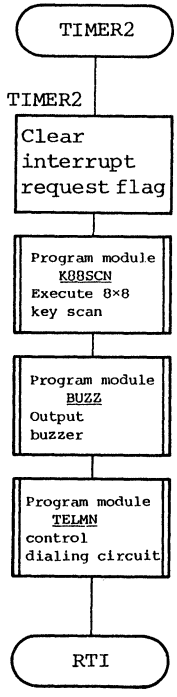
"Control Timer 2" controls key scan, buzzer output, and controlling dialing circuit.

(2) Arguments

There are no common entry and return arguments for controlling timer 2. Particular entry and return arguments for each module are explained in each module's description in detail.

	Contents	Storage Location	No. of Bytes
Entry	———	———	———
Returns	———	———	———

(3) Flowchart



Key Scan	MODULE NO.	T21.0	LABEL	K88SCN

(1) Function

Executes key scan of 8×8 key matrix. After processed chatter prevention routine, valid data is stored in KEYDAT (RAM) and flag is set indicating key data is contained.

(2) Arguments

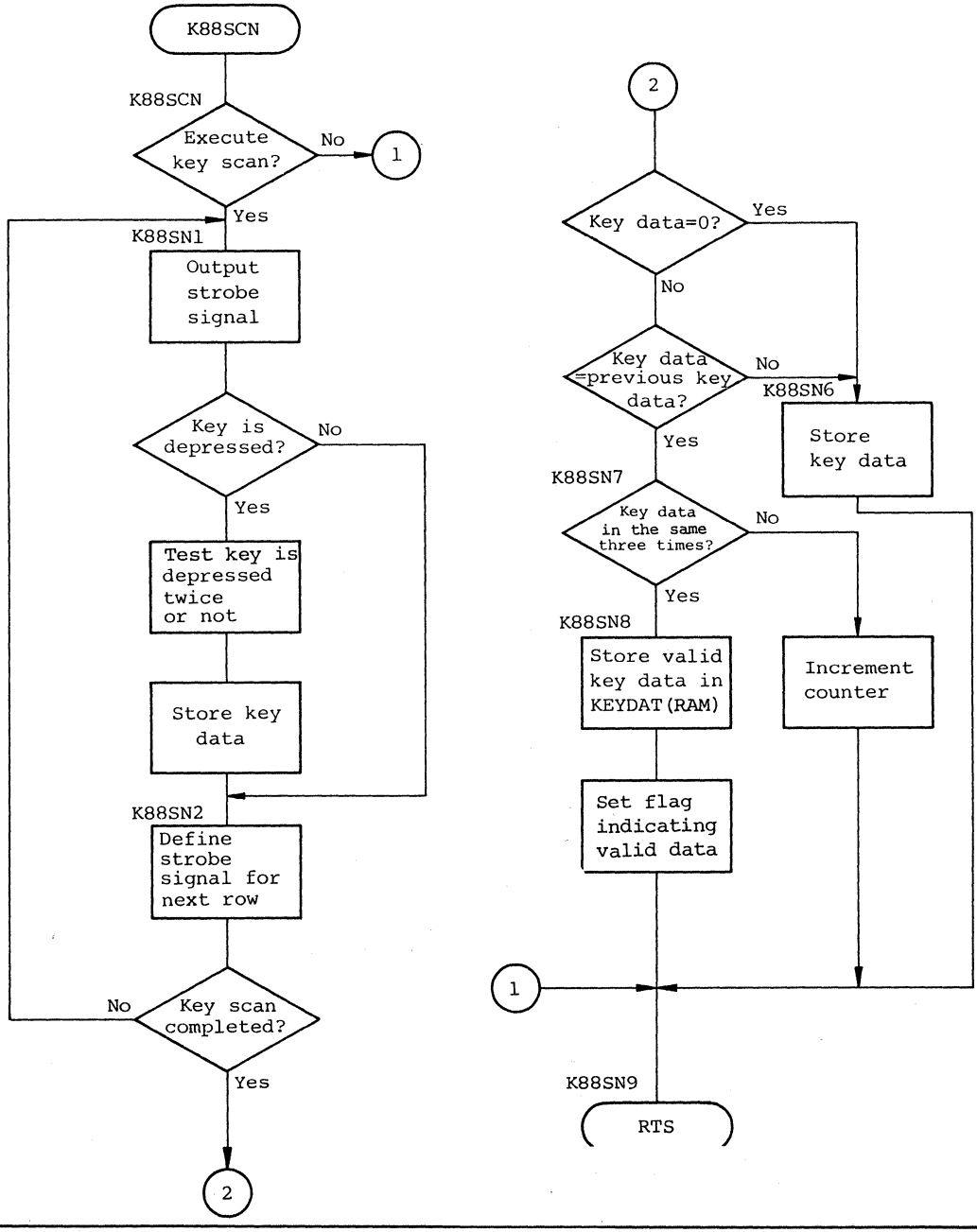
	Contents		Storage Location	No. of Bytes
Entry	———		———	———
Returns	Key data	(Note)	KEYDAT (RAM)	1
	Key data/no key data		(0, FLAG)	(1 bit)

Note:

(0, FLAG) = 0 : No key data is contained

(0, FLAG) = 1 : Key data is contained

(3) Flowchart


K88SCN
K88SCN
Execute key scan?
No
1
Yes
K88SN1
Output strobe signal
Key is depressed?
No
Yes
Test key is depressed twice or not
Store key data
K88SN2
Define strobe signal for next row
No
Key scan completed?
Yes
2
2
Key data=0?
Yes
No
Key data =previous key data?
No
K88SN6
Store key data
Yes
K88SN7
Key data in the same three times?
No
Yes
K88SN8
Store valid key data in KEYDAT(RAM)
Increment counter
Set flag indicating valid data
1
K88SN9
RTS

Buzzing	MODULE NO.	T22.0	LABEL	BUZZ

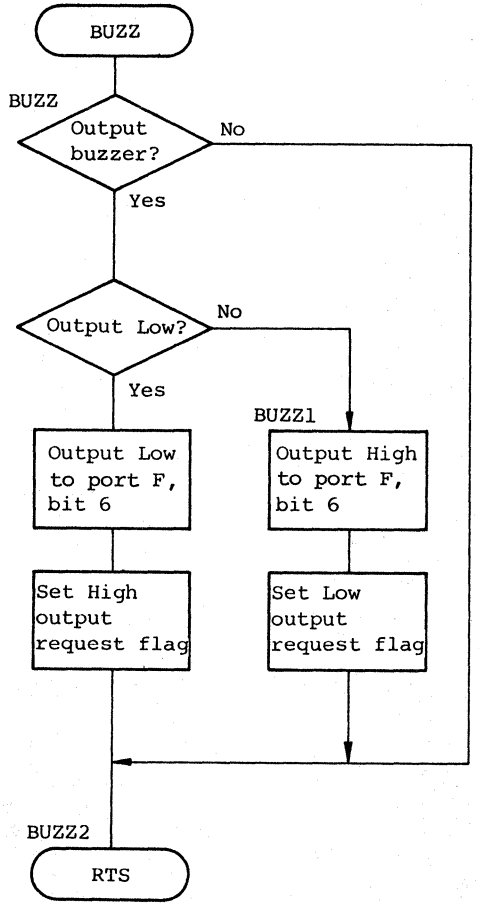
(1) Function

Output High and Low to port F, bit 6 by the same cycle of timer 2, if buzzer output request flag is set.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Buzzer output request flag	(3, BZRFLG)	(1 bit)
	High·Low output request flag	(0, BZR)	(1 bit)
Returns	High·Low output request flag	(0, BZR)	(1 bit)

(3) Flowchart



Control Dialing Circuit	MODULE NO.	T23.0	LABEL	TELMN

(1) Function

"Control Dialing Circuit" controls six routine (While Talking, while Receiving, Generate Tone, Relay ON, while Retrying, and Connect Lines) as shown in (3) Flowchart. Each routine has its flag to indicate dialing circuit status. Flag is checked at the beginning of this routine and each module is executed depending on the flag.

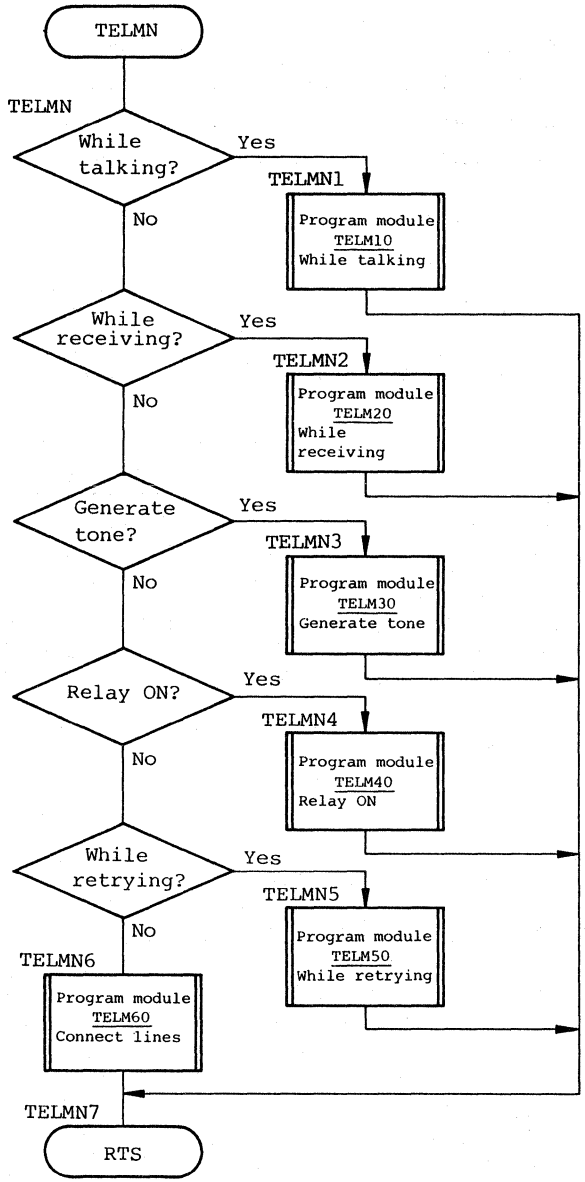
(2) Arguments

Common entry and return arguments for controlling dialing circuit are shown below. Particular entry and return arguments for each module are explained in each modules description in detail.

	Contents	Storage Location	No. of Bytes
Entry	Talking flag	(1, TELFLG)	(1 bit)
	Receiving flag	(1, PDDTR)	(1 bit)
	Tone output flag	(2, BZR)	(1 bit)
	Tone output request flag	(1, BZR)	(1 bit)
	Retrying flag	(0, BZRFLG)	(1 bit)
	Handset up?	(2, PDDTR) (3, PDDTR)	(2 bits)
Returns	———	———	———

Control Dialing Circuit	MODULE NO.	T23.0	LABEL	TELMN

(3) Flowchart



While Talking	MODULE NO.	T23.1	LABEL	TELM10

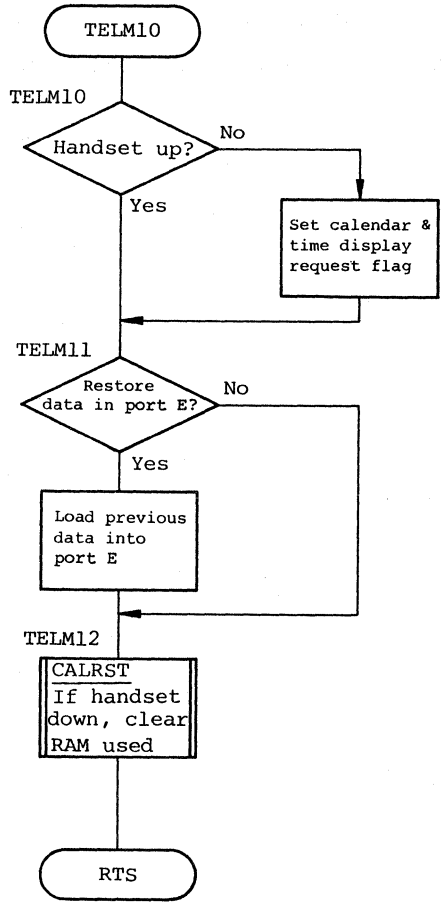
(1) Function

Tests if conversation is continuing. If handset is down, clear talking flag and set calendar & time display request flag.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Handset up?	(2,PDDTR)(3,PDDTR)	(2 bits)
	Port E status	(0, WAIT3)	(1 bit)
	Contents of port E before tone output	MEMOP (RAM)	1
Returns	Calendar & time display request flag	(1, FLAG)	(1 bit)
	Port E	PEDTR	1
	Port E status	(0, WAIT3)	(1 bit)

(3) Flowchart


TELM10
TELM10
Handset up?
No
Yes
Set calendar & time display request flag
TELM11
Restore data in port E?
No
Yes
Load previous data into port E
TELM12
CALRST
If handset down, clear RAM used
RTS

While Receiving	MODULE NO.	T23.2	LABEL	TELM20

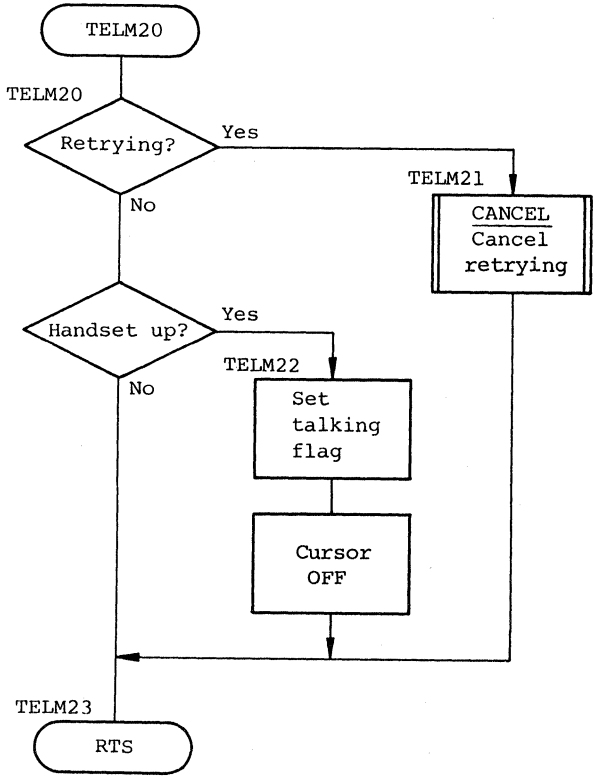
(1) Function

Tests whether or not handset is up. If handset is up, set talking flag. If called, whether handset is up or not, retrying is canceled.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Retrying flag	(0, BZRFLG)	(1 bit)
	Handset up?	(2,PDDTR)(3,PDDTR)	(2 bits)
Returns	Talking flag	(1, TELFLG)	(1 bit)
	Retrying flag	(0, BZRFLG)	(1 bit)
	Cursor OFF	CURADP (RAM)	1

(3) Flowchart



Generate Tone	MODULE NO.	T23.3	LABEL	TELM30

(1) Function

Generates tone for normal dialing, speed dialing, redialing, retrying. Dialing number for output is stored in DIALNO (RAM) until 20th digit. From 21st digit, number is stored in DIADAT (RAM).

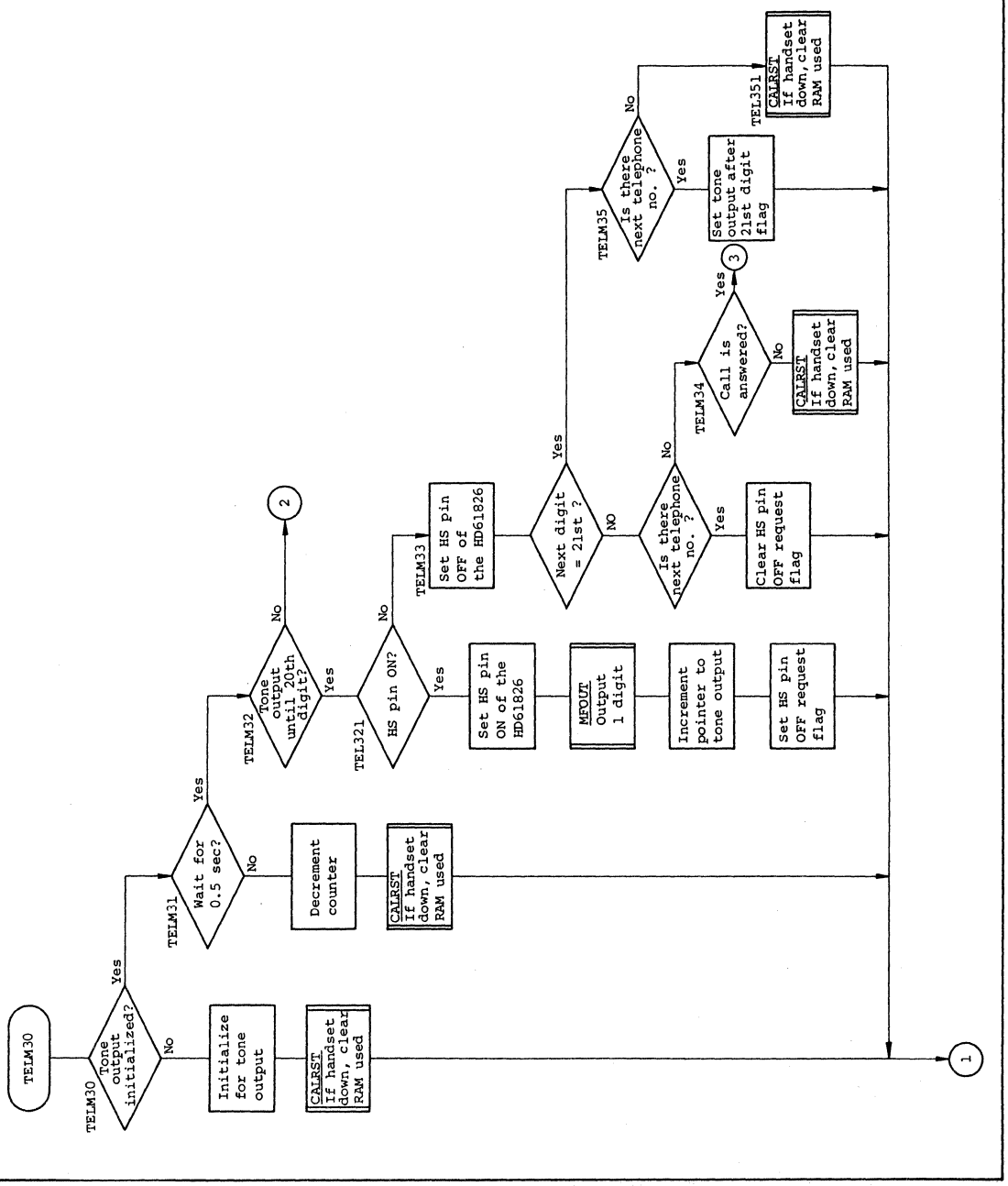
When tone output is finished, test when conversation begins. If conversation begins, set talking flag.

(2) Arguments

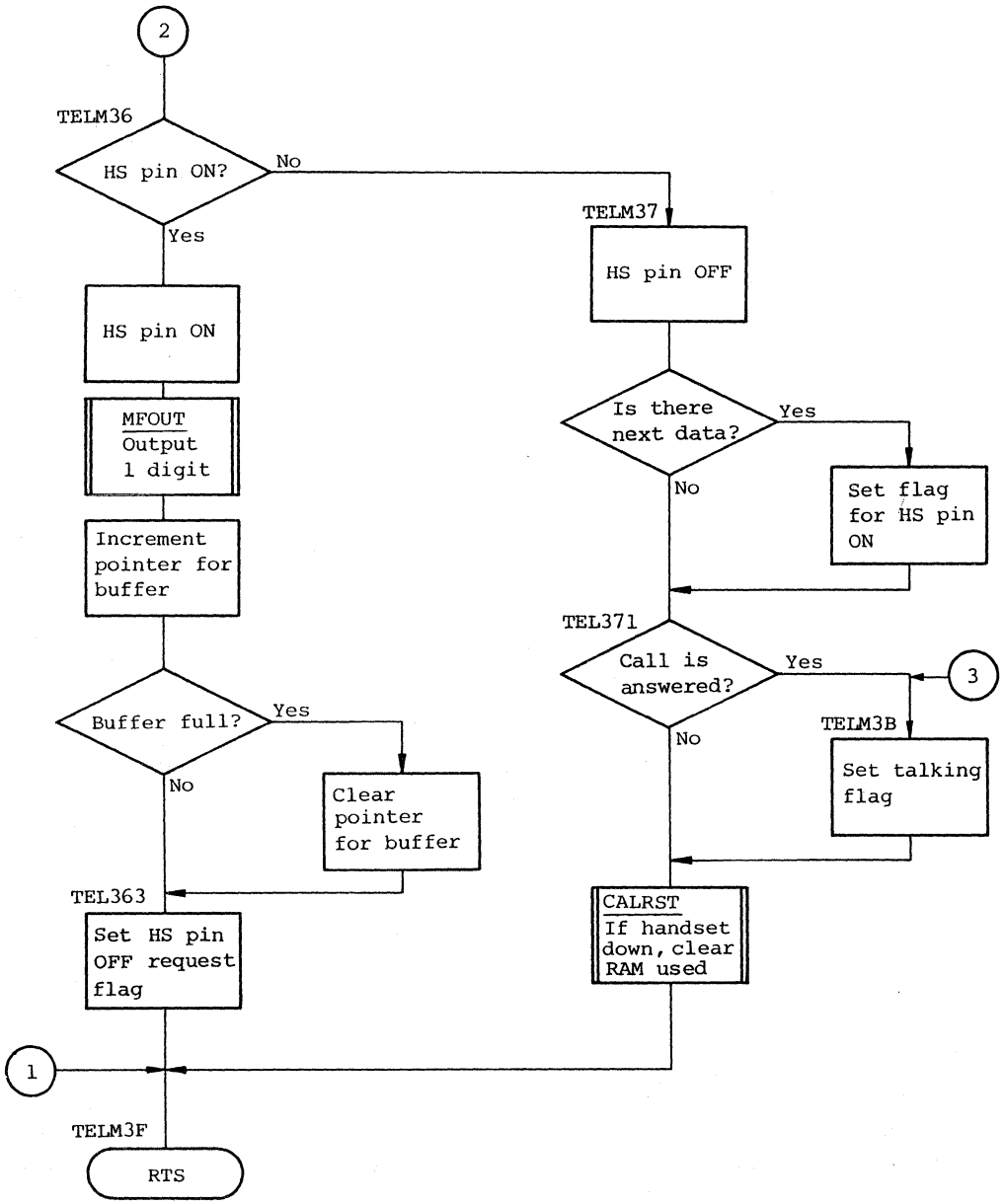
	Contents	Storage Location	No. of Bytes
Entry	Process flag	TELWAT (RAM)	1
	Tone output until 20th digit flag	(5, WAIT5)	(1 bit)
	Tone output after 21st digit flag	(5, WAIT5)	(1 bit)
	Telephone number (until 20th digit)	DIALNO (RAM)	20
	Telephone number (from 21st digit)	DIADAT (RAM)	5
	Pointer of telephone number (until 20th digit)	TELNO (RAM)	1
	Pointer of telephone number (from 21st digit)	DIAP2 (RAM)	1
	Retrying flag	(0, BZRFLG)	(1 bit)
	Handset up?	(6,TELFLG) (7,TELFLG)	(2 bits)
	Next data existance flag	(3, WAIT3)	(1 bit)
	Contents of port E before tone output	PEDTR	1
	Call is answered?	(2,PDDTR) (3,PDDTR)	(2 bits)
Returns	Process flag	TELWAT (RAM)	1
	Tone output data	MFDAT (RAM)	1
	Talking flag	(1, TELFLG)	(1 bit)
	Handset up flag	(0, TELFLG)	(1 bit)
	Contents of port E	MEMOP (RAM)	1
	Contents of port E reserved flag	(0, WAIT3)	(1 bit)

Generate Tone	MODULE NO.	T23.3	LABEL	TELM30

(3) Flowchart



Generate Tone	MODULE NO.	T23.3	LABEL	TELM30



Relay ON	MODULE NO.	T23.4	LABEL	TELM40

(1) Function

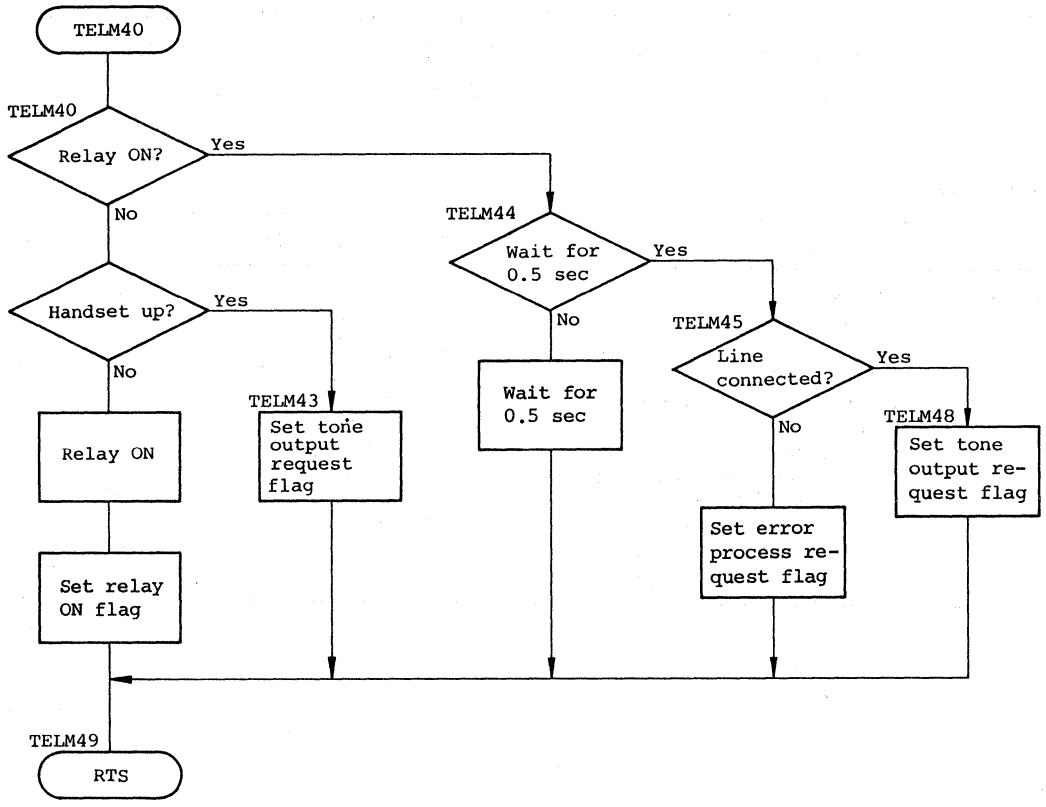
Confirms whether or not handset is up for tone output. If handset is up, set tone output request flag. If handset is down, connect line using relay. After confirming line connected, set tone output request flag.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Relay ON flag	(6, BZR)	(1 bit)
	Handset up? (Line connected?)	(2,PDDTR),(3,PDDTR)	(2 bits)
	Wait for 0.5 sec flag	TELWAT (RAM)	1
Returns	Relay ON request flag	(5, PFDTR)	(1 bit)
	Relay ON flag	(6, BZR)	(1 bit)
	Tone output request flag	(2, BZR)	(1 bit)
	Error process request flag	(0,ERFLG)(1,ERFLG)	(2 bits)
	Handset up	(0, TELFLG)	(1 bit)
	Which photo-coupler ON	(6,TELFLG)(7,TELFLG)	(2 bits)
	Relay OFF request flag	(5, BZR)	(1 bit)
	Dialing flag	(1, BZR)	(1 bit)

Relay ON	MODULE NO.	T23.4	LABEL	TELM40

(3) Flowchart



While Retrying	MODULE NO.	T23.5	LABEL	TELM50

(1) Function

Retries for 45 seconds every two minutes.

(a) While waiting for two minutes; blink the telephone number on and off to indicate retrying.

(b) If call is answered while ringing for 45 seconds,

(i) Set buzzer request flag to notify caller.

(ii) Set talking flag

(iii) Complete retrying

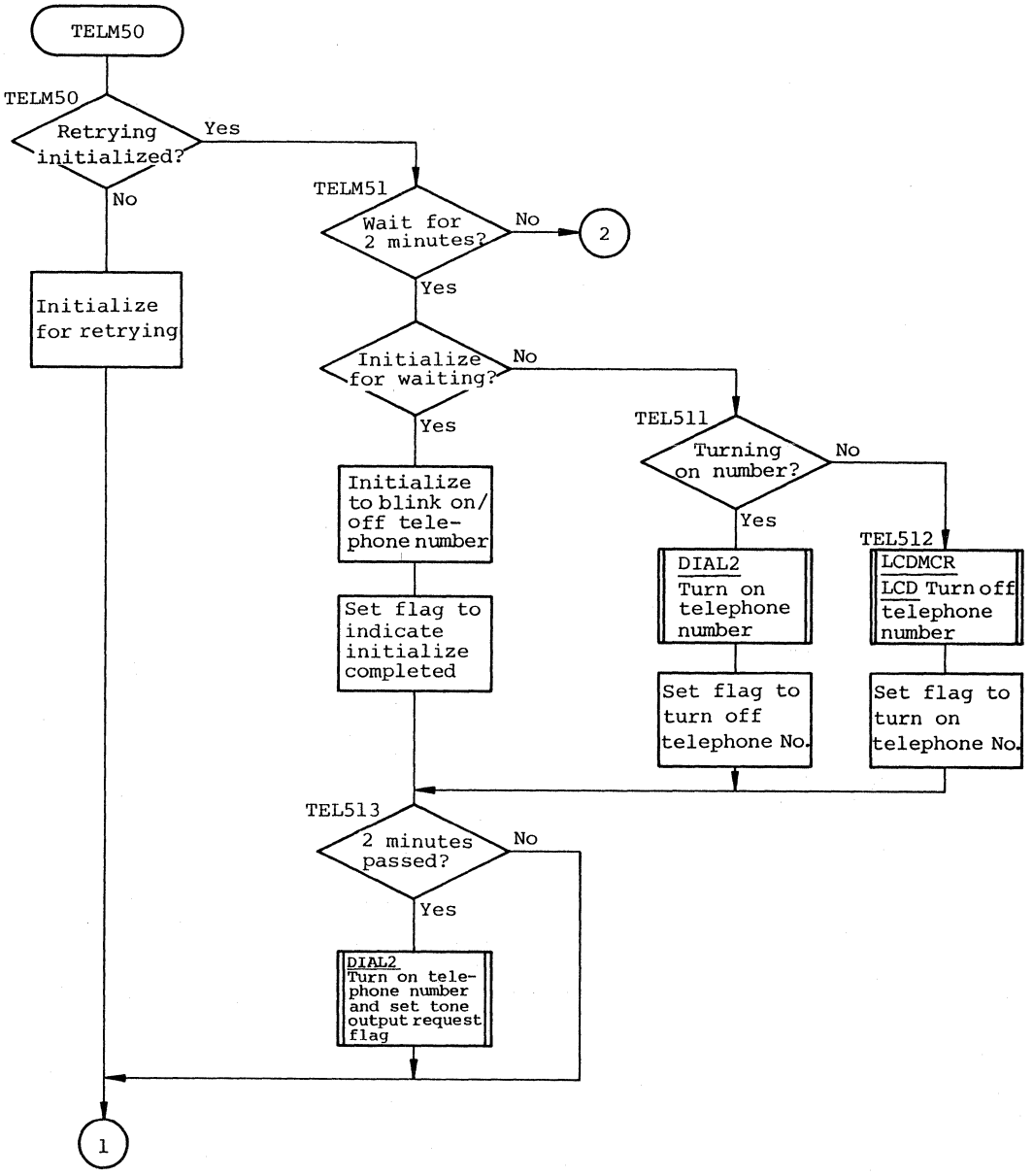
(c) If ringing is executed for 10 times with no answer, cancel retrying.

(2) Arguments

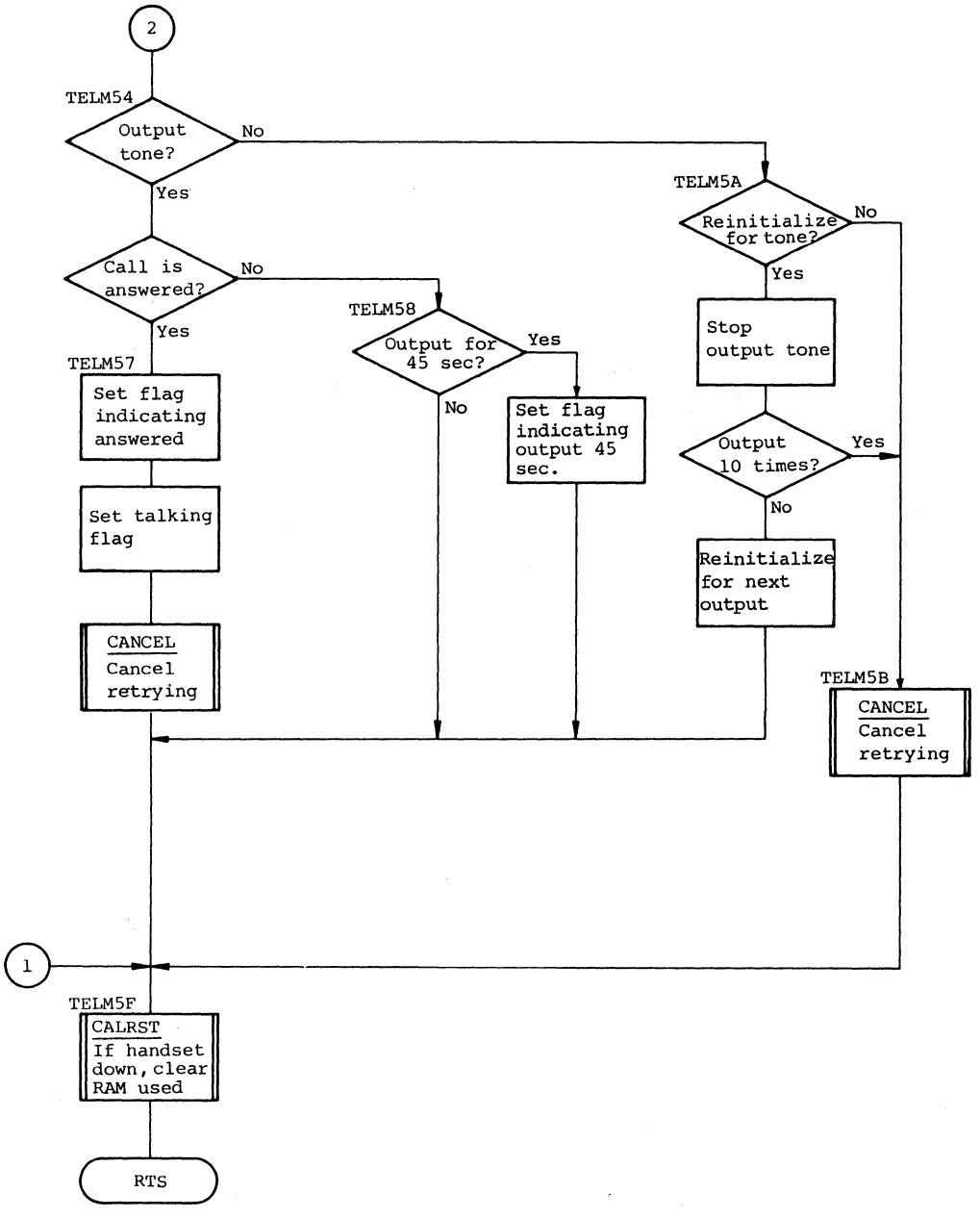
	Contents	Storage Location	No. of Bytes
Entry	Process flag for tone output	TELWAT (RAM)	1
	Counter for retrying function	REC (RAM)	1
	Counting how many times retrying is executed	(7, BZR)	(1 bit)
	Which photo-coupler is high	(6,TELFLG) (7,TELFLG)	(2 bits)
	Call is answered	(2,PDDTR) (3,PDDTR)	(2 bits)
Returns	Process flag for tone output	TELWAT (RAM)	1
	Telephone number	LCDM (RAM)	40
	Tone output request flag	(1, BZR)	(1 bit)
	Tone outputting flag	(4, WAIT3)	(1 bit)
	Talking flag	(1, TELFLG)	(1 bit)
	Handset up?	(0, TELFLG)	(1 bit)
	Buzzer output request flag	(3,BZRFLG)	(1 bit)
	Buzzing time	BZRCNT (RAM)	2
	Error message display OFF request counter	ERR1 (RAM), ERR2 (RAM)	2

While Retrying	MODULE NO.	T23.5	LABEL	TELM50

(3) Flowchart



While Retrying	MODULE NO.	T23.5	LABEL	TELM50



Connect Lines	MODULE NO.	T23.6	LABEL	TELM60

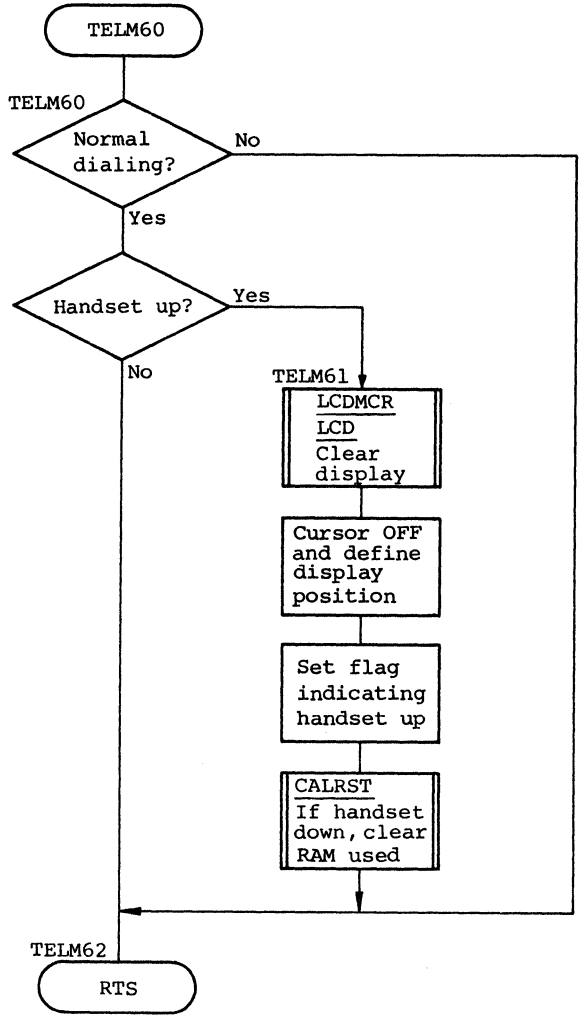
(1) Function

Clears display and sets flag indicating handset is up and prepare for tone output.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Mode flag	MODFLG (RAM)	1
	Handset up	(2,PDDTR)(3,PDDTR)	(2 bits)
Returns	LCD display RAM	LCDM (RAM)	40
	RAM for redialing	MEMO (RAM)	21
	Handset up?	(0, TELFLG)	(1 bit)
	Tone outputting flag	(4, WAIT3)	(1 bit)
	Cursor OFF	CURADP (RAM)	1
	LCD display pointer	LCDMP (RAM)	1

(3) Flowchart


TELM60
TELM60
Normal dialing?
No
Yes
Handset up?
Yes
No
TELM61
LCDMCR
LCD
Clear display
Cursor OFF and define display position
Set flag indicating handset up
CALRST
If handset down, clear RAM used
TELM62
RTS

Display on LCD	MODULE NO.	EX1.0	LABEL	LCD

(1) Function

Displays ASCII, stored in LCDM (RAM), on LCD.

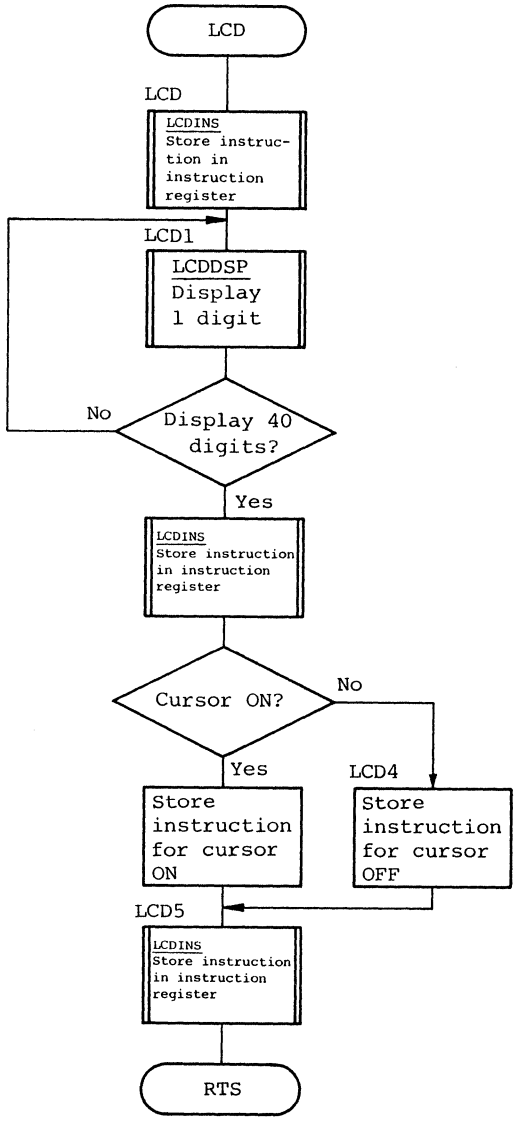
(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Display data (ASCII)	LCDM (RAM)	40
	Cursor position (Note)	CURADP (RAM)	1
Returns	———	———	———

Note: CURADP indicates cursor position.

Example: CURADP=0 No cursor displayed.
CURADP=1 Cursor is displayed on 1st digit.
CURADP=2 Cursor is displayed on 2nd digit.
:

(3) Flowchart


LCD
LCD
LCDINS
Store instruc-
tion in
instruction
register
LCD1
LCDDSP
Display
1 digit
No
Display 40
digits?
Yes
LCDINS
Store instruction
in instruction
register
Cursor ON?
No
Yes
LCD4
Store
instruction
for cursor
ON
Store
instruction
for cursor
OFF
LCD5
LCDINS
Store instruction
in instruction
register
RTS

Microcomputer → Telephone Numbers Storing RAM	MODULE NO.	EX2.0	LABEL	OUTDAT

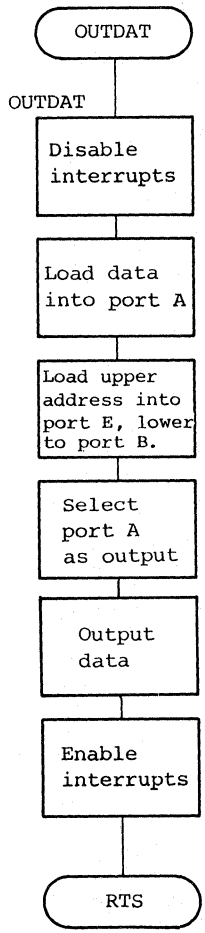
(1) Function

Loads data from microcomputer to RAM for storing telephone numbers.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Data	DATA (RAM)	1
	Address	DTDP (RAM)	2
Returns	Data	(RAM for storing telephone numbers)	1

(3) Flowchart



Microcomputer ← Telephone Numbers Storing RAM	MODULE NO.	EX2.1	LABEL	INPDAT

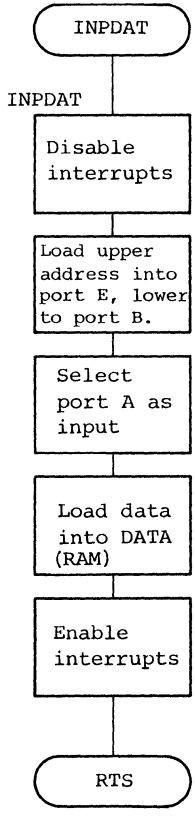
(1) Function

Loads data from RAM for storing telephone numbers to microcomputer.

(2) Arguments

	Contents	Storage Location	No. of Bytes
Entry	Address	DTDP (RAM)	2
Returns	Data	DATA (RAM)	1

(3) Flowchart



3.5 RAM Table

RAM allocation for the Intelligent Telephone is Shown in table 3.6.

Table 3.6 RAM Table

Label	No. of Bytes	Description
ADDRAT	4	Counter for rate of charge
BZRCNT	2	Counter for buzz time
CALCNT	1	Address pointer for modifying data
CALNDR	5	Counter for calendar & time
CNT	1	Address pointer for modifying data
CNTBIT	1	Indicator determining what should be changed from BCD to ASCII
CURADP	1	Current cursor position
DATA	1	Temporary storage for telephone no. data
DIADAT	5	Buffer of telephone no. for more than 21 digits tone output
DIALNO	21	Currently dialed telephone no.
DIAP1	1	Pointer to buffer of telephone no. for more than 21 digits tone output
DIAP2	1	
DTADS	2	Starting address of review data
DTCNT	1	Pointer indicating what digit of review data is being reviewed
DTDP	2	Address pointer for telephone no. data
ERR1	1	Counter for error message display time
ERR2	1	
HOLD	1	Pointer to display RAM converting blank spaces
KEYDAT	1	Valid key data
KEYNUM	1	Data for identifying depressed key
KEYSET	1	ASCII code corresponding key data
LCDM	40	LCD display RAM
LCDMP	1	Pointer to LCD display RAM
MEMO	21	Previously called telephone number
MEMOP	1	Contents of port E before tone output
MFDAT	1	Tone data to be output
MODFLG	1	Mode flag
NEWKEY	1	Current key data
OLDKEY	1	Previous key data
POINTR	1	Pointer for moving data to display RAM
REC	1	Counter for retrying function
REFDAT	20	Data being reviewed

Table 3.6 RAM Table (cont.)

Label	No. of Bytes	Description
SCOUNT	1	Counter used in determining depressed key
SDP	2	Pointer to next review data
SEC	2	Charge per unit time
SECCNT	2	Counter for unit time of charge
SHUCNT	1	Address pointer for modifying data
SHUSEC	2	Charge per time to be modified
SHUSTA	5	Calendar & time to be modified
SINGL	4	Counter of charge
STBDAT	1	Data for strobe signal output
TCNTR	1	Counter for timer 1 interrupts
TEL	3	Counter for elapsed time
TELCNT	1	Pointer to what digit of telephone no. is output
TELNO	1	Pointer to what digit of telephone no. is output
TELWAT	1	Process flag for tone output
TIMSTA	1	Time counter to be modified
TOTAL	4	Counter for accumulated charge
TOTLKY	1	Number of depressed key
TRMFLG	1	Process flag
TRNS	1	Counter of blank spaces
WAIT1	1	Counter for HS pin ON
WAIT2	1	Counter for HS pin ON/OFF
WAIT4	1	Counter for blinking telephone no. ON/OFF during retrying

3.6 Flag Table

The flag used by the Intelligent Telephone are shown in Table 3.7.

Note : Flag function of bits 1 ∿ 3 of port D data register is described at the last of Flag Table.

Table 3.7 Flag Table

Flag	Bit	Description
BZR	0	0 = Output High for buzzing 1 = Output Low for buzzing
	1	0 = Not requesting tone output 1 = Requesting tone output
	2	0 = Not tone outputting 1 = Tone outputting
	3	0 = Not requesting HS pin ON 1 = Requesting HS pin ON
	4	Not used
	5	0 = Not requesting relay OFF 1 = Requesting relay OFF
	6	0 = Relay OFF 1 = Relay ON
	7	0 = Not counting how many times retrying function is executed 1 = Counting how many times retrying function is executed
BZRFLG	0	0 = Not retrying 1 = Retrying
	1	0 = Not waiting for 2 minutes during retrying 1 = Waiting for 2 minutes during retrying
	2	0 = Not ringing for 45 sec during retrying 1 = Ringing for 45 sec during retrying
	3	0 = Not requesting buzzing 1 = Requesting buzzing
	4	0 = Not buzzing 1 = Buzzing
	5	Not used
	6	Not used
	7	0 = Cursor movement enable 1 = Cursor movement disable
CHATFL	0	Counter to indicate how many times current key data is compared with previous key data
	1	
	2	
	3	
	4	Not used
	5	Not used
	6	Not used
	7	0 = Chatter prevension incompletion 1 = Chatter prevension completion

Table 3.7 Flag Table (cont.)

Flag	Bit	Description
ERFLG	0	bit 2 1 0 Description
	1	0 0 0 No error
		0 0 1 Buzzing only
	2	0 1 1 Buzzing and displaying "ERROR"
		1 0 1 Buzzing and displaying "FULL"
	3	Not used
	4	Not used
	5	Not used
	6	Not used
	7	Not used
FLAG	0	0 = No key data input 1 = Key data input
	1	0 = Not displaying calendar & time 1 = Displaying calendar & time
	2	0 = Reviewing in forward direction 1 = Reviewing in reverse direction
	3	0 = Not displaying elapsed time 1 = Displaying elapsed time
	4	0 = Not reviewing data 1 = Reviewing data
	5	0 = Not desired data 1 = Desired data
	6	0 = Not displaying accumulated charge 1 = Displaying accumulated charge
	7	0 = Not displaying charge 1 = Displaying charge
TELFLG	0	0 = Handset down 1 = Handset up
	1	0 = Not talking 1 = Talking
	2	0 = Not receiving 1 = Receiving
	3	0 = No ringing 1 = Ringing
	4	0 = Not waiting for 3 sec before tone output 1 = Waiting for 3 sec before tone output
	5	0 = Not dialing 1 = Dialing
	6	0 = Photo-coupler (1) OFF 1 = Photo-coupler (1) ON
	7	0 = Photo-coupler (2) OFF 1 = Photo-coupler (2) ON

Table 3.7 Flag Table (cont.)

Flag	Bit	Description
WAIT3	0	0 = Not contents of port E restored 1 = Contents of port E restored
	1	0 = Retrying not initialized 1 = Retrying initialized
	2	0 = Turn on telephone number during retrying 1 = Turn off telephone number during retrying
	3	Not used
	4	0 = No tone output 1 = Tone output
	5	0 = Not initialized for tone output 1 = Initialized for tone output
	6	Not used
	7	Not used
WAIT5	0	Not used
	1	0 = Not receiving 1 = Receiving
	2	Not used
	3	0 = Not preparing for tone output after 21st digit 1 = Preparing for tone output after 21st digit
	4	0 = LCD display not full 1 = LCD display full
	5	0 = Tone output until 20th digit 1 = Tone output after 21st digit
	6	Not used
	7	0 = Not displaying "ERROR" 1 = Displaying "ERROR"

Note : Bits 1 ∿ 3 of port D data register indicate dialing circuit status as follows;

Flag	Bit	Description
PDDTR	1	0 = Not receiving 1 = Receiving
	2 3	Handset up bit 2 3 (1 0) ⇨ Talking bit 2 3 (0 1) or Handset up bit 2 3 (0 1) ⇨ Talking bit 2 3 (1 0)

3.7 Subroutine Table

The subroutines used by the Intelligent Telephone are shown in Table 3.8 with input and output arguments. (Refer to "3.5 RAM Table" for argument details.)

Table 3.8 Subroutine Table

Subroutine Name	Entry		Returns		Function
	Storage Location	Bytes	Storage Location	Bytes	
ARGINT	——	——	——	——	Clear internal RAM
CALLCD	CALNDR (RAM)	5	LCDM (RAM)	40	Load calendar data into display RAM
CALRST	——	——	——	——	If handset is down, clear RAM used
CALSHU	LCDM (RAM)	40	CALNDR (RAM) TELFLG (RAM) ERFLG (RAM)	5 1 1	Determine whether or not modified data is correct format as calendar & time. If data is correct, set modified calendar & time. If incorrect, store "0" in TRMFLG (RAM).
CANCEL	——	——	——	——	Cancel retrying
CLEAR	——	——	LCDM (RAM)	40	Store $00 in display RAM
CRGSET	ADDRAT (RAM) SEC (RAM)	4 2	LCDM (RAM)	40	Load charge per unit time into display RAM
CURCAL	LCDMP (RAM) KEYDAT (RAM)	1 1	LCDMP (RAM) CURADP (RAM)	1 1	Define cursor position for setting calendar & time
CURRAT	LCDMP (RAM)	1	LCDMP (RAM) CURADP (RAM)	1 1	Define cursor position for setting charge
DIACLR	——	——	DIALNO (RAM)	21	Clear RAM for tone output
DIAL2	MEMO (RAM)	21	LCDM (RAM)	40	Display telephone number for retrying
DTADD	DTDP (RAM)	2	DTDP (RAM)	2	Increment pointer to data table
DTLCD	DTDP (RAM)	2	LCDM (RAM)	40	Display name and telephone number
DTSUB	DTDP (RAM)	2	DTDP (RAM)	2	Decrement pointer to data table

Table 3.8 Subroutine Table (cont.)

Subroutine Name	Entry		Returns		Function
	Storage Location	Bytes	Storage Location	Bytes	
END	(4, WAIT3)	(1 bit)	(1, FLAG)	(1 bit)	Clear RAM used in Mode Process and set calendar & time display request flag. In the case of tone outputting, display telephone no.
	DIALNO (RAM)	21	LCDM (RAM)	40	
KENSAK	REFDAT (RAM)	20	LCDM (RAM)	40	Compare data string
	FLAG (RAM)	1	FLAG (RAM)	1	
LCDBSY	——	——	——	——	Check LCD busy flag
LCDDSP	IX	1	——	——	Store ASCII in DDRAM of LCD-II
LCDINS	IX	1	——	——	Store instruction in instruction register
LCDINT	——	——	——	——	Select display mode of LCD-II
LCDMCR	——	——	LCDM (RAM)	40	Clear display RAM
LCDMST	FLAG (RAM)	1	LCDM (RAM)	40	Clear display RAM and send display format
LCDRES	——	——	——	——	Reset LCD-II by instruction
LCDSET	CNT (RAM)	1	LCDM (RAM)	40	Send calendar & time or charge to display RAM
	FLAG (RAM)	1			
LFCRSR	CURADP (RAM)	1	CURADP (RAM)	1	Move cursor to the left
	LCDMP (RAM)	1	LCDMP (RAM)	1	
	MODFLG (RAM)	1			
	TRMFLG (RAM)	1			
MEMCLR	——	——	MEMO (RAM)	21	Clear telephone number for redialing
MFOUT	ACCA	1	MFDAT (RAM)	1	Output tone data corresponding to ASCII
MNYASC	TOTAL (RAM)	4	LCDM (RAM)	40	Convert data to ASCII and load in display RAM
	SINGL (RAM)	4			
	ADDRAT (RAM)	4			
	SEC (RAM)	2			
MONEY	ADDRAT (RAM)	4	TOTAL (RAM)	4	Count charge
	SEC (RAM)	2	SINGL (RAM)	4	
MOVE1	LCDM (RAM)	40	REFDAT (RAM)	20	Move to review data area

Table 3.8 Subroutine Table (cont.)

Subroutine Name	Entry Storage Location	Bytes	Returns Storage Location	Bytes	Function
MOVE2	REFDAT (RAM)	20	LCDM (RAM)	40	Move review data to display RAM
RAMCLR	——	——	——	——	Clear RAM for storing telephone numbers
RATSET	LCDM (RAM)	40	ADDRAT (RAM) SEC (RAM) TRMFLG (RAM) ERFLG (RAM)	4 2 1 1	Determine whether or not modified data is correct format as charge per unit time. If data is correct, set modified charge per unit time. If incorrect, store "0" in TRMFLG (RAM).
RTCRSR	CURADP (RAM) LCDMP (RAM) MODFLG (RAM) TRMFLG (RAM)	1 1 1 1	CURADP (RAM) LCDMP (RAM)	1 1	Move cursor to the right
SHUCAL	CALNDR (RAM) CURADP (RAM)	5 1	LCDM (RAM)	40	Load calendar & time or cursor into display RAM
SNGCLR	——	——	SINGL (RAM)	4	Clear charge
SNGSTA	——	——	LCDM (RAM)	40	Load charge into display RAM
TAN	LCDM (RAM)	40	DIALNO (RAM) MEMO (RAM) LCDMP (RAM) CURADP (RAM)	21 21 1 1	Display speed dial and corresponding telephone number
TELCLR	——	——	TEL (RAM)	3	Clear elapsed time
TLCNT	TEL (RAM)	3	TEL (RAM)	3	Count elapsed time
TELLCD	CNT (RAM) FLAG (RAM)	1 1	LCDM (RAM)	40	Load elapsed time into display RAM
TELNO1	LCDM (RAM)	40	DATA (RAM)	1	Change 2 bytes of data into 1 byte
TELNO2	DATA (RAM)	1	LCDM (RAM)	40	Change 1 byte of data into 2 bytes
TIMCNT	CNTBIT (RAM)	1	CALNDR (RAM) TEL (RAM)	5 3	Count calendar & time charge
TOLCLR	——	——	TOTAL (RAM)	4	Clear accumulated charge

Table 3.8 Subroutine Table (cont.)

Subroutine Name	Entry		Returns		Function
	Storage Location	Bytes	Storage Location	Bytes	
TOLSTA	TOTAL (RAM)	4	LCDM (RAM)	40	Load accumulated charge into display RAM
TOROKU	DTDP (RAM) LCDM (RAM)	2 40	(RAM for storing telephone no.)	29	Enter name and telephone number in RAM for storing telephone no.
YERCNT	CALNDR (RAM)	5	CALNDR (RAM)	5	Count calendar & time

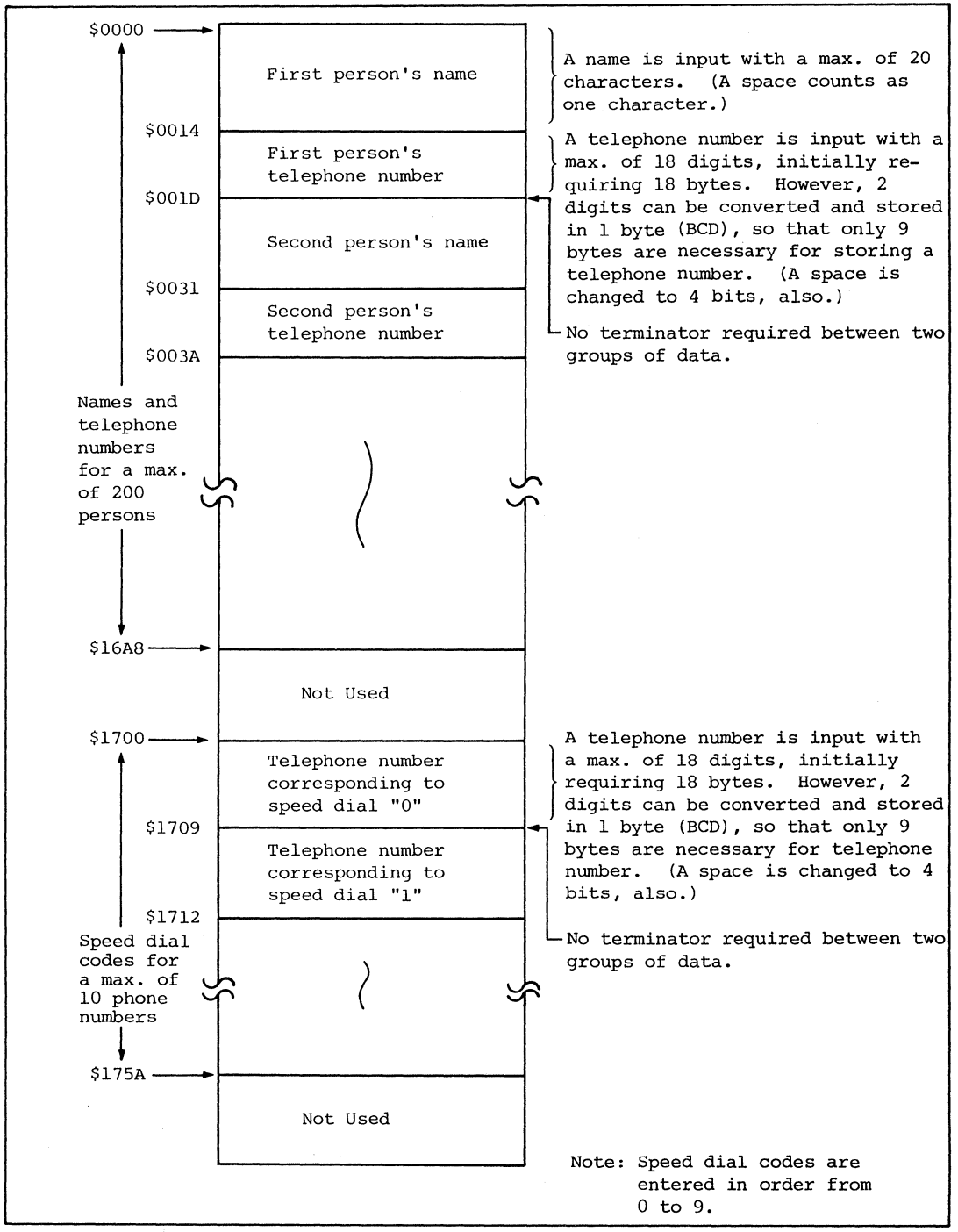


Fig. 3.4 RAM Memory Map for Storing Telephone Numbers

3.9 Ports Labels Table

The ports used by the Intelligent Telephone are shown in Table 3.9.

Table 3.9 Ports Labels Table

Ports	Label	Address
Port A data register	PADTR	$00
Port A data direction register	PADDR	$04
Port B data register	PBDTR	$01
Port B data direction register	PBDDR	$05
Port C data register	PCDTR	$02
Port C data direction register	PCDDR	$06
Port D data register	PDDTR	$03
Port E data register	PEDTR	$0B
Port F data register	PFDTR	$0C
Port G data register	PGDTR	$0D
Port G data direction register	PGDDR	$07

4.1 Program Listing

```
00001                    *
00002                    ****    RAM ALLOCATION    ************************
00003                    *
00004 0040               *       ORG     $40
00005                    *
00006 0040 0001          LCDMP   RMB     1       Pointer to diplay RAM
00007 0041 0001          MODFLG  RMB     1       Mode flag
00008 0042 0001          FLAG    RMB     1       Flag for information to be displayed
00009 0043 0001          TRMFLG  RMB     1       Process flag
00010 0044 0001          TELFLG  RMB     1       Flag for output tone
00011 0045 0002          DTDP    RMB     2       Address pointer to telephone no. data
00012 0047 0002          SDP     RMB     2       Pointer to next review data
00013 0049 0001          DTCNT   RMB     1       Pointer to review data digit
00014 004A 0001          DATA    RMB     1       Temporary storage for telephone no. data
00015 004B 0001          CURADP  RMB     1       Current cursor pointer
00016 004C 0002          BZRCNT  RMB     2       Buzzeing time counter
00017 004E 0001          BZRFLG  RMB     1       Flag for tone output
00018 004F 0001          BZR     RMB     1       Flag for tone output
00019 0050 0002          DTADS   RMB     2       Start address of reiview data
00020 0052 0002          SEC     RMB     2       Charge per unit time
00021                    *
00022 0054 0001          KEYSET  RMB     1       ASCII code
00023 0055 0001          OLDKEY  RMB     1       Previous key data
00024 0056 0001          NEWKEY  RMB     1       Current key data
00025 0057 0001          CHATFL  RMB     1       Chatter counter
00026 0058 0001          KEYDAT  RMB     1       Valid key data
00027 0059 0001          TOTLKY  RMB     1       Total key number
00028 005A 0001          KEYNUM  RMB     1       Data for indentifying depressed key
00029 005B 0001          STBDAT  RMB     1       Data for strobe signal output
00030 005C 0001          SCOUNT  RMB     1       Counter used in determining depressed key
00031                    *
00032 005D 0005          CALNDR  RMB     5       Counter for calendar & time
00033 0062 0003          TEL     RMB     3       Counter for elapsed time
00034 0065 0005          SHUSTA  RMB     5       Calendar & tme to be modified
00035 006A 0001          TIMSTA  RMB     1       Time counter to be modified
00036 006B 0001          CNTBIT  RMB     1       Indicator for BCD/ASCII conversion
00037 006C 0001          POINTR  RMB     1       Pointer to moving data to display RAM
00038 006D 0001          CNT     RMB     1       Address pointer for modifying data
00039 006E 0001          CALCNT  RMB     1       Address pointer for modifying data
00040 006F 0001          SHUCNT  RMB     1       Address pointer for modifying data
00041 0070 0001          TCNTR   RMB     1       Counter for tomer 1 interrupts
00042 0071 0028          LCDM    RMB     40      Display RAM
00043 0099 0015          DIALNO  RMB     21      Currently dialed telephone no.
00044                    *
00045 00AE 0001          WAIT1   RMB     1       Counter for  HS pin ON
00046 00AF 0001          WAIT2   RMB     1       Counter for HS pin ON/OFF time
00047 00B0 0001          WAIT3   RMB     1       Flag for tone output
00048 00B1 0001          WAIT4   RMB     1       Counter for blinking telephone no. ON/OFF
00049 00B2 0001          WAIT5   RMB     1       Flag for tone output
00050 00B3 0001          MEMOP   RMB     1       Contents of port E
00051 00B4 0001          REC     RMB     1       Counter for retrying function
00052 00B5 0001          MFDAT   RMB     1       Tone data to be output
00053 00B6 0001          TELWAT  RMB     1       Process flag for tone output
00054 00B7 0005          DIADAT  RMB     5       Buffer of telephone no.
00055 00BC 0001          DIAP1   RMB     1       Pointer to buffer of telephone no.
```

```
00056 00BD 0001      DIAP2  RMB    1         Pointer to buffer of telephone no.
00057 00BE 0001      ERFLG  RMB    1         Flag for error process
00058                *
00059 00BF 0001      HOLD   RMB    1         Pointer to display RAM for blank space process
00060 00C0 0001      TRNS   RMB    1         Counter of blank space
00061                *
00062 0100                ORG    $100
00063                *
00064 0100 0015      MEMO   RMB    21        Previously called telephone no.
00065                *
00066 0115 0014      REFDAT RMB    20        Data being reviewed
00067 0129 0004      ADDRAT RMB    4         Counter for rate of charge
00068 012D 0004      SINGL  RMB    4         Counter for charge
00069 0131 0004      TOTAL  RMB    4         Counter for accumulated charge
00070 0135 0002      SECCNT RMB    2         Counter for unit time of charge
00071 0137 0002      SHUSEC RMB    2         Charge per time to be modified
00072                *
00073 0139 0001      TELNO  RMB    1         Digit pointer to telephone no.
00074 013A 0001      TELCNT RMB    1         Digit pointer to telephone no.
00075 013B 0001      ERR1   RMB    1         Count error message display time
00076 013C 0001      ERR2   RMB    1         Count error message display time
00077                *
00078                * JUMP FROM TABLE
00079                *
00080 013D 0001      JTBL   RMB    1         Store instruction for jump
00081 013E 0002      JMPADR RMB    2         Store ADDR for jump
00082                *
00083                ****   SYMBOL DEFINITION   **********************
00084                *
00085       0000     PADTR  EQU    $00       Port A data reg.
00086       0004     PADDR  EQU    $04       Port A data direction reg.
00087       0001     PBDTR  EQU    $01       Port B data reg.
00088       0005     PBDDR  EQU    $05       Port B data direction reg.
00089       0002     PCDTR  EQU    $02       Port C data reg.
00090       0006     PCDDR  EQU    $06       Port C data direction reg.
00091       0003     PDDTR  EQU    $03       Port D data reg.
00092       000B     PEDTR  EQU    $0B       Port E data reg.
00093       000C     PFDTR  EQU    $0C       Port F data reg.
00094       000D     PGDTR  EQU    $0D       Port G data reg.
00095       0007     PGDDR  EQU    $07       Port G data direction reg.
00096       0010     SCR    EQU    $10       SCI CTRL reg.
00097       0011     SSR    EQU    $11       SCI STS reg.
00098       0008     TDR    EQU    $08       Timer data reg.
00099       0009     TCR    EQU    $09       Timer control reg.
00100       16A8     RAMEND EQU    29*200    External RAM end address
00101       0003     CMPNUM EQU    $03
00102                *
00103                ************************************************
00104                *                                              *
00105                *       MAIN  PROGRAM      (MAINPR)            *
00106                *                                              *
00107                ************************************************
00108                *
00109 0500                ORG    $500
00110                *
```

```
00111      0500      MAINPR EQU    *
00112 0500 9C                RSP              Initialize stack pointer
00113 0501 CD 0523           JSR     SYSINT   Initialize system
00114 0504 01 42 FD MAINKS   BRCLR   0,FLAG,MAINKS  Key data input?
00115 0507 A6 2E             LDA     #$2E     Key data >= $2E
00116 0509 B1 58             CMP     KEYDAT
00117 050B 22 05             BHI     MAIN1    If < $2E, execute input data check
00118 050D CD 057B           JSR     MODE     If >= $2E, execute mode process
00119 0510 20 05             BRA     MAIN2
00120 0512 CD 05E9 MAIN1     JSR     CHECK
00121 0515 20 00             BRA     MAIN2
00122 0517 00 BE 02 MAIN2    BRSET   0,ERFLG,MAIN3  Error?
00123 051A 20 03             BRA     MAIN4
00124 051C CD 0695 MAIN3     JSR     ERROR
00125 051F 11 42   MAIN4     BCLR    0,FLAG   Reintialize flag
00126 0521 20 E1             BRA     MAINKS
00127                        *
00128                        ************************************************
00129                        *                                              *
00130                        *       NAME    : INITIALIZE SYSTEM (SYSINT)   *
00131                        *                                              *
00132                        ************************************************
00133                        *                                              *
00134                        *       ENTRY   : NOTHING                      *
00135                        *       RETURNS : NOTHING                      *
00136                        *                                              *
00137                        ************************************************
00138 0523 3F 55    SYSINT   CLR     OLDKEY
00139 0525 A6 FA             LDA     #$FA     Initialize timer1
00140 0527 B7 08             STA     TDR
00141 0529 A6 85             LDA     #$85
00142 052B B7 09             STA     TCR
00143 052D A6 0E             LDA     #$0E     Initialize timer2
00144 052F B7 10             STA     SCR
00145 0531 A6 20             LDA     #$20
00146 0533 B7 11             STA     SSR
00147 0535 A6 FF             LDA     #$FF     Initialize ports
00148 0537 B7 05             STA     PBDDR
00149 0539 3F 02             CLR     PCDTR
00150 053B A6 60             LDA     #$60
00151 053D B7 0B             STA     PEDTR
00152 053F CD 1541           JSR     ARGINT   Initialize internal RAM
00153 0542 CD 18DA           JSR     END
00154 0545 A6 1F             LDA     #$1F     Check data in external RAM
00155 0547 B7 45             STA     DTDP
00156 0549 3F 46             CLR     DTDP+1
00157 054B CD 187C           JSR     INPDAT
00158 054E B6 4A             LDA     DATA
00159 0550 A1 4C             CMP     #$4C
00160 0552 26 23             BNE     RAMCL
00161 0554 3C 46             INC     DTDP+1
00162 0556 CD 187C           JSR     INPDAT
00163 0559 B6 4A             LDA     DATA
00164 055B A1 4F             CMP     #$4F
00165 055D 26 18             BNE     RAMCL
```

```
00166 055F 3C 46           INC    DTDP+1
00167 0561 CD 187C         JSR    INPDAT
00168 0564 B6 4A           LDA    DATA
00169 0566 A1 56           CMP    #$56
00170 0568 26 0D           BNE    RAMCL
00171 056A 3C 46           INC    DTDP+1
00172 056C CD 187C         JSR    INPDAT
00173 056F B6 4A           LDA    DATA
00174 0571 A1 45           CMP    #$45
00175 0573 26 02           BNE    RAMCL
00176 0575 20 03           BRA    RAMCLE
00177 0577 CD 1970 RAMCL   JSR    RAMCLR   Clear external RAM
00178 057A 81      RAMCLE  RTS
00179                      *
00180                      ************************************************
00181                      *                                              *
00182                      *      NAME    : MODE PROCESS (MODE)           *
00183                      *                                              *
00184                      ************************************************
00185                      *                                              *
00186                      *      ENTRY   : KEYDAT, TRMFLG, MODFLG        *
00187                      *      RETURNS : MODFLG, TRMFLG, ERFLG         *
00188                      *                                              *
00189                      ************************************************
00190 057B B6 58    MODE   LDA    KEYDAT   Execute key?
00191 057D A1 40           CMP    #$40
00192 057F 27 06           BEQ    MODE2
00193 0581 A0 2D           SUB    #$2D     If not, define mode flag
00194 0583 B7 41           STA    MODFLG
00195 0585 20 0A           BRA    MODE4
00196 0587 B6 41    MODE2  LDA    MODFLG   If so, test if it is error
00197 0589 26 04           BNE    MODE3
00198 058B 10 BE           BSET   0,ERFLG  If error, set error process request flag
00199 058D 20 29           BRA    MODEF
00200 058F 3C 43    MODE3  INC    TRMFLG   Increment process flag
00201 0591 5F       MODE4  CLR    X
00202 0592 D1 05B9  MODE5  CMP    MDTBL,X  Jump from table
00203 0595 27 0B           BEQ    MODE6
00204 0597 5C              INC    X
00205 0598 5C              INC    X
00206 0599 5C              INC    X
00207 059A A3 30           CPX    #MDTBLE-MDTBL
00208 059C 26 F4           BNE    MODE5
00209 059E 10 BE           BSET   0,ERFLG
00210 05A0 20 16           BRA    MODEF
00211 05A2 5C       MODE6  INC    X
00212 05A3 D6 05B9         LDA    MDTBL,X
00213 05A6 C7 013E         STA    JMPADR
00214 05A9 5C              INC    X
00215 05AA D6 05B9         LDA    MDTBL,X
00216 05AD C7 013F         STA    JMPADR+1
00217 05B0 A6 CC           LDA    #$CC
00218 05B2 C7 013D         STA    JTBL     Call specified module
00219 05B5 CD 013D         JSR    JTBL
00220 05B8 81       MODEF  RTS
```

```
00221                    *
00222                    ************************************************
00223                    *                                              *
00224                    *       NAME    : JUMP TABLE FOR MODE PROCESS  *
00225                    *                                              *
00226                    ************************************************
00227 05B9 01       MDTBL  FCB    $1
00228 05BA 06EE             FDB    MOD10    Set charge display request flag
00229 05BC 02               FCB    $2
00230 05BD 06F7             FDB    MOD20    Set accumulated charge display request flag
00231 05BF 03               FCB    $3
00232 05C0 0700             FDB    MOD30    Set elapsed time display request flag
00233 05C2 04               FCB    $4
00234 05C3 0709             FDB    MOD40    Entry a telephone number
00235 05C5 05               FCB    $5
00236 05C6 0739             FDB    MOD50    Review/modify/delete a telephone number
00237 05C8 06               FCB    $6
00238 05C9 0739             FDB    MOD50    Review/modify/delete a telephone number
00239 05CB 07               FCB    $7
00240 05CC 0739             FDB    MOD50    Review/modify/delete a telephone number
00241 05CE 08               FCB    $8
00242 05CF 0801             FDB    MOD80    Prepare for speed dialing
00243 05D1 09               FCB    $9
00244 05D2 084C             FDB    MOD90    Enter a speed dial number
00245 05D4 0A               FCB    $A
00246 05D5 08B2             FDB    MOD100   Prepare for redialing
00247 05D7 0B               FCB    $B
00248 05D8 08E0             FDB    MOD110   Prepare for retrying
00249 05DA 0C               FCB    $C
00250 05DB 090E             FDB    MOD120   Prepare for normal dialing
00251 05DD 0D               FCB    $D
00252 05DE 0929             FDB    MOD130   Set calendar & time
00253 05E0 0E               FCB    $E
00254 05E1 0955             FDB    MOD140   Set rate for charge
00255 05E3 0F               FCB    $F
00256 05E4 0980             FDB    MOD150   Clear accumulated charge
00257 05E6 10               FCB    $10
00258 05E7 098A             FDB    MOD160   End
00259       05E9     MDTBLE EQU    *
00260                    *
00261                    ************************************************
00262                    *                                              *
00263                    *       NAME    : CHECK INPUT DATA (CHECK)     *
00264                    *                                              *
00265                    ************************************************
00266                    *                                              *
00267                    *       ENTRY   : KEYDAT, MODFLG, FLAG         *
00268                    *       RETURNS : KEYSET                       *
00269                    *                                              *
00270                    ************************************************
00271 05E9 03 42 03 CHECK  BRCLR  1,FLAG,CHECK1  Calendar & time?
00272 05EC CC 0651          JMP    CHECKF
00273 05EF A6 29    CHECK1 LDA    #$29     Key data = numeral?
00274 05F1 B1 58           CMP    KEYDAT
00275 05F3 22 02           BHI    CHECK2   If so, execute numeral process
```

```
00276 05F5 20 33           BRA    CHECK5   If not, execute cursor process
00277 05F7 BE 58    CHECK2 LDX    KEYDAT   For numeral process
00278 05F9 D6 0651         LDA    KEYCD-1,X
00279 05FC B7 54           STA    KEYSET   Change key data into ASCII
00280 05FE B6 41           LDA    MODFLG
00281 0600 5F              CLR    X        Search module to be executed
00282 0601 D1 067A  CHECK3 CMP    CATBL,X
00283 0604 27 0C           BEQ    CHECK4
00284 0606 5C              INC    X
00285 0607 5C              INC    X
00286 0608 5C              INC    X
00287 0609 A3 12           CPX    #CATBLE-CATBL
00288 060B 26 F4           BNE    CHECK3
00289 060D CD 0A8C         JSR    CHEK4
00290 0610 20 3F           BRA    CHECKF
00291 0612 5C       CHECK4 INC    X
00292 0613 D6 067A         LDA    CATBL,X
00293 0616 C7 013E         STA    JMPADR
00294 0619 5C              INC    X
00295 061A D6 067A         LDA    CATBL,X
00296 061D C7 013F         STA    JMPADR+1
00297 0620 A6 CC           LDA    #$CC
00298 0622 C7 013D         STA    JTBL     Call specified module
00299 0625 CD 013D         JSR    JTBL
00300 0628 20 27           BRA    CHECKF
00301 062A 5F       CHECK5 CLR    X        For cursor process
00302 062B B6 58           LDA    KEYDAT
00303 062D D1 068C  CHECK6 CMP    CSTBL,X  Search module to be executed
00304 0630 27 09           BEQ    CHECK7
00305 0632 5C              INC    X
00306 0633 5C              INC    X
00307 0634 5C              INC    X
00308 0635 A3 09           CPX    #CSTBLE-CSTBL
00309 0637 26 F4           BNE    CHECK6
00310 0639 20 16           BRA    CHECKF
00311 063B 5C       CHECK7 INC    X
00312 063C D6 068C         LDA    CSTBL,X
00313 063F C7 013E         STA    JMPADR
00314 0642 5C              INC    X
00315 0643 D6 068C         LDA    CSTBL,X
00316 0646 C7 013F         STA    JMPADR+1
00317 0649 A6 CC           LDA    #$CC
00318 064B C7 013D         STA    JTBL     Call specified module
00319 064E CD 013D         JSR    JTBL
00320 0651 81       CHECKF RTS
00321                      *
00322                      ************************************************
00323                      *                                              *
00324                      *       DATA TABLE                             *
00325                      *                                              *
00326                      ************************************************
00327                      *
00328 0652 41       KEYCD  FCC    "ABCDE123"       Data for 8x8
00329 065A 46              FCC    "FGHIJ456"          key scan
00330 0662 4B              FCC    "KLMNO789"
```

```
00331 066A 50              FCC     "PQRST*0#"
00332 0672 55              FCC     "UVWXYZ. "
00333                 *
00334                 ************************************************
00335                 *                                              *
00336                 *       NAME    : JUMP TABLE FOR DATA CHECK    *
00337                 *                                              *
00338                 ************************************************
00339 067A 00        CATBL  FCB     $0
00340 067B 098E            FDB     CHEK1     Check for normal dialing
00341 067D 0C              FCB     $C
00342 067E 098E            FDB     CHEK1     Check for normal dialing
00343 0680 08              FCB     $8
00344 0681 0A20            FDB     CHEK2     Check for speed dialing
00345 0683 09              FCB     $9
00346 0684 0A47            FDB     CHEK3     Check for entering speed dial number
00347 0686 0D              FCB     $D
00348 0687 0ADD            FDB     CHEK5     Check for setting calendar & time, charge
00349 0689 0E              FCB     $E
00350 068A 0ADD            FDB     CHEK5     Check for setting calendar & time, charge
00351      068C      CATBLE EQU     *
00352                 *
00353                 ************************************************
00354                 *                                              *
00355                 *       NAME    : JUMP TABLE FOR CURSOR        *
00356                 *                                              *
00357                 ************************************************
00358 068C 29        CSTBL  FCB     $29
00359 068D 0B12            FDB     CURLR     Move cursor to right or left
00360 068F 2A              FCB     $2A
00361 0690 0B12            FDB     CURLR     Move cursor to right or left
00362 0692 2B              FCB     $2B
00363 0693 0B72            FDB     MVECUR    Move cursor to 21st digit
00364      0695      CSTBLE EQU     *
00365                 *
00366                 ************************************************
00367                 *                                              *
00368                 *       NAME    : ERROR PROCESS (ERROR)        *
00369                 *                                              *
00370                 ************************************************
00371                 *                                              *
00372                 *       ENTRY   : BZRCNT, ERR1, ERR2, ERFLG    *
00373                 *       RETURNS : ERFLG                        *
00374                 *                                              *
00375                 ************************************************
00376 0695 B6 4D     ERROR  LDA     BZRCNT+1 Buzzing time defined?
00377 0697 26 12           BNE     ERROR2
00378 0699 C6 013B         LDA     ERR1     Displaying time defined?
00379 069C 26 0D           BNE     ERROR2
00380 069E A6 60           LDA     #$60
00381 06A0 B7 4D           STA     BZRCNT+1 Define buzzing time
00382 06A2 C7 013B         STA     ERR1     Define displaying time
00383 06A5 4F              CLR     A
00384 06A6 B7 4C           STA     BZRCNT
00385 06A8 C7 013C         STA     ERR2
```

```
00386 06AB 16 4E     ERROR2 BSET   3,BZRFLG
00387 06AD 03 BE 37         BRCLR  1,ERFLG,ERROR6  Only buzzer?
00388 06B0 04 BE 1B         BRSET  2,ERFLG,ERROR4  Display "ERROR"
00389 06B3 B6 42            LDA    FLAG      Clear display
00390 06B5 A4 35            AND    #$35
00391 06B7 B7 42            STA    FLAG
00392 06B9 CD 1966          JSR    LCDMCR
00393 06BC CD 110E          JSR    LCD
00394 06BF AE 09            LDX    #9
00395 06C1 D6 1A6B   ERROR3 LDA    ERRORD-1,X  Display "ERROR"
00396 06C4 E7 81            STA    LCDM+16,X
00397 06C6 5A               DEC    X
00398 06C7 26 F8            BNE    ERROR3
00399 06C9 CD 110E          JSR    LCD
00400 06CC 20 19            BRA    ERROR6
00401 06CE B6 42     ERROR4 LDA    FLAG      Clear display
00402 06D0 A4 35            AND    #$35
00403 06D2 B7 42            STA    FLAG
00404 06D4 CD 1966          JSR    LCDMCR
00405 06D7 CD 110E          JSR    LCD
00406 06DA AE 08            LDX    #8
00407 06DC D6 1A63   ERROR5 LDA    FULLD-1,X  Display "FULL"
00408 06DF E7 81            STA    LCDM+16,X
00409 06E1 5A               DEC    X
00410 06E2 26 F8            BNE    ERROR5
00411 06E4 CD 110E          JSR    LCD
00412 06E7 B6 BE     ERROR6 LDA    ERFLG
00413 06E9 A4 F8            AND    #%11111000
00414 06EB B7 BE            STA    ERFLG
00415 06ED 81               RTS
00416                *
00417                ****   PROGRAM MODULE FOR MODE PROCESS   *******
00418                *
00419                *
00420                ************************************************
00421                *                                              *
00422                *      NAME    : SET CHARGE DISPLAY REQUEST    *
00423                *                      FLAG (MOD10)            *
00424                *                                              *
00425                ************************************************
00426                *                                              *
00427                *      ENTRY   : NOTHING                       *
00428                *      RETURNS : FLAG                          *
00429                *                                              *
00430                ************************************************
00431 06EE B6 42     MOD10  LDA    FLAG      Clear display
00432 06F0 A4 35            AND    #$35
00433 06F2 B7 42            STA    FLAG
00434 06F4 1E 42            BSET   7,FLAG    Set charge display request flag
00435 06F6 81               RTS
00436                *
00437                ************************************************
00438                *                                              *
00439                *      NAME    : SET ACCUMULATED CHARGE DISPLAY *
00440                *                      REQUEST FLAG (MOD20)    *
```

```
00441                   *                                                *
00442                   ************************************************
00443                   *                                                *
00444                   *       ENTRY   : NOTHIMG                        *
00445                   *       RETURNS : FLAG                           *
00446                   *                                                *
00447                   ************************************************
00448 06F7 B6 42       MOD20   LDA     FLAG      Clear display
00449 06F9 A4 35               AND     #$35
00450 06FB B7 42               STA     FLAG
00451 06FD 1C 42               BSET    6,FLAG    Set accumulated charge display request flag
00452 06FF 81                  RTS
00453                   *
00454                   ************************************************
00455                   *                                                *
00456                   *       NAME    : SET ELAPSED TIME DISPLAY       *
00457                   *                 REQUEST FLAG (MOD30)           *
00458                   *                                                *
00459                   ************************************************
00460                   *                                                *
00461                   *       ENTRY   : NOTHING                        *
00462                   *       RETURNS : FLAG                           *
00463                   *                                                *
00464                   ************************************************
00465 0700 B6 42       MOD30   LDA     FLAG      Clear display
00466 0702 A4 35               AND     #$35
00467 0704 B7 42               STA     FLAG
00468 0706 16 42               BSET    3,FLAG    Set elapsed time display request flag
00469 0708 81                  RTS
00470                   *
00471                   ************************************************
00472                   *                                                *
00473                   *       NAME    : ENTER A TELEPHONE NUMBER       *
00474                   *                           (MOD40)              *
00475                   *                                                *
00476                   ************************************************
00477                   *                                                *
00478                   *       ENTRY   : TRMFLG, LCDM, WAIT3            *
00479                   *       RETURNS : LCDM, CURADP, FLAG             *
00480                   *                                                *
00481                   ************************************************
00482 0709 08 B0 2C MOD40  BRSET  4,WAIT3,MOD42  Output tone?
00483 070C B6 43           LDA     TRMFLG    Check process flag
00484 070E 26 1E           BNE     MOD41
00485 0710 B6 42           LDA     FLAG      If 0, clear display
00486 0712 A4 35           AND     #$35
00487 0714 B7 42           STA     FLAG
00488 0716 A6 01           LDA     #$1
00489 0718 B7 4B           STA     CURADP
00490 071A CD 1966         JSR     LCDMCR
00491 071D CD 110E         JSR     LCD
00492 0720 15 42           BCLR    2,FLAG    Define search direction flag
00493 0722 3F 45           CLR     DTDP
00494 0724 3F 46           CLR     DTDP+1
00495 0726 CD 12C1         JSR     KENSAK    Search empty area
```

```
00496 0729 CD 1966          JSR     LCDMCR
00497 072C 20 0A            BRA     MOD42
00498 072E A1 01      MOD41 CMP     #$1       If 1, enter data
00499 0730 26 06            BNE     MOD42
00500 0732 CD 139A          JSR     TOROKU
00501 0735 CD 18DA          JSR     END
00502 0738 81         MOD42 RTS
00503                       *
00504                       ************************************************
00505                       *                                              *
00506                       *       NAME    : REVIEW / MODIFY / DELETE     *
00507                       *                   A TELEPHONE NUMBER (MOD50) *
00508                       *                                              *
00509                       ************************************************
00510                       *                                              *
00511                       *       ENTRY   : WAIT3, TRMFLG, LCDM,KEYDAT   *
00512                       *       RETURNS : LCDM, CURADP, BZRFLG, FLAG   *
00513                       *                 LCDMP                        *
00514                       *                                              *
00515                       ************************************************
00516 0739 09 B0 03 MOD50 BRCLR 4,WAIT3,MOD51  Output tone?
00517 073C CC 0800          JMP     MOD7F     If so, ignore this routine
00518 073F B6 43      MOD51 LDA     TRMFLG    Check process flag
00519 0741 26 16            BNE     MOD52
00520 0743 B6 42            LDA     FLAG      If 0, clear display
00521 0745 A4 35            AND     #$35
00522 0747 B7 42            STA     FLAG
00523 0749 A6 01            LDA     #$1
00524 074B B7 4B            STA     CURADP
00525 074D CD 1966          JSR     LCDMCR
00526 0750 CD 110E          JSR     LCD
00527 0753 CD 1480          JSR     CLEAR
00528 0756 CC 0800          JMP     MOD7F
00529 0759 A1 01      MOD52 CMP     #$1       If 1, refer to data
00530 075B 26 1E            BNE     MOD53
00531 075D 1E 4E            BSET    7,BZRFLG
00532 075F 3F 45            CLR     DTDP
00533 0761 3F 46            CLR     DTDP+1
00534 0763 A6 01            LDA     #$1
00535 0765 B7 4B            STA     CURADP
00536 0767 15 42            BCLR    2,FLAG
00537 0769 CD 19CB          JSR     MOVE1
00538 076C CD 12C1          JSR     KENSAK
00539 076F 3C 43            INC     TRMFLG
00540 0771 A6 01            LDA     #$1
00541 0773 B7 4B            STA     CURADP
00542 0775 4A               DEC     A
00543 0776 B7 40            STA     LCDMP
00544 0778 CC 0800          JMP     MOD7F
00545 077B A1 02      MOD53 CMP     #$2       If 2, define data table start address
00546 077D 26 31            BNE     MOD6
00547 077F B6 58            LDA     KEYDAT
00548 0781 A1 2A            CMP     #$2A
00549 0783 26 10            BNE     MOD54
00550 0785 15 42            BCLR    2,FLAG
```

```
00551 0787 B6 51          LDA    DTADS+1
00552 0789 AB 1D          ADD    #29
00553 078B B7 46          STA    DTDP+1
00554 078D B6 50          LDA    DTADS
00555 078F A9 00          ADC    #0
00556 0791 B7 45          STA    DTDP
00557 0793 20 15          BRA    MOD56
00558 0795 A1 29    MOD54 CMP    #$29
00559 0797 27 03          BEQ    MOD55
00560 0799 CC 0800        JMP    MOD7F
00561 079C 14 42    MOD55 BSET   2,FLAG
00562 079E B6 51          LDA    DTADS+1
00563 07A0 A0 1D          SUB    #29
00564 07A2 B7 46          STA    DTDP+1
00565 07A4 B6 50          LDA    DTADS
00566 07A6 A2 00          SBC    #0
00567 07A8 B7 45          STA    DTDP
00568 07AA CD 12C1  MOD56 JSR    KENSAK
00569 07AD CC 0800        JMP    MOD7F
00570 07B0 B6 41    MOD6  LDA    MODFLG   Check mode flag
00571 07B2 A1 06          CMP    #$6
00572 07B4 27 0A          BEQ    MOD61    If 3, jump to MOD61
00573 07B6 A1 07          CMP    #$7
00574 07B8 27 2C          BEQ    MOD7     If 4, jump to MOD7
00575 07BA CD 18DA        JSR    END
00576 07BD CC 0800        JMP    MOD7F
00577 07C0 B6 43    MOD61 LDA    TRMFLG   Test if process flag is 3
00578 07C2 A1 03          CMP    #$3
00579 07C4 26 08          BNE    MOD62
00580 07C6 CD 110E        JSR    LCD
00581 07C9 1F 4E          BCLR   7,BZRFLG
00582 07CB CC 0800        JMP    MOD7F
00583 07CE A1 04    MOD62 CMP    #$4      Test if process flag is 4
00584 07D0 27 03          BEQ    MOD63
00585 07D2 CC 0800        JMP    MOD7F
00586 07D5 B6 50    MOD63 LDA    DTADS    If 4, enter modified data
00587 07D7 B7 45          STA    DTDP
00588 07D9 B6 51          LDA    DTADS+1
00589 07DB B7 46          STA    DTDP+1
00590 07DD CD 139A        JSR    TOROKU
00591 07E0 CD 18DA        JSR    END
00592 07E3 CC 0800        JMP    MOD7F
00593 07E6 B6 43    MOD7  LDA    TRMFLG   Test if process flag is 3
00594 07E8 A1 03          CMP    #$3
00595 07EA 27 03          BEQ    MOD71
00596 07EC CC 0800        JMP    MOD7F
00597 07EF CD 1480  MOD71 JSR    CLEAR    Delete data
00598 07F2 B6 50          LDA    DTADS
00599 07F4 B7 45          STA    DTDP
00600 07F6 B6 51          LDA    DTADS+1
00601 07F8 B7 46          STA    DTDP+1
00602 07FA CD 139A        JSR    TOROKU
00603 07FD CD 18DA        JSR    END
00604 0800 81       MOD7F RTS
00605               *
```

```
00606                ************************************************
00607                *                                              *
00608                *       NAME    : PREPARE FOR SPEED DIALING    *
00609                *                                 (MOD80)      *
00610                *                                              *
00611                ************************************************
00612                *                                              *
00613                *       ENTRY   : WAIT3, TRMFLG, LCDM          *
00614                *       RETURNS : LCDM, CURADP, BZR, LCDMP     *
00615                *                 WAIT3                        *
00616                *                                              *
00617                ************************************************
00618 0801 08 B0 47 MOD80  BRSET  4,WAIT3,MOD85  Output tone?
00619 0804 B6 43           LDA    TRMFLG
00620 0806 26 2E           BNE    MOD83
00621 0808 B6 42           LDA    FLAG      If so, clear display
00622 080A A4 35           AND    #$35
00623 080C B7 42           STA    FLAG
00624 080E CD 1966         JSR    LCDMCR
00625 0811 CD 110E         JSR    LCD
00626 0814 AE 04           LDX    #$4
00627 0816 D6 1A3F MOD82   LDA    NO1-1,X
00628 0819 E7 73           STA    LCDM+2,X Display "NO?  - "
00629 081B 5A              DEC    X
00630 081C 26 F8           BNE    MOD82
00631 081E A6 09           LDA    #9
00632 0820 B7 40           STA    LCDMP
00633 0822 A6 0A           LDA    #$A
00634 0824 B7 4B           STA    CURADP
00635 0826 CD 110E         JSR    LCD
00636 0829 BE 40           LDX    LCDMP
00637 082B 5A              DEC    X         Define cursor position
00638 082C BF 40           STX    LCDMP
00639 082E BE 4B           LDX    CURADP
00640 0830 5A              DEC    X
00641 0831 BF 4B           STX    CURADP
00642 0833 CC 084B         JMP    MOD85
00643 0836 A1 01    MOD83  CMP    #$1
00644 0838 26 11           BNE    MOD85
00645 083A AE 08           LDX    #$8
00646 083C D6 1A43 MOD84   LDA    NO2-1,X  Display "NO."
00647 083F E7 71           STA    LCDM,X
00648 0841 5A              DEC    X
00649 0842 26 F8           BNE    MOD84
00650 0844 CD 1409         JSR    TAN      Review speed dial no.
00651 0847 12 4F           BSET   1,BZR    Set tone output request flag
00652 0849 18 B0           BSET   4,WAIT3  Set tone outputting flag
00653 084B 81       MOD85  RTS
00654                *
00655                ************************************************
00656                *                                              *
00657                *       NAME    : ENTER A SPEED DIAL NUMBER    *
00658                *                                              *
00659                ************************************************
00660                *                                              *
```

```
00661                    *       ENTRY   : WAIT3, TRMFLG, LCDM           *
00662                    *       RETURNS : LCDM, CURADP, LCDMP           *
00663                    *                                               *
00664                    ************************************************
00665 084C 08 B0 62 MOD90  BRSET  4,WAIT3,MOD94  Output tone?
00666 084F B6 43           LDA    TRMFLG
00667 0851 26 23           BNE    MOD91
00668 0853 B6 42           LDA    FLAG      Clear display
00669 0855 A4 35           AND    #$35
00670 0857 B7 42           STA    FLAG
00671 0859 CD 1966         JSR    LCDMCR
00672 085C CD 110E         JSR    LCD
00673 085F A6 09           LDA    #9        Define cursor position
00674 0861 B7 40           STA    LCDMP
00675 0863 A6 0A           LDA    #$A
00676 0865 B7 4B           STA    CURADP    Cursor ON
00677 0867 CD 110E         JSR    LCD
00678 086A BE 40           LDX    LCDMP
00679 086C 5A              DEC    X
00680 086D BF 40           STX    LCDMP
00681 086F A6 0A           LDA    #$A
00682 0871 B7 4B           STA    CURADP
00683 0873 CC 08B1         JMP    MOD94
00684 0876 A1 01     MOD91 CMP    #$1
00685 0878 26 14           BNE    MOD92
00686 087A A6 2A           LDA    #$2A
00687 087C B7 78           STA    LCDM+7
00688 087E B7 7C           STA    LCDM+11   Display " *  *  - "
00689 0880 A6 15           LDA    #$15
00690 0882 B7 4B           STA    CURADP
00691 0884 A6 14           LDA    #$14
00692 0886 B7 40           STA    LCDMP
00693 0888 CD 110E         JSR    LCD
00694 088B CC 08B1         JMP    MOD94
00695 088E A1 02     MOD92 CMP    #$2
00696 0890 26 1F           BNE    MOD94
00697 0892 B6 7A           LDA    LCDM+9    Load speed dial no.
00698 0894 A4 0F           AND    #$0F
00699 0896 B7 46           STA    DTDP+1    Define address to be entered
00700 0898 BE 46           LDX    DTDP+1
00701 089A A6 17           LDA    #$17
00702 089C B7 45           STA    DTDP
00703 089E A6 F7           LDA    #$F7
00704 08A0 B7 46           STA    DTDP+1
00705 08A2 AB 09     MOD93 ADD    #$9
00706 08A4 B7 46           STA    DTDP+1
00707 08A6 5A              DEC    X
00708 08A7 A3 FF           CPX    #$FF
00709 08A9 26 F7           BNE    MOD93
00710 08AB CD 13BD         JSR    MOD95     Enter data
00711 08AE CD 18DA         JSR    END
00712 08B1 81        MOD94 RTS
00713                    *
00714                    ************************************************
00715                    *                                               *
```

```
00716                   *       NAME    : PREPARE FOR REDIALING (MOD100)*
00717                   *                                               *
00718                   *************************************************
00719                   *                                               *
00720                   *       ENTRY   : WAIT3, MEMO                   *
00721                   *       RETURNS : LCDM, BZR, WAIT3              *
00722                   *                                               *
00723                   *************************************************
00724 08B2 08 B0 2A MOD100 BRSET  4,WAIT3,MOD103  Output tone?
00725 08B5 B6 42            LDA     FLAG      Clear display
00726 08B7 A4 35            AND     #$35
00727 08B9 B7 42            STA     FLAG
00728 08BB CD 1966          JSR     LCDMCR
00729 08BE CD 110E          JSR     LCD
00730 08C1 AE 0A            LDX     #10
00731 08C3 D6 1A4B  MOD101  LDA     DIAND1-1,X  Display "REDIAL"
00732 08C6 E7 71            STA     LCDM,X
00733 08C8 5A               DEC     X
00734 08C9 26 F8            BNE     MOD101
00735 08CB 5F               CLR     X
00736 08CC D6 0100  MOD102  LDA     MEMO,X    Display number
00737 08CF E7 85            STA     LCDM+20,X
00738 08D1 E7 99            STA     DIALNO,X
00739 08D3 5C               INC     X
00740 08D4 A3 14            CPX     #20
00741 08D6 26 F4            BNE     MOD102
00742 08D8 CD 110E          JSR     LCD
00743 08DB 12 4F            BSET    1,BZR     Set tone output request flag
00744 08DD 18 B0            BSET    4,WAIT3   Set tone outputting flag
00745 08DF 81       MOD103  RTS
00746                   *
00747                   *************************************************
00748                   *                                               *
00749                   *       NAME    : PREPARE FOR RETRYING (MOD110) *
00750                   *                                               *
00751                   *************************************************
00752                   *                                               *
00753                   *       ENTRY   : WAIT3, MEMO                   *
00754                   *       RETURNS : LCDM, BZRFLG, WAIT3           *
00755                   *                                               *
00756                   *************************************************
00757 08E0 08 B0 2A MOD110 BRSET  4,WAIT3,MOD113  Output tone?
00758 08E3 B6 42            LDA     FLAG      Clear display
00759 08E5 A4 35            AND     #$35
00760 08E7 B7 42            STA     FLAG
00761 08E9 CD 1966          JSR     LCDMCR
00762 08EC CD 110E          JSR     LCD
00763 08EF AE 0E            LDX     #14
00764 08F1 D6 1A55  MOD111  LDA     DIAND2-1,X  Display "RETRY"
00765 08F4 E7 71            STA     LCDM,X
00766 08F6 5A               DEC     X
00767 08F7 26 F8            BNE     MOD111
00768 08F9 5F               CLR     X
00769 08FA D6 0100  MOD112  LDA     MEMO,X    Display number
00770 08FD E7 85            STA     LCDM+20,X
```

```
00771 08FF E7 99           STA    DIALNO,X
00772 0901 5C              INC    X
00773 0902 A3 14           CPX    #20
00774 0904 26 F4           BNE    MOD112
00775 0906 CD 110E         JSR    LCD
00776 0909 18 B0           BSET   4,WAIT3  Set tone outputting flag
00777 090B 10 4E           BSET   0,BZRFLG Set retry request flag
00778 090D 81        MOD113 RTS
00779                *
00780                ************************************************
00781                *                                              *
00782                *      NAME    : PREPARE FOR NORMAL DIALING    *
00783                *                                (MOD120)      *
00784                *                                              *
00785                ************************************************
00786                *                                              *
00787                *      ENTRY   : WAIT3                         *
00788                *      RETURNS : LCDM, CURADP, MEMO, BZR       *
00789                *                                              *
00790                ************************************************
00791 090E 08 B0 17 MOD120 BRSET  4,WAIT3,MOD121  Output tone?
00792 0911 B6 42           LDA    FLAG     Clear display
00793 0913 A4 35           AND    #$35
00794 0915 B7 42           STA    FLAG
00795 0917 CD 1966         JSR    LCDMCR
00796 091A CD 110E         JSR    LCD
00797 091D A6 02           LDA    #$2
00798 091F B7 40           STA    LCDMP    Set cursor position
00799 0921 3F 4B           CLR    CURADP
00800 0923 CD 1476         JSR    MEMCLR   Clear RAM for redialing
00801 0926 12 4F           BSET   1,BZR    Set tone output request flag
00802 0928 81        MOD121 RTS
00803                *
00804                ************************************************
00805                *                                              *
00806                *      NAME    : SET CALENDAR & TIME (MOD130)  *
00807                *                                              *
00808                ************************************************
00809                *                                              *
00810                *      ENTRY   : WAIT3, TRMFLG, LCDM           *
00811                *      RETURNS : FLAG, LCDM, CALNDR            *
00812                *                                              *
00813                ************************************************
00814 0929 08 B0 28 MOD130 BRSET  4,WAIT3,MOD134  Output tone?
00815 092C B6 43           LDA    TRMFLG   Check process flag
00816 092E 26 10           BNE    MOD132
00817 0930 B6 42     MOD131 LDA    FLAG     If 0, clear display
00818 0932 A4 35           AND    #$35
00819 0934 B7 42           STA    FLAG
00820 0936 CD 1491         JSR    SHUCAL
00821 0939 CD 110E         JSR    LCD
00822 093C 3C 43           INC    TRMFLG
00823 093E 20 14           BRA    MOD134
00824 0940 A1 01     MOD132 CMP    #$1
00825 0942 26 02           BNE    MOD133
```

```
00826 0944 20 0E           BRA    MOD134
00827 0946 A1 02    MOD133 CMP    #$2       If 2, set calendar & time
00828 0948 26 07           BNE    MOD136
00829 094A CD 149F         JSR    CALSHU
00830 094D B6 43           LDA    TRMFLG
00831 094F 27 DF           BEQ    MOD131
00832 0951 CD 18DA  MOD136 JSR    END
00833 0954 81       MOD134 RTS
00834                      *
00835                      ************************************************
00836                      *                                              *
00837                      *       NAME    : SET RATE FOR CHARGE (MOD140) *
00838                      *                                              *
00839                      ************************************************
00840                      *                                              *
00841                      *       ENTRY   : TRMFLG, LCDM                 *
00842                      *       RETURNS : FLAG, LCDM, ADDRAT, SEC      *
00843                      *                                              *
00844                      ************************************************
00845 0955 B6 43    MOD140 LDA    TRMFLG   Check process flag
00846 0957 26 11           BNE    MOD142
00847 0959 B6 42    MOD141 LDA    FLAG
00848 095B A4 35           AND    #$35
00849 095D B7 42           STA    FLAG     Clear display
00850 095F CD 16EA         JSR    CRGSET
00851 0962 CD 110E         JSR    LCD      Diaplay current charge
00852 0965 3C 43           INC    TRMFLG
00853 0967 CC 097F         JMP    MOD144
00854 096A A1 01    MOD142 CMP    #$1
00855 096C 26 03           BNE    MOD143
00856 096E CC 097F         JMP    MOD144
00857 0971 A1 02    MOD143 CMP    #$2
00858 0973 26 07           BNE    MOD145
00859 0975 CD 172C         JSR    RATSET   Enter modified charge
00860 0978 B6 43           LDA    TRMFLG
00861 097A 27 DD           BEQ    MOD141
00862 097C CD 18DA  MOD145 JSR    END
00863 097F 81       MOD144 RTS
00864                      *
00865                      ************************************************
00866                      *                                              *
00867                      *       NAME    : CLEAR ACCUMULATED CHARGE     *
00868                      *                          (MOD150)            *
00869                      *                                              *
00870                      ************************************************
00871                      *                                              *
00872                      *       ENTRY   : NOTHING                      *
00873                      *       RETURNS : TOTAL                        *
00874                      *                                              *
00875                      ************************************************
00876 0980 AE 04    MOD150 LDX    #4
00877 0982 4F       MOD151 CLR    A        Clear TOTAL
00878 0983 D7 0130         STA    TOTAL-1,X
00879 0986 5A              DEC    X
00880 0987 26 F9           BNE    MOD151
```

```
00881 0989 81              RTS
00882                 *
00883                 ************************************************
00884                 *                                              *
00885                 *       NAME    :  END  (MOD160)               *
00886                 *                                              *
00887                 ************************************************
00888                 *                                              *
00889                 *       ENTRY   : NOTHING                      *
00890                 *       RETURNS : INTERNAL RAM                 *
00891                 *                                              *
00892                 ************************************************
00893 098A CD 18DA  MOD160 JSR    END       Clear RAM
00894 098D 81              RTS
00895                 *
00896                 *****  PROGRAM MODULE FOR CHECK INPUT DATA  ******
00897                 *
00898                 ************************************************
00899                 *                                              *
00900                 *       NAME    : CHECK FOR NORMAL DIALING     *
00901                 *                                 (CHEK1)      *
00902                 *                                              *
00903                 ************************************************
00904                 *                                              *
00905                 *       ENTRY   : KEYSET, WAIT3, WAIT5, LCDMP  *
00906                 *                 DIAP1                        *
00907                 *       RETURNS : DIALNO, DIADAT, MEMO, BZR    *
00908                 *                 WAIT3, LCDM, DIAP1, LCDMP    *
00909                 *                 WAIT5, CURADP, ERFLG         *
00910                 *                                              *
00911                 ************************************************
00912 098E B6 54     CHEK1  LDA    KEYSET    Key data = " * " ?
00913 0990 A1 2A            CMP    #$2A
00914 0992 27 11            BEQ    CHK12     Key data = " # " ?
00915 0994 A1 23            CMP    #$23
00916 0996 27 0D            BEQ    CHK12     Key data = 0 - 9 ?
00917 0998 A1 39            CMP    #$39
00918 099A 22 04            BHI    CHK11
00919 099C A1 30            CMP    #$30
00920 099E 24 05            BCC    CHK12
00921 09A0 10 BE     CHK11  BSET   0,ERFLG   If not, buzzing
00922 09A2 CC 0A1F          JMP    CHK16
00923 09A5 0A B0 21  CHK12  BRSET  5,WAIT3,CHK13  Prepared for tone output
00924 09A8 A6 02            LDA    #$2
00925 09AA B7 40            STA    LCDMP
00926 09AC 3F 4B            CLR    CURADP
00927 09AE CD 1476          JSR    MEMCLR    Clear RAM for redialing
00928 09B1 1A B0            BSET   5,WAIT3
00929 09B3 BE 40            LDX    LCDMP
00930 09B5 B6 54            LDA    KEYSET
00931 09B7 E7 71            STA    LCDM,X
00932 09B9 E7 98            STA    DIALNO-1,X  Store key data in DIALNO
00933 09BB E7 FF            STA    MEMO-1,X
00934 09BD 3C 40            INC    LCDMP
00935 09BF CD 110E          JSR    LCD
```

```
00936 09C2 12 4F           BSET   1,BZR    Set tone output request flag
00937 09C4 18 B0           BSET   4,WAIT3  Set tone output flag
00938 09C6 CC 0A1F         JMP    CHK16
00939 09C9 08 B2 3F CHK13  BRSET  4,WAIT5,CHK15  Display full?
00940 09CC 06 B2 18        BRSET  3,WAIT5,CHK14  21st digit?
00941 09CF 1A B0           BSET   5,WAIT3
00942 09D1 BE 40           LDX    LCDMP
00943 09D3 A3 16           CPX    #22
00944 09D5 27 10           BEQ    CHK14
00945 09D7 B6 54           LDA    KEYSET
00946 09D9 E7 71           STA    LCDM,X
00947 09DB E7 98           STA    DIALNO-1,X  Store key data in DIALNO
00948 09DD E7 FF           STA    MEMO-1,X
00949 09DF 3C 40           INC    LCDMP
00950 09E1 CD 110E         JSR    LCD      Dispay key data
00951 09E4 CC 0A1F         JMP    CHK16
00952 09E7 16 B2    CHK14  BSET   3,WAIT5
00953 09E9 B6 54           LDA    KEYSET
00954 09EB BE BC           LDX    DIAP1
00955 09ED E7 B7           STA    DIADAT,X Store key data in DIADAT
00956 09EF BE 40           LDX    LCDMP
00957 09F1 E7 71           STA    LCDM,X
00958 09F3 CD 110E         JSR    LCD      Display key data
00959 09F6 3C BC           INC    DIAP1
00960 09F8 B6 BC           LDA    DIAP1
00961 09FA A1 04           CMP    #4       Buffer full?
00962 09FC 26 02           BNE    CHK17
00963 09FE 3F BC           CLR    DIAP1    If so, clear buffer pointer
00964 0A00 3C 40    CHK17  INC    LCDMP    If not, increment buffer pointer
00965 0A02 B6 40           LDA    LCDMP
00966 0A04 A1 28           CMP    #40      Display full?
00967 0A06 27 03           BEQ    CHK15
00968 0A08 CC 0A1F         JMP    CHK16
00969 0A0B 17 B2    CHK15  BCLR   3,WAIT5  If so, define flag
00970 0A0D 18 B2           BSET   4,WAIT5
00971 0A0F B6 54           LDA    KEYSET
00972 0A11 BE BC           LDX    DIAP1
00973 0A13 E7 B7           STA    DIADAT,X Store key data in DIADAT
00974 0A15 3C BC           INC    DIAP1
00975 0A17 B6 BC           LDA    DIAP1
00976 0A19 A1 04           CMP    #4       Buffer full?
00977 0A1B 26 02           BNE    CHK16
00978 0A1D 3F BC           CLR    DIAP1    If so, clear buffer pointer
00979 0A1F 81       CHK16  RTS
00980                      *
00981                      ************************************************
00982                      *                                              *
00983                      *      NAME    : CHECK FOR SPEED DIALING       *
00984                      *                              (CHEK2)         *
00985                      *                                              *
00986                      ************************************************
00987                      *                                              *
00988                      *      ENTRY   : KEYSET, TRMFLG                *
00989                      *      RETURNS : CURADP, LCDM, ERFLG, BZR      *
00990                      *                LCDMP                         *
```

```
00991                   *                                               *
00992                   ************************************************
00993 0A20 B6 43  CHEK2  LDA     TRMFLG   Check process flag
00994 0A22 26 22         BNE     CHK23
00995 0A24 A6 09         LDA     #$9      If 0, set cursor position
00996 0A26 B7 40         STA     LCDMP
00997 0A28 A6 0A         LDA     #$A
00998 0A2A B7 4B         STA     CURADP
00999 0A2C B6 54         LDA     KEYSET
01000 0A2E A1 39         CMP     #$39
01001 0A30 22 12         BHI     CHK22    Key data = 0 - 9 ?
01002 0A32 A1 30         CMP     #$30
01003 0A34 24 02         BCC     CHK21
01004 0A36 20 0C         BRA     CHK22
01005 0A38 B6 54  CHK21  LDA     KEYSET   If so, display number
01006 0A3A B7 7A         STA     LCDM+9
01007 0A3C CD 110E       JSR     LCD
01008 0A3F CD 0587       JSR     MODE2
01009 0A42 20 02         BRA     CHK23
01010 0A44 10 BE  CHK22  BSET    0,ERFLG  If not, buzzing
01011 0A46 81     CHK23  RTS
01012                   *
01013                   ************************************************
01014                   *                                               *
01015                   *       NAME    : CHECK FOR ENTERING SPEED DIAL  *
01016                   *                        NUMBER (CHEK3)         *
01017                   *                                               *
01018                   ************************************************
01019                   *                                               *
01020                   *       ENTRY   : KEYSET, TRMFLG, LCDMP, CURADP *
01021                   *       RETURNS : LCDM, ERFLG, CURADP, LCDMP    *
01022                   *                                               *
01023                   ************************************************
01024 0A47 B6 43  CHEK3  LDA     TRMFLG   Check process flag
01025 0A49 26 0E         BNE     CHK31
01026 0A4B B6 54         LDA     KEYSET
01027 0A4D B7 7A         STA     LCDM+9
01028 0A4F A1 39         CMP     #$39     Speed dial no.= 0-9?
01029 0A51 22 36         BHI     CHK35
01030 0A53 A1 30         CMP     #$30
01031 0A55 24 2D         BCC     CHK34
01032 0A57 20 30         BRA     CHK35
01033 0A59 B6 54  CHK31  LDA     KEYSET   Key data = " * " ?
01034 0A5B A1 2A         CMP     #$2A
01035 0A5D 27 0E         BEQ     CHK32
01036 0A5F A1 23         CMP     #$23     Key data = " # " ?
01037 0A61 27 0A         BEQ     CHK32
01038 0A63 A1 39         CMP     #$39     Key data = $30 - $39 ?
01039 0A65 22 22         BHI     CHK35
01040 0A67 A1 30         CMP     #$30
01041 0A69 24 02         BCC     CHK32
01042 0A6B 20 1C         BRA     CHK35
01043 0A6D 3C 4B  CHK32  INC     CURADP
01044 0A6F BE 40         LDX     LCDMP
01045 0A71 B6 54         LDA     KEYSET
```

```
01046 0A73 E7 71          STA    LCDM,X   Store key data in LCDM
01047 0A75 B6 4B          LDA    CURADP
01048 0A77 A1 27          CMP    #39      Cursor position > 39 ?
01049 0A79 25 07          BCS    CHK33
01050 0A7B B7 4B          STA    CURADP   If so, stop cursor position
01051 0A7D 4A             DEC    A
01052 0A7E B7 40          STA    LCDMP
01053 0A80 20 02          BRA    CHK34
01054 0A82 3C 40    CHK33 INC    LCDMP    If not, increment cursor position.
01055 0A84 CD 110E  CHK34 JSR    LCD
01056 0A87 20 02          BRA    CHK36
01057 0A89 10 BE    CHK35 BSET   0,ERFLG  If invalid data input, buzzing
01058 0A8B 81       CHK36 RTS
01059                     *
01060                     ************************************************
01061                     *                                              *
01062                     *      NAME    : CHECK FOR TELEPHONE NUMBERS   *
01063                     *                          (CHEK4)             *
01064                     *                                              *
01065                     ************************************************
01066                     *                                              *
01067                     *      ENTRY   : KEYSET, LCDMP, MODFLG         *
01068                     *      RETURNS : ERFLG, LCDM, LCDMP, CURADP    *
01069                     *                                              *
01070                     ************************************************
01071 0A8C B6 40    CHEK4 LDA    LCDMP    Cursor positon > Line 20?
01072 0A8E A1 13          CMP    #19
01073 0A90 22 0C          BHI    CHK42
01074 0A92 BE 40          LDX    LCDMP    Process for Line 1-20
01075 0A94 B6 54          LDA    KEYSET
01076 0A96 E7 71          STA    LCDM,X   Store key data in LCDM
01077 0A98 3C 40          INC    LCDMP
01078 0A9A 3C 4B          INC    CURADP
01079 0A9C 20 37          BRA    CHK45
01080 0A9E B6 54    CHK42 LDA    KEYSET   Process for Line 21-38
01081 0AA0 A1 2A          CMP    #$2A     Key data = " * " ?
01082 0AA2 27 1A          BEQ    CHK43
01083 0AA4 A1 23          CMP    #$23     Key data = " # " ?
01084 0AA6 27 16          BEQ    CHK43
01085 0AA8 A1 39          CMP    #$39     Key data = 0 - 9 ?
01086 0AAA 22 2E          BHI    CHK46
01087 0AAC A1 30          CMP    #$30
01088 0AAE 24 0E          BCC    CHK43
01089 0AB0 B6 41          LDA    MODFLG
01090 0AB2 A1 06          CMP    #$6      Modify data?
01091 0AB4 26 24          BNE    CHK46
01092 0AB6 B6 54          LDA    KEYSET
01093 0AB8 A1 20          CMP    #$20     If so, SP is valid data
01094 0ABA 27 02          BEQ    CHK43
01095 0ABC 20 1C          BRA    CHK46
01096 0ABE BE 40    CHK43 LDX    LCDMP
01097 0AC0 B6 54          LDA    KEYSET
01098 0AC2 E7 71          STA    LCDM,X
01099 0AC4 B6 4B          LDA    CURADP
01100 0AC6 A1 27          CMP    #39      Cursor position > Line 39?
```

```
01101 0AC8 25 07           BCS    CHK44
01102 0ACA B7 4B           STA    CURADP
01103 0ACC 4A              DEC    A
01104 0ACD B7 40           STA    LCDMP
01105 0ACF 20 04           BRA    CHK45
01106 0AD1 3C 40     CHK44 INC    LCDMP    Increment cursror position
01107 0AD3 3C 4B           INC    CURADP
01108 0AD5 CD 110E   CHK45 JSR    LCD      Display figures
01109 0AD8 20 02           BRA    CHK47
01110 0ADA 10 BE     CHK46 BSET   0,ERFLG  If invalid data input, buzzer
01111 0ADC 81        CHK47 RTS
01112                      *
01113                      ************************************************
01114                      *                                              *
01115                      *      NAME    : CHECK FOR CALENDAR & TIME,    *
01116                      *                        CHARGE (CHEK5)        *
01117                      *                                              *
01118                      ************************************************
01119                      *                                              *
01120                      *      ENTRY   : KEYSET, TRMFLG, MODFLG        *
01121                      *      RETURNS : ERFLG, LCDM, LCDMP, CURADP    *
01122                      *                                              *
01123                      ************************************************
01124 0ADD B6 54     CHEK5 LDA    KEYSET   Key data = 0 - 9?
01125 0ADF A1 39           CMP    #$39
01126 0AE1 22 2C           BHI    CHK53
01127 0AE3 A1 30           CMP    #$30
01128 0AE5 24 02           BCC    CHK51
01129 0AE7 20 26           BRA    CHK53
01130 0AE9 BE 40     CHK51 LDX    LCDMP    Store key data in LCDM
01131 0AEB B6 54           LDA    KEYSET
01132 0AED E7 71           STA    LCDM,X
01133 0AEF B6 43           LDA    TRMFLG
01134 0AF1 A1 01           CMP    #1
01135 0AF3 26 1C           BNE    CHK5F
01136 0AF5 3C 40           INC    LCDMP
01137 0AF7 3C 4B           INC    CURADP
01138 0AF9 B6 41           LDA    MODFLG
01139 0AFB A1 0D           CMP    #$D
01140 0AFD 26 08           BNE    CHK52
01141 0AFF CD 17EC         JSR    CURCAL   Define cursor position for
01142 0B02 CD 110E         JSR    LCD      -setting calendar & time
01143 0B05 20 0A           BRA    CHK5F
01144 0B07 CD 17BB   CHK52 JSR    CURRAT   Define cursor position for
01145 0B0A CD 110E         JSR    LCD      -setting charge
01146 0B0D 20 02           BRA    CHK5F
01147 0B0F 10 BE     CHK53 BSET   0,ERFLG  Set error process request flag
01148 0B11 81        CHK5F RTS
01149                      *
01150                      ************************************************
01151                      *                                              *
01152                      *      NAME    : MOVE CURSOR TO RIGHT OR LEFT  *
01153                      *                              (CURLR)         *
01154                      *                                              *
01155                      ************************************************
```

```
01156                    *                                                 *
01157                    *       ENTRY   : KEYDAT, FLAG, BZRFLG, MODFLG    *
01158                    *                 TRMFLG, WAIT3, LCDMP, CURADP    *
01159                    *       RETURNS : LCDM, LCDMP, CURADP             *
01160                    *                                                 *
01161                    ***************************************************
01162 0B12 08 B0 5C CURLR  BRSET  4,WAIT3,CURLRF  Output tone?
01163 0B15 B6 41           LDA    MODFLG
01164 0B17 A1 0D           CMP    #$D       Set calendar & time?
01165 0B19 27 2B           BEQ    CURLR4
01166 0B1B A1 0E           CMP    #$E       Set charge ?
01167 0B1D 27 3C           BEQ    CURLR7
01168 0B1F B6 58           LDA    KEYDAT    If not, move cursor normally
01169 0B21 A1 29           CMP    #$29
01170 0B23 26 0B           BNE    CURLR2
01171 0B25 02 42 49        BRSET  1,FLAG,CURLRF
01172 0B28 0E 4E 14        BRSET  7,BZRFLG,CURLR3
01173 0B2B CD 1911         JSR    LFCRSR    Move cursor to left?
01174 0B2E 20 41           BRA    CURLRF
01175 0B30 A1 2A     CURLR2 CMP    #$2A
01176 0B32 26 3D           BNE    CURLRF
01177 0B34 02 42 3A        BRSET  1,FLAG,CURLRF
01178 0B37 0E 4E 05        BRSET  7,BZRFLG,CURLR3
01179 0B3A CD 1934         JSR    RTCRSR
01180 0B3D 20 32           BRA    CURLRF
01181 0B3F 3A 43     CURLR3 DEC    TRMFLG
01182 0B41 CD 0587         JSR    MODE2     Search data
01183 0B44 20 2B           BRA    CURLRF
01184 0B46 B6 58     CURLR4 LDA    KEYDAT    Move cursor for setting calendar & time
01185 0B48 A1 29           CMP    #$29
01186 0B4A 26 06           BNE    CURLR5
01187 0B4C 3A 40           DEC    LCDMP     Move cursor to left
01188 0B4E 3A 4B           DEC    CURADP
01189 0B50 20 04           BRA    CURLR6
01190 0B52 3C 40     CURLR5 INC    LCDMP     Move cursor to right
01191 0B54 3C 4B           INC    CURADP
01192 0B56 CD 17EC   CURLR6 JSR    CURCAL
01193 0B59 20 13           BRA    CURLRA
01194 0B5B B6 58     CURLR7 LDA    KEYDAT    Move cursor for setting charge
01195 0B5D A1 29           CMP    #$29
01196 0B5F 26 06           BNE    CURLR8
01197 0B61 3A 40           DEC    LCDMP     Move cursor to left
01198 0B63 3A 4B           DEC    CURADP
01199 0B65 20 04           BRA    CURLR9
01200 0B67 3C 40     CURLR8 INC    LCDMP     Move cursor to right
01201 0B69 3C 4B           INC    CURADP
01202 0B6B CD 17BB   CURLR9 JSR    CURRAT
01203 0B6E CD 110E   CURLRA JSR    LCD
01204 0B71 81        CURLRF RTS
01205                    *
01206                    ***************************************************
01207                    *                                                 *
01208                    *    NAME    : MOVE CURSOR TO 21ST DIGIT          *
01209                    *                                  (MVECUR)       *
01210                    *                                                 *
```

```
01211                  ************************************************
01212                  *                                              *
01213                  *       ENTRY   : WAIT3                        *
01214                  *       RETURNS : LCDMP, CURADP                *
01215                  *                                              *
01216                  ************************************************
01217 0B72 08 B0 0A MVECUR BRSET  4,WAIT3,MVECUF  Output tone?
01218 0B75 A6 14          LDA    #20       Move cursor on 21st digit
01219 0B77 B7 40          STA    LCDMP
01220 0B79 4C             INC    A
01221 0B7A B7 4B          STA    CURADP
01222 0B7C CD 110E        JSR    LCD
01223 0B7F 81        MVECUF RTS
01224                  *
01225                  ****   PROGRAM MODULE FOR TIMER1  ***************
01226                  *
01227                  *
01228                  ************************************************
01229                  *                                              *
01230                  *       NAME    : CONTROL TIMER1 (TIMER1)      *
01231                  *                                              *
01232                  ************************************************
01233                  *                                              *
01234                  *       ENTRY   : NOTHING                      *
01235                  *       RETURNS : NOTHING                      *
01236                  *                                              *
01237                  ************************************************
01238 0B80 1F 09     TIMER1 BCLR   7,TCR
01239 0B82 A6 FA            LDA    #$FA
01240 0B84 B7 08            STA    TDR
01241 0B86 CD 0B8D          JSR    BUZC
01242 0B89 CD 0BDB          JSR    FIGURE
01243 0B8C 80               RTI
01244                  *
01245                  ************************************************
01246                  *                                              *
01247                  *       NAME : COUNT BUZZING TIME (BUZC)       *
01248                  *                                              *
01249                  ************************************************
01250                  *                                              *
01251                  *       ENTRY   : BZRFLG, BZRCNT, ERR1, ERR2   *
01252                  *                 WAITS                        *
01253                  *       RETURNS : BZRFLG, BZRCNT, ERR1, ERR2   *
01254                  *                 FLAG, WAITS                  *
01255                  *                                              *
01256                  ************************************************
01257 0B8D 06 4E 06 BUZC   BRSET  3,BZRFLG,BUZC1  Test it buzzer end
01258 0B90 0E B2 1D        BRSET  7,WAITS,BUZC2
01259 0B93 CC 0BDA         JMP    BUZC6
01260 0B96 BE 4D     BUZC1 LDX    BZRCNT+1 Count buzzer time
01261 0B98 5A              DEC    X
01262 0B99 BF 4D           STX    BZRCNT+1
01263 0B9B A3 00           CPX    #0
01264 0B9D 26 11           BNE    BUZC2
01265 0B9F B6 4C           LDA    BZRCNT
```

```
01266 0BA1 27 09           BEQ    BUZC3
01267 0BA3 4A              DEC    A
01268 0BA4 B7 4C           STA    BZRCNT
01269 0BA6 A6 FF           LDA    #$FF
01270 0BA8 B7 4D           STA    BZRCNT+1
01271 0BAA 20 04           BRA    BUZC2
01272 0BAC 17 4E     BUZC3 BCLR   3,BZRFLG Count display time
01273 0BAE 1E B2           BSET   7,WAIT5
01274 0BB0 C6 013B   BUZC2 LDA    ERR1
01275 0BB3 27 07           BEQ    BUZC5
01276 0BB5 4A              DEC    A
01277 0BB6 C7 013B         STA    ERR1
01278 0BB9 CC 0BDA         JMP    BUZC6
01279 0BBC C6 013C   BUZC5 LDA    ERR2
01280 0BBF 27 19           BEQ    BUZC6
01281 0BC1 4A              DEC    A
01282 0BC2 C7 013C         STA    ERR2
01283 0BC5 A1 01           CMP    #1
01284 0BC7 26 0C           BNE    BUZC4
01285 0BC9 CD 1966         JSR    LCDMCR
01286 0BCC CD 110E         JSR    LCD
01287 0BCF 12 42           BSET   1,FLAG   Define flag for calendar & time
01288 0BD1 1F B2           BCLR   7,WAIT5
01289 0BD3 20 05           BRA    BUZC6
01290 0BD5 A6 FF     BUZC4 LDA    #$FF
01291 0BD7 C7 013B         STA    ERR1
01292 0BDA 81        BUZC6 RTS
01293                      *
01294                      ************************************************
01295                      *                                              *
01296                      *       NAME    : COUNT TIME FOR DIALING       *
01297                      *                    INFORMATION ON LCD (FIGURE)  *
01298                      *                                              *
01299                      ************************************************
01300                      *                                              *
01301                      *       ENTRY   : TCNTR, TELFLG, FLAG, CALNDR  *
01302                      *                 TEL, SINGL, TOTAL            *
01303                      *       RETURNS : TCNTR, CALNDR, TEL, SINGL    *
01304                      *                 TOTAL, LCDM, CURADP, SECCNT  *
01305                      *                                              *
01306                      ************************************************
01307 0BDB 3A 70     FIGURE DEC    TCNTR    Decrement 1-second counter
01308 0BDD 26 45           BNE    FIG8     1-second?
01309 0BDF A6 7D           LDA    #$7D     Reinitialize 1-second counter
01310 0BE1 B7 70           STA    TCNTR
01311 0BE3 CD 11E4         JSR    YERCNT   Count calender
01312 0BE6 01 44 0D        BRCLR  0,TELFLG,FIG1  Hand-set up?
01313 0BE9 CD 1489         JSR    TELCLR   Clear elapsed time
01314 0BEC CD 17B1         JSR    SNGCLR   Clear charge
01315 0BEF 4F              CLR    A
01316 0BF0 C7 0135         STA    SECCNT   Clear charge time counter
01317 0BF3 C7 0136         STA    SECCNT+1
01318 0BF6 03 44 06 FIG1   BRCLR  1,TELFLG,FIG2  Talking?
01319 0BF9 CD 11EA         JSR    TLCNT    Count elapsed time
01320 0BFC CD 15CE         JSR    MONEY    Count charge
```

```
01321 0BFF 02 42 0B FIG2   BRSET  1,FLAG,FIG3  Display calendar & time?
01322 0C02 06 42 0D        BRSET  3,FLAG,FIG4  Elapsed time?
01323 0C05 0C 42 0F        BRSET  6,FLAG,FIG5  Accumulated charge?
01324 0C08 0E 42 11        BRSET  7,FLAG,FIG6  Charge?
01325 0C0B 20 17           BRA    FIG8
01326 0C0D CD 1258  FIG3   JSR    CALLCD   Load calendar & time data
01327 0C10 20 0D           BRA    FIG7
01328 0C12 CD 1267  FIG4   JSR    TELLCD   Load elapsed time data
01329 0C15 20 08           BRA    FIG7
01330 0C17 CD 16CD  FIG5   JSR    TOLSTA   Load accumulated charge data
01331 0C1A 20 03           BRA    FIG7
01332 0C1C CD 167E  FIG6   JSR    SNGSTA   Load charge data
01333 0C1F 3F 4B    FIG7   CLR    CURADP   Cursor OFF
01334 0C21 CD 110E         JSR    LCD      Display information
01335 0C24 81       FIG8   RTS
01336                      *
01337                      ****    PROGRAM MODULE FOR TIMER2   ************
01338                      *
01339                      *
01340                      ************************************************
01341                      *                                              *
01342                      *       NAME   : CONTROL TIMER2 (TIMER2)       *
01343                      *                                              *
01344                      ************************************************
01345                      *                                              *
01346                      *       ENTRY    : NOTHING                     *
01347                      *       RETURNS  : NOTHING                     *
01348                      *                                              *
01349                      ************************************************
01350 0C25 1D 11    TIMER2 BCLR   6,SSR
01351 0C27 CD 0C31         JSR    K88SCN
01352 0C2A CD 0C95         JSR    BUZZ
01353 0C2D CD 0CAF         JSR    TELMN
01354 0C30 80              RTI
01355                      *
01356                      ************************************************
01357                      *                                              *
01358                      *       NAME     : KEY SCAN (K88SCN)           *
01359                      *                                              *
01360                      ************************************************
01361                      *                                              *
01362                      *       ENTRY    :  NOTHING                    *
01363                      *       RETURNS  :  KEYDAT, FLAG               *
01364                      *                                              *
01365                      ************************************************
01366 0C31 00 42 60 K88SCN BRSET  0,FLAG,K88SN9  Test if there is key data
01367 0C34 A6 80           LDA    #$80     Initialize strobe data
01368 0C36 B7 5B           STA    STBDAT
01369 0C38 A6 01           LDA    #$01     Initialize key number
01370 0C3A B7 5A           STA    KEYNUM
01371 0C3C 3F 59           CLR    TOTLKY   Clear total key
01372 0C3E 3F 56           CLR    NEWKEY   Clear new key
01373 0C40 B6 5B    K88SN1 LDA    STBDAT   Output strobe
01374 0C42 B7 06           STA    PCDDR
01375 0C44 B6 0D           LDA    PGDTR    Load key data
```

```
01376 0C46 A1 FF          CMP    #$FF      Test if key is pressed?
01377 0C48 26 08          BNE    K88SN2
01378 0C4A A6 08          LDA    #8        Store next key number
01379 0C4C BB 5A          ADD    KEYNUM
01380 0C4E B7 5A          STA    KEYNUM
01381 0C50 20 19          BRA    K88SN5
01382 0C52 AE 08    K88SN2 LDX   #8        Store shift counter
01383 0C54 BF 5C          STX    SCOUNT
01384 0C56 44       K88SN3 LSR   A         Test if key is pressed
01385 0C57 25 0C          BCS    K88SN4
01386 0C59 3C 59          INC    TOTLKY    Increment total key
01387 0C5B BE 59          LDX    TOTLKY    Test if 2 keys are pressed
01388 0C5D A3 01          CPX    #1
01389 0C5F 25 33          BCS    K88SN9
01390 0C61 BE 5A          LDX    KEYNUM    Store key data in new key
01391 0C63 BF 56          STX    NEWKEY
01392 0C65 3C 5A    K88SN4 INC   KEYNUM    Increment key number
01393 0C67 3A 5C          DEC    SCOUNT    Decrement shift counter
01394 0C69 26 EB          BNE    K88SN3    Test if shift 8 bits
01395 0C6B 34 5B    K88SN5 LSR   STBDAT    Output next strobe
01396 0C6D 26 D1          BNE    K88SN1    Test if stobe output is completed
01397 0C6F B6 56          LDA    NEWKEY
01398 0C71 27 04          BEQ    K88SN6    New key = 0 ?
01399 0C73 B1 55          CMP    OLDKEY    New key = old key ?
01400 0C75 27 06          BEQ    K88SN7    Branch if equal
01401 0C77 B7 55    K88SN6 STA   OLDKEY    Store new key in old key
01402 0C79 3F 57          CLR    CHATFL    Clear chatter counter
01403 0C7B 20 17          BRA    K88SN9
01404 0C7D 0E 57 14 K88SN7 BRSET 7,CHATFL,K88SN9  Chatter generation?
01405 0C80 A6 0F          LDA    #$0F      Check key data
01406 0C82 B4 57          AND    CHATFL
01407 0C84 A1 03          CMP    #CMPNUM
01408 0C86 27 04          BEQ    K88SN8
01409 0C88 3C 57          INC    CHATFL    Increment chatter counter
01410 0C8A 20 08          BRA    K88SN9
01411 0C8C 1E 57    K88SN8 BSET  7,CHATFL  Set flag for chatter prevention complete
01412 0C8E B6 56          LDA    NEWKEY    Store new key in key data
01413 0C90 B7 58          STA    KEYDAT
01414 0C92 10 42          BSET   0,FLAG    Set flag for valid key data
01415 0C94 81       K88SN9 RTS
01416                     *
01417                     ************************************************
01418                     *                                              *
01419                     *      NAME : BUZZING (BUZZ)                   *
01420                     *                                              *
01421                     ************************************************
01422                     *                                              *
01423                     *      ENTRY   : BZRFLG, BZR                   *
01424                     *      RETURNS : BZR                           *
01425                     *                                              *
01426                     ************************************************
01427 0C95 07 4E 16 BUZZ   BRCLR  3,BZRFLG,BUZZ2  Output buzzer?
01428 0C98 00 4F 08        BRSET  0,BZR,BUZZ1  Check output condition
01429 0C9B B6 0C           LDA    PFDTR
01430 0C9D A4 BF           AND    #%10111111  Output Low
```

```
01431 0C9F B7 0C          STA    PFDTR
01432 0CA1 10 4F          BSET   0,BZR    Set flag to output High
01433 0CA3 CC 0CAE        JMP    BUZZ2
01434 0CA6 B6 0C    BUZZ1 LDA    PFDTR
01435 0CA8 AA 40          ORA    #%01000000  Output High
01436 0CAA B7 0C          STA    PFDTR
01437 0CAC 11 4F          BCLR   0,BZR    Clear flag to output Low
01438 0CAE 81       BUZZ2 RTS
01439                     *
01440                     ************************************************
01441                     *                                              *
01442                     *       NAME    : CONTROL DIALING CIRCUIT      *
01443                     *                          (TELMN)             *
01444                     *                                              *
01445                     ************************************************
01446                     *                                              *
01447                     *       ENTRY   : TELFLG, PDDTR, BZR, BZRFLG   *
01448                     *       RETURNS : NOTHING                      *
01449                     *                                              *
01450                     ************************************************
01451 0CAF 02 44 0E TELMN  BRSET  1,TELFLG,TELMN1  While talking?
01452 0CB2 03 03 10        BRCLR  1,PDDTR,TELMN2  While receiving?
01453 0CB5 04 4F 12        BRSET  2,BZR,TELMN3  Generating tone?
01454 0CB8 02 4F 14        BRSET  1,BZR,TELMN4  Relay ON
01455 0CBB 00 4E 16        BRSET  0,BZRFLG,TELMN5  While retrying?
01456 0CBE 20 19           BRA    TELMN6   Handset up?
01457 0CC0 CD 0CDD  TELMN1 JSR    TELM10
01458 0CC3 20 17           BRA    TELMN7
01459 0CC5 CD 0D00  TELMN2 JSR    TELM20
01460 0CC8 20 12           BRA    TELMN7
01461 0CCA CD 0D15  TELMN3 JSR    TELM30
01462 0CCD 20 0D           BRA    TELMN7
01463 0CCF CD 0E96  TELMN4 JSR    TELM40
01464 0CD2 20 08           BRA    TELMN7
01465 0CD4 CD 0EF2  TELMN5 JSR    TELM50
01466 0CD7 20 03           BRA    TELMN7
01467 0CD9 CD 0FDB  TELMN6 JSR    TELM60
01468 0CDC 81       TELMN7 RTS
01469                     *
01470                     ************************************************
01471                     *                                              *
01472                     *       NAME    : WHILE TALKING (TELM10)       *
01473                     *                                              *
01474                     ************************************************
01475                     *                                              *
01476                     *       ENTRY   : PDDTR, WAIT3, MEMOP          *
01477                     *       RETURNS : FLAG, PEDTR, WAIT3           *
01478                     *                                              *
01479                     ************************************************
01480 0CDD 04 03 07 TELM10 BRSET  2,PDDTR,TELM11  Cut line?
01481 0CE0 06 03 04        BRSET  3,PDDTR,TELM11
01482 0CE3 17 42           BCLR   3,FLAG
01483 0CE5 12 42           BSET   1,FLAG   Display calender & time
01484 0CE7 01 B0 12 TELM11 BRCLR  0,WAIT3,TELM12
01485 0CEA B6 42           LDA    FLAG
```

```
01486 0CEC A4 35           AND    #$35
01487 0CEE B7 42           STA    FLAG
01488 0CF0 B6 0B           LDA    PEDTR    Restore contents of port E
01489 0CF2 A4 7F           AND    #%01111111
01490 0CF4 B7 0B           STA    PEDTR
01491 0CF6 B6 B3           LDA    MEMOP
01492 0CF8 B7 0B           STA    PEDTR
01493 0CFA 11 B0           BCLR   0,WAIT3
01494 0CFC CD 1038  TELM12 JSR    CALRST
01495 0CFF 81              RTS
01496                *
01497                ************************************************
01498                *                                              *
01499                *      NAME    : WHILE RECEIVING (TELM20)      *
01500                *                                              *
01501                ************************************************
01502                *                                              *
01503                *      ENTRY   : BZRFLG, PDDTR                 *
01504                *      RETURNS : TELFLG, BZRFLG, CURADP        *
01505                *                                              *
01506                ************************************************
01507 0D00 00 4E 08 TELM20 BRSET  0,BZRFLG,TELM21  While retrying?
01508 0D03 04 03 0A        BRSET  2,PDDTR,TELM22  Handset up?
01509 0D06 06 03 07        BRSET  3,PDDTR,TELM22
01510 0D09 20 09           BRA    TELM23
01511 0D0B CD 1001  TELM21 JSR    CANCEL   Cancel retrying
01512 0D0E 20 04           BRA    TELM23
01513 0D10 12 44    TELM22 BSET   1,TELFLG Define flag for talking
01514 0D12 3F 4B           CLR    CURADP   Cursor OFF
01515 0D14 81       TELM23 RTS
01516                *
01517                ************************************************
01518                *                                              *
01519                *      NAME    : GENEATING TONE (TELM30)       *
01520                *                                              *
01521                ************************************************
01522                *                                              *
01523                *      ENTRY   : TELWAT, WAIT5, BZR, DIALNO    *
01524                *                TELNO, WAIT3, TELFLG, DIADAT  *
01525                *                DIAP2, BZRFLG, TELFLG, PEDTR  *
01526                *                PDDTR                         *
01527                *      RETURNS : TELWAT, BZR, MFDAT, TELFLG    *
01528                *                DIAP2, MEMOP, WAIT3           *
01529                *                                              *
01530                ************************************************
01531 0D15 B6 B6    TELM30 LDA    TELWAT
01532 0D17 26 15           BNE    TELM31
01533 0D19 A6 5F           LDA    #95      Initialize for tone output
01534 0D1B B7 B6           STA    TELWAT
01535 0D1D B6 0B           LDA    PEDTR
01536 0D1F B7 B3           STA    MEMOP    Save contents of PEDTR
01537 0D21 A6 01           LDA    #$1
01538 0D23 C7 0139         STA    TELNO
01539 0D26 10 B0           BSET   0,WAIT3
01540 0D28 CD 1038         JSR    CALRST   Test if handset is down
```

```
01541 0D2B CC 0E95          JMP    TELM3F
01542 0D2E A1 01     TELM31 CMP    #1
01543 0D30 27 08            BEQ    TELM32
01544 0D32 3A B6            DEC    TELWAT   Wait 0.5s
01545 0D34 CD 1038          JSR    CALRST   Test if handset is down
01546 0D37 CC 0E95          JMP    TELM3F
01547 0D3A 0B B2 03 TELM32 BRCLR  5,WAIT5,TEL321  Output less than 20th no.?
01548 0D3D CC 0DE3          JMP    TELM36
01549 0D40 06 4F 44 TEL321 BRSET  3,BZR,TELM33
01550 0D43 A6 80            LDA    #%10000000  HS pin ON
01551 0D45 B7 0B            STA    PEDTR
01552 0D47 B6 AE            LDA    WAIT1    Wait for 80ms for HS pin
01553 0D49 A1 05            CMP    #5
01554 0D4B 26 32            BNE    TEL324
01555 0D4D B6 AF            LDA    WAIT2    Output tone for 200ms
01556 0D4F 26 10            BNE    TEL322
01557 0D51 CE 0139          LDX    TELNO    Load dial no.
01558 0D54 E6 99            LDA    DIALNO,X
01559 0D56 CD 10DF          JSR    MFOUT    Find output data
01560 0D59 B6 0B            LDA    PEDTR
01561 0D5B A4 80            AND    #%10000000  Define output tone data
01562 0D5D BA B5            ORA    MFDAT
01563 0D5F B7 0B            STA    PEDTR
01564 0D61 3C AF     TEL322 INC    WAIT2
01565 0D63 B6 AF            LDA    WAIT2    Count 200ms
01566 0D65 A1 0C            CMP    #12
01567 0D67 27 03            BEQ    TEL323
01568 0D69 CC 0E95          JMP    TELM3F
01569 0D6C C6 0139  TEL323 LDA    TELNO    Increment dial no. pointer
01570 0D6F 4C               INC    A
01571 0D70 C7 0139          STA    TELNO
01572 0D73 3F AE            CLR    WAIT1
01573 0D75 3F AF            CLR    WAIT2
01574 0D77 16 4F            BSET   3,BZR    Define flag for HS pin OFF
01575 0D79 CD 1038          JSR    CALRST
01576 0D7C CC 0E95          JMP    TELM3F
01577 0D7F 3C AE     TEL324 INC    WAIT1    Count 80ms
01578 0D81 CD 1038          JSR    CALRST   Test if handset is down
01579 0D84 CC 0E95          JMP    TELM3F
01580 0D87 B6 0B     TELM33 LDA    PEDTR    HS pin OFF
01581 0D89 A4 00            AND    #%00000000
01582 0D8B B7 0B            STA    PEDTR
01583 0D8D B6 AF            LDA    WAIT2
01584 0D8F A1 06            CMP    #6       Not output tone for 100ms
01585 0D91 26 1C            BNE    TEL332
01586 0D93 C6 0139          LDA    TELNO    Next digit is 21?
01587 0D96 A1 15            CMP    #21
01588 0D98 26 02            BNE    TEL331
01589 0D9A 20 33            BRA    TELM35
01590 0D9C CE 0139 TEL331 LDX    TELNO    If not, load next dial no.
01591 0D9F E6 99            LDA    DIALNO,X
01592 0DA1 A1 20            CMP    #$20     Is there next dial no.?
01593 0DA3 27 12            BEQ    TELM34   If not, wait until next no. input
01594 0DA5 17 4F            BCLR   3,BZR    If so, define flag to output next tone
01595 0DA7 3F AF            CLR    WAIT2
```

```
01596 0DA9 CD 1038          JSR    CALRST   Test if handset is down
01597 0DAC CC 0E95          JMP    TELM3F
01598 0DAF 3C AF     TEL332 INC    WAIT2    Count 100ms
01599 0DB1 CD 1038          JSR    CALRST   Test if handset is down
01600 0DB4 CC 0E95          JMP    TELM3F
01601 0DB7 00 4E 0C TELM34 BRSET  0,BZRFLG,TEL341  While retrying?
01602 0DBA 0C 44 0C         BRSET  6,TELFLG,TEL342  Answered?
01603 0DBD 0E 44 0C         BRSET  7,TELFLG,TEL343
01604 0DC0 CD 1038          JSR    CALRST
01605 0DC3 CC 0E95          JMP    TELM3F
01606 0DC6 CC 0E64   TEL341 JMP    TELM38
01607 0DC9 CC 0E7B   TEL342 JMP    TELM39
01608 0DCC CC 0E84   TEL343 JMP    TELM3A
01609 0DCF BE BD     TELM35 LDX    DIAP2    Output more than 21th digit
01610 0DD1 E6 B7            LDA    DIADAT,X
01611 0DD3 A1 20            CMP    #$20     Is there next data?
01612 0DD5 27 06            BEQ    TEL351
01613 0DD7 1A B2            BSET   5,WAIT5  Define flag for output tone
01614 0DD9 3F AF            CLR    WAIT2
01615 0DDB 17 4F            BCLR   3,BZR
01616 0DDD CD 1038   TEL351 JSR    CALRST   Test if handset is down
01617 0DE0 CC 0E95          JMP    TELM3F
01618 0DE3 06 4F 4F TELM36 BRSET  3,BZR,TELM37
01619 0DE6 B6 0B            LDA    PEDTR
01620 0DE8 AA 80            ORA    #%10000000  HS pin ON
01621 0DEA B7 0B            STA    PEDTR
01622 0DEC B6 AE            LDA    WAIT1
01623 0DEE A1 05            CMP    #5       Wait for 80ms for HS pin
01624 0DF0 26 3B            BNE    TEL364
01625 0DF2 B6 AF            LDA    WAIT2
01626 0DF4 26 17            BNE    TEL361
01627 0DF6 BE BD            LDX    DIAP2    Outout tone for 200ms
01628 0DF8 E6 B7            LDA    DIADAT,X
01629 0DFA CD 10DF          JSR    MFOUT    Define tone data
01630 0DFD B6 0B            LDA    PEDTR
01631 0DFF A4 80            AND    #%10000000
01632 0E01 BA B5            ORA    MFDAT
01633 0E03 B7 0B            STA    PEDTR
01634 0E05 BE BD            LDX    DIAP2    Clear dial no. already output
01635 0E07 A6 20            LDA    #$20
01636 0E09 E7 B7            STA    DIADAT,X
01637 0E0B 20 0C            BRA    TEL362
01638 0E0D B6 AF     TEL361 LDA    WAIT2    Count 200ms
01639 0E0F 4C               INC    A
01640 0E10 B7 AF            STA    WAIT2
01641 0E12 A1 0C            CMP    #12
01642 0E14 27 03            BEQ    TEL362
01643 0E16 CC 0E95          JMP    TELM3F
01644 0E19 B6 BD     TEL362 LDA    DIAP2    Test if buffer is full
01645 0E1B 4C               INC    A
01646 0E1C B7 BD            STA    DIAP2
01647 0E1E A1 04            CMP    #4
01648 0E20 26 02            BNE    TEL363
01649 0E22 3F BD            CLR    DIAP2
01650 0E24 16 4F     TEL363 BSET   3,BZR    Define flag for HS pin OFF
```

```
01651 0E26 3F AE           CLR    WAIT1
01652 0E28 3F AF           CLR    WAIT2
01653 0E2A CC 0E95         JMP    TELM3F
01654 0E2D 3C AE    TEL364 INC    WAIT1    Count 80ms
01655 0E2F CD 1038         JSR    CALRST
01656 0E32 CC 0E95         JMP    TELM3F
01657 0E35 B6 0B    TELM37 LDA    PEDTR
01658 0E37 A4 00           AND    #%00000000  HS pin OFF
01659 0E39 B7 0B           STA    PEDTR
01660 0E3B B6 AF           LDA    WAIT2
01661 0E3D A1 07           CMP    #7
01662 0E3F 26 1B           BNE    TEL372
01663 0E41 BE BD           LDX    DIAP2    Is there next dial no.?
01664 0E43 E6 B7           LDA    DIADAT,X
01665 0E45 A1 20           CMP    #$20
01666 0E47 27 07           BEQ    TEL371
01667 0E49 17 4F           BCLR   3,BZR    Define flag for HS pin ON
01668 0E4B 3F AF           CLR    WAIT2
01669 0E4D CC 0E95         JMP    TELM3F
01670 0E50 0C 44 28 TEL371 BRSET  6,TELFLG,TELM39  Answered?
01671 0E53 0E 44 2E        BRSET  7,TELFLG,TELM3A
01672 0E56 CD 1038         JSR    CALRST   Test if handset is down
01673 0E59 CC 0E95         JMP    TELM3F
01674 0E5C 3C AF    TEL372 INC    WAIT2
01675 0E5E CD 1038         JSR    CALRST
01676 0E61 CC 0E95         JMP    TELM3F
01677 0E64 15 4F    TELM38 BCLR   2,BZR    Initialize for retrying
01678 0E66 A6 FF           LDA    #$FF
01679 0E68 C7 013A         STA    TELCNT
01680 0E6B A6 02           LDA    #2
01681 0E6D B7 B6           STA    TELWAT
01682 0E6F A6 0A           LDA    #10
01683 0E71 C7 0139         STA    TELNO
01684 0E74 B6 B3           LDA    MEMOP
01685 0E76 B7 0B           STA    PEDTR
01686 0E78 CC 0E95         JMP    TELM3F
01687 0E7B 06 03 0F TELM39 BRSET  3,PDDTR,TELM3B  Test if answered
01688 0E7E CD 1038         JSR    CALRST   Test if handset is down
01689 0E81 CC 0E95         JMP    TELM3F
01690 0E84 04 03 06 TELM3A BRSET  2,PDDTR,TELM3B
01691 0E87 CD 1038         JSR    CALRST   Test if handset is down
01692 0E8A CC 0E95         JMP    TELM3F
01693 0E8D 12 44    TELM3B BSET   1,TELFLG If answered, define flag for talking
01694 0E8F 11 44           BCLR   0,TELFLG
01695 0E91 B6 B3           LDA    MEMOP
01696 0E93 B7 0B           STA    PEDTR    Restore contents of port E
01697 0E95 81       TELM3F RTS
01698                      *
01699                      ************************************************
01700                      *                                              *
01701                      *       NAME    : RELAY ON (TELM40)            *
01702                      *                                              *
01703                      ************************************************
01704                      *                                              *
01705                      *       ENTRY   : BZR, PDDTR, TELWAT           *
```

```
01706                   *       RETURNS : PFDTR, BZR, ERFLG, TELFLG     *
01707                   *                                               *
01708                   *************************** *********************
01709 0E96 0C 4F 22 TELM40 BRSET  6,BZR,TELM44  Relay ON?
01710 0E99 04 03 11        BRSET  2,PDDTR,TELM41  Handset up?
01711 0E9C 06 03 12        BRSET  3,PDDTR,TELM42
01712 0E9F B6 0C           LDA    PFDTR    If not, connect line
01713 0EA1 AA 20           ORA    #%00100000
01714 0EA3 B7 0C           STA    PFDTR
01715 0EA5 A6 40           LDA    #64      Initialize counter
01716 0EA7 B7 B6           STA    TELWAT
01717 0EA9 1C 4F           BSET   6,BZR
01718 0EAB 20 44           BRA    TELM49
01719 0EAD 1C 44    TELM41 BSET   6,TELFLG Define flag
01720 0EAF 20 02           BRA    TELM43
01721 0EB1 1E 44    TELM42 BSET   7,TELFLG
01722 0EB3 14 4F    TELM43 BSET   2,BZR    Define flag for tone output
01723 0EB5 13 4F           BCLR   1,BZR
01724 0EB7 3F B6           CLR    TELWAT
01725 0EB9 20 36           BRA    TELM49
01726 0EBB B6 B6    TELM44 LDA    TELWAT
01727 0EBD A1 01           CMP    #1
01728 0EBF 27 04           BEQ    TELM45
01729 0EC1 3A B6           DEC    TELWAT   Wait for 0.5s
01730 0EC3 20 2C           BRA    TELM49
01731 0EC5 04 03 15 TELM45 BRSET  2,PDDTR,TELM46  Line connected?
01732 0EC8 06 03 16        BRSET  3,PDDTR,TELM47
01733 0ECB 10 BE           BSET   0,ERFLG  Set error process request flag
01734 0ECD 12 BE           BSET   1,ERFLG
01735 0ECF B6 0C           LDA    PFDTR
01736 0ED1 A4 DF           AND    #%11011111
01737 0ED3 B7 0C           STA    PFDTR
01738 0ED5 1D 4F           BCLR   6,BZR
01739 0ED7 CD 1038         JSR    CALRST
01740 0EDA CC 0EF1         JMP    TELM49
01741 0EDD 1C 44    TELM46 BSET   6,TELFLG Define flag
01742 0EDF 20 02           BRA    TELM48
01743 0EE1 1E 44    TELM47 BSET   7,TELFLG
01744 0EE3 14 4F    TELM48 BSET   2,BZR    Define flag for tone output
01745 0EE5 13 4F           BCLR   1,BZR
01746 0EE7 17 4F           BCLR   3,BZR
01747 0EE9 1D 4F           BCLR   6,BZR
01748 0EEB 1A 4F           BSET   5,BZR
01749 0EED 10 44           BSET   0,TELFLG
01750 0EEF 3F B6           CLR    TELWAT
01751 0EF1 81       TELM49 RTS
01752                   *
01753                   ************************************************
01754                   *                                              *
01755                   *       NAME    : WHILE RETRYING (TELM50)      *
01756                   *                                              *
01757                   ************************************************
01758                   *                                              *
01759                   *       ENTRY   : TELWAT, TELFLG, REC, BZR     *
01760                   *                 PDDTR                        *
```

```
01761                    *       RETURNS : TELWAT, WAIT3, BZR, TELFLG    *
01762                    *                 REC, LCDM, BZRFLG, ERR1, ERR2 *
01763                    *                                               *
01764                    ************************************************
01765 0EF2 B6 B6     TELMS0 LDA     TELWAT   Initialize retrying
01766 0EF4 26 1A            BNE     TELMS1
01767 0EF6 A6 1E            LDA     #30
01768 0EF8 C7 0139          STA     TELNO
01769 0EFB A6 FF            LDA     #$FF
01770 0EFD C7 013A          STA     TELCNT
01771 0F00 A6 01            LDA     #1
01772 0F02 B7 B6            STA     TELWAT
01773 0F04 0E 4F 06         BRSET   7,BZR,TEL501  Initialize tone output counter?
01774 0F07 A6 0A            LDA     #10
01775 0F09 B7 B4            STA     REC
01776 0F0B 1E 4F            BSET    7,BZR
01777 0F0D CC 0FD7   TEL501 JMP     TELMSF
01778 0F10 A1 01     TELMS1 CMP     #1
01779 0F12 26 59            BNE     TELMS4
01780 0F14 02 B0 08         BRSET   1,WAIT3,TEL511
01781 0F17 A6 35            LDA     #$35     Initialize for blinking retry no.
01782 0F19 B7 B1            STA     WAIT4
01783 0F1B 12 B0            BSET    1,WAIT3
01784 0F1D 20 28            BRA     TEL513
01785 0F1F 04 B0 0F  TEL511 BRSET   2,WAIT3,TEL512
01786 0F22 3A B1            DEC     WAIT4    Turn on retry no.
01787 0F24 B6 B1            LDA     WAIT4
01788 0F26 26 1F            BNE     TEL513
01789 0F28 13 B0            BCLR    1,WAIT3
01790 0F2A 14 B0            BSET    2,WAIT3
01791 0F2C CD 1526          JSR     DIAL2
01792 0F2F 20 16            BRA     TEL513
01793 0F31 3A B1     TEL512 DEC     WAIT4    Turn off retry no.
01794 0F33 B6 B1            LDA     WAIT4
01795 0F35 26 10            BNE     TEL513
01796 0F37 13 B0            BCLR    1,WAIT3
01797 0F39 15 B0            BCLR    2,WAIT3
01798 0F3B B6 42            LDA     FLAG
01799 0F3D A4 35            AND     #$35
01800 0F3F B7 42            STA     FLAG
01801 0F41 CD 1966          JSR     LCDMCR
01802 0F44 CD 110E          JSR     LCD
01803 0F47 C6 013A   TEL513 LDA     TELCNT   Wait 2 minutes
01804 0F4A 4A               DEC     A
01805 0F4B C7 013A          STA     TELCNT
01806 0F4E 26 12            BNE     TELMS2
01807 0F50 C6 0139          LDA     TELNO
01808 0F53 4A               DEC     A
01809 0F54 C7 0139          STA     TELNO
01810 0F57 26 0C            BNE     TELMS3
01811 0F59 3F B6            CLR     TELWAT
01812 0F5B 12 4F            BSET    1,BZR    If waiting finished, set flag
01813 0F5D 18 B0            BSET    4,WAIT3
01814 0F5F CD 1526          JSR     DIAL2
01815 0F62 CC 0FD7   TELMS2 JMP     TELMSF
```

```
01816 0F65 A6 FF    TELM53 LDA    #$FF
01817 0F67 C7 013A         STA    TELCNT
01818 0F6A CC 0FD7         JMP    TELM5F
01819 0F6D A1 02    TELM54 CMP    #2
01820 0F6F 26 4C           BNE    TELM5A
01821 0F71 0C 44 09        BRSET  6,TELFLG,TELM55
01822 0F74 0E 44 0C        BRSET  7,TELFLG,TELM56
01823 0F77 CD 1001         JSR    CANCEL
01824 0F7A CC 0FD7         JMP    TELM5F
01825 0F7D 06 03 09 TELM55 BRSET  3,PDDTR,TELM57  Test if answered
01826 0F80 CC 0FA4         JMP    TELM58
01827 0F83 04 03 03 TELM56 BRSET  2,PDDTR,TELM57
01828 0F86 CC 0FA4         JMP    TELM58
01829 0F89 12 44    TELM57 BSET   1,TELFLG Define flag for talking
01830 0F8B 11 44           BCLR   0,TELFLG
01831 0F8D 16 4E           BSET   3,BZRFLG Initialize buzzering
01832 0F8F A6 07           LDA    #$7
01833 0F91 B7 4C           STA    BZRCNT
01834 0F93 A6 FF           LDA    #$FF
01835 0F95 B7 4D           STA    BZRCNT+1
01836 0F97 4F              CLR    A
01837 0F98 C7 013B         STA    ERR1
01838 0F9B C7 013C         STA    ERR2
01839 0F9E CD 1001         JSR    CANCEL
01840 0FA1 CC 0FD7         JMP    TELM5F
01841 0FA4 C6 013A  TELM58 LDA    TELCNT   Output tone 45s
01842 0FA7 4A              DEC    A
01843 0FA8 C7 013A         STA    TELCNT
01844 0FAB 26 0D           BNE    TELM59
01845 0FAD C6 0139         LDA    TELNO
01846 0FB0 4A              DEC    A
01847 0FB1 C7 0139         STA    TELNO
01848 0FB4 26 04           BNE    TELM59
01849 0FB6 A6 03           LDA    #3
01850 0FB8 B7 B6           STA    TELWAT
01851 0FBA CC 0FD7  TELM59 JMP    TELM5F
01852 0FBD A1 03    TELM5A CMP    #3
01853 0FBF 26 0C           BNE    TELM5B
01854 0FC1 B6 0C           LDA    PFDTR    Stop tone output
01855 0FC3 A4 CF           AND    #%11001111
01856 0FC5 B7 0C           STA    PFDTR
01857 0FC7 3A B4           DEC    REC      Decrement repeat counter
01858 0FC9 B6 B4           LDA    REC
01859 0FCB 26 06           BNE    TELM5C
01860 0FCD CD 1001  TELM5B JSR    CANCEL   If 10 times,cancel retrying
01861 0FD0 CC 0FD7         JMP    TELM5F
01862 0FD3 3F B6    TELM5C CLR    TELWAT   If not, reinitialize for next output
01863 0FD5 19 B0           BCLR   4,WAIT3
01864 0FD7 CD 1038  TELM5F JSR    CALRST
01865 0FDA 81              RTS
01866              *
01867              ************************************************
01868              *                                              *
01869              *      NAME    : CONNECT LINES (TELM60)        *
01870              *                                              *
```

```
                        ************************************************
                        *                                              *
                        *       ENTRY   : MODFLG, PDDTR                *
                        *       RETURNS : LCDM, MEMO, TELFLG, WAIT3    *
                        *                 CURADP, LCDMP                *
                        *                                              *
                        ************************************************
OFDB B6 41      TELM60  LDA     MODFLG
OFDD 26 1E              BNE     TELM62
OFDF 04 03 05           BRSET   2,PDDTR,TELM61  Handset up?
OFE2 06 03 02           BRSET   3,PDDTR,TELM61
OFE5 20 16              BRA     TELM62
OFE7 B6 42      TELM61  LDA     FLAG      If so, clear display
OFE9 A4 35              AND     #$35
OFEB B7 42              STA     FLAG
OFED CD 1966            JSR     LCDMCR
OFF0 CD 110E            JSR     LCD
OFF3 A6 02              LDA     #2
OFF5 B7 40              STA     LCDMP
OFF7 3F 4B              CLR     CURADP
OFF9 10 B0              BSET    0,WAIT3  Define flag for tone output
OFFB 10 44              BSET    0,TELFLG
OFFD CD 1038    TELM62  JSR     CALRST
1000 81                 RTS
                        *
                        ************************************************
                        *                                              *
                        *       NAME : RESET RETRYING (CANCEL)         *
                        *                                              *
                        ************************************************
                        *                                              *
                        *       ENTRY   : NOTHING                      *
                        *       RETURNS : RAM(USED FOR RETRYING)       *
                        *                                              *
                        ************************************************
1001 B6 4E      CANCEL  LDA     BZRFLG   Reset retrying
1003 A4 FE              AND     #%11111110
1005 B7 4E              STA     BZRFLG
1007 02 44 06           BRSET   1,TELFLG,CANCL2
100A B6 0C              LDA     PFDTR
100C A4 DF              AND     #%11011111
100E B7 0C              STA     PFDTR
1010 4F         CANCL2  CLR     A         Clear RAM
1011 B7 B6              STA     TELWAT
1013 C7 0139            STA     TELND
1016 C7 013A            STA     TELCNT
1019 B7 B4              STA     REC
101B B7 4C              STA     BZRCNT
101D B7 B0              STA     WAIT3
101F B7 B1              STA     WAIT4
1021 B7 B2              STA     WAIT5
1023 B6 4F              LDA     BZR
1025 A4 7F              AND     #%01111111
1027 B7 4F              STA     BZR
1029 B6 0B              LDA     PEDTR
```

```
01926 102B A4 7F           AND    #%01111111
01927 102D B7 0B           STA    PEDTR
01928 102F B6 B3           LDA    MEMOP
01929 1031 B7 0B           STA    PEDTR
01930 1033 10 B0           BSET   0,WAIT3
01931 1035 12 42           BSET   1,FLAG
01932 1037 81              RTS
01933                 *
01934                 ************************************************
01935                 *                                              *
01936                 *      NAME : RESET HANDSET (CALRST)           *
01937                 *                                              *
01938                 ************************************************
01939                 *                                              *
01940                 *      ENTRY   : NOTHING                       *
01941                 *      RETURNS : NOTHING                       *
01942                 *                                              *
01943                 ************************************************
01944 1038 01 4E 03 CALRST BRCLR  0,BZRFLG,RTS11  Test if retrying
01945 103B CC 10CE         JMP    RTS18
01946 103E 0B 4F 0B RTS11  BRCLR  5,BZR,RTS12
01947 1041 09 03 08        BRCLR  4,PDDTR,RTS12
01948 1044 B6 0C           LDA    PFDTR    Relay OFF
01949 1046 A4 DF           AND    #%11011111
01950 1048 B7 0C           STA    PFDTR
01951 104A 1B 4F           BCLR   5,BZR
01952 104C 04 03 07 RTS12  BRSET  2,PDDTR,RTS13  Test if talking
01953 104F 06 03 04        BRSET  3,PDDTR,RTS13
01954 1052 19 B0           BCLR   4,WAIT3
01955 1054 20 03           BRA    RTS14
01956 1056 CC 10DE  RTS13  JMP    RSTEND
01957 1059 01 B0 34 RTS14  BRCLR  0,WAIT3,RTS17
01958 105C B6 0B           LDA    PEDTR    If not, HS pin OFF
01959 105E A4 7F           AND    #%01111111
01960 1060 B7 0B           STA    PEDTR
01961 1062 B6 B3           LDA    MEMOP
01962 1064 B7 0B           STA    PEDTR
01963 1066 17 42           BCLR   3,FLAG
01964 1068 12 42           BSET   1,FLAG
01965 106A 11 B0           BCLR   0,WAIT3
01966 106C AE 05           LDX    #5
01967 106E A6 20    RTS15  LDA    #$20
01968 1070 E7 B6           STA    DIADAT-1,X
01969 1072 5A              DEC    X
01970 1073 26 F9           BNE    RTS15
01971 1075 CD 18DA         JSR    END
01972 1078 B6 41           LDA    MODFLG
01973 107A A1 01           CMP    #$1
01974 107C 27 0C           BEQ    RTS16
01975 107E A1 02           CMP    #$2
01976 1080 27 08           BEQ    RTS16
01977 1082 A1 03           CMP    #$3
01978 1084 27 04           BEQ    RTS16
01979 1086 A1 0C           CMP    #$C
01980 1088 27 00           BEQ    RTS16
```

```
01981 108A 09 B0 03 RTS16  BRCLR  4,WAIT3,RTS17
01982 108D CD 18DA         JSR    END
01983 1090 4F       RTS17  CLR    A         Clear RAM
01984 1091 B7 B6           STA    TELWAT
01985 1093 C7 0139         STA    TELNO
01986 1096 C7 013A         STA    TELCNT
01987 1099 B7 B4           STA    REC
01988 109B B7 AE           STA    WAIT1
01989 109D B7 AF           STA    WAIT2
01990 109F B7 B0           STA    WAIT3
01991 10A1 B7 B2           STA    WAIT5
01992 10A3 B7 BC           STA    DIAP1
01993 10A5 B7 BD           STA    DIAP2
01994 10A7 B6 44           LDA    TELFLG
01995 10A9 A4 04           AND    #%00000100
01996 10AB B7 44           STA    TELFLG
01997 10AD B6 4E           LDA    BZRFLG
01998 10AF A4 F9           AND    #%11111001
01999 10B1 B7 4E           STA    BZRFLG
02000 10B3 B6 4F           LDA    BZR
02001 10B5 A4 01           AND    #%00000001
02002 10B7 B7 4F           STA    BZR
02003 10B9 CD 146C         JSR    DIACLR
02004 10BC CD 1489         JSR    TELCLR
02005 10BF CD 17B1         JSR    SNGCLR
02006 10C2 4F              CLR    A
02007 10C3 C7 0135         STA    SECCNT
02008 10C6 C7 0136         STA    SECCNT+1
02009 10C9 0C B0 0E        BRSET  6,WAIT3,RTS19
02010 10CC 20 10           BRA    RSTEND
02011 10CE B6 41    RTS18  LDA    MODFLG
02012 10D0 A1 10           CMP    #$10
02013 10D2 26 0A           BNE    RSTEND
02014 10D4 CD 1001         JSR    CANCEL
02015 10D7 CC 104C         JMP    RTS12
02016 10DA 3F 43    RTS19  CLR    TRMFLG
02017 10DC 1D B0           BCLR   6,WAIT3
02018 10DE 81       RSTEND RTS
02019                      *
02020                      ************************************************
02021                      *                                              *
02022                      *      NAME : REFER TO DIAL NUMBER (MFOUT)     *
02023                      *                                              *
02024                      ************************************************
02025                      *                                              *
02026                      *      ENTRY   : ACCA                          *
02027                      *      RETURNS : MFDAT, Bit C                  *
02028                      *                                              *
02029                      ************************************************
02030 10DF 5F       MFOUT  CLR    X
02031 10E0 D1 10F6  MFOUT1 CMP    MFTBL,X
02032 10E3 27 09           BEQ    MFOUT2
02033 10E5 5C              INC    X
02034 10E6 5C              INC    X
02035 10E7 A3 18           CPX    #MFTBLE-MFTBL
```

```
02036 10E9 26 F5          BNE    MFOUT1
02037 10EB 98             CLC
02038 10EC 20 07          BRA    MFOUTE
02039 10EE 5C       MFOUT2 INC    X
02040 10EF D6 10F6        LDA    MFTBL,X
02041 10F2 B7 B5          STA    MFDAT    Define dial no.
02042 10F4 99             SEC
02043 10F5 81       MFOUTE RTS
02044                     *
02045                     ************************************************
02046                     *              /                               *
02047                     *       NAME    : DATA TABLE                   *
02048                     *                                              *
02049                     ************************************************
02050 10F6 31       MFTBL  FCC    "1"              Output tone data
02051 10F7 09              FCB    %00001001
02052 10F8 32              FCC    "2"
02053 10F9 0A              FCB    %00001010
02054 10FA 33              FCC    "3"
02055 10FB 0C              FCB    %00001100
02056 10FC 34              FCC    "4"
02057 10FD 11              FCB    %00010001
02058 10FE 35              FCC    "5"
02059 10FF 12              FCB    %00010010
02060 1100 36              FCC    "6"
02061 1101 14              FCB    %00010100
02062 1102 37              FCC    "7"
02063 1103 21              FCB    %00100001
02064 1104 38              FCC    "8"
02065 1105 22              FCB    %00100010
02066 1106 39              FCC    "9"
02067 1107 24              FCB    %00100100
02068 1108 2A              FCC    "*"
02069 1109 41              FCB    %01000001
02070 110A 30              FCC    "0"
02071 110B 42              FCB    %01000010
02072 110C 23              FCC    "#"
02073 110D 44              FCB    %01000100
02074       110E   MFTBLE EQU    *
02075                     *
02076                     ****   PROGRAM MODULE FOR LCD DISPLAY   ********
02077                     *
02078                     *
02079                     ************************************************
02080                     *                                              *
02081                     *       NAME    : LCD PROGRAM (LCD)            *
02082                     *                                              *
02083                     ************************************************
02084                     *                                              *
02085                     *       ENTRY   : LCDM, CURADP                 *
02086                     *       RETURNS : NOTHING                      *
02087                     *                                              *
02088                     ************************************************
02089 110E 9B       LCD    SEI
02090 110F AE 02           LDX    #2
```

```
02091 1111 CD 1148          JSR    LCDINS   Clear address counter
02092 1114 3F 6C            CLR    POINTR   Clear pointer
02093 1116 BE 6C            LDX    POINTR
02094 1118 EE 71     LCD1   LDX    LCDM,X   Load ASCII into IX
02095 111A CD 115C          JSR    LCDDSP   Display figure
02096 111D 3C 6C            INC    POINTR   Increment pointer
02097 111F BE 6C            LDX    POINTR
02098 1121 A3 28            CPX    #40      Pointer=40?
02099 1123 26 F3            BNE    LCD1     Branch if pointer =/ 40
02100 1125 B6 4B            LDA    CURADP
02101 1127 B7 6C            STA    POINTR   Load cursor position data
02102 1129 27 19            BEQ    LCD4     Display cursor?
02103 112B AE 02            LDX    #02
02104 112D CD 1148          JSR    LCDINS   Load instruction
02105 1130 3A 6C            DEC    POINTR
02106 1132 27 09            BEQ    LCD3     Display on 1st digit?
02107 1134 AE 14     LCD2   LDX    #$14
02108 1136 CD 1148          JSR    LCDINS   Shift cursor to right
02109 1139 3A 6C            DEC    POINTR
02110 113B 26 F7            BNE    LCD2
02111 113D AE 0E     LCD3   LDX    #$0E     Cursor ON
02112 113F CD 1148   LCD5   JSR    LCDINS
02113 1142 9A               CLI
02114 1143 81               RTS
02115 1144 AE 0C     LCD4   LDX    #$0C     Cursor OFF
02116 1146 20 F7            BRA    LCD5
02117                *
02118                ************************************************
02119                *                                              *
02120                *      NAME    : LOAD INSTRUCTION INTO         *
02121                *                  INSTRUCTION REGISTER(LCDINS)*
02122                *                                              *
02123                ************************************************
02124                *                                              *
02125                *      ENTRY   : IX                            *
02126                *      RETURNS : NOTHING                       *
02127                *                                              *
02128                ************************************************
02129 1148 CD 11C4  LCDINS JSR    LCDBSY   Check busy flag
02130 114B BF 00            STX    PADTR
02131 114D B6 0C            LDA    PFDTR
02132 114F A4 F0            AND    #$F0
02133 1151 B7 0C            STA    PFDTR    R¥W=0 RS=0 E=0
02134 1153 AA 01            ORA    #$01     R¥W=0 RS=0 E=1
02135 1155 B7 0C            STA    PFDTR
02136 1157 A4 F0            AND    #$F0
02137 1159 B7 0C            STA    PFDTR    R¥W=0 RS=0 E=0
02138 115B 81               RTS
02139                *
02140                ************************************************
02141                *                                              *
02142                *      NAME    : DISPLAY FIGURE ON LCD(LCDDSP) *
02143                *                                              *
02144                ************************************************
02145                *                                              *
```

```
02146                   *       ENTRY   : IX                            *
02147                   *       RETURNS : NOTHING                       *
02148                   *                                               *
02149                   ************************************************
02150 115C CD 11C4  LCDDSP JSR    LCDBSY   Check busy flag
02151 115F BF 00           STX    PADTR
02152 1161 B6 0C           LDA    PFDTR
02153 1163 A4 F0           AND    #$F0
02154 1165 AA 02           ORA    #$02     R¥W=0 RS=1 E=0
02155 1167 B7 0C           STA    PFDTR
02156 1169 AA 01           ORA    #$01     R¥W=0 RS=1 E=1
02157 116B B7 0C           STA    PFDTR
02158 116D A4 F2           AND    #$F2     R/W=0 RS=1 E=0
02159 116F B7 0C           STA    PFDTR
02160 1171 81              RTS
02161                   *
02162                   ************************************************
02163                   *                                               *
02164                   *       NAME    : RESET LCD-II (LCDRES)         *
02165                   *                                               *
02166                   ************************************************
02167                   *                                               *
02168                   *       ENTRY   : NOTHING                       *
02169                   *       RETURNS : NOTHING                       *
02170                   *                                               *
02171                   ************************************************
02172 1172 A6 03    LCDRES LDA    #3
02173 1174 B7 6D           STA    CNT      Initialize CNT
02174 1176 A6 0C    LCDRS1 LDA    #$0C     Execute of software timer of 15ms
02175 1178 AE FF    LCDRS2 LDX    #$FF
02176 117A 5A       LCDRS3 DEC    X
02177 117B 26 FD           BNE    LCDRS3
02178 117D 4A              DEC    A
02179 117E 26 F8           BNE    LCDRS2
02180 1180 A6 FF           LDA    #$FF
02181 1182 B7 04           STA    PADDR    Select port A as output
02182 1184 3F 0C           CLR    PFDTR    R¥W=0 RS=0 E=0
02183 1186 A6 01           LDA    #$01
02184 1188 B7 0C           STA    PFDTR    R¥W=0 RS=0 E=1
02185 118A A6 30           LDA    #$30
02186 118C B7 00           STA    PADTR    Output instruction
02187 118E 3F 0C           CLR    PFDTR    R¥W=0 RS=0 E=0
02188 1190 3A 6D           DEC    CNT
02189 1192 26 E2           BNE    LCDRS1   Branch if not CNT=0
02190 1194 81              RTS
02191                   *
02192                   ************************************************
02193                   *                                               *
02194                   *       NAME    : INITIALIZE DISPLAY MODE OF LCD*
02195                   *                                 (LCDINT)      *
02196                   *                                               *
02197                   ************************************************
02198                   *                                               *
02199                   *       ENTRY   : NOTHING                       *
02200                   *       RETURNS : NOTHING                       *
```

```
02201                   *                                               *
02202                   *************************************************
02203 1195 AE 06        LCDINT LDX    #6
02204 1197 A3 02        LCDIN1 CPX    #2
02205 1199 27 14               BEQ    LCDIN2
02206 119B CD 11C4      LCDIN5 JSR    LCDBSY   Check busy flag
02207 119E 3F 0C               CLR    PFDTR    R¥W=0 RS=0 E=0
02208 11A0 A6 01               LDA    #1
02209 11A2 B7 0C               STA    PFDTR    R¥W=0 RS=0 E=1
02210 11A4 D6 1A0C             LDA    YOKO-1,X
02211 11A7 B7 00               STA    PADTR    Store instruction data
02212 11A9 3F 0C               CLR    PFDTR    R¥W=0 RS=0 E=0
02213 11AB 5A                  DEC    X
02214 11AC 26 E9               BNE    LCDIN1   Branch if not IX=0
02215 11AE 81                  RTS
02216 11AF AE 40        LCDIN2 LDX    #$40
02217 11B1 CD 1148             JSR    LCDINS   Output instruction
02218 11B4 AE 0C               LDX    #12
02219 11B6 B7 6C               STA    POINTR
02220 11B8 5F           LCDIN3 CLR    X
02221 11B9 CD 115C             JSR    LCDDSP
02222 11BC 3A 6C               DEC    POINTR
02223 11BE 26 F8               BNE    LCDIN3   Test if data is written 12 times
02224 11C0 AE 02               LDX    #2
02225 11C2 20 D7               BRA    LCDIN5
02226                   *
02227                   *************************************************
02228                   *                                               *
02229                   *       NAME    : CHECK BUSY (LCDBSY)           *
02230                   *                                               *
02231                   *************************************************
02232                   *                                               *
02233                   *       ENTRY   : NOTHING                       *
02234                   *       RETURNS : NOTHING                       *
02235                   *                                               *
02236                   *************************************************
02237 11C4 3F 04        LCDBSY CLR    PADDR    Select port A as input
02238 11C6 B6 0C               LDA    PFDTR
02239 11C8 A4 F0               AND    #$F0     R¥W=1 RS=0 E=0
02240 11CA AA 04               ORA    #$04
02241 11CC B7 0C               STA    PFDTR
02242 11CE B6 0C        LCDBY1 LDA    PFDTR
02243 11D0 AA 01               ORA    #$01     R¥W=1 RS=0 E=1
02244 11D2 B7 0C               STA    PFDTR
02245 11D4 B6 00               LDA    PADTR    Load busy flag into ACCA
02246 11D6 48                  LSL    A        Load MSB into carry bit
02247 11D7 B6 0C               LDA    PFDTR
02248 11D9 A4 F4               AND    #$F4
02249 11DB B7 0C               STA    PFDTR    R¥W=1 RS=0 E=0
02250 11DD 25 EF               BCS    LCDBY1
02251 11DF A6 FF               LDA    #$FF     Select port A as output
02252 11E1 B7 04               STA    PADDR
02253 11E3 81                  RTS
02254                   *
02255                   *************************************************
```

```
02256                   *                                               *
02257                   *       NAME    : COUNT CALENDAR & TIME (YERCNT)*
02258                   *                                               *
02259                   ************************************************
02260                   *                                               *
02261                   *       ENTRY   : CALNDR                        *
02262                   *       RETURNS : CALNDR                        *
02263                   *                                               *
02264                   ************************************************
02265 11E4 AE 05       YERCNT LDX    #$5
02266 11E6 CD 11F4            JSR    TIMCNT   Count calendar & time
02267 11E9 81                 RTS
02268                   *
02269                   ************************************************
02270                   *                                               *
02271                   *       NAME    : COUNT ELAPSED TIME (TLCNT)    *
02272                   *                                               *
02273                   ************************************************
02274                   *                                               *
02275                   *       ENTRY   : TEL                           *
02276                   *       RETURNS : TEL                           *
02277                   *                                               *
02278                   ************************************************
02279 11EA AE 03       TLCNT  LDX    #3
02280 11EC 10 6B              BSET   0,CNTBIT Set bit
02281 11EE CD 11F4            JSR    TIMCNT   Count time
02282 11F1 11 6B              BCLR   0,CNTBIT Clear bit
02283 11F3 81                 RTS
02284                   *
02285                   ************************************************
02286                   *                                               *
02287                   *       NAME    : COUNT TIME (TIMCNT)           *
02288                   *                                               *
02289                   ************************************************
02290                   *                                               *
02291                   *       ENTRY   : CNTBIT                        *
02292                   *       RETURNS : CALNDR, TEL                   *
02293                   *                                               *
02294                   ************************************************
02295 11F4 99           TIMCNT SEC             Set carry bit
02296 11F5 A6 01        TIMCT1 LDA    #1
02297 11F7 B5 6B              BIT    CNTBIT   Elapsed time?
02298 11F9 26 4A              BNE    TIMCT2
02299 11FB 4F                 CLR    A        Clear acca
02300 11FC E9 5C              ADC    CALNDR-1,X  Count up
02301 11FE 8D           ,TIMCT3 DAA
02302 11FF 01 6B 08           BRCLR  0,CNTBIT,TIMCT4
02303 1202 B7 6A              STA    TIMSTA   Store TIMSTA
02304 1204 9F                 TXA             IX->ACCA
02305 1205 AB 02              ADD    #2       IX+2
02306 1207 97                 TAX             ACCA->IX
02307 1208 B6 6A              LDA    TIMSTA
02308 120A A3 02        TIMCT4 CPX    #$2      IX=2?
02309 120C 26 3C              BNE    TIMCT5
02310 120E B7 6A              STA    TIMSTA
```

```
02311 1210 B6 5D          LDA    CALNDR
02312 1212 44             LSR    A
02313 1213 44             LSR    A
02314 1214 44             LSR    A
02315 1215 44             LSR    A
02316 1216 48             LSL    A
02317 1217 B7 5A          STA    KEYNUM
02318 1219 48             LSL    A
02319 121A 48             LSL    A
02320 121B BB 5A          ADD    KEYNUM
02321 121D B7 5A          STA    KEYNUM
02322 121F B6 5D          LDA    CALNDR
02323 1221 A4 OF          AND    #$OF
02324 1223 BB 5A          ADD    KEYNUM
02325 1225 97             TAX
02326 1226 B6 6A          LDA    TIMSTA
02327 1228 D1 1A00        CMP    MONTH-1,X
02328 122B AE 02          LDX    #2
02329 122D 25 20    TIMCT6 BCS    TIMCT7
02330 122F 99             SEC              Bit set
02331 1230 D6 1A17        LDA    SHUNEW-1,X
02332 1233 B7 6A    TIMCT8 STA    TIMSTA   Store count data
02333 1235 A6 01          LDA    #1
02334 1237 B5 6B          BIT    CNTBIT   Elapsed time?
02335 1239 27 17          BEQ    TIMCT9
02336 123B 5A             DEC    X
02337 123C 5A             DEC    X        X-2->X
02338 123D B6 6A          LDA    TIMSTA
02339 123F E7 61          STA    TEL-1,X  Store TEL
02340 1241 5A       TIMC10 DEC    X        Decrement pointer
02341 1242 26 B1          BNE    TIMCT1   Branch if not IX=0
02342 1244 81             RTS
02343 1245 4F       TIMCT2 CLR    A        Clear ACCA
02344 1246 E9 61          ADC    TEL-1,X  Count up
02345 1248 20 B4          BRA    TIMCT3
02346 124A D1 1A12  TIMCT5 CMP    SHUSE-1,X  Compare table
02347 124D 20 DE          BRA    TIMCT6
02348 124F 98       TIMCT7 CLC
02349 1250 20 E1          BRA    TIMCT8   0->C
02350 1252 B6 6A    TIMCT9 LDA    TIMSTA
02351 1254 E7 5C          STA    CALNDR-1,X  Store CALNDR counter
02352 1256 20 E9          BRA    TIMC10
02353                *
02354                ************************************************
02355                *                                              *
02356                *       NAME    : LOAD CALENDAR & TIME DATA    *
02357                *                       INTO DISPLAY RAM (CALLCD)*
02358                *                                              *
02359                ************************************************
02360                *                                              *
02361                *       ENTRY   : CALNDR                       *
02362                *       RETURNS : LCDM                         *
02363                *                                              *
02364                ************************************************
02365 1258 CD 12AB  CALLCD JSR    LCDMST
```

```
02366 125B A6 05           LDA    #5
02367 125D B7 6D           STA    CNT       5->CNT
02368 125F A6 1B           LDA    #27
02369 1261 B7 6E           STA    CALCNT    27->CALCNT
02370 1263 CD 1276         JSR    LCDSET    Store CALN
02371 1266 81              RTS
02372                *
02373                ************************************************
02374                *                                              *
02375                *       NAME    : LOAD ELAPSED TIME DATA       *
02376                *                    INTO DISPLAY RAM(TELLCD)  *
02377                *                                              *
02378                ************************************************
02379                *                                              *
02380                *       ENTRY   : CNT, FLAG                    *
02381                *       RETURNS : LCDM                         *
02382                *                                              *
02383                ************************************************
02384 1267 CD 12AB  TELLCD JSR    LCDMST
02385 126A A6 03           LDA    #3
02386 126C B7 6D           STA    CNT       3->CNT
02387 126E A6 1B           LDA    #27
02388 1270 B7 6E           STA    CALCNT    8->CALCNT
02389 1272 CD 1276         JSR    LCDSET
02390 1275 81              RTS
02391                *
02392                ************************************************
02393                *                                              *
02394                *       NAME    : LOAD DATA INTO DISPLAY RAM   *
02395                *                                    (LCDSET)  *
02396                *                                              *
02397                ************************************************
02398                *                                              *
02399                *       ENTRY   : CNT, FLAG                    *
02400                *       RETURNS: LCDM                          *
02401                *                                              *
02402                ************************************************
02403 1276 BE 6D    LCDSET LDX    CNT
02404 1278 06 42 28 LCDSE1 BRSET  3,FLAG,LCDSE2
02405 127B E6 5C           LDA    CALNDR-1,X  Load calendar data
02406 127D A4 0F    LCDSE3 AND    #$0F
02407 127F AB 30           ADD    #'0       Data->ASCII
02408 1281 BE 6E           LDX    CALCNT
02409 1283 E7 71           STA    LCDM,X
02410 1285 3A 6E           DEC    CALCNT
02411 1287 BE 6D           LDX    CNT
02412 1289 06 42 1B        BRSET  3,FLAG,LCDSE6
02413 128C E6 5C           LDA    CALNDR-1,X
02414 128E 44       LCDSE7 LSR    A         Shift 4 bits to right
02415 128F 44              LSR    A
02416 1290 44              LSR    A
02417 1291 44              LSR    A
02418 1292 AB 30           ADD    #'0       ->ASCII
02419 1294 BE 6E           LDX    CALCNT
02420 1296 E7 71           STA    LCDM,X
```

```
02421 1298 3A 6D          DEC    CNT      Decrement CNT
02422 129A 3A 6E          DEC    CALCNT   Decrement 2 CALCNT
02423 129C 3A 6E          DEC    CALCNT
02424 129E BE 6D          LDX    CNT      Load pointer
02425 12A0 26 D6          BNE    LCDSE1
02426 12A2 81             RTS
02427 12A3 E6 61    LCDSE2 LDA    TEL-1,X  Load charge counter
02428 12A5 20 D6          BRA    LCDSE3
02429 12A7 E6 61    LCDSE6 LDA    TEL-1,X  Load charge counter
02430 12A9 20 E3          BRA    LCDSE7
02431                *
02432                ************************************************
02433                *                                              *
02434                *      NAME    : LOAD DISPLAY FORMAT (LCDMST)  *
02435                *                                              *
02436                ************************************************
02437                *                                              *
02438                *      ENTRY   : FLAG                          *
02439                *      RETURNS : LCDM                          *
02440                *                                              *
02441                ************************************************
02442 12AB CD 1966 LCDMST JSR    LCDMCR   Clear diaplay RAM
02443 12AE AE 0F          LDX    #$F      15->IX
02444 12B0 06 42 09 LCDMS1 BRSET  3,FLAG,LCDMS2  Calendar or charge?
02445 12B3 D6 19E2        LDA    CALDAT-1,X  For calendar & time display
02446 12B6 E7 7D    LCDMS3 STA    LCDM+12,X  Store in display RAM
02447 12B8 5A             DEC    X
02448 12B9 26 F5          BNE    LCDMS1
02449 12BB 81             RTS
02450 12BC D6 19F1  LCDMS2 LDA    TELDAT-1,X  For elapsed time display
02451 12BF 20 F5          BRA    LCDMS3
02452                *
02453                ****  SUBROUTINES   ****************************
02454                *
02455                *
02456                ************************************************
02457                *                                              *
02458                *      NAME    : REVIEW DATA (KENSAK)          *
02459                *                                              *
02460                ************************************************
02461                *                                              *
02462                *      ENTRY   : RFDAT, FLAG                   *
02463                *      RETURNS : FLAG, LCDM                    *
02464                *                                              *
02465                ************************************************
02466 12C1 A6 20    KENSAK LDA    #$20     Clear " * "
02467 12C3 B7 98           STA    LCDM+39
02468 12C5 19 42           BCLR   4,FLAG
02469 12C7 B6 45    KENSK1 LDA    DTDP     Load start address of the data
02470 12C9 B7 47           STA    SDP
02471 12CB B6 46           LDA    DTDP+1
02472 12CD B7 48           STA    SDP+1
02473 12CF 3F 49           CLR    DTCNT
02474 12D1 CD 187C         JSR    INPDAT   Load data from  external RAM
02475 12D4 B6 4A           LDA    DATA     Search space area
```

```
02476 12D6 26 09           BNE    KENSK2
02477 12D8 B6 41           LDA    MODFLG
02478 12DA A1 04           CMP    #$4
02479 12DC 26 55           BNE    KENSK9
02480 12DE CC 1399         JMP    KNSRTS
02481 12E1 B6 43    KENSK2 LDA    TRMFLG
02482 12E3 27 4E           BEQ    KENSK9
02483 12E5 B6 4A    KENSK3 LDA    DATA      Data = SP?
02484 12E7 A1 20           CMP    #$20
02485 12E9 26 0A           BNE    KENSK4
02486 12EB CD 18C4         JSR    DTADD
02487 12EE 3C 49           INC    DTCNT
02488 12F0 CD 187C         JSR    INPDAT
02489 12F3 20 F0           BRA    KENSK3
02490 12F5 5F       KENSK4 CLR    X
02491 12F6 D6 0115  KENSK5 LDA    REFDAT,X
02492 12F9 A1 20           CMP    #$20
02493 12FB 26 13           BNE    KENSK6
02494 12FD 5C              INC    X
02495 12FE A3 14           CPX    #20
02496 1300 26 F4           BNE    KENSK5
02497 1302 A6 10           LDA    #$10
02498 1304 B7 4C           STA    BZRCNT
02499 1306 3F 4F           CLR    BZR
02500 1308 16 4E           BSET   3,BZRFLG If all data is SP, buzzing
02501 130A CD 1966         JSR    LCDMCR
02502 130D CC 1399         JMP    KNSRTS
02503 1310 1B 42    KENSK6 BCLR   5,FLAG   Clear equal flag
02504 1312 B6 49    KENSK7 LDA    DTCNT    Store next data address
02505 1314 BB 48           ADD    SDP+1
02506 1316 B7 46           STA    DTDP+1
02507 1318 B6 47           LDA    SDP
02508 131A A9 00           ADC    #0
02509 131C B7 45           STA    DTDP
02510 131E CD 187C  KENSK8 JSR    INPDAT
02511 1321 B6 4A           LDA    DATA
02512 1323 D1 0115         CMP    REFDAT,X Compare data
02513 1326 27 3F           BEQ    KNSKEQ
02514 1328 3C 49           INC    DTCNT
02515 132A 0A 42 C8        BRSET  5,FLAG,KENSK4  Check equal flag
02516 132D B6 49           LDA    DTCNT
02517 132F A1 14           CMP    #20
02518 1331 26 DF           BNE    KENSK7
02519 1333 B6 48    KENSK9 LDA    SDP+1    Store pointer of next data
02520 1335 AB 1D           ADD    #29
02521 1337 B7 46           STA    DTDP+1
02522 1339 B6 47           LDA    SDP
02523 133B A9 00           ADC    #0
02524 133D B7 45           STA    DTDP
02525 133F 05 42 07        BRCLR  2,FLAG,KENSKA  Check search direction flag
02526 1342 CD 18CF         JSR    DTSUB
02527 1345 B6 45           LDA    DTDP
02528 1347 2B 3E           BMI    KENSKD
02529 1349 B6 45    KENSKA LDA    DTDP     Test if pointer shows last address
02530 134B A1 16           CMP    #RAMEND/256
```

```
02531 134D 27 03           BEQ    KENSKB
02532 134F CC 12C7         JMP    KENSK1
02533 1352 B6 46    KENSKB LDA    DTDP+1
02534 1354 A1 A8           CMP    #RAMEND*256/256
02535 1356 25 03           BCS    KENSKC
02536 1358 CC 12C7         JMP    KENSK1
02537 135B B6 41    KENSKC LDA    MODFLG
02538 135D A1 04           CMP    #$4
02539 135F 26 2A           BNE    KENSKE
02540 1361 10 BE           BSET   0,ERFLG  If RAM is full, display "FULL"
02541 1363 14 BE           BSET   2,ERFLG
02542 1365 20 32           BRA    KNSRTS
02543 1367 5C       KNSKEQ INC    X
02544 1368 CD 18C4         JSR    DTADD
02545 136B 1A 42           BSET   5,FLAG
02546 136D D6 0115         LDA    REFDAT,X
02547 1370 26 AC           BNE    KENSK8
02548 1372 09 42 06        BRCLR  4,FLAG,KNSEQ1
02549 1375 A6 2A           LDA    #$2A
02550 1377 B7 98           STA    LCDM+39
02551 1379 20 10           BRA    KENSKE
02552 137B 18 42    KNSEQ1 BSET   4,FLAG
02553 137D B6 47           LDA    SDP
02554 137F B7 50           STA    DTADS
02555 1381 B6 48           LDA    SDP+1
02556 1383 B7 51           STA    DTADS+1
02557 1385 20 AC           BRA    KENSK9
02558 1387 3F 45    KENSKD CLR    DTDP     Clear data table pointer
02559 1389 3F 46           CLR    DTDP+1
02560 138B 09 42 0B KENSKE BRCLR  4,FLAG,KNSRTS  Check pl. flag
02561 138E B6 50           LDA    DTADS    Display data reviewed
02562 1390 B7 45           STA    DTDP
02563 1392 B6 51           LDA    DTADS+1
02564 1394 B7 46           STA    DTDP+1
02565 1396 CD 1449         JSR    DTLCD
02566 1399 81       KNSRTS RTS
02567                      *
02568                      ************************************************
02569                      *                                              *
02570                      *      NAME    : ENTER DATA   (TOROKU)         *
02571                      *                                              *
02572                      ************************************************
02573                      *                                              *
02574                      *      ENTRY   : DTDP, LCDM                    *
02575                      *      RETURNS : NOTHING                       *
02576                      *                                              *
02577                      ************************************************
02578 139A 5F       TOROKU CLR    X        Clear pointer
02579 139B E6 71    TOROK1 LDA    LCDM,X
02580 139D A1 20           CMP    #$20     Data = SP?
02581 139F 26 0C           BNE    TOROK3
02582 13A1 A3 13           CPX    #19      SP = 20 digit?
02583 13A3 27 03           BEQ    TOROK2
02584 13A5 5C              INC    X        Increment pointer
02585 13A6 20 F3           BRA    TOROK1
```

```
02586 13A8 CD 1966  TOROK2 JSR    LCDMCR   Clear display RAM
02587 13AB 20 5B           BRA    TRKRTS
02588 13AD 5F       TOROK3 CLR    X
02589 13AE E6 71    TOROK4 LDA    LCDM,X
02590 13B0 B7 4A           STA    DATA     Store data in output RAM
02591 13B2 CD 189F         JSR    OUTDAT   Outout data
02592 13B5 5C              INC    X
02593 13B6 CD 18C4         JSR    DTADD
02594 13B9 A3 14           CPX    #20      Test if 20 digits are entered
02595 13BB 26 F1           BNE    TOROK4
02596 13BD 3F C0    MOD95  CLR    TRNS
02597 13BF A6 14           LDA    #20      Initialize pointer for external RAM
02598 13C1 B7 BF           STA    HOLD
02599 13C3 BE BF    TRKC1  LDX    HOLD
02600 13C5 E6 71           LDA    LCDM,X
02601 13C7 A1 20           CMP    #$20     Data = SP?
02602 13C9 26 04           BNE    TRKC2
02603 13CB 3C BF           INC    HOLD     If SP, increment pointer
02604 13CD 20 0F           BRA    TRKC3
02605 13CF A1 23    TRKC2  CMP    #$23
02606 13D1 26 02           BNE    TRKC5
02607 13D3 A6 2E           LDA    #$2E
02608 13D5 BE C0    TRKC5  LDX    TRNS     Load data into saving RAM
02609 13D7 D7 0115         STA    REFDAT,X
02610 13DA 3C BF           INC    HOLD     Increment pointer
02611 13DC 3C C0           INC    TRNS
02612 13DE B6 BF    TRKC3  LDA    HOLD
02613 13E0 A1 26           CMP    #38      Complete entering?
02614 13E2 26 DF           BNE    TRKC1
02615 13E4 B6 C0    TRKC4  LDA    TRNS     Complete saving?
02616 13E6 A1 12           CMP    #18
02617 13E8 27 0B           BEQ    TOROK5
02618 13EA BE C0           LDX    TRNS
02619 13EC A6 FF           LDA    #$FF
02620 13EE D7 0115         STA    REFDAT,X Load $FF into rest area
02621 13F1 3C C0           INC    TRNS
02622 13F3 20 EF           BRA    TRKC4
02623 13F5 CD 19D7  TOROK5 JSR    MOVE2    Load data
02624 13F8 AE 14           LDX    #20
02625 13FA CD 1823  TOROK6 JSR    TELNO1   Change 2bytes into 1byte
02626 13FD CD 189F         JSR    OUTDAT
02627 1400 5C              INC    X
02628 1401 CD 18C4         JSR    DTADD    Output data
02629 1404 A3 26           CPX    #38
02630 1406 26 F2           BNE    TOROK6   Test if entering is complete
02631 1408 81       TRKRTS RTS
02632                  *
02633                  ************************************************
02634                  *                                              *
02635                  *      NAME    : SEARCH SPEED DIAL NO.(TAN)    *
02636                  *                                              *
02637                  ************************************************
02638                  *                                              *
02639                  *      ENTRY   : LCDM                          *
02640                  *      RETURNS : DIALNO, MEMO, LCDM, CURADP    *
```

```
02641                    *                                                 *
02642                    ************************************************
02643 1409 B6 7A     TAN     LDA     LCDM+9    Load speed dial no.
02644 140B A4 0F             AND     #$0F
02645 140D B7 46             STA     DTDP+1
02646 140F BE 46             LDX     DTDP+1
02647 1411 A6 17             LDA     #$17
02648 1413 B7 45             STA     DTDP
02649 1415 A6 F7             LDA     #$F7
02650 1417 B7 46             STA     DTDP+1
02651 1419 AB 09     TAN1    ADD     #9        Search data in external RAM
02652 141B B7 46             STA     DTDP+1
02653 141D 5A                DEC     X
02654 141E A3 FF             CPX     #$FF
02655 1420 26 F7             BNE     TAN1
02656 1422 AE 14             LDX     #20
02657 1424 CD 187C   TAN2    JSR     INPDAT    Load speed dial no.
02658 1427 CD 183B           JSR     TELNO2    Change 1 byte to 2 bytes
02659 142A 5C                INC     X
02660 142B 5C                INC     X
02661 142C CD 18C4           JSR     DTADD
02662 142F A3 26             CPX     #38
02663 1431 26 F1             BNE     TAN2
02664 1433 CD 1476           JSR     MEMCLR    Clear memo for redialing
02665 1436 5F                CLR     X
02666 1437 E6 84     TAN3    LDA     LCDM+19,X Display speed dial no.
02667 1439 E7 99             STA     DIALNO,X
02668 143B D7 0100           STA     MEMO,X
02669 143E 5C                INC     X
02670 143F A3 14             CPX     #20
02671 1441 26 F4             BNE     TAN3
02672 1443 3F 4B             CLR     CURADP
02673 1445 CD 110E           JSR     LCD
02674 1448 81                RTS
02675                    *
02676                    ************************************************
02677                    *                                               *
02678                    *       NAME    : DISPLAY DATA FOR ONE PERSON   *
02679                    *                           (DTLCD)             *
02680                    *                                               *
02681                    ************************************************
02682                    *                                               *
02683                    *       ENTRY   : DTDP                          *
02684                    *       RETURNS : LCDM                          *
02685                    *                                               *
02686                    ************************************************
02687 1449 5F        DTLCD   CLR     X
02688 144A CD 187C   DTLCD1  JSR     INPDAT    Load name from external RAM
02689 144D B6 4A             LDA     DATA
02690 144F E7 71             STA     LCDM,X    Display name
02691 1451 CD 18C4           JSR     DTADD
02692 1454 5C                INC     X
02693 1455 A3 14             CPX     #20
02694 1457 26 F1             BNE     DTLCD1
02695 1459 CD 187C   DTLCD2  JSR     INPDAT    Load dial no. from external RAM
```

```
02696 145C CD 1838          JSR     TELNO2   Change 2 bytes to 1 byte
02697 145F CD 18C4          JSR     DTADD
02698 1462 5C               INC     X
02699 1463 5C               INC     X
02700 1464 A3 26            CPX     #38
02701 1466 26 F1            BNE     DTLCD2   Test if input data is complete
02702 1468 CD 110E          JSR     LCD      Display name and telephone no.
02703 146B 81               RTS
02704                  *
02705                  ************************************************
02706                  *                                              *
02707                  *       NAME    : CLEAR DIAL NUMBER (DIACLR)   *
02708                  *                                              *
02709                  ************************************************
02710                  *                                              *
02711                  *       ENTRY   : NOTHING                      *
02712                  *       RETURNS : DIALNO                       *
02713                  *                                              *
02714                  ************************************************
02715 146C AE 15       DIACLR LDX     #21
02716 146E A6 20       DIACL1 LDA     #$20
02717 1470 E7 98            STA     DIALNO-1,X  Clear dial no.
02718 1472 5A               DEC     X
02719 1473 26 F9            BNE     DIACL1
02720 1475 81               RTS
02721                  *
02722                  ************************************************
02723                  *                                              *
02724                  *       NAME    : CLEAR MEMO FOR REDIALING     *
02725                  *                                 (MEMCLR)     *
02726                  *                                              *
02727                  ************************************************
02728                  *                                              *
02729                  *       ENTRY   : NOTHING                      *
02730                  *       RETURNS : MEMO                         *
02731                  *                                              *
02732                  ************************************************
02733 1476 AE 15       MEMCLR LDX     #21
02734 1478 A6 20       MEMCL1 LDA     #$20
02735 147A E7 FF            STA     MEMO-1,X Clear memo for redialing
02736 147C 5A               DEC     X
02737 147D 26 F9            BNE     MEMCL1
02738 147F 81               RTS
02739                  *
02740                  ************************************************
02741                  *                                              *
02742                  *       NAME    : CLEAR DISPLAY RAM (CLEAR)    *
02743                  *                                              *
02744                  ************************************************
02745                  *                                              *
02746                  *       ENTRY   : NOTHING                      *
02747                  *       RETURNS : LCDM                         *
02748                  *                                              *
02749                  ************************************************
02750 1480 AE 28       CLEAR  LDX     #40
```

```
02751 1482 4F       CLR1   CLR    A
02752 1483 E7 70           STA    LCDM-1,X Clear display RAM
02753 1485 5A              DEC    X
02754 1486 26 FA           BNE    CLR1
02755 1488 81              RTS
02756                 *
02757                 ************************************************
02758                 *                                              *
02759                 *      NAME    : CLEAR ELAPSED TIME (TELCLR)   *
02760                 *                                              *
02761                 ************************************************
02762                 *                                              *
02763                 *      ENTRY   : NOTHING                       *
02764                 *      RETURNS : TEL                           *
02765                 *                                              *
02766                 ************************************************
02767 1489 AE 03     TELCLR LDX    #3
02768 148B 6F 61    TELCL1 CLR    TEL-1,X  Clear elapsed time counter
02769 148D 5A              DEC    X
02770 148E 26 FB           BNE    TELCL1
02771 1490 81              RTS
02772                 *
02773                 ************************************************
02774                 *                                              *
02775                 *      NAME    : LOAD CALENDAR & TIME DATA     *
02776                 *                    INTO DIAPLAY RAM (SHUCAL) *
02777                 *                                              *
02778                 ************************************************
02779                 *                                              *
02780                 *      ENTRY   : CALNDR, CURADP                *
02781                 *      RETURNS : LCDM                          *
02782                 *                                              *
02783                 ************************************************
02784 1491 17 42    SHUCAL BCLR   3,FLAG   Clear flag
02785 1493 CD 1258         JSR    CALLCD
02786 1496 A6 0F           LDA    #$F
02787 1498 B7 4B           STA    CURADP   Define cursor position
02788 149A A6 0E           LDA    #14
02789 149C B7 40           STA    LCDMP
02790 149E 81              RTS
02791                 *
02792                 ************************************************
02793                 *                                              *
02794                 *      NAME    : CHECK CALNDAR & TIME (CALSHU) *
02795                 *                                              *
02796                 ************************************************
02797                 *                                              *
02798                 *      ENTRY   : LCDM                          *
02799                 *      RETURNS : CALNDR, TELFLG, ERFLG         *
02800                 *                                              *
02801                 ************************************************
02802 149F A6 02    CALSHU LDA    #2
02803 14A1 B7 6D           STA    CNT      Set digit counter
02804 14A3 3F 6F           CLR    SHUCNT   Clear pointer
02805 14A5 3F 6E           CLR    CALCNT
```

```
02806 14A7 BE 6E           LDX    CALCNT
02807 14A9 4F       CALSH1 CLR    A         Clear ACCA
02808 14AA 98              CLC               Clear bit C
02809 14AB E6 7E           LDA    LCDM+13,X
02810 14AD A1 30           CMP    #'0       Key data = 0-9?
02811 14AF 25 6D           BCS    CALS10
02812 14B1 AB C6           ADD    #$C6
02813 14B3 25 69           BCS    CALS10
02814 14B5 AB 0A           ADD    #$0A      Convert ASCII into BCD
02815 14B7 BE 6D           LDX    CNT       Load digit counter
02816 14B9 27 22           BEQ    CALSH2    Check digit
02817 14BB 48              LSL    A         Shift 4 bits left
02818 14BC 48              LSL    A
02819 14BD 48              LSL    A
02820 14BE 48              LSL    A
02821 14BF BE 6F           LDX    SHUCNT
02822 14C1 E7 65           STA    SHUSTA,X  Store modified data
02823 14C3 3A 6D    CALSH3 DEC    CNT       Decrement digit pointer
02824 14C5 3C 6E    CALSH4 INC    CALCNT    LCDM+13 increment
02825 14C7 BE 6E           LDX    CALCNT
02826 14C9 A3 0F           CPX    #$F       CALCNT=15?
02827 14CB 26 DC           BNE    CALSH1
02828 14CD AE 05           LDX    #5
02829 14CF E6 64    CALSH5 LDA    SHUSTA-1,X  Load SHUSE DATA
02830 14D1 A3 03           CPX    #3        MONTH and DAY ?
02831 14D3 24 02           BCC    CALSH7
02832 14D5 27 3E           BEQ    CALSH8    Error
02833 14D7 E7 5C    CALSH7 STA    CALNDR-1,X  store CALNDR
02834 14D9 5A              DEC    X
02835 14DA 26 F3           BNE    CALSH5    IX=0 ?
02836 14DC 81       CALSH9 RTS
02837 14DD BE 6F    CALSH2 LDX    SHUCNT
02838 14DF EB 65           ADD    SHUSTA,X  1st digit?
02839 14E1 A3 01           CPX    #1
02840 14E3 27 0F           BEQ    DAY
02841 14E5 D1 1A13         CMP    SHUSE,X
02842 14E8 24 2B    CALSH6 BCC    CALSH8    Error
02843 14EA E7 65           STA    SHUSTA,X  SHUSEdata->SHUSESTA
02844 14EC A6 02           LDA    #2
02845 14EE B7 6D           STA    CNT       2->CNT
02846 14F0 3C 6F           INC    SHUCNT    Increment SHUCNT
02847 14F2 20 D1           BRA    CALSH4
02848 14F4 B7 BF    DAY    STA    HOLD
02849 14F6 B6 65           LDA    SHUSTA
02850 14F8 44              LSR    A
02851 14F9 44              LSR    A
02852 14FA 44              LSR    A
02853 14FB 44              LSR    A
02854 14FC 48              LSL    A
02855 14FD B7 C0           STA    TRNS
02856 14FF 48              LSL    A
02857 1500 48              LSL    A
02858 1501 BB C0           ADD    TRNS
02859 1503 B7 C0           STA    TRNS
02860 1505 B6 65           LDA    SHUSTA
```

```
02861 1507 A4 0F          AND    #$0F
02862 1509 BB C0          ADD    TRNS
02863 150B 97             TAX
02864 150C B6 BF          LDA    HOLD
02865 150E D1 1A00        CMP    MONTH-1,X  MONTH data table
02866 1511 AE 01          LDX    #1         1->IX
02867 1513 20 D3          BRA    CALSH6
02868 1515 4F       CALSH8 CLR    A          Prepare for error process
02869 1516 B7 4F          STA    BZR
02870 1518 B7 43          STA    TRMFLG
02871 151A 10 BE          BSET   0,ERFLG
02872 151C 20 BE          BRA    CALSH9
02873 151E B6 6D    CALS10 LDA    CNT
02874 1520 A1 02          CMP    #2
02875 1522 26 F1          BNE    CALSH8
02876 1524 20 9D          BRA    CALSH3
02877                *
02878                ************************************************
02879                *                                              *
02880                *       NAME    : DISPLAY RETRYING NUMBER      *
02881                *                                 (DIAL2)      *
02882                *                                              *
02883                ************************************************
02884                *                                              *
02885                *       ENTRY   : DIANO2, MEMO                 *
02886                *       RETURNS : LCDM                         *
02887                *                                              *
02888                ************************************************
02889 1526 AE 0E    DIAL2  LDX    #14
02890 1528 D6 1A55  DIAL21 LDA    DIANO2-1,X  Display "RETRY"
02891 152B E7 71          STA    LCDM,X
02892 152D 5A             DEC    X
02893 152E 26 F8          BNE    DIAL21
02894 1530 5F             CLR    X
02895 1531 D6 0100  DIAL22 LDA    MEMO,X   Load retry no.
02896 1534 E7 85          STA    LCDM+20,X
02897 1536 E7 99          STA    DIALNO,X
02898 1538 5C             INC    X
02899 1539 A3 14          CPX    #20
02900 153B 26 F4          BNE    DIAL22
02901 153D CD 110E        JSR    LCD      Display retry no.
02902 1540 81             RTS
02903                *
02904                ************************************************
02905                *                                              *
02906                *       NAME    : INITIALIZE INTERNAL RAM      *
02907                *                                 (ARGINT)     *
02908                *                                              *
02909                ************************************************
02910                *                                              *
02911                *       ENTRY   : NOTHING                      *
02912                *       RETURNS : NOTHING                      *
02913                *                                              *
02914                ************************************************
02915 1541 9B       ARGINT SEI
```

```
02916 1542 CD 1172          JSR    LCDRES
02917 1545 CD 1195          JSR    LCDINT   Initialize LCD
02918 1548 9A               CLI
02919 1549 CD 1489          JSR    TELCLR   Clear TEL counter
02920 154C CD 146C          JSR    DIACLR
02921 154F CD 1476          JSR    MEMCLR
02922 1552 4F               CLR    A
02923 1553 B7 6C            STA    POINTR   Clear RAM
02924 1555 B7 6E            STA    CALCNT
02925 1557 B7 6D            STA    CNT
02926 1559 B7 4B            STA    CURADP
02927 155B B7 6F            STA    SHUCNT
02928 155D C7 0137          STA    SHUSEC
02929 1560 B7 6B            STA    CNTBIT
02930 1562 B7 44            STA    TELFLG
02931 1564 B7 B6            STA    TELWAT
02932 1566 C7 0139          STA    TELNO
02933 1569 C7 013A          STA    TELCNT
02934 156C B7 B4            STA    REC
02935 156E B7 B5            STA    MFDAT
02936 1570 B7 42            STA    FLAG
02937 1572 B7 AE            STA    WAIT1
02938 1574 B7 AF            STA    WAIT2
02939 1576 B7 B0            STA    WAIT3
02940 1578 B7 B1            STA    WAIT4
02941 157A B7 B2            STA    WAIT5
02942 157C B7 B3            STA    MEMOP
02943 157E B7 4E            STA    BZRFLG
02944 1580 B7 4F            STA    BZR
02945 1582 B7 BC            STA    DIAP1
02946 1584 B7 BD            STA    DIAP2
02947 1586 C7 013B          STA    ERR1
02948 1589 C7 013C          STA    ERR2
02949 158C B7 BE            STA    ERFLG
02950 158E AE 04            LDX    #4
02951 1590 4F         ARG16 CLR    A
02952 1591 D7 0128          STA    ADDRAT-1,X  Clear ADDRAT
02953 1594 5A               DEC    X
02954 1595 26 F9            BNE    ARG16
02955 1597 AE 02            LDX    #2
02956 1599 4F         ARG17 CLR    A
02957 159A E7 51            STA    SEC-1,X  Clear SEC
02958 159C 5A               DEC    X
02959 159D 26 FA            BNE    ARG17
02960 159F AE 02            LDX    #2
02961 15A1 4F         ARG18 CLR    A
02962 15A2 D7 0134          STA    SECCNT-1,X  Clear SECCNT
02963 15A5 5A               DEC    X
02964 15A6 26 F9            BNE    ARG18
02965 15A8 CD 17A7          JSR    TOLCLR   Clear TOTAL
02966 15AB CD 17B1          JSR    SNGCLR   Clear SINGL
02967 15AE AE 05            LDX    #5
02968 15B0 6F 5C      ARG19 CLR    CALNDR-1,X
02969 15B2 5A               DEC    X
02970 15B3 26 FB            BNE    ARG19
```

```
02971 15B5 AE 04           LDX    #4
02972 15B7 A6 01     ARG100 LDA    #1
02973 15B9 E7 5C           STA    CALNDR-1,X
02974 15BB 5A              DEC    X
02975 15BC 26 F9           BNE    ARG100
02976 15BE AE 05           LDX    #5
02977 15C0 A6 20     ARG110 LDA    #$20
02978 15C2 E7 B6           STA    DIADAT-1,X
02979 15C4 5A              DEC    X
02980 15C5 26 F9           BNE    ARG110
02981 15C7 CD 1001         JSR    CANCEL   Initialize for telephone circuit
02982 15CA CD 1038         JSR    CALRST
02983 15CD 81              RTS
02984                *
02985                ************************************************
02986                *                                              *
02987                *      NAME    : COUNT CHARGE (MONEY)          *
02988                *                                              *
02989                ************************************************
02990                *                                              *
02991                *      ENTRY   : ADDRAT, SEC                   *
02992                *      RETURNS : SINGL, TOTAL                  *
02993                *                                              *
02994                ************************************************
02995 15CE AE 02     MONEY  LDX    #2       2->IX
02996 15D0 99               SEC             1->C
02997 15D1 4F        MONEY1 CLR    A
02998 15D2 D9 0134          ADC    SECCNT-1,X  1 sec up
02999 15D5 8D               DAA
03000 15D6 D7 0134          STA    SECCNT-1,X
03001 15D9 5A               DEC    X
03002 15DA 26 F5            BNE    MONEY1   IX=0?
03003 15DC B6 53            LDA    SEC+1    SEC=SECCNT?
03004 15DE C1 0136          CMP    SECCNT+1
03005 15E1 26 2E            BNE    MONEY4
03006 15E3 B6 52            LDA    SEC
03007 15E5 C1 0135          CMP    SECCNT
03008 15E8 26 27            BNE    MONEY4
03009 15EA 4F               CLR    A
03010 15EB C7 0136          STA    SECCNT+1 Clear SECCNT
03011 15EE C7 0135          STA    SECCNT
03012 15F1 AE 04            LDX    #4
03013 15F3 98               CLC             Clear bit C
03014 15F4 D6 0128   MONEY2 LDA    ADDRAT-1,X  Rate+accumulated charge
03015 15F7 D9 0130          ADC    TOTAL-1,X
03016 15FA 8D               DAA
03017 15FB D7 0130          STA    TOTAL-1,X
03018 15FE 5A               DEC    X
03019 15FF 26 F3            BNE    MONEY2   IX=0?
03020 1601 AE 04            LDX    #4
03021 1603 98               CLC             0->C
03022 1604 D6 0128   MONEY3 LDA    ADDRAT-1,X
03023 1607 D9 012C          ADC    SINGL-1,X  rate+SINGL CHARGE
03024 160A 8D               DAA
03025 160B D7 012C          STA    SINGL-1,X
```

```
03026 160E 5A              DEC     X
03027 160F 26 F3           BNE     MONEY3
03028 1611 81        MONEY4 RTS
03029                     *
03030                     ************************************************
03031                     *                                              *
03032                     *       NAME    : CHANGE TO ASCII (MNYASC)     *
03033                     *                                              *
03034                     ************************************************
03035                     *                                              *
03036                     *       ENTRY   : TOTAL, SINGL, ADDRAT, SEC    *
03037                     *       RETURNS : LCDM                         *
03038                     *                                              *
03039                     ************************************************
03040 1612 BE 6D     MNYASC LDX     CNT
03041 1614 04 6B 37  MNYAS1 BRSET   2,CNTBIT,MNYAS6  TOTAL->AccA
03042 1617 06 6B 39         BRSET   3,CNTBIT,MNYAS7  SINGL->ACCA
03043 161A 08 6B 3B         BRSET   4,CNTBIT,MNYAS8  ADDRAT->ACCA
03044 161D 0A 6B 3D         BRSET   5,CNTBIT,MNYAS9  SEC->ACCA
03045 1620 A4 0F     MNYAS2 AND     #$0F
03046 1622 AB 30            ADD     #'0
03047 1624 BE 6E            LDX     CALCNT
03048 1626 E7 71            STA     LCDM,X
03049 1628 3A 6E            DEC     CALCNT
03050 162A BE 6D            LDX     CNT
03051 162C 04 6B 32         BRSET   2,CNTBIT,MNYA13
03052 162F 06 6B 34         BRSET   3,CNTBIT,MNYA14
03053 1632 08 6B 36         BRSET   4,CNTBIT,MNYA15
03054 1635 0A 6B 38         BRSET   5,CNTBIT,MNYA16
03055 1638 44        MNYAS4 LSR     A
03056 1639 44               LSR     A
03057 163A 44               LSR     A
03058 163B 44               LSR     A
03059 163C AB 30            ADD     #'0
03060 163E BE 6E            LDX     CALCNT
03061 1640 0A 6B 35         BRSET   5,CNTBIT,MNYA17
03062 1643 E7 71     MNYA10 STA     LCDM,X
03063 1645 3A 6E     MNYAS5 DEC     CALCNT
03064 1647 3A 6D            DEC     CNT
03065 1649 BE 6D            LDX     CNT
03066 164B 26 C7            BNE     MNYAS1
03067 164D 81               RTS
03068 164E D6 0130   MNYAS6 LDA     TOTAL-1,X
03069 1651 20 CD            BRA     MNYAS2
03070 1653 D6 012C   MNYAS7 LDA     SINGL-1,X
03071 1656 20 C8            BRA     MNYAS2
03072 1658 D6 0128   MNYAS8 LDA     ADDRAT-1,X
03073 165B 20 C3            BRA     MNYAS2
03074 165D E6 51     MNYAS9 LDA     SEC-1,X
03075 165F 20 BF            BRA     MNYAS2
03076 1661 D6 0130   MNYA13 LDA     TOTAL-1,X
03077 1664 20 D2            BRA     MNYAS4
03078 1666 D6 012C   MNYA14 LDA     SINGL-1,X
03079 1669 20 CD            BRA     MNYAS4
03080 166B D6 0128   MNYA15 LDA     ADDRAT-1,X
```

```
03081 166E 20 C8           BRA    MNYAS4
03082 1670 A3 01     MNYA16 CPX    #1
03083 1672 27 C4           BEQ    MNYAS4
03084 1674 E6 51           LDA    SEC-1,X
03085 1676 20 C0           BRA    MNYAS4
03086 1678 A3 1B     MNYA17 CPX    #27
03087 167A 27 C9           BEQ    MNYAS5
03088 167C 20 C5           BRA    MNYA10
03089                      *
03090                      ************************************************
03091                      *                                              *
03092                      *       NAME    : LOAD CHARGE DATA INTO        *
03093                      *                 DISPLAY RAM (SNGSTA)         *
03094                      *                                              *
03095                      ************************************************
03096                      *                                              *
03097                      *       ENTRY   : NOTHING                      *
03098                      *       RETURNS : LCDM                         *
03099                      *                                              *
03100                      ************************************************
03101 167E CD 1966   SNGSTA JSR    LCDMCR
03102 1681 A6 02           LDA    #2
03103 1683 B7 6D           STA    CNT
03104 1685 A6 16           LDA    #22
03105 1687 B7 6E           STA    CALCNT
03106 1689 1A 6B           BSET   5,CNTBIT
03107 168B CD 1612         JSR    MNYASC
03108 168E 1B 6B           BCLR   5,CNTBIT
03109 1690 AE 0D           LDX    #13
03110 1692 D6 1A2B   SNGS1  LDA    RATDAT-1,X
03111 1695 E7 72           STA    LCDM+1,X
03112 1697 5A              DEC    X
03113 1698 26 F8           BNE    SNGS1
03114 169A AE 04           LDX    #4
03115 169C D6 1A38   SNGS2  LDA    SECDAT-1,X
03116 169F E7 80           STA    LCDM+15,X
03117 16A1 5A              DEC    X
03118 16A2 26 F8           BNE    SNGS2
03119 16A4 AE 0F           LDX    #15
03120 16A6 D6 1A1C   SNGS3  LDA    CRGDAT-1,X
03121 16A9 E7 88           STA    LCDM+23,X
03122 16AB 5A              DEC    X
03123 16AC 26 F8           BNE    SNGS3
03124 16AE A6 04           LDA    #4          4->CNT
03125 16B0 B7 6D           STA    CNT
03126 16B2 A6 0E           LDA    #14         13->CALCNT
03127 16B4 B7 6E           STA    CALCNT
03128 16B6 18 6B           BSET   4,CNTBIT RATE SET
03129 16B8 CD 1612         JSR    MNYASC      data->ASCII
03130 16BB 19 6B           BCLR   4,CNTBIT
03131 16BD A6 04           LDA    #4
03132 16BF B7 6D           STA    CNT
03133 16C1 A6 26           LDA    #38
03134 16C3 B7 6E           STA    CALCNT
03135 16C5 1A 6C           BSET   3,CNTBIT CHARGE SET
```

```
03136 16C7 CD 1612          JSR    MNYASC   data->ASCII
03137 16CA 17 6B            BCLR   3,CNTBIT
03138 16CC 81               RTS
03139                  *
03140                  ************************************************
03141                  *                                              *
03142                  *      NAME    : LOAD ACCUMULATED CHARGE       *
03143                  *                    INTO DISPLAY RAM (TOLSTA) *
03144                  *                                              *
03145                  ************************************************
03146                  *                                              *
03147                  *      ENTRY   : TOTAL                         *
03148                  *      RETURNS : LCDM                          *
03149                  *                                              *
03150                  ************************************************
03151 16CD CD 1966  TOLSTA JSR    LCDMCR
03152 16D0 AE 0F            LDX    #15
03153 16D2 D6 1A1C  TOLST1 LDA    CRGDAT-1,X
03154 16D5 E7 7D            STA    LCDM+12,X
03155 16D7 5A               DEC    X
03156 16D8 26 F8            BNE    TOLST1
03157 16DA A6 04            LDA    #4
03158 16DC B7 6D            STA    CNT
03159 16DE A6 1B            LDA    #27
03160 16E0 B7 6E            STA    CALCNT
03161 16E2 14 6B            BSET   2,CNTBIT Set charge
03162 16E4 CD 1612          JSR    MNYASC
03163 16E7 15 6B            BCLR   2,CNTBIT
03164 16E9 81               RTS
03165                  *
03166                  ************************************************
03167                  *                                              *
03168                  *      NAME    : SET CHARGE PER UNIT TIME      *
03169                  *                                (CRGSET)      *
03170                  *                                              *
03171                  ************************************************
03172                  *                                              *
03173                  *      ENTRY   : ADDRAT, SEC                   *
03174                  *      RETURNS : LCDM                          *
03175                  *                                              *
03176                  ************************************************
03177 16EA CD 1966  CRGSET JSR    LCDMCR
03178 16ED A6 A0            LDA    #$A0
03179 16EF B7 88            STA    LCDM+23
03180 16F1 A6 04            LDA    #4
03181 16F3 B7 6D            STA    CNT
03182 16F5 A6 16            LDA    #22
03183 16F7 B7 6E            STA    CALCNT
03184 16F9 18 6B            BSET   4,CNTBIT Set rate
03185 16FB CD 1612          JSR    MNYASC   Data->ASCII
03186 16FE 19 6B            BCLR   4,CNTBIT
03187 1700 A6 02            LDA    #2
03188 1702 B7 6D            STA    CNT
03189 1704 A6 1E            LDA    #30
03190 1706 B7 6E            STA    CALCNT
```

```
03191 1708 1A 6B           BSET   5,CNTBIT Set rate
03192 170A CD 1612          JSR    MNYASC   Data->ASCII
03193 170D 1B 6B            BCLR   5,CNTBIT
03194 170F AE 05            LDX    #5
03195 1711 D6 1A2B  CRGST1  LDA    RATDAT-1,X
03196 1714 E7 7A            STA    LCDM+9,X
03197 1716 5A               DEC    X
03198 1717 26 F8            BNE    CRGST1
03199 1719 AE 04            LDX    #4
03200 171B D6 1A38  CRGST2  LDA    SECDAT-1,X
03201 171E E7 88            STA    LCDM+23,X
03202 1720 5A               DEC    X
03203 1721 26 F8            BNE    CRGST2
03204 1723 A6 10            LDA    #16
03205 1725 B7 4B            STA    CURADP
03206 1727 A6 0F            LDA    #15
03207 1729 B7 40            STA    LCDMP
03208 172B 81               RTS
03209                *
03210                ************************************************
03211                *                                              *
03212                *      NAME    : STORE RATE COUNTER (RATSET)   *
03213                *                                              *
03214                ************************************************
03215                *                                              *
03216                *      ENTRY   : LCDM                          *
03217                *      RETURNS : ADDRAT, SEC, TRMFLG, ERFLG    *
03218                *                                              *
03219                ************************************************
03220 172C 3F 6F    RATSET CLR    SHUCNT
03221 172E 3F 6D           CLR    CNT      Clear counter
03222 1730 4F              CLR    A
03223 1731 BE 6F           LDX    SHUCNT
03224 1733 E6 80    RATSE1 LDA    LCDM+15,X  Store ASCII
03225 1735 A1 30           CMP    #'0      ASCII>0 ?
03226 1737 25 65           BCS    RATSE5
03227 1739 AB C6           ADD    #$C6     ASCII<9 ?
03228 173B 25 61           BCS    RATSE5
03229 173D AB 0A           ADD    #$0A     0-ASCII-9
03230 173F 48              LSL    A
03231 1740 48              LSL    A        Second digit
03232 1741 48              LSL    A
03233 1742 48              LSL    A
03234 1743 BE 6D           LDX    CNT
03235 1745 E7 65           STA    SHUSTA,X Store SHUSTA
03236 1747 3C 6F           INC    SHUCNT
03237 1749 BE 6F           LDX    SHUCNT
03238 174B E6 80           LDA    LCDM+15,X  Store ASCII
03239 174D A1 30           CMP    #'0      ASCII>0 ?
03240 174F 25 4D           BCS    RATSE5
03241 1751 AB C6           ADD    #$C6     ASCII<9 ?
03242 1753 25 49           BCS    RATSE5
03243 1755 AB 0A           ADD    #$0A
03244 1757 BE 6D           LDX    CNT
03245 1759 EB 65           ADD    SHUSTA,X
```

```
03246 175B E7 65           STA    SHUSTA,X
03247 175D 3C 6F           INC    SHUCNT
03248 175F 3C 6D           INC    CNT
03249 1761 BE 6F           LDX    SHUCNT
03250 1763 A3 08           CPX    #8          Rate end ?
03251 1765 26 CC           BNE    RATSE1
03252 1767 B6 8D           LDA    LCDM+28     SEC ASCII->deta
03253 1769 A0 30           SUB    #'0         Third digit
03254 176B 25 31           BCS    RATSE5
03255 176D C7 0137         STA    SHUSEC
03256 1770 B6 8E           LDA    LCDM+29
03257 1772 A0 30           SUB    #'0
03258 1774 25 28           BCS    RATSE5
03259 1776 48              LSL    A           Second digit
03260 1777 48              LSL    A
03261 1778 48              LSL    A
03262 1779 48              LSL    A
03263 177A C7 0138         STA    SHUSEC+1
03264 177D B6 8F           LDA    LCDM+30
03265 177F A0 30           SUB    #'0         First digit
03266 1781 25 1B           BCS    RATSE5
03267 1783 CB 0138         ADD    SHUSEC+1
03268 1786 C7 0138         STA    SHUSEC+1
03269 1789 AE 04           LDX    #4
03270 178B E6 64    RATSE2 LDA    SHUSTA-1,X
03271 178D D7 0128         STA    ADDRAT-1,X  Store ADDRAT
03272 1790 5A              DEC    X
03273 1791 26 F8           BNE    RATSE2
03274 1793 AE 02           LDX    #2
03275 1795 D6 0136  RATSE3 LDA    SHUSEC-1,X
03276 1798 E7 51           STA    SEC-1,X     Store SEC
03277 179A 5A              DEC    X
03278 179B 26 F8           BNE    RATSE3
03279 179D 81       RATSE4 RTS
03280 179E 4F       RATSE5 CLR    A
03281 179F B7 4F           STA    BZR         Clear BZR
03282 17A1 B7 43           STA    TRMFLG      Clear TRMFLG
03283 17A3 10 BE           BSET   0,ERFLG     Set error process request flag
03284 17A5 20 F6           BRA    RATSE4
03285               *
03286               ************************************************
03287               *                                              *
03288               *       NAME    : CLEAR ACCUMULATED CHARGE     *
03289               *                                (TOLCLR)      *
03290               *                                              *
03291               ************************************************
03292               *                                              *
03293               *       ENTRY   : NOTHING                      *
03294               *       RETURNS : TOTAL                        *
03295               *                                              *
03296               ************************************************
03297 17A7 AE 04    TOLCLR LDX    #4
03298 17A9 4F       TOLCL1 CLR    A
03299 17AA D7 0130         STA    TOTAL-1,X   Clear accumulated charge
03300 17AD 5A              DEC    X
```

```
03301 17AE 26 F9           BNE    TOLCL1
03302 17B0 81              RTS
03303                 *
03304                 ************************************************
03305                 *                                              *
03306                 *      NAME    : CLEAR CHARGE (SNGCLR)         *
03307                 *                                              *
03308                 ************************************************
03309                 *                                              *
03310                 *      ENTRY   : NOTHING                       *
03311                 *      RETURNS : SINGL                         *
03312                 *                                              *
03313                 ************************************************
03314 17B1 AE 04     SNGCLR LDX    #4
03315 17B3 4F        SNGCL1 CLR    A
03316 17B4 D7 012C         STA    SINGL-1,X  Clear charge
03317 17B7 5A              DEC    X
03318 17B8 26 F9           BNE    SNGCL1
03319 17BA 81              RTS
03320                 *
03321                 ************************************************
03322                 *                                              *
03323                 *      NAME    : MOVE CURSOR FOR SETTING CHARGE*
03324                 *                                (CURRAT)      *
03325                 *                                              *
03326                 ************************************************
03327                 *                                              *
03328                 *      ENTRY   : LCDMP                         *
03329                 *      RETURNS : LCDMP, CURADP                 *
03330                 *                                              *
03331                 ************************************************
03332 17BB B6 40     CURRAT LDA    LCDMP    Load LCDM pointer
03333 17BD A1 0E           CMP    #14      LCDMP=14 ?
03334 17BF 27 11           BEQ    CURRA1
03335 17C1 A1 17           CMP    #23      LCDMP=23 ?
03336 17C3 27 13           BEQ    CURRA2
03337 17C5 A1 1B           CMP    #27      LCDMP=27 ?
03338 17C7 27 19           BEQ    CURRA3
03339 17C9 A1 1F           CMP    #31      LCDMP=31 ?
03340 17CB 26 04           BNE    CURRA4
03341 17CD 3A 40           DEC    LCDMP    30->LCDMP
03342 17CF 3A 4B           DEC    CURADP   31->CURADP
03343 17D1 81        CURRA4 RTS
03344 17D2 3C 40     CURRA1 INC    LCDMP    15->LCDMP
03345 17D4 3C 4B           INC    CURADP   16->CURADP
03346 17D6 20 F9           BRA    CURRA4
03347 17D8 A6 1C     CURRA2 LDA    #28      28->LCDMP
03348 17DA B7 40           STA    LCDMP
03349 17DC A6 1D           LDA    #29      29->CURADP
03350 17DE B7 4B           STA    CURADP
03351 17E0 20 EF           BRA    CURRA4
03352 17E2 A6 16     CURRA3 LDA    #22      22->LCDMP
03353 17E4 B7 40           STA    LCDMP
03354 17E6 A6 17           LDA    #23      23->CURADP
03355 17E8 B7 4B           STA    CURADP
```

```
03356 17EA 20 E5          BRA    CURRA4
03357                  *
03358                  ************************************************
03359                  *                                              *
03360                  *      NANE    : MOVE CURSOR FOR SETTING       *
03361                  *                  CALENDAR & CHARGE (CURCAL)  *
03362                  *                                              *
03363                  ************************************************
03364                  *                                              *
03365                  *      ENTRY   : LCDMP, KEYDAT                 *
03366                  *      RETURNS : LCDMP, CURADP                 *
03367                  *                                              *
03368                  ************************************************
03369 17EC B6 40     CURCAL LDA    LCDMP    Load LCDM pointer
03370 17EE A1 0D          CMP    #13      LCDMP=13 ?
03371 17F0 27 19          BEQ    CURCL1
03372 17F2 A1 10          CMP    #16      LCDMP=16 ?
03373 17F4 27 1B          BEQ    CURCL2
03374 17F6 A1 13          CMP    #19      LCDMP=19 ?
03375 17F8 27 17          BEQ    CURCL2
03376 17FA A1 16          CMP    #22      LCDMP=22 ?
03377 17FC 27 13          BEQ    CURCL2
03378 17FE A1 19          CMP    #25      LCDMP=25 ?
03379 1800 27 0F          BEQ    CURCL2
03380 1802 A1 1C          CMP    #28      LCDMP=28 ?
03381 1804 26 04          BNE    CURCL5
03382 1806 3A 4B          DEC    CURADP   28->CURADP
03383 1808 3A 40          DEC    LCDMP    27->LCDMP
03384 180A 81          CURCL5 RTS
03385 180B 3C 40       CURCL1 INC    LCDMP    14->LCDMP
03386 180D 3C 4B          INC    CURADP   15->CURADP
03387 180F 20 F9          BRA    CURCL5
03388 1811 B6 58       CURCL2 LDA    KEYDAT   RIGHT key or LEFT key ?
03389 1813 A1 29          CMP    #$29
03390 1815 27 06          BEQ    CURCL3
03391 1817 3C 40          INC    LCDMP    LCDMP+1
03392 1819 3C 4B          INC    CURADP   CURADP+1
03393 181B 20 ED          BRA    CURCL5
03394 181D 3A 40       CURCL3 DEC    LCDMP    LCDMP-1
03395 181F 3A 4B          DEC    CURADP   CURADP-1
03396 1821 20 E7          BRA    CURCL5
03397                  *
03398                  ************************************************
03399                  *                                              *
03400                  *      NAME    : CONVERT 2 BYTES INTO 1 BYTE   *
03401                  *                        (TELNO1)              *
03402                  *                                              *
03403                  ************************************************
03404                  *                                              *
03405                  *      ENTRY   : LCDM                          *
03406                  *      RETURNS : DATA                          *
03407                  *                                              *
03408                  ************************************************
03409 1823 E6 71       TELNO1 LDA    LCDM,X   Load data
03410 1825 A4 0F          AND    #$0F
```

```
03411 1827 E7 71            STA     LCDM,X
03412 1829 5C               INC     X
03413 182A E6 71            LDA     LCDM,X
03414 182C A1 20            CMP     #$20
03415 182E 26 02            BNE     TEL11
03416 1830 A6 FF            LDA     #$FF
03417 1832 48         TEL11 ASL     A
03418 1833 48               ASL     A
03419 1834 48               ASL     A
03420 1835 48               ASL     A
03421 1836 EB 70            ADD     LCDM-1,X
03422 1838 B7 4A            STA     DATA      Store data in output RAM
03423 183A 81               RTS
03424                 *
03425                 ************************************************
03426                 *                                              *
03427                 *       NAME    : CONVERT 1 BYTE INTO 2 BYTES  *
03428                 *                       (TELNO2)               *
03429                 *                                              *
03430                 ************************************************
03431                 *                                              *
03432                 *       ENTRY   : DATA                         *
03433                 *       RETURNS : LCDM                         *
03434                 *                                              *
03435                 ************************************************
03436 183B B6 4A     TELNO2 LDA    DATA
03437 183D E7 72            STA     LCDM+1,X
03438 183F A4 0F            AND     #$0F
03439 1841 AB 30            ADD     #$30      Change to ASCII
03440 1843 A1 3F            CMP     #$3F
03441 1845 26 04            BNE     TEL21
03442 1847 A6 20            LDA     #$20
03443 1849 20 0E            BRA     TEL222
03444 184B A1 3A      TEL21 CMP     #$3A
03445 184D 26 04            BNE     TEL22
03446 184F A6 2A            LDA     #$2A
03447 1851 20 06            BRA     TEL222
03448 1853 A1 3E      TEL22 CMP     #$3E
03449 1855 26 02            BNE     TEL222
03450 1857 A6 23            LDA     #$23
03451 1859 E7 71      TEL222 STA    LCDM,X
03452 185B E6 72            LDA     LCDM+1,X
03453 185D 44               LSR     A
03454 185E 44               LSR     A
03455 185F 44               LSR     A
03456 1860 44               LSR     A
03457 1861 AB 30            ADD     #$30
03458 1863 A1 3F            CMP     #$3F      Change to ASCII
03459 1865 26 04            BNE     TEL23
03460 1867 A6 20            LDA     #$20
03461 1869 20 0E            BRA     TEL25
03462 186B A1 3A      TEL23 CMP     #$3A
03463 186D 26 04            BNE     TEL24
03464 186F A6 2A            LDA     #$2A
03465 1871 20 06            BRA     TEL25
```

```
03466 1873 A1 3E    TEL24  CMP    #$3E
03467 1875 26 02           BNE    TEL25
03468 1877 A6 23           LDA    #$23
03469 1879 E7 72    TEL25  STA    LCDM+1,X
03470 187B 81              RTS
03471               *
03472               ****     DATA TRNSFER BETWEEN MICROCOMPUTER AND
03473               *                    EXTERNAL RAM
03474               *
03475               ************************************************
03476               *                                              *
03477               *      NAME    : INPUT DATA FROM EXTERNAL RAM  *
03478               *                              (INPDAT)        *
03479               *                                              *
03480               ************************************************
03481               *                                              *
03482               *      ENTRY   : DTDP                          *
03483               *      RETURNS : DATA                          *
03484               *                                              *
03485               ************************************************
03486 187C 9B       INPDAT SEI            Disable interrupts
03487 187D B6 46           LDA    DTDP+1
03488 187F B7 01           STA    PBDTR
03489 1881 B6 45           LDA    DTDP
03490 1883 AA 60           ORA    #$60
03491 1885 B7 0B           STA    PEDTR
03492 1887 3F 04           CLR    PADDR
03493 1889 B6 0B           LDA    PEDTR
03494 188B AA E0           ORA    #$E0
03495 188D B7 0B           STA    PEDTR
03496 188F A4 DF           AND    #$DF     Input data
03497 1891 B7 0B           STA    PEDTR
03498 1893 B6 00           LDA    PADTR
03499 1895 B7 4A           STA    DATA
03500 1897 B6 45           LDA    DTDP
03501 1899 AA 60           ORA    #$60
03502 189B B7 0B           STA    PEDTR
03503 189D 9A              CLI            Enable interrupts
03504 189E 81              RTS
03505               *
03506               ************************************************
03507               *                                              *
03508               *      NAME    : OUTPUT DATA TO EXTERNAL RAM   *
03509               *                              (OUTDAT)        *
03510               *                                              *
03511               ************************************************
03512               *                                              *
03513               *      ENTRY   : DATA, DTDP                    *
03514               *      RETURNS : NOTHING                       *
03515               *                                              *
03516               ************************************************
03517 189F 9B       OUTDAT SEI            Disable interrupts
03518 18A0 B6 4A           LDA    DATA
03519 18A2 B7 00           STA    PADTR
03520 18A4 B6 45           LDA    DTDP
```

```
03521 18A6 AA 60           ORA    #$60
03522 18A8 B7 0B           STA    PEDTR
03523 18AA B6 46           LDA    DTDP+1
03524 18AC B7 01           STA    PBDTR
03525 18AE A6 FF           LDA    #$FF
03526 18B0 B7 04           STA    PADDR    Output data
03527 18B2 B6 0B           LDA    PEDTR
03528 18B4 AA E0           ORA    #$E0
03529 18B6 B7 0B           STA    PEDTR
03530 18B8 A4 BF           AND    #$BF
03531 18BA B7 0B           STA    PEDTR
03532 18BC B6 45           LDA    DTDP
03533 18BE AA 60           ORA    #$60
03534 18C0 B7 0B           STA    PEDTR
03535 18C2 9A              CLI               Enable interrupts
03536 18C3 81              RTS
03537                *
03538                ****  SUBROUTINES     ***************************
03539                *
03540                *
03541                ************************************************
03542                *                                              *
03543                *      NAME    : INCREMENT POINTER TO EXTERNAL *
03544                *                        RAM (DTADD)           *
03545                *                                              *
03546                ************************************************
03547                *                                              *
03548                *      ENTRY   : DTDP                          *
03549                *      RETURNS : DTDP                          *
03550                *                                              *
03551                ************************************************
03552 18C4 B6 46     DTADD  LDA    DTDP+1
03553 18C6 AB 01            ADD    #$1       Increment pointer
03554 18C8 B7 46            STA    DTDP+1
03555 18CA 24 02            BCC    DTADD1
03556 18CC 3C 45            INC    DTDP
03557 18CE 81        DTADD1 RTS
03558                *
03559                ************************************************
03560                *                                              *
03561                *      NAME    : DECREMENT POINTER FROM        *
03562                *                      EXTERNAL RAM (DTSUB)    *
03563                *                                              *
03564                ************************************************
03565                *                                              *
03566                *      ENTRY   : DTDP                          *
03567                *      RETURNS : DTDP                          *
03568                *                                              *
03569                ************************************************
03570 18CF B6 46     DTSUB  LDA    DTDP+1
03571 18D1 A0 3A            SUB    #58       Decrement pointer
03572 18D3 B7 46            STA    DTDP+1
03573 18D5 24 02            BCC    DTSUB1
03574 18D7 3A 45            DEC    DTDP
03575 18D9 81        DTSUB1 RTS
```

```
03576                   *
03577                   ************************************************
03578                   *                                              *
03579                   *       NAME    : CLEAR RAM USED (END)         *
03580                   *                                              *
03581                   ************************************************
03582                   *                                              *
03583                   *       ENTRY   : NOTHING                      *
03584                   *       RETURNS : NOTHING                      *
03585                   *                                              *
03586                   ************************************************
03587 18DA 08 B0 1D END     BRSET  4,WAIT3,END1  Output tone?
03588 18DD 4F               CLR    A             If not, clear RAM
03589 18DE B7 41            STA    MODFLG
03590 18E0 B7 43            STA    TRMFLG
03591 18E2 B7 40            STA    LCDMP
03592 18E4 B7 49            STA    DTCNT
03593 18E6 B7 45            STA    DTDP
03594 18E8 B7 46            STA    DTDP+1
03595 18EA 5F               CLR    X
03596 18EB A6 01            LDA    #$1
03597 18ED B7 4B            STA    CURADP
03598 18EF A6 02            LDA    #$02
03599 18F1 B7 42            STA    FLAG
03600 18F3 1F 4E            BCLR   7,BZRFLG
03601 18F5 CD 1966          JSR    LCDMCR
03602 18F8 20 16            BRA    END3
03603 18FA B6 42    END1    LDA    FLAG          If so, display dial no.
03604 18FC A4 35            AND    #$35
03605 18FE B7 42            STA    FLAG
03606 1900 CD 1966          JSR    LCDMCR
03607 1903 5F               CLR    X
03608 1904 E6 99    END2    LDA    DIALNO,X
03609 1906 E7 73            STA    LCDM+2,X
03610 1908 5C               INC    X
03611 1909 A3 14            CPX    #20
03612 190B 26 F7            BNE    END2
03613 190D CD 110E          JSR    LCD
03614 1910 81       END3    RTS
03615                   *
03616                   ************************************************
03617                   *                                              *
03618                   *       NAME    : MOVE CURSOR TO LEFT (LFCRSR) *
03619                   *                                              *
03620                   ************************************************
03621                   *                                              *
03622                   *       ENTRY   : LCDMP, CURADP, MODFLG, TRMFLG *
03623                   *       RETURNS : LCDMP, CURADP                *
03624                   *                                              *
03625                   ************************************************
03626 1911 B6 41    LFCRSR LDA     MODFLG
03627 1913 A1 08            CMP    #$8
03628 1915 27 07            BEQ    LFCR2
03629 1917 A1 09            CMP    #$9
03630 1919 27 03            BEQ    LFCR2
```

```
03631 191B CC 1922          JMP     LFCR4
03632 191E B6 43     LFCR2  LDA     TRMFLG
03633 1920 27 11            BEQ     LFCR3
03634 1922 3A 40     LFCR4  DEC     LCDMP     Move cursor to left
03635 1924 3A 4B            DEC     CURADP
03636 1926 A6 01            LDA     #$1
03637 1928 B1 4B            CMP     CURADP
03638 192A 23 04            BLS     LFCR1
03639 192C B7 4B            STA     CURADP
03640 192E 3F 40            CLR     LCDMP
03641 1930 CD 110E   LFCR1  JSR     LCD       Display cursor
03642 1933 81        LFCR3  RTS
03643                *
03644                ************************************************
03645                *                                              *
03646                *       NAME    : MOVE CURSOR TO RIGHT (RTCRSR) *
03647                *                                              *
03648                ************************************************
03649                *                                              *
03650                *       ENTRY   : LCDMP, CURADP, MODFLG, TRMFLG *
03651                *       RETURNS : LCDMP, CURADP                *
03652                *                                              *
03653                ************************************************
03654 1934 B6 41     RTCRSR LDA     MODFLG
03655 1936 A1 08            CMP     #$8
03656 1938 27 07            BEQ     RTCR3
03657 193A A1 09            CMP     #$9
03658 193C 27 03            BEQ     RTCR3
03659 193E CC 1945          JMP     RTCR5
03660 1941 B6 43     RTCR3  LDA     TRMFLG
03661 1943 27 20            BEQ     RTCR4
03662 1945 B6 4B     RTCR5  LDA     CURADP
03663 1947 A1 14            CMP     #20
03664 1949 23 08            BLS     RTCR1
03665 194B BE 4B            LDX     CURADP
03666 194D E6 71            LDA     LCDM,X
03667 194F A1 20            CMP     #$20
03668 1951 27 0F            BEQ     RTCR2
03669 1953 3C 40     RTCR1  INC     LCDMP     Move cursor to right
03670 1955 3C 4B            INC     CURADP
03671 1957 A6 26            LDA     #38
03672 1959 B1 4B            CMP     CURADP
03673 195B 24 05            BCC     RTCR2
03674 195D B7 4B            STA     CURADP
03675 195F 4A               DEC     A
03676 1960 B7 40            STA     LCDMP
03677 1962 CD 110E   RTCR2  JSR     LCD       Display cursor
03678 1965 81        RTCR4  RTS
03679                *
03680                ************************************************
03681                *                                              *
03682                *       NAME    : CLEAR DIAPLAY RAM (LCDMCR)    *
03683                *                                              *
03684                ************************************************
03685                *                                              *
```

```
03686                 *       ENTRY   : NOTHING                       *
03687                 *       RETURNS : LCDM                          *
03688                 *                                               *
03689                 ************************************************
03690 1966 AE 28     LCDMCR LDX    #40       Clear diaplay RAM
03691 1968 A6 20     MCR1   LDA    #$20
03692 196A E7 70            STA    LCDM-1,X
03693 196C 5A               DEC    X
03694 196D 26 F9            BNE    MCR1
03695 196F 81               RTS
03696                 *
03697                 ************************************************
03698                 *                                               *
03699                 *       NAME    : CLEAR EXTERNAL RAM (RAMCLR)   *
03700                 *                                               *
03701                 ************************************************
03702                 *                                               *
03703                 *       ENTRY   : NOTHING                       *
03704                 *       RETURNS : NOTHING                       *
03705                 *                                               *
03706                 ************************************************
03707 1970 3F 4A     RAMCLR CLR    DATA
03708 1972 3F 45            CLR    DTDP
03709 1974 3F 46            CLR    DTDP+1
03710 1976 CD 189F  RAMCL1 JSR    OUTDAT    Clear external RAM
03711 1979 CD 18C4          JSR    DTADD
03712 197C A6 16            LDA    #RAMEND/256
03713 197E B1 45            CMP    DTDP
03714 1980 26 F4            BNE    RAMCL1
03715 1982 A6 A8            LDA    #RAMEND*256/256
03716 1984 B1 46            CMP    DTDP+1
03717 1986 26 EE            BNE    RAMCL1
03718 1988 A6 17            LDA    #$17
03719 198A B7 45            STA    DTDP
03720 198C 5F               CLR    X
03721 198D BF 46            STX    DTDP+1
03722 198F A6 FF    RAMCL2 LDA    #$FF
03723 1991 B7 4A            STA    DATA
03724 1993 CD 189F          JSR    OUTDAT
03725 1996 B6 46            LDA    DTDP+1
03726 1998 AB 01            ADD    #$1
03727 199A B7 46            STA    DTDP+1
03728 199C BE 46            LDX    DTDP+1
03729 199E A3 64            CPX    #$64
03730 19A0 26 ED            BNE    RAMCL2
03731 19A2 A6 1F            LDA    #$1F      Store RAM check data
03732 19A4 B7 45            STA    DTDP
03733 19A6 3F 46            CLR    DTDP+1
03734 19A8 A6 4C            LDA    #$4C
03735 19AA B7 4A            STA    DATA
03736 19AC CD 189F          JSR    OUTDAT
03737 19AF 3C 46            INC    DTDP+1
03738 19B1 A6 4F            LDA    #$4F
03739 19B3 B7 4A            STA    DATA
03740 19B5 CD 189F          JSR    OUTDAT
```

```
03741 19B8 3C 46          INC    DTDP+1
03742 19BA A6 56          LDA    #$56
03743 19BC B7 4A          STA    DATA
03744 19BE CD 189F        JSR    OUTDAT
03745 19C1 3C 46          INC    DTDP+1
03746 19C3 A6 45          LDA    #$45
03747 19C5 B7 4A          STA    DATA
03748 19C7 CD 189F        JSR    OUTDAT
03749 19CA 81             RTS
03750                *
03751                ************************************************
03752                *                                              *
03753                *      NAME    : MOVE TO REVIEW DATA AREA      *
03754                *                              (MOVE1)         *
03755                *                                              *
03756                ************************************************
03757                *                                              *
03758                *      ENTER   : LCDM                          *
03759                *      RETURNS : REFDAT                        *
03760                *                                              *
03761                ************************************************
03762 19CB 5F        MOVE1  CLR    X
03763 19CC E6 71     MOV11  LDA    LCDM,X
03764 19CE D7 0115          STA    REFDAT,X Store review data
03765 19D1 5C               INC    X
03766 19D2 A3 14            CPX    #20
03767 19D4 26 F6            BNE    MOV11
03768 19D6 81               RTS
03769                *
03770                ************************************************
03771                *                                              *
03772                *      NAME    : MOVE REVIEW DATA TO DISPLAY   *
03773                *                        RAM (MOVE2)           *
03774                *                                              *
03775                ************************************************
03776                *                                              *
03777                *      ENTER   : REFDAT                        *
03778                *      RETURNS : LCDM                          *
03779                *                                              *
03780                ************************************************
03781 19D7 5F        MOVE2  CLR    X
03782 19D8 D6 0115   MOV21  LDA    REFDAT,X
03783 19DB E7 85            STA    LCDM+20,X  Store display data
03784 19DD 5C               INC    X
03785 19DE A3 12            CPX    #18
03786 19E0 26 F6            BNE    MOV21
03787 19E2 81               RTS
03788                *
03789                ************************************************
03790                *                                              *
03791                *      DATA TABLE                              *
03792                *                                              *
03793                ************************************************
03794 19E3 2A        CALDAT FCC    "*  /     :  :  "
03795 19F2 2A        TELDAT FCC    "* TIME   :  :  "
```

```
03796 1A01 32        MONTH  FCB    $32,$29,$32,$31,$32,$31
03797 1A07 32               FCB    $32,$32,$31,$32,$31,$32
03798 1A0D 0C        YOKO   FCB    $0C,$80,$06,$01,$08,$34
03799 1A13 13        SHUSE  FCB    $13,$00,$24,$60,$60
03800 1A18 01        SHUNEW FCB    $01,$01,$00,$00,$00
03801 1A1D 43        CRGDAT FCC    "CHARGE:         "
03802 1A2C 52        RATDAT FCC    "RATE:          "
03803 1A39 53        SECDAT FCC    "SEC:    "
03804 1A40 4E        NO1    FCC    "NO.?"
03805 1A44 20        NO2    FCC    "    NO. "
03806 1A4C 20        DIANO1 FCC    " REDIAL    "
03807 1A56 20        DIANO2 FCC    " RETRY          "
03808 1A64 2A        FULLD  FCC    "* FULL *"
03809 1A6C 2A        ERRORD FCC    "* ERROR *"
03810                *
03811                ************************************************
03812                *                                              *
03813                *       VECTOR ADDRESSES                       *
03814                *                                              *
03815                ************************************************
03816                *
03817 1FF6                  ORG    $1FF6
03818                *
03819 1FF6 0C25             FDB    TIMER2   K88SCN/BUZZER
03820 1FF8 0B80             FDB    TIMER1   INT2/TIMER
03821 1FFA 0500             FDB    MAINPR   INT
03822 1FFC 0500             FDB    MAINPR   SWI
03823 1FFE 0500             FDB    MAINPR   RES
03824                *
03825                       END
```

ADDRAT	0129	ARGINT	1541	ARG100	15B7	ARG110	15C0	ARG16	1590
ARG17	1599	ARG18	15A1	ARG19	15B0	BUZC	0B8D	BUZC1	0B96
BUZC2	0BB0	BUZC3	0BAC	BUZC4	0BD5	BUZC5	0BBC	BUZC6	0BDA
BUZZ	0C95	BUZZ1	0CA6	BUZZ2	0CAE	BZR	004F	BZRCNT	004C
BZRFLG	004E	CALCNT	006E	CALDAT	19E3	CALLCD	1258	CALNDR	005D
CALRST	1038	CALSHU	149F	CALSH1	14A9	CALSH2	14DD	CALSH3	14C3
CALSH4	14C5	CALSH5	14CF	CALSH6	14E8	CALSH7	14D7	CALSH8	1515
CALSH9	14DC	CALS10	151E	CANCEL	1001	CANCL2	1010	CATBL	067A
CATBLE	068C	CHATFL	0057	CHECK	05E9	CHECKF	0651	CHECK1	05EF
CHECK2	05F7	CHECK3	0601	CHECK4	0612	CHECK5	062A	CHECK6	062D
CHECK7	063B	CHEK1	098E	CHEK2	0A20	CHEK3	0A47	CHEK4	0A8C
CHEK5	0ADD	CHK11	09A0	CHK12	09A5	CHK13	09C9	CHK14	09E7
CHK15	0A0B	CHK16	0A1F	CHK17	0A00	CHK21	0A38	CHK22	0A44
CHK23	0A46	CHK31	0A59	CHK32	0A6D	CHK33	0A82	CHK34	0A84
CHK35	0A89	CHK36	0A8B	CHK42	0A9E	CHK43	0ABE	CHK44	0AD1
CHK45	0AD5	CHK46	0ADA	CHK47	0ADC	CHK5F	0B11	CHK51	0AE9
CHK52	0B07	CHK53	0B0F	CLEAR	1480	CLR1	1482	CMPNUM	0003
CNT	006D	CNTBIT	006B	CRGDAT	1A1D	CRGSET	16EA	CRGST1	1711
CRGST2	171B	CSTBL	068C	CSTBLE	0695	CURADP	004B	CURCAL	17EC
CURCL1	180B	CURCL2	1811	CURCL3	181D	CURCL5	180A	CURLR	0B12
CURLRA	0B6E	CURLRF	0B71	CURLR2	0B30	CURLR3	0B3F	CURLR4	0B46
CURLR5	0B52	CURLR6	0B56	CURLR7	0B5B	CURLR8	0B67	CURLR9	0B6B
CURRAT	17BB	CURRA1	17D2	CURRA2	17D8	CURRA3	17E2	CURRA4	17D1
DATA	004A	DAY	14F4	DIACLR	146C	DIACL1	146E	DIADAT	00B7
DIALNO	0099	DIAL2	1526	DIAL21	1528	DIAL22	1531	DIANO1	1A4C
DIANO2	1A56	DIAP1	00BC	DIAP2	00BD	DTADD	18C4	DTADD1	18CE
DTADS	0050	DTCNT	0049	DTDP	0045	DTLCD	1449	DTLCD1	144A
DTLCD2	1459	DTSUB	18CF	DTSUB1	18D9	END	18DA	END1	18FA
END2	1904	END3	1910	ERFLG	00BE	ERROR	0695	ERRORD	1A6C
ERROR2	06AB	ERROR3	06C1	ERROR4	06CE	ERROR5	06DC	ERROR6	06E7
ERR1	013B	ERR2	013C	FIGURE	0BDB	FIG1	0BF6	FIG2	0BFF
FIG3	0C0D	FIG4	0C12	FIG5	0C17	FIG6	0C1C	FIG7	0C1F
FIG8	0C24	FLAG	0042	FULLD	1A64	HOLD	00BF	INPDAT	187C
JMPADR	013E	JTBL	013D	KENSAK	12C1	KENSKA	1349	KENSKB	1352
KENSKC	135B	KENSKD	1387	KENSKE	138B	KENSK1	12C7	KENSK2	12E1
KENSK3	12E5	KENSK4	12F5	KENSK5	12F6	KENSK6	1310	KENSK7	1312
KENSK8	131E	KENSK9	1333	KEYCD	0652	KEYDAT	0058	KEYNUM	005A
KEYSET	0054	KNSEQ1	137B	KNSKEQ	1367	KNSRTS	1399	K88SCN	0C31
K88SN1	0C40	K88SN2	0C52	K88SN3	0C56	K88SN4	0C65	K88SN5	0C6B
K88SN6	0C77	K88SN7	0C7D	K88SN8	0C8C	K88SN9	0C94	LCD	110E
LCDBSY	11C4	LCDBY1	11CE	LCDDSP	115C	LCDINS	1148	LCDINT	1195
LCDIN1	1197	LCDIN2	11AF	LCDIN3	11B8	LCDIN5	119B	LCDM	0071
LCDMCR	1966	LCDMP	0040	LCDMST	12AB	LCDMS1	12B0	LCDMS2	12BC
LCDMS3	12B6	LCDRES	1172	LCDRS1	1176	LCDRS2	1178	LCDRS3	117A
LCDSET	1276	LCDSE1	1278	LCDSE2	12A3	LCDSE3	127D	LCDSE6	12A7
LCDSE7	128E	LCD1	1118	LCD2	1134	LCD3	113D	LCD4	1144
LCD5	113F	LFCRSR	1911	LFCR1	1930	LFCR2	191E	LFCR3	1933
LFCR4	1922	MAINKS	0504	MAINPR	0500	MAIN1	0512	MAIN2	0517
MAIN3	051C	MAIN4	051F	MCR1	'968	MDTBL	05B9	MDTBLE	05E9
MEMCLR	1476	MEMCL1	1478	MEMO	0100	MEMOP	00B3	MFDAT	00B5
MFOUT	10DF	MFOUTE	10F5	MFOUT1	10E0	MFOUT2	10EE	MFTBL	10F6
MFTBLE	110E	MNYASC	1612	MNYAS1	1614	MNYAS2	1620	MNYAS4	1638
MNYAS5	1645	MNYAS6	164E	MNYAS7	1653	MNYAS8	1658	MNYAS9	165D
MNYA10	1643	MNYA13	1661	MNYA14	1666	MNYA15	166B	MNYA16	1670

MNYA17	1678	MODE	057B	MODEF	05B8	MODE2	0587	MODE3	058F
MODE4	0591	MODE5	0592	MODE6	05A2	MODFLG	0041	MOD10	06EE
MOD100	08B2	MOD101	08C3	MOD102	08CC	MOD103	08DF	MOD110	08E0
MOD111	08F1	MOD112	08FA	MOD113	090D	MOD120	090E	MOD121	0928
MOD130	0929	MOD131	0930	MOD132	0940	MOD133	0946	MOD134	0954
MOD136	0951	MOD140	0955	MOD141	0959	MOD142	096A	MOD143	0971
MOD144	097F	MOD145	097C	MOD150	0980	MOD151	0982	MOD160	098A
MOD20	06F7	MOD30	0700	MOD40	0709	MOD41	072E	MOD42	0738
MOD50	0739	MOD51	073F	MOD52	0759	MOD53	077B	MOD54	0795
MOD55	079C	MOD56	07AA	MOD6	07B0	MOD61	07C0	MOD62	07CE
MOD63	07D5	MOD7	07E6	MOD7F	0800	MOD71	07EF	MOD80	0801
MOD82	0816	MOD83	0836	MOD84	083C	MOD85	084B	MOD90	084C
MOD91	0876	MOD92	088E	MOD93	08A2	MOD94	08B1	MOD95	13BD
MONEY	15CE	MONEY1	15D1	MONEY2	15F4	MONEY3	1604	MONEY4	1611
MONTH	1A01	MOVE1	19CB	MOVE2	19D7	MOV11	19CC	MOV21	19D8
MVECUF	0B7F	MVECUR	0B72	NEWKEY	0056	NO1	1A40	NO2	1A44
OLDKEY	0055	OUTDAT	189F	PADDR	0004	PADTR	0000	PBDDR	0005
PBDTR	0001	PCDDR	0006	PCDTR	0002	PDDTR	0003	PEDTR	000B
PFDTR	000C	PGDDR	0007	PGDTR	000D	POINTR	006C	RAMCL	0577
RAMCLE	057A	RAMCLR	1970	RAMCL1	1976	RAMCL2	198F	RAMEND	16A8
RATDAT	1A2C	RATSET	172C	RATSE1	1733	RATSE2	178B	RATSE3	1795
RATSE4	179D	RATSE5	179E	REC	00B4	REFDAT	0115	RSTEND	10DE
RTCRSR	1934	RTCR1	1953	RTCR2	1962	RTCR3	1941	RTCR4	1965
RTCR5	1945	RTS11	103E	RTS12	104C	RTS13	1056	RTS14	1059
RTS15	106E	RTS16	108A	RTS17	1090	RTS18	10CE	RTS19	10DA
SCOUNT	005C	SCR	0010	SDP	0047	SEC	0052	SECCNT	0135
SECDAT	1A39	SHUCAL	1491	SHUCNT	006F	SHUNEW	1A18	SHUSE	1A13
SHUSEC	0137	SHUSTA	0065	SINGL	012D	SNGCLR	17B1	SNGCL1	17B3
SNGSTA	167E	SNGS1	1692	SNGS2	169C	SNGS3	16A6	SSR	0011
STBDAT	005B	SYSINT	0523	TAN	1409	TAN1	1419	TAN2	1424
TAN3	1437	TCNTR	0070	TCR	0009	TDR	0008	TEL	0062
TELCLR	1489	TELCL1	148B	TELCNT	013A	TELDAT	19F2	TELFLG	0044
TELLCD	1267	TELMN	0CAF	TELMN1	0CC0	TELMN2	0CC5	TELMN3	0CCA
TELMN4	0CCF	TELMN5	0CD4	TELMN6	0CD9	TELMN7	0CDC	TELM10	0CDD
TELM11	0CE7	TELM12	0CFC	TELM20	0D00	TELM21	0D0B	TELM22	0D10
TELM23	0D14	TELM3A	0E84	TELM3B	0E8D	TELM3F	0E95	TELM30	0D15
TELM31	0D2E	TELM32	0D3A	TELM33	0D87	TELM34	0D87	TELM35	0DCF
TELM36	0DE3	TELM37	0E35	TELM38	0E64	TELM39	0E7B	TELM40	0E96
TELM41	0EAD	TELM42	0EB1	TELM43	0EB3	TELM44	0EBB	TELM45	0EC5
TELM46	0EDD	TELM47	0EE1	TELM48	0EE3	TELM49	0EF1	TELM5A	0FBD
TELM5B	0FCD	TELM5C	0FD3	TELM5F	0FD7	TELM50	0EF2	TELM51	0F10
TELM52	0F62	TELM53	0F65	TELM54	0F6D	TELM55	0F7D	TELM56	0F83
TELM57	0F89	TELM58	0FA4	TELM59	0FBA	TELM60	0FDB	TELM61	0FE7
TELM62	0FFD	TELNO	0139	TELNO1	1823	TELNO2	183B	TELWAT	00B6
TEL11	1832	TEL21	184B	TEL22	1853	TEL222	1859	TEL23	186B
TEL24	1873	TEL25	1879	TEL321	0D40	TEL322	0D61	TEL323	0D6C
TEL324	0D7F	TEL331	0D9C	TEL332	0DAF	TEL341	0DC6	TEL342	0DC9
TEL343	0DCC	TEL351	0DDD	TEL361	0E0D	TEL362	0E19	TEL363	0E24
TEL364	0E2D	TEL371	0E50	TEL372	0E5C	TEL501	0F0D	TEL511	0F1F
TEL512	0F31	TEL513	0F47	TIMCNT	11F4	TIMCT1	11F5	TIMCT2	1245
TIMCT3	11FE	TIMCT4	120A	TIMCT5	124A	TIMCT6	122D	TIMCT7	124F
TIMCT8	1233	TIMCT9	1252	TIMC10	1241	TIMER1	0B80	TIMER2	0C25
TIMSTA	006A	TLCNT	11EA	TOLCLR	17A7	TOLCL1	17A9	TOLSTA	16CD
TOLST1	16D2	TOROKU	139A	TOROK1	139B	TOROK2	13A8	TOROK3	13AD
TOROK4	13AE	TOROK5	13F5	TOROK6	13FA	TOTAL	0131	TOTLKY	0059
TRKC1	13C3	TRKC2	13CF	TRKC3	13DE	TRKC4	13E4	TRKC5	13D5
TRKRTS	1408	TRMFLG	0043	TRNS	00C0	WAIT1	00AE	WAIT2	00AF
WAIT3	00B0	WAIT4	00B1	WAIT5	00B2	YERCNT	11E4	YOKO	1A0D

```
ADDRAT 0129  00067* 02952  03014  03022  03072  03080  03271
ARGINT 1541  00152  02915*
ARG100 15B7  02972* 02975
ARG110 15C0  02977* 02980
ARG16  1590  02951* 02954
ARG17  1599  02956* 02959
ARG18  15A1  02961* 02964
ARG19  15B0  02968* 02970
BUZC   0B8D  01241  01257*
BUZC1  0B96  01257  01260*
BUZC2  0BB0  01258  01264  01271  01274*
BUZC3  0BAC  01266  01272*
BUZC4  0BD5  01284  01290*
BUZC5  0BBC  01275  01279*
BUZC6  0BDA  01259  01278  01280  01289  01292*
BUZZ   0C95  01352  01427*
BUZZ1  0CA6  01428  01434*
BUZZ2  0CAE  01427  01433  01438*
BZR    004F  00018* 00651  00743  00801  00936  01428  01432  01437
             01453  01454  01549  01574  01594  01615  01618  01650
             01667  01677  01709  01717  01722  01723  01738  01744
             01745  01746  01747  01748  01773  01776  01812  01922
             01924  01946  01951  02000  02002  02499  02869  02944
             03281
BZRCNT 004C  00016* 00376  00381  00384  01260  01262  01265  01268
             01270  01833  01835  01918  02498
BZRFLG 004E  00017* 00386  00531  00581  00777  01172  01178  01257
             01272  01427  01455  01507  01601  01831  01906  01908
             01944  01997  01999  02500  02943  03600
CALCNT 006E  00039* 02369  02388  02408  02410  02419  02422  02423
             02805  02806  02824  02825  02924  03047  03049  03060
             03063  03105  03127  03134  03160  03183  03190
CALDAT 19E3  02445  03794*
CALLCD 1258  01326  02365* 02785
CALNDR 005D  00032* 02300  02311  02322  02351  02405  02413  02833
             02968  02973
CALRST 1038  01494  01540  01545  01575  01578  01596  01599  01604
             01616  01655  01672  01675  01688  01691  01739  01864
             01893  01944* 02982
CALSHU 149F  00829  02802*
CALSH1 14A9  02807* 02827
CALSH2 14DD  02816  02837*
CALSH3 14C3  02823* 02876
CALSH4 14C5  02824* 02847
CALSH5 14CF  02829* 02835
CALSH6 14E8  02842* 02867
CALSH7 14D7  02831  02833*
CALSH8 1515  02832  02842  02868* 02875
CALSH9 14DC  02836* 02872
CALS10 151E  02811  02813  02873*
CANCEL 1001  01511  01823  01839  01860  01906* 02014  02981
CANCL2 1010  01909  01913*
CATBL  067A  00282  00287  00292  00295  00339*
CATBLE 068C  00287  00351*
CHATFL 0057  00025* 01402  01404  01406  01409  01411
CHECK  05E9  00120  00271*
CHECKF 0651  00272  00290  00300  00310  00320*
```

		01197	01200	01219	01889	02789	03207	03332	03341
		03344	03348	03353	03369	03383	03385	03391	03394
		03591	03634	03640	03669	03676			
LCDMST	12AB	02365	02384	02442*					
LCDMS1	12B0	02444*	02448						
LCDMS2	12BC	02444	02450*						
LCDMS3	12B6	02446*	02451						
LCDRES	1172	02172*	02916						
LCDRS1	1176	02174*	02189						
LCDRS2	1178	02175*	02179						
LCDRS3	117A	02176*	02177						
LCDSET	1276	02370	02389	02403*					
LCDSE1	1278	02404*	02425						
LCDSE2	12A3	02404	02427*						
LCDSE3	127D	02406*	02428						
LCDSE6	12A7	02412	02429*						
LCDSE7	128E	02414*	02430						
LCD1	1118	02094*	02099						
LCD2	1134	02107*	02110						
LCD3	113D	02106	02111*						
LCD4	1144	02102	02115*						
LCD5	113F	02112*	02116						
LFCRSR	1911	01173	03626*						
LFCR1	1930	03638	03641*						
LFCR2	191E	03628	03630	03632*					
LFCR3	1933	03633	03642*						
LFCR4	1922	03631	03634*						
MAINKS	0504	00114*	00114	00126					
MAINPR	0500	00111*	03821	03822	03823				
MAIN1	0512	00117	00120*						
MAIN2	0517	00119	00121	00122*					
MAIN3	051C	00122	00124*						
MAIN4	051F	00123	00125*						
MCR1	1968	03691*	03694						
MDTBL	05B9	00202	00207	00212	00215	00227*			
MDTBLE	05E9	00207	00259*						
MEMCLR	1476	00800	00927	02664	02733*	02921			
MEMCL1	1478	02734*	02737						
MEMO	0100	00064*	00736	00769	00933	00948	02668	02735	02895
MEMOP	00B3	00050*	01491	01536	01684	01695	01928	01961	02942
MFDAT	00B5	00052*	01562	01632	02041	02935			
MFOUT	10DF	01559	01629	02030*					
MFOUTE	10F5	02038	02043*						
MFOUT1	10E0	02031*	02036						
MFOUT2	10EE	02032	02039*						
MFTBL	10F6	02031	02035	02040	02050*				
MFTBLE	110E	02035	02074*						
MNYASC	1612	03040*	03107	03129	03136	03162	03185	03192	
MNYAS1	1614	03041*	03066						
MNYAS2	1620	03045*	03069	03071	03073	03075			
MNYAS4	1638	03055*	03077	03079	03081	03083	03085		
MNYAS5	1645	03063*	03087						
MNYAS6	164E	03041	03068*						
MNYAS7	1653	03042	03070*						
MNYAS8	1658	03043	03072*						
MNYAS9	165D	03044	03074*						
MNYA10	1643	03062*	03088						

```
MNYA13 1661  03051  03076*
MNYA14 1666  03052  03078*
MNYA15 166B  03053  03080*
MNYA16 1670  03054  03082*
MNYA17 1678  03061  03086*
MODE   057B  00118  00190*
MODEF  05B8  00199  00210  00220*
MODE2  0587  00192  00196* 01008  01182
MODE3  058F  00197  00200*
MODE4  0591  00195  00201*
MODE5  0592  00202* 00208
MODE6  05A2  00203  00211*
MODFLG 0041  00007* 00194  00196  00280  00570  01089  01138  01163
             01878  01972  02011  02477  02537  03589  03626  03654
MOD10  06EE  00228  00431*
MOD100 08B2  00246  00724*
MOD101 08C3  00731* 00734
MOD102 08CC  00736* 00741
MOD103 08DF  00724  00745*
MOD110 08E0  00248  00757*
MOD111 08F1  00764* 00767
MOD112 08FA  00769* 00774
MOD113 090D  00757  00778*
MOD120 090E  00250  00791*
MOD121 0928  00791  00802*
MOD130 0929  00252  00814*
MOD131 0930  00817* 00831
MOD132 0940  00816  00824*
MOD133 0946  00825  00827*
MOD134 0954  00814  00823  00826  00833*
MOD136 0951  00828  00832*
MOD140 0955  00254  00845*
MOD141 0959  00847* 00861
MOD142 096A  00846  00854*
MOD143 0971  00855  00857*
MOD144 097F  00853  00856  00863*
MOD145 097C  00858  00862*
MOD150 0980  00256  00876*
MOD151 0982  00877* 00880
MOD160 098A  00258  00893*
MOD20  06F7  00230  00448*
MOD30  0700  00232  00465*
MOD40  0709  00234  00482*
MOD41  072E  00484  00498*
MOD42  0738  00482  00497  00499  00502*
MOD50  0739  00236  00238  00240  00516*
MOD51  073F  00516  00518*
MOD52  0759  00519  00529*
MOD53  077B  00530  00545*
MOD54  0795  00549  00558*
MOD55  079C  00559  00561*
MOD56  07AA  00557  00568*
MOD6   07B0  00546  00570*
MOD61  07C0  00572  00577*
MOD62  07CE  00579  00583*
MOD63  07D5  00584  00586*
MOD7   07E6  00574  00593*
```

```
TAN1    1419  02651* 02655
TAN2    1424  02657* 02663
TAN3    1437  02666* 02671
TCNTR   0070  00041* 01307  01310
TCR     0009  00099* 00142  01238
TDR     0008  00098* 00140  01240
TEL     0062  00033* 02339  02344  02427  02429  02768
TELCLR  1489  01313  02004  02767* 02919
TELCL1  148B  02768* 02770
TELCNT  013A  00074* 01679  01770  01803  01805  01817  01841  01843
              01916  01986  02933
TELDAT  19F2  02450  03795*
TELFLG  0044  00010* 01312  01318  01451  01513  01602  01603  01670
              01671  01693  01694  01719  01721  01741  01743  01749
              01821  01822  01829  01830  01892  01909  01994  01996
              02930
TELLCD  1267  01328  02384*
TELMN   0CAF  01353  01451*
TELMN1  0CC0  01451  01457*
TELMN2  0CC5  01452  01459*
TELMN3  0CCA  01453  01461*
TELMN4  0CCF  01454  01463*
TELMN5  0CD4  01455  01465*
TELMN6  0CD9  01456  01467*
TELMN7  0CDC  01458  01460  01462  01464  01466  01468*
TELM10  0CDD  01457  01480*
TELM11  0CE7  01480  01481  01484*
TELM12  0CFC  01484  01494*
TELM20  0D00  01459  01507*
TELM21  0D08  01507  01511*
TELM22  0D10  01508  01509  01513*
TELM23  0D14  01510  01512  01515*
TELM3A  0E84  01608  01671  01690*
TELM3B  0E8D  01687  01690  01693*
TELM3F  0E95  01541  01546  01568  01576  01579  01597  01600  01605
              01617  01643  01653  01656  01669  01673  01676  01686
              01689  01692  01697*
TELM30  0D15  01461  01531*
TELM31  0D2E  01532  01542*
TELM32  0D3A  01543  01547*
TELM33  0D87  01549  01580*
TELM34  0DB7  01593  01601*
TELM35  0DCF  01589  01609*
TELM36  0DE3  01548  01618*
TELM37  0E35  01618  01657*
TELM38  0E64  01606  01677*
TELM39  0E7B  01607  01670  01687*
TELM40  0E96  01463  01709*
TELM41  0EAD  01710  01719*
TELM42  0EB1  01711  01721*
TELM43  0EB3  01720  01722*
TELM44  0EBB  01709  01726*
TELM45  0EC5  01728  01731*
TELM46  0EDD  01731  01741*
TELM47  0EE1  01732  01743*
TELM48  0EE3  01742  01744*
TELM49  0EF1  01718  01725  01730  01740  01751*
```

TELM5A	0FBD	01820	01852*						
TELM5B	0FCD	01853	01860*						
TELM5C	0FD3	01859	01862*						
TELM5F	0FD7	01777	01815	01818	01824	01840	01851	01861	01864*
TELM50	0EF2	01465	01765*						
TELM51	0F10	01766	01778*						
TELM52	0F62	01806	01815*						
TELM53	0F65	01810	01816*						
TELM54	0F6D	01779	01819*						
TELM55	0F7D	01821	01825*						
TELM56	0F83	01822	01827*						
TELM57	0F89	01825	01827	01829*					
TELM58	0FA4	01826	01828	01841*					
TELM59	0FBA	01844	01848	01851*					
TELM60	0FDB	01467	01878*						
TELM61	0FE7	01880	01881	01883*					
TELM62	0FFD	01879	01882	01893*					
TELNO	0139	00073*	01538	01557	01569	01571	01586	01590	01683
		01768	01807	01809	01845	01847	01915	01985	02932
TELNO1	1823	02625	03409*						
TELNO2	183B	02658	02696	03436*					
TELWAT	00B6	00053*	01531	01534	01544	01681	01716	01724	01726
		01729	01750	01765	01772	01811	01850	01862	01914
		01984	02931						
TEL11	1832	03415	03417*						
TEL21	184B	03441	03444*						
TEL22	1853	03445	03448*						
TEL222	1859	03443	03447	03449	03451*				
TEL23	186B	03459	03462*						
TEL24	1873	03463	03466*						
TEL25	1879	03461	03465	03467	03469*				
TEL321	0D40	01547	01549*						
TEL322	0D61	01556	01564*						
TEL323	0D6C	01567	01569*						
TEL324	0D7F	01554	01577*						
TEL331	0D9C	01588	01590*						
TEL332	0DAF	01585	01598*						
TEL341	0DC6	01601	01606*						
TEL342	0DC9	01602	01607*						
TEL343	0DCC	01603	01608*						
TEL351	0DDD	01612	01616*						
TEL361	0E0D	01626	01638*						
TEL362	0E19	01637	01642	01644*					
TEL363	0E24	01648	01650*						
TEL364	0E2D	01624	01654*						
TEL371	0E50	01666	01670*						
TEL372	0E5C	01662	01674*						
TEL501	0F0D	01773	01777*						
TEL511	0F1F	01780	01785*						
TEL512	0F31	01785	01793*						
TEL513	0F47	01784	01788	01792	01795	01803*			
TIMCNT	11F4	02266	02281	02295*					
TIMCT1	11F5	02296*	02341						
TIMCT2	1245	02298	02343*						

```
TIMCT3  11FE  02301* 02345
TIMCT4  120A  02302  02308*
TIMCT5  124A  02309  02346*
TIMCT6  122D  02329* 02347
TIMCT7  124F  02329  02348*
TIMCT8  1233  02332* 02349
TIMCT9  1252  02335  02350*
TIMC10  1241  02340* 02352
TIMER1  0B80  01238* 03820
TIMER2  0C25  01350* 03819
TIMSTA  006A  00035* 02303  02307  02310  02326  02332  02338  02350
TLCNT   11EA  01319  02279*
TOLCLR  17A7  02965  03297*
TOLCL1  17A9  03298* 03301
TOLSTA  16CD  01330  03151*
TOLST1  16D2  03153* 03156
TOROKU  139A  00500  00590  00602  02578*
TOROK1  139B  02579* 02585
TOROK2  13A8  02583  02586*
TOROK3  13AD  02581  02588*
TOROK4  13AE  02589* 02595
TOROK5  13F5  02617  02623*
TOROK6  13FA  02625* 02630
TOTAL   0131  00069* 00878  03015  03017  03068  03076  03299
TOTLKY  0059  00027* 01371  01386  01387
TRKC1   13C3  02599* 02614
TRKC2   13CF  02602  02605*
TRKC3   13DE  02604  02612*
TRKC4   13E4  02615* 02622
TRKC5   13D5  02606  02608*
TRKRTS  1408  02587  02631*
TRMFLG  0043  00009* 00200  00483  00518  00539  00577  00593  00619
              00666  00815  00822  00830  00845  00852  00860  00993
              01024  01133  01181  02016  02481  02870  03282  03590
              03632  03660
TRNS    00C0  00060* 02596  02608  02611  02615  02618  02621  02855
              02858  02859  02862
WAIT1   00AE  00045* 01552  01572  01577  01622  01651  01654  01988
              02937
WAIT2   00AF  00046* 01555  01564  01565  01573  01583  01595  01598
              01614  01625  01638  01640  01652  01660  01668  01674
              01989  02938
WAIT3   00B0  00047* 00482  00516  00618  00652  00665  00724  00744
              00757  00776  00791  00814  00923  00928  00937  00941
              01162  01217  01484  01493  01539  01780  01783  01785
              01789  01790  01796  01797  01813  01863  01891  01919
              01930  01954  01957  01965  01981  01990  02009  02017
              02939  03587
WAIT4   00B1  00048* 01782  01786  01787  01793  01794  01920  02940
WAIT5   00B2  00049* 00939  00940  00952  00969  00970  01258  01273
              01288  01547  01613  01921  01991  02941
YERCNT  11E4  01311  02265*
YOKO    1A0D  02210  03798*
```

Circuit Diagram of the Intelligent Telephone

5.1 Circuit Diagrams

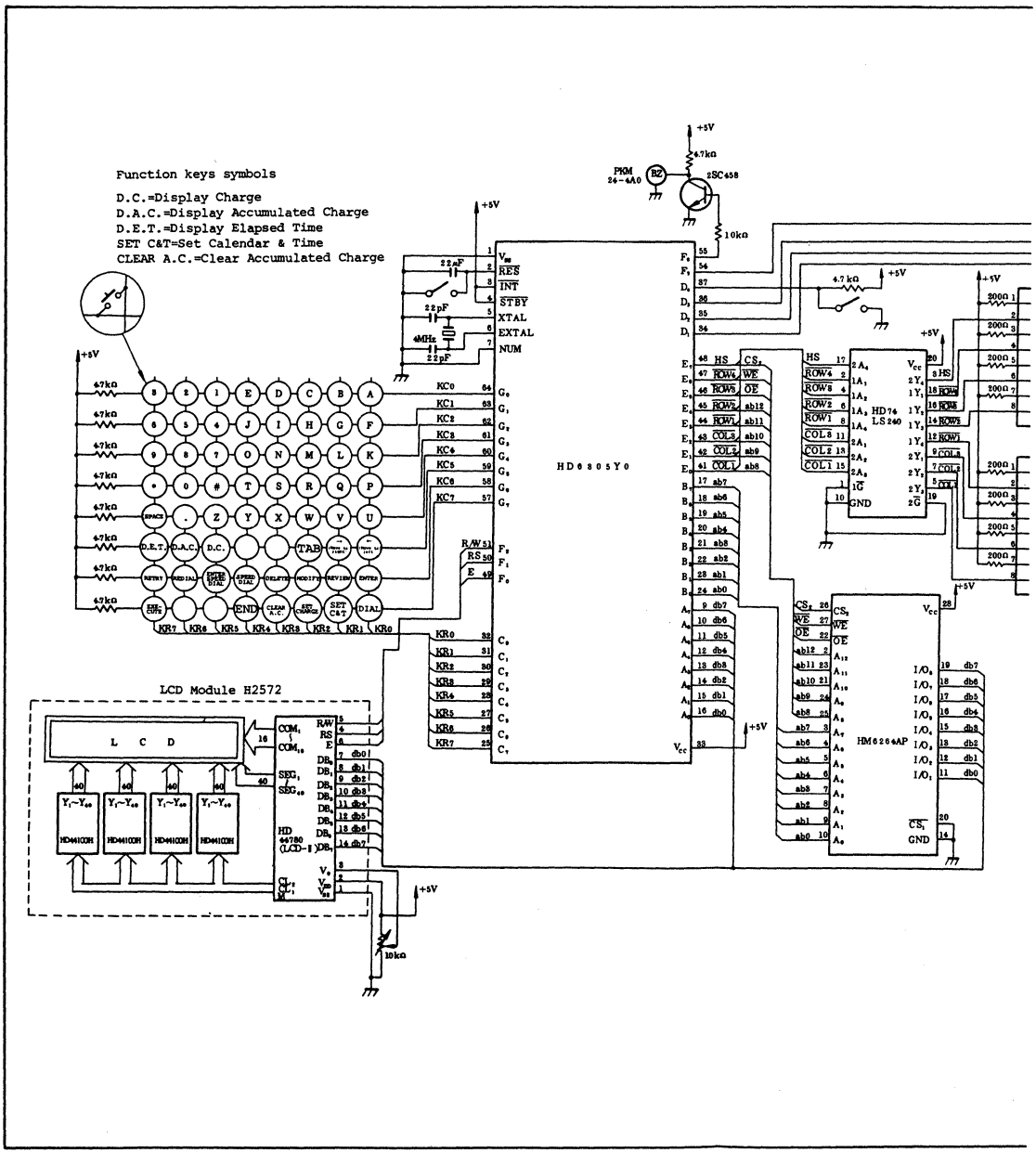
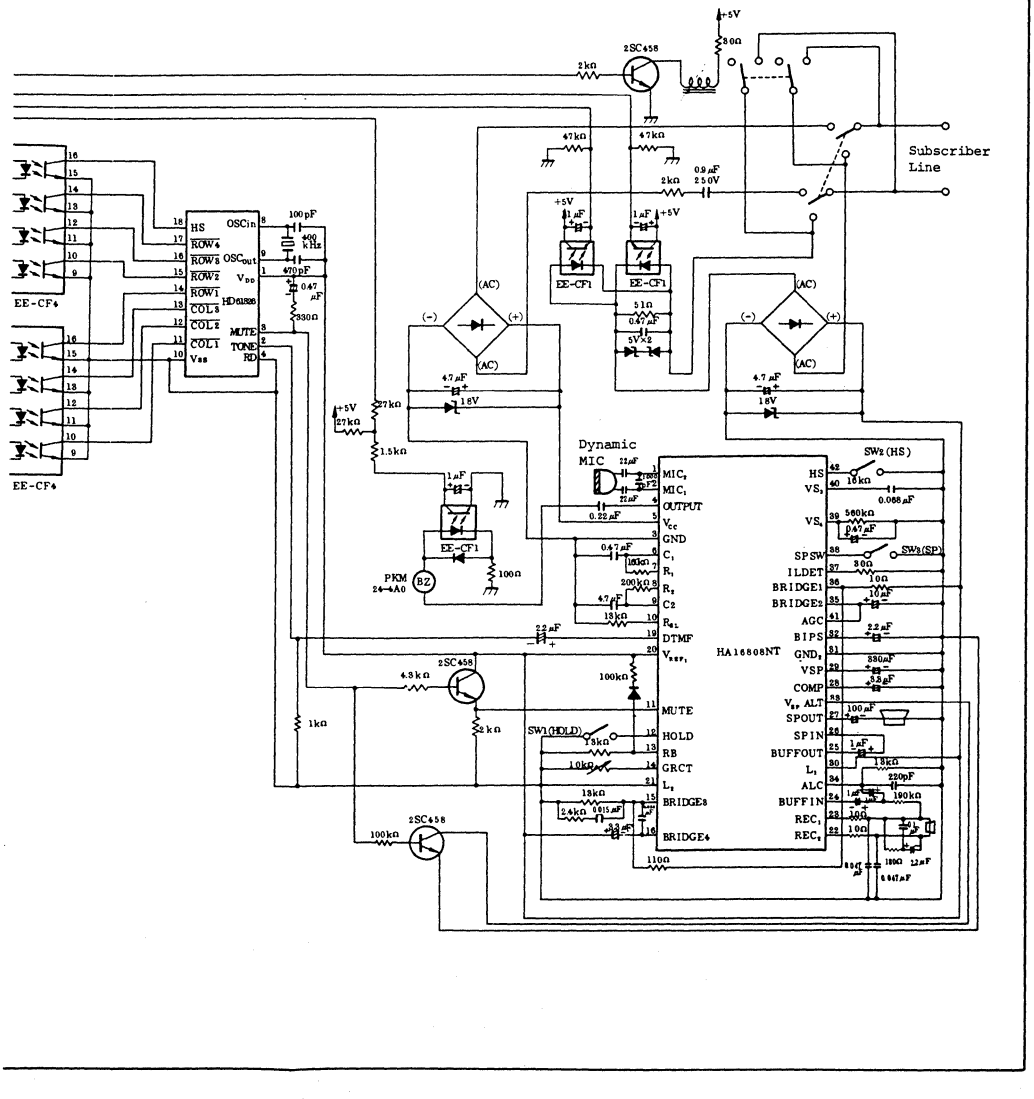


Fig. 5.1 Circuit Diagram



of the Intelligent Telephone

5.2 Pin Location of the HD6305Y0

Pin location of the HD6305Y0 for the Intelligent Telephone is shown in Fig. 5.2.

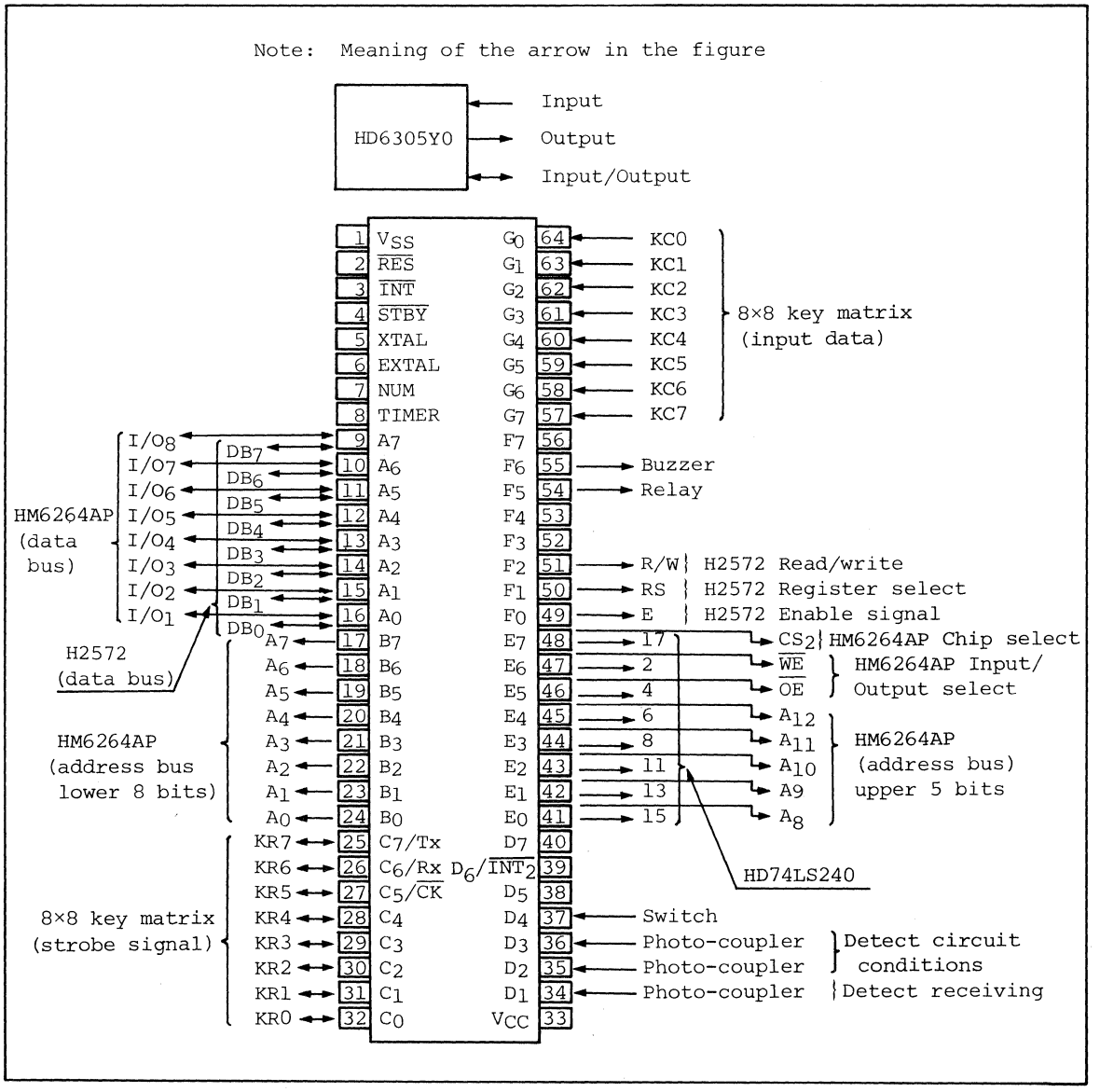


Fig. 5.2 Pin Location of the Intelligent Telephone

5.3 Pin Functions

Pin functions of the Intelligent Telephone is shown in Table 5.1.

"Active Level" in Table 5.1 indicates the following;

High : logical 1

Low : logical 0

— : logical 1 or logical 0

Table 5.1 Pin Functions of the Intelligent Telephone

Pin Name (HD6305Y0)	Input/ Output	Active Level (High or Low)	Function	Devices	Pin Name of Devices	Program Label
A_0	Input/ Output	—	Data bus for RAM (Note)	HM6264AP	I/O_1	PADTR
A_1	Input/ Output	—			I/O_2	
A_2	Input/ Output	—			I/O_3	
A_3	Input/ Output	—			I/O_4	
A_4	Input/ Output	—			I/O_5	
A_5	Input/ Output	—			I/O_6	
A_6	Input/ Output	—			I/O_7	
A_7	Input/ Output	—			I/O_8	
A_0	Input/ Output	—	Data bus for LCD (Note)	H2572	DB_0	PADTR
A_1	Input/ Output	—			DB_1	
A_2	Input/ Output	—			DB_2	
A_3	Input/ Output	—			DB_3	
A_4	Input/ Output	—			DB_4	
A_5	Input/ Output	—			DB_5	
A_6	Input/ Output	—			DB_6	
A_7	Input/ Output	—			DB_7	
B_0	Output	—	Address bus for RAM (Lower 8 bits)	HM6264AP	A_0	PBDTR
B_1	Output	—			A_1	
B_2	Output	—			A_2	
B_3	Output	—			A_3	
B_4	Output	—			A_4	
B_5	Output	—			A_5	
B_6	Output	—			A_6	
B_7	Output	—			A_7	

Note: Data bus for RAM and data bus for LCD use the same port.

Table 5.1 Pin Functions of the Intelligent Telephone (cont.)

Pin Name (HD6305Y0)	Input/ Output	Active Level (High or Low)	Function	Devices	Pin Name of Devices	Program Label
C_0	Input/ Output	Low	Key scan strobe signal	8×8 key matrix	KR0	PCDDR
C_1	Input/ Output	Low			KR1	
C_2	Input/ Output	Low			KR2	
C_3	Input/ Output	Low			KR3	
C_4	Input/ Output	Low			KR4	
C_5	Input/ Output	Low			KR5	
C_6	Input/ Output	Low			KR6	
C_7	Input/ Output	Low			KR7	
D_1	Input	High	Detect receiving	Photo-coupler	Collecter	PDDTR
D_2	Input	—	Detect circuit conditions	Photo-coupler	Emitter	
D_3	Input	—		Photo-coupler	Emitter	
D_4	Input	High	Control hook switch	Switch		
E_0	Output	—	Address bus for RAM (Note) (upper 5 bits)	HM6264AP	A8	PEDTR
E_1	Output	—			A9	
E_2	Output	—			A_{10}	
E_3	Output	—			A_{11}	
E_4	Output	—			A_{12}	
E_5	Output	Low	Data output enable (Note)		$\overline{OE}$	
E_6	Output	Low	Data write enable (Note)		$\overline{WE}$	
E_7	Output	High	RAM chip select (Note)		CS_2	
E_0	Output	—	Data bus for tone output (Note)	HD61826	$\overline{COL1}$	PEDTR
E_1	Output	—			$\overline{COL2}$	
E_2	Output	—			$\overline{COL3}$	
E_3	Output	—			$\overline{ROW1}$	
E_4	Output	—			$\overline{ROW2}$	
E_5	Output	—			$\overline{ROW3}$	
E_6	Output	—			$\overline{ROW4}$	
E_7	Output	Low	Hook switch ON/OFF (Note)		HS	
F_0	Output	High	Enable signal	LCD-II	E	PFDTR
F_1	Output	Low	Select instruction register		RS	
		High	Select data register			
F_2	Output	Low	Write data (microcomputer→ LCD-II)		R/W	
		High	Read data (microcomputer← LCD-II)			
F_5	Output	High	Control relay	Relay		
F_6	Output	—	Output buzzer	Buzzer		

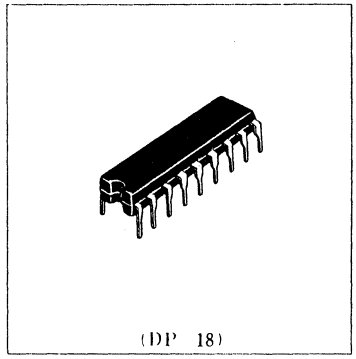
Note: Address bus for RAM (upper 5 bits), control signal and data bus for tone output use the same port.

Table 5.1 Pin Functions of the Intelligent Telephone (cont.)

Pin Name (HD6305Y0)	Input/ Output	Active Level (High or Low)	Function	Devices	Pin Name of Devices	Program Label
G_0	Input	—	Key input data	8×8	KC0	PGDTR
G_1	Input	—		key	KC1	
G_2	Input	—		matrix	KC2	
G_3	Input	—			KC3	
G_4	Input	—			KC4	
G_5	Input	—			KC5	
G_6	Input	—			KC6	
G_7	Input	—			KC7	

Appendix I. HD61826 Data Sheet

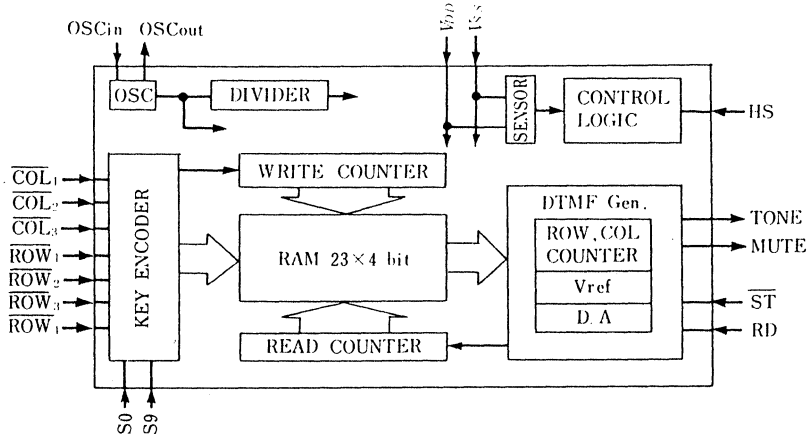
Tone Generator with Redial



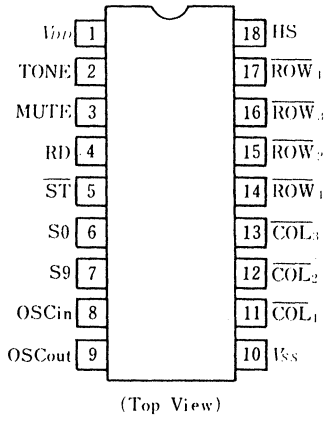
■ FEATURES

- Direct telephone-line operation
- CMOS process for low-power and low-voltage operation
- Uses either a standard 2-of-7 keyboard or the inexpensive matrix keyboard
- Stable operation by using ceramic resonator
- Redial function (# key)
- Pause input (# key)
- 0 or 9 dialing in inhibition pins for PABX system
- 23-digit redial memory
- Redial memory overflow protection (inhibit redial)
- On chip power supply voltage sense circuit
  - Memory clear voltage
  - Reset voltage
- Internal voltage reference circuit for stable Tone output
- Tone output with low distortion

■ BLOCK DIAGRAM



■ PIN ASSIGNMENT



■ ABSOLUTE MAXIMUM RATINGS (Vss = 0V)

Item	Symbol	Value	Unit
Power supply voltage	V_{DD}	6.0	V
Terminal voltage	V_T	V_{SS} −0.3 to V_{DD} +0.3	V
Operating temperature	T_{opr}	−20 to +75	°C
Storage temperature	T_{stg}	−55 to +125	°C

■ ELECTRICAL CHARACTERISTICS

• **DC Characteristics** (V_{SS} = 0V, V_{DD} = 2.0 to 5.5V, T_a = −20 to +75°C)

Item	Symbol	Test Condition	min.	typ. *1	max.	Unit
Operating voltage (1)	V_{DD}	Tone Out Mode	2.5	–	5.5	V
Operating voltage (2)	V_{DD}	Non Tone Out Mode	1.7	–	5.5	V
Reset voltage	V_{DR}		–	1.5	–	V
Memory clear voltage	V_{DC}		–	1.15	–	V
Operating current	I_{DD}	Tone Out Mode, no load	–	300	–	μA
Memory retention (reset) current	I_{DR}		–	0.5	–	μA
Input High voltage	V_{IH}	except HS pin	80% V_{DD}	–	–	V
		HS pin V_{DD} = 3.0 to 5.5V	90% V_{DD}	–	–	
		HS pin V_{DD} = 2.0 to 3.0V	V_{DD}-0.1	–	–	
Input Low voltage	V_{IL}		–	–	20% V_{DD}	V
Input leak current	$\|I_{LI}\|$	Pull-up MOS off, V_{IN} = 0 to V_{DD}	–	–	1	μA
Input Pull-up MOS current	$-I_P$	V_{IN} = 0 (RD, ST, S0, S9, ROW, COL, HS)	–	10	–	μA
Output High voltage	V_{OH}	$-I_{OH}$ = 0.1mA (MUTE)	V_{DD}-0.5	–	–	V
Output Low voltage	V_{OL}	I_{OL} = 0.1mA (ROW, COL)	–	–	0.3	V
Output leak current	$\|I_{LO}\|$	Output MOS off, V_{IN} = 0 to V_{DD} (TONE MUTE)	–	–	1	μA

• **AC Characteristics** (V_{SS} = 0V, V_{DD} = 2.0 to 5.5V, T_a = −20 to +75°C)

Item		Symbol	Test Condition	min.	typ.*1	max.	Unit
Oscillation frequency		f_{osc}		–	400	–	kHz
Oscillation start up time		t_{str}		–	5	–	ms
Tone Out	ROW TONE	V_{OR}	Single Tone Mode, 600Ω to V_{SS}	200	245	290	mVrms
	COLUM TONE	V_{OC}	V_{DD} = 2.5 to 5.5V, T_a = 25°C	270	310	360	mVrms
Tone Out	ROW TONE	V_{OR}	Single Tone Mode, 10kΩ to V_{SS}	245	270	300	mVrms
	COLUM TONE	V_{OC}	V_{DD} = 2.5 to 5.5V, T_a = 25°C	310	340	370	mVrms
ROW/COLUM Tone Out ratio		dB_{CR}	V_{DD} = 2.5 to 5.5V	–	2	–	dB
Output distortion		D_{IS}	10kΩ to V_{SS}, V_{DD} = 2.5 to 5.5V	–	5	7	%

*1 Typ. value is the design value (the standard value at V_{DD} = 2.5V and T_a = 25°C)

■ Description

The HD61826 is specifically designed IC to implement a DTMF (Dual-Tone Multi Frequency) telephone dialing system. With low voltage and low-power consumption CMOS process, it can be operated directly from the telephone line. This IC generates each DTMF signal by digitally synthesizing the sinusoidal waveform for the individual frequencies, using a 400 kHz ceramic oscillation as frequency reference. The last dial numbers can be redialed by the simple key operation using an internal redial memory. The HD61826 can also be used as a normal DTMF dialer without the redial memory by mode select input.

In the HD61826, ON HOOK/OFF HOOK is detected by the HS pin. When the supply voltage is lower than reset voltage, the HD61826 does not accept any key inputs independent of the HS pin. When the power supply voltage is lower than memory clear voltage, the internal memory data is cleared.

While the telephone is in the OFF HOOK and the supply voltage is higher than the reset voltage, the oscillator is enabled by a key input and then this key input is implemented with the key debounce circuit, In this case, if the first key after the reset (note 1) is other than # and *, the internal memory data is cleared, and then this input key is encoded and stored into the memory. The following keys are in turn stored into the memory and converted into DTMF signal outputs. (note 2)

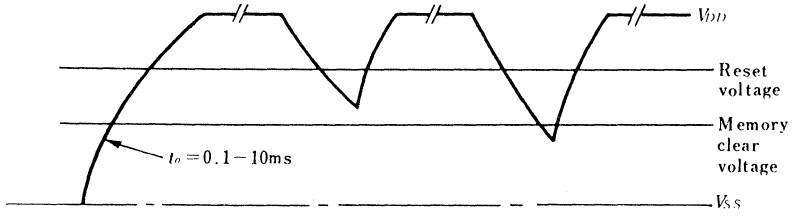
When the HD61826 is reset after dialing, it will be in the redial mode with the first # key. However, if the 24 or more keys have been dialed previously or the memory has already been cleared, it cannot be in the redial mode. During the redial mode, any key input is not accepted, but after the completion of redial, the HD61826 can be used as a usual dialer. The signal output will stop with the pause during the redial and the redial starts again with # key. # key is used to insert the pause data in the memory. In this case, # key does not influence the output of signal but is stored in the memory as one digit.

NOTES:
1. In the HD61826, the reset means the clearing of all logic (counter, etc.) except RAM. HD61826 is reset when the telephone is in the ON HOOK or the supply voltage is lower than the reset voltage.
2. While the key is pushed, the DTMF signal is kept generating.

■ PIN FUNCTION

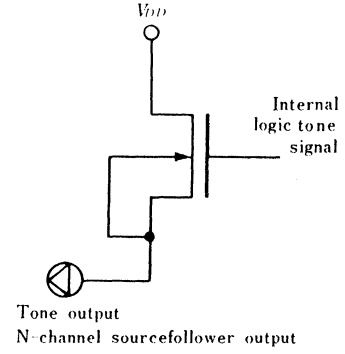
• V_{DD} (Pin 1)

This is a positive voltage supply pin which applies voltage to the basis of the V_{SS} pin. HD61826 provides the internal sense circuit for the supply voltage. To make this circuit operate stable, the following rising time is necessary.

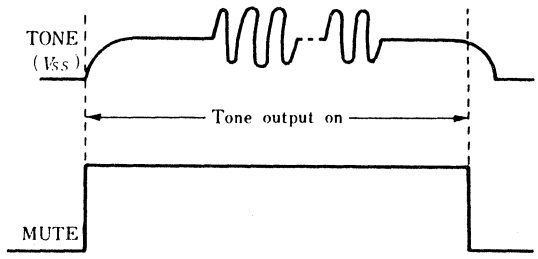


• TONE (Pin 2)

This is a Tone output (DTMF signal output) pin. Output circuit is N-Channel MOS source-follower and the resistor to Vss is necessary. Further, to realize low power consumption in the standby mode, TONE output MOS and internal V_{ref} circuit are turned off after tone output completed. And this is synchronous with MUTE signal. DTMF signal is digitally synthesized by using the 400 kHz oscillation as frequency reference. Tone output frequency of HD61826 and its deviation from standard DTMF are as follows:



		Standard DTMF (Hz)	Tone Output Frequency Using 400 kHz Oscillation	% Deviation from Standard
ROW	f_1	697	694,44	−0.37
	f_2	770	769,23	−0.10
	f_3	852	851,06	−0.11
	f_4	941	938,97	−0.22
COL	f_5	1209	1212,12	0.26
	f_6	1336	1333,33	−0.20
	f_7	1477	1481,48	0.30



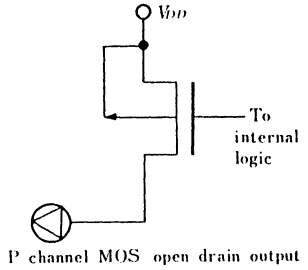
As the HD61826 contains an voltage reference (V_{ref}) circuit, it always generates stable Tone output amplitude even if supply voltage and temperature change.

- **MUTE (Pin 3)**

This is a pin which mutes the receiver and the transmitter. Output circuit is P-Channel MOS open drain.

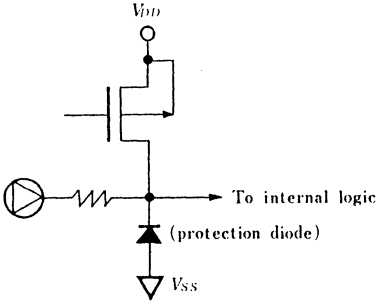
MUTE . . . High level

While reset, the output voltage is held to Low level.



- **RD (Pin 4) Redial Operation**

This is an input pin which selects HD61826 operations: a tone dialer with redial providing memory function, and a simple tone dialer in which only normal key input is converted into tone output. This pin is implemented with the pull-up MOS, and to realize the low power dissipation on reset, the pull-up MOS is turned off at the reset. (note 1)



RD pin	Operation Mode	Redial	Operation
High (to V_{DD}) or open	Tone Dialer with Redial	available	• no Tone out with * or # • Redial/Pause with #
Low (to V_{SS})	Simple Tone Dialer	not available	• Tone out with * or #

NOTES:
1. The input pins, with pull-up MOS which is turned off at the reset, can be applied to RD, $\overline{ST}$, S0, S9, HS.
2. The logic is positive. PMOS is ON with the low voltage and OFF with the high voltage.

- **$\overline{ST}$ (Pin 5) Single Tone Test**

This is an input pin to put HD61826 in the single tone mode for tone output test. This pin is implemented with the pull-up MOS which is turned off at the reset. Further, in the single tone mode, digital signal for test is output on MUTE. Usually $\overline{ST}$ pin should be fixed to High level or open.

$\overline{ST}$ pin	S0 pin	S9 pin	Operation Mode	Tone Output
High (to V_{DD}) or open	–	–	Dual Tone	DTMF Tone out
Low (to V_{SS})	Low	High	Single Tone	to ROW Single tone out
	High	Low		to COL Single tone out

NOTE: S0 and S9 pins should not be Low level at the same time.

- **S0, S9 (Pin 6, 7) Selection**

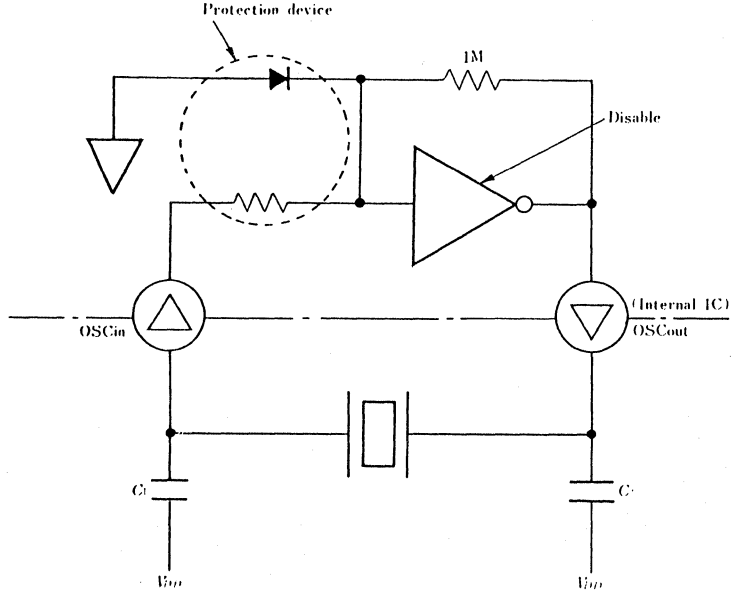
These are input pins to select functions and each of them has pull-up MOS which is turned OFF at the reset. The function of this terminal is to prevent the 0 dialing in and 9 dialing in, which is applied to the telephone subset under the PBX system. When the first key input after the reset is 0 or 9, all the key inputs including the 0 or 9 key become invalid after then. In other words, the signals are not output to TONE and MUTE. Then the telephone is initialized by the reset.

S0 Pin	S9 Pin	Function
High (to V_{DD}) or open	High (to V_{DD}) or open	Normal dialing mode
High (to V_{DD}) or open	Low (to V_{SS})	9 dialing in inhibition mode
Low (to V_{SS})	High (to V_{DD}) or open	0 dialing in inhibition mode
Low (to V_{SS})	Low (to V_{SS})	Test mode (for testing IC. Not use.)

- **OSCin, OSCout (Pin 8, 9) Oscillation Input, Output**

These are the input pins for the oscillator and construct the inverter (with disable function to stop the oscillation). The frequency is stable in the circuit by using the ceramic resonator. Then the oscillator section needs two external capacitors. The ceramic resonator should be 400 kHz and High Q.

Recommended ceramic resonator:
KYOCERA CO.
KBR-400H
ex. ceramic oscillation
C_1 = 100 pF
C_2 = 470 pF



Oscillation Circuit

When OSCout is open, a 400 kHz external pulse can be applied to OSCin.

- **V_{SS} (Pin 10)**

This is a negative power supply pin.

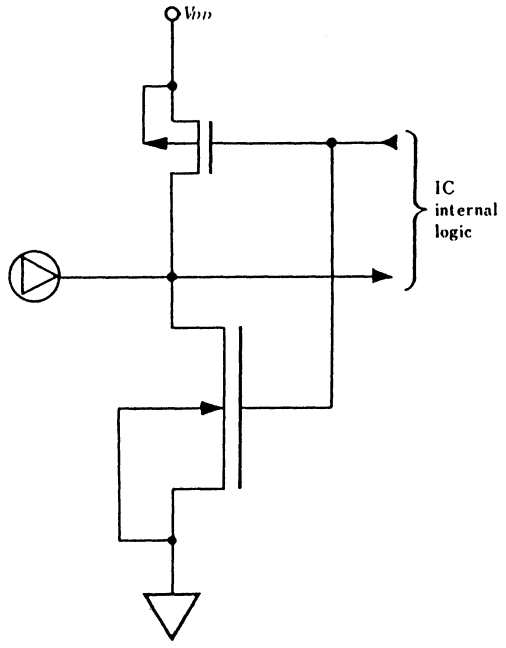
- **$\overline{COL}_1$ to $\overline{COL}_3$ (Pin 11 to 13) Colum Input**
  $\overline{ROW}_1$ to $\overline{ROW}_4$ (Pin 14 to 17) Row Input

These are input/output pins for a key board which consist of PMOS pull-up and NMOS driver. (As a matter of form, these are CMOS.)

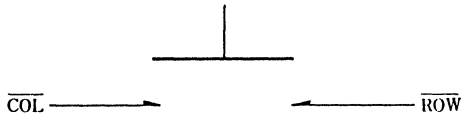
As the row and column are alternately scanned, the HD61826 can be connected both to the matrix-type keyboard and the 2-of-7 keyboard. While waiting for the key input, Rows are High level and Columns are Low level. And in the reset mode, both Row and Column are Low level.

The hold time of the key should be more than 10 ms. (While the oscillation stops, the period for starting oscillation should be added.) (note 1) The key debounce time is 20 ms. Tone is remaining while pushing the key (precisely speaking, after key operation, tone continues during the debounce time). But the key operation should meet the DTMF receiver specification.

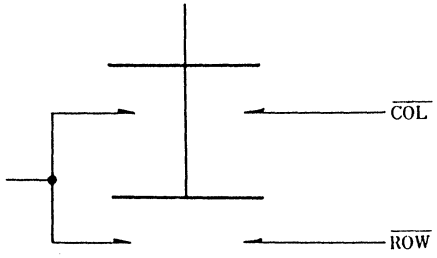
NOTES:
1. The oscillation stops in HD61826;
   1. After the reset
   2. After the completkon of Tone output
   3. On Pause



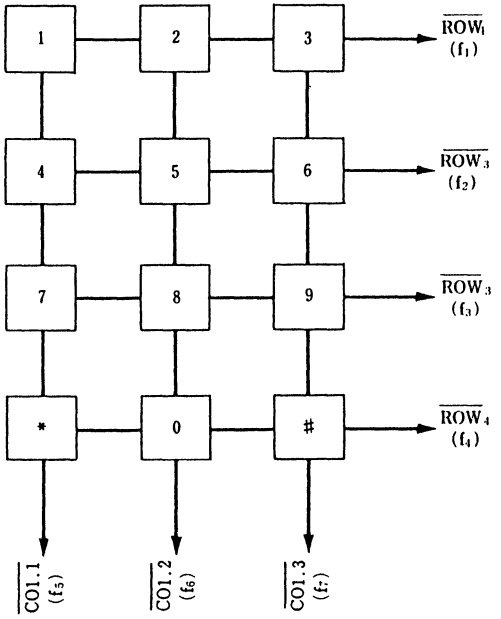
I/O Circuit for Keyboard



Matrix Keyboard



2–of–7 Keyboard



Keyboard Configuration

When two keys are pushed at the same time, the key of the smaller number of ROW and COL is given priority and is entered.

Ex. When 2 and 5 are pushed at the same time, 2 is accepted.

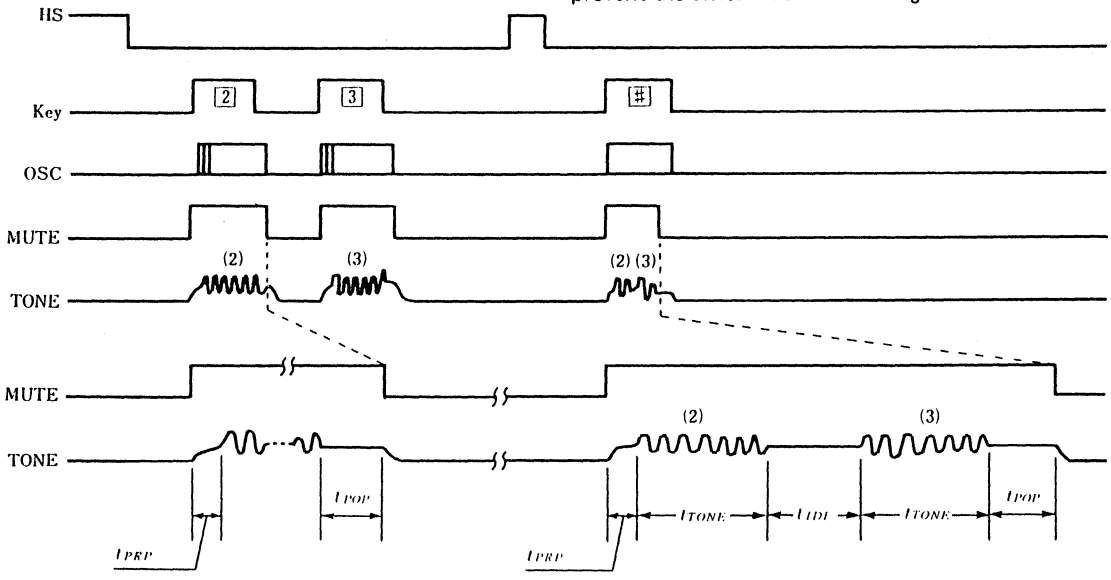
- **HS (Pin 18) Hook Switch**

This is an input pin for detecting ON-HOOK/OFF-HOOK switch. It has a pull-up MOS which is turned off at the reset.

ON HOOK . . . . High (to V_{DD}) or open
OFF HOOK . . . Low (to V_{SS})

HD61826 is reset by setting the HS terminal High level. Even if the HS terminal is fixed to Low level, the HD61826 is reset when supply voltage is lower than reset voltage. So without using the HS pin the HD61826 can control the operation mode by the monitor of supply voltage. Then, as the pull-up MOS is turned off with the reset signal, the current does not increase on reset. An external capacitor must be provided between the HS pin and V_{SS} to prevent the switch from chattering.

■ TIMING CHART



Mode	Item	Symbol	min.	typ.	max.	Unit
Normal Dial	Pre-Digital Pause	t_{PRP}	–	5	–	ms
	Post-Digital Pause	t_{POP}	–	44	–	ms
Redial	Pre-Digital Pause	t_{PRP}	–	5	–	ms
	Tone Output time	t_{TONE}	–	133	–	ms
	Inter-Digital Pause	t_{IDP}	–	87	–	ms
	Post-Digital Pause	t_{POP}	–	44	–	ms

■ AN EXAMPLE OF KEY OPERATION (RD = High)

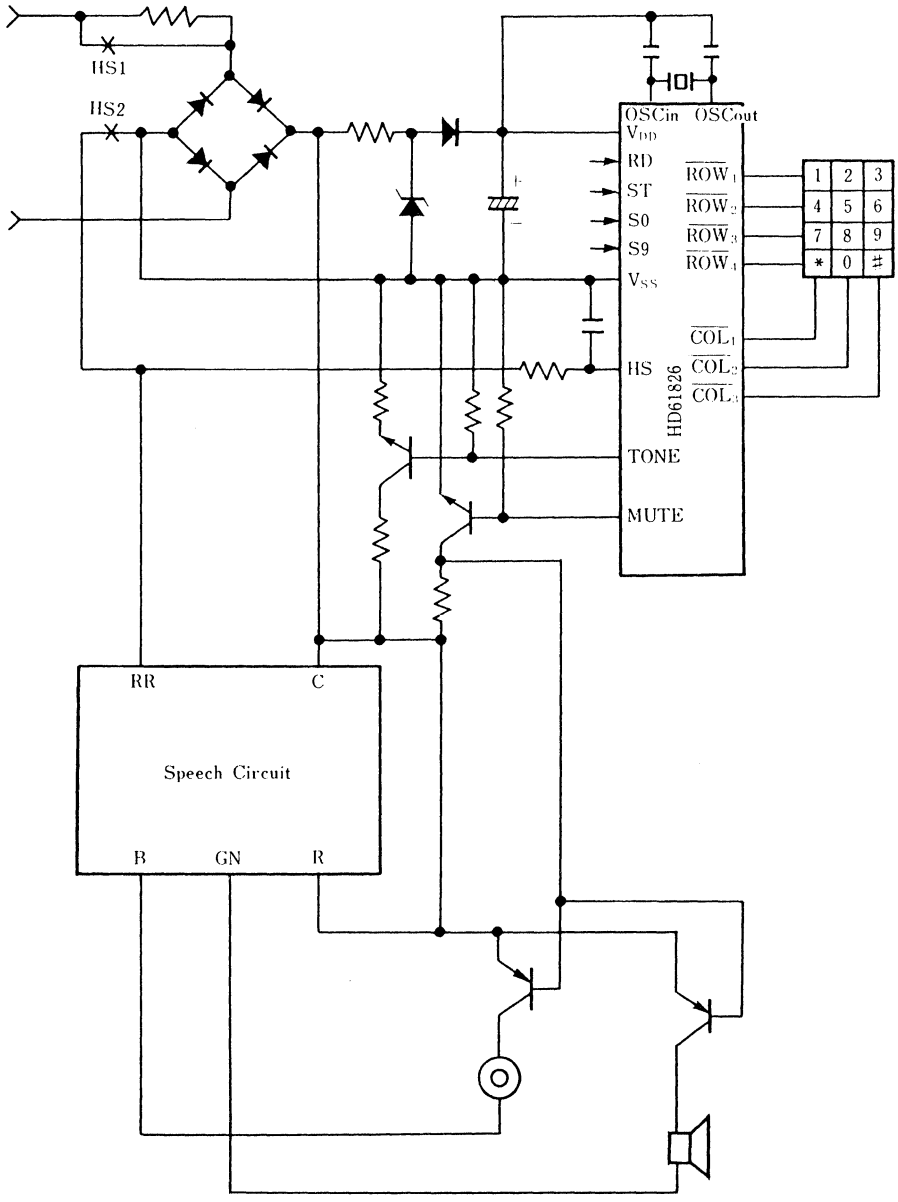
	HOOK	KEY	TONE
Normal Dial	ON OFF	0 1 2 3 4 5 6	0, 1, 2, 3, 4, 5, 6, (note)
Redial	ON OFF	↑	0 1 2 3 4 5 6 (note)
Dial after Redial		7 8	7, 8,
Redial	ON OFF	↑	0 1 2 3 4 5 6 7 8
Normal Dial (Pause Entry)	ON OFF	9 ↑ 1 2	9, 1, 2,
Redial (including Pause)	ON OFF	↑ #	9, 1 -2,
Dial of 24 digits or more	ON OFF	1 1 . . . . . 1 24 times	1, 1, ... 1, 24 times
Prevention of Over Flow	ON OFF	# 0 1	0, 1

(0 dialing in inhibition mode with S0 = Low and S9 = High)

ON OFF	1 2 3 0	
ON OFF	0 4 5 6 7 ↑	
ON OFF	#	1 2 3 0,

NOTE: , shows that after tone output Mute is Low level and - shows that Mute is High level.

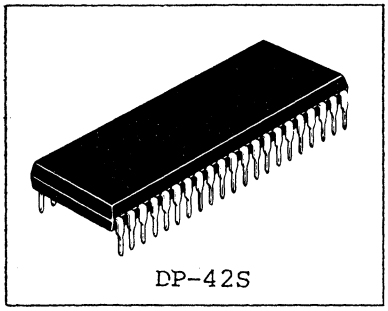
■ APPLICATION CIRCUIT



-Preliminary-

■ Abstract

The HA16808NT is a monolithic IC including speech network, tone ringer, and speaker amp. Therefore the telephone with speaker can be composed of HA16808NT and dialer IC.

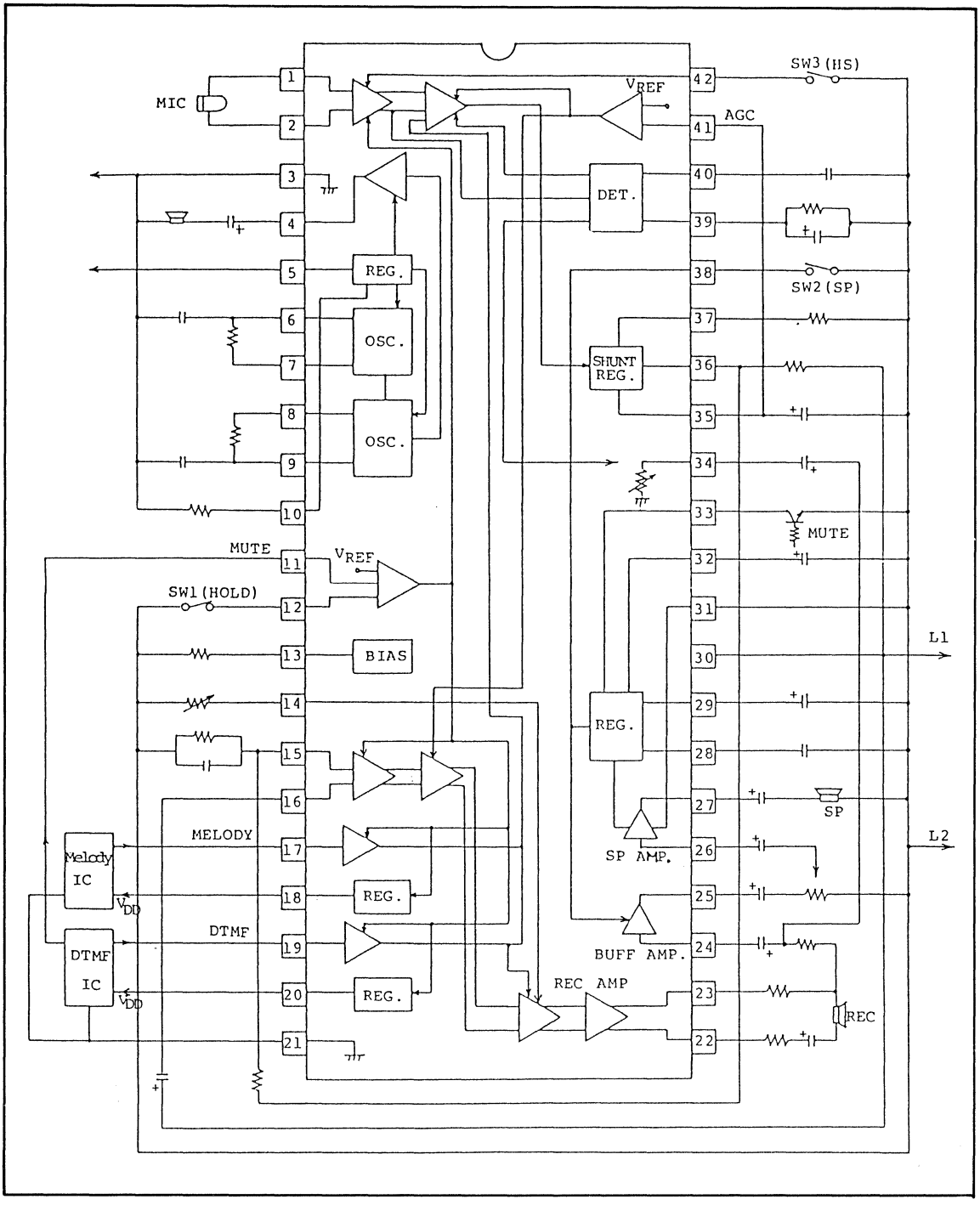


DP-42S

■ Features

- Low voltage operation ------ (1.8V)
- AGC according to the line current (Gains of sending, receiving, DTMF, and melody)
- Adjustable receiving gain by external resistor (0 ~ +10dB)
- Interface for DTMF (V_{DD}, MUTE, Sending amp. of DTMF signal)
- Interface for melody IC (V_{DD}, Sending amp. of melody signal)
- Noise suppression (No output for the input below noise level. The noise level can be controlled by external components.)
- Possible to dial under the on-hook condition (Included speaker amp.)
- Possible to receive by speaker (Also possible to send with handset in speaker mode.)
- At the time to send DTMF or melody, backtone comes out from receiver or speaker.
- Variable oscillation freq. of ringer by external R, C.
- Variable supply initiation current.

■ Block Diagram



■ Maximum Rating (Ta=25℃)

NO	Item	Symbol	Rating	Unit	Note
1	Supply Voltage (OFF HOOK)	V_L	20	V	1
2	Supply Current (OFF HOOK)	I_L	120	mA	
3	Supply Voltage(Tone Ringer)	V_{TR}	30	V	
4	Operation Temperature	Topr	−20 ~ +70	℃	
5	Storage Temperature	Tstg	−55 ~ +125	℃	
6	Power Dissipation	P_T	950	mW	

Note 1) 3mS Pulse duration

■ Electrical Characteristics

(handset mode)

NO	Item		Symbol	Test Condition		Spec.			Unit
				I_L mA		min.	typ.	max.	
1	Supply Voltage	Speaking	V_L	20		2.3	2.6	2.8	V
2				80		5.0	6.5	8.0	V
3		Dialing		20		5.5	6.2	6.6	V
4				80		7.0		10.0	V
5	Receiving Gain		G_R	30	f=1kHz	−6.5	−5	−3.5	dB
6				80		−12.5	−11	−9.5	dB
7		Up Mode		30	R_{EXBs}=560Ω	5	10	15	dB
8	Sending Gain		G_T	30	f=1kHz	50	51.5	53	dB
9				80		44.5	46	47.5	dB
10	Side tone		G_{SID}	30	f=1kHz			40	dB
11	DTMF Sending Gain		G_{MF}	30		22.5	24	25.5	dB
12				80		19	20.5	22	dB
13	Sending Dynamic Range		DR_T	30	f=1kHz	2.2	2.7		V_{P-P}
14				80		4.5	6.0		V_{P-P}
15	Receiving Dynamic		DR_R	20	f=1kHz	0.4	0.6		V_{P-P}
16	Range			80		0.8			V_{P-P}
17	DTMF Dynamic Range		DR_{MF}	20		2.8	4.0		V_{P-P}
18				80		3.0			V_{P-P}
19	DTMF Supply Voltage	Stand−by	V_{DD}	20		1.6	1.8		V

NO	Item		Symbol	Test Conditions		Spec.			Unit
				I_L mA		min.	typ.	max.	
21	DTMF Supply Voltage	Mute	V_{DD}	20		3.5	3.8		V
22	DTMF Supply Current	Stand−by	I_{DD}	20		200			μA
23		Mute		20		2			mA
24	DTMF Backtone		BT_{MF}	80	V_{IN}=170mVp-p	40	60	80	mVp-p
25	Line Matching Impedance		Z_{IN}	20,80	f=1kHz	500	600	700	Ω

(speaker mode)

NO	Item		Symbol	Test Conditions		Spec.			Unit
				I_L mA		min.	typ.	max.	
1	Supply Voltage	Speaking	V_{LSP}	20		3.7	4.1	4.5	V
2				80		6		9	V
3		Dialing		20		5.2	6.0	6.8	V
4				80		7		10	V
5	Receiving Gain		G_{RSP}	30	f=1kHz	−12.5	−10	−7.5	dB
6				80		−17.5	−15	−12.5	dB
7	Sending Gain		G_{TSP}	30	f=1kHz	47.5	49.5	51.5	dB
8				80		40	42	44	dB
9	Side tone		G_{SIDSP}	30	f=1kHz			45	dB
10	DTMF Sending Gain		G_{MFSP}	30		21.5	23.5	25.5	dB
11				80		17.5	19.5	21.5	dB
12	Sending Dynamic Range		DR_{TSP}	50	f=1kHz	2.3			Vp-p
13	Receiving (SP) Dynamic Range		DR_{SP}	50	Speaker Output	0.8			Vp-p
14	DTMF Dynamic Range		DR_{MFSP}	20		2.5			Vp-p
15	DTMF Backtone	Receiver	BT_{MFSP}	25	Vin=120mVp-p	40	60	80	mVp-p
16		Speaker		50		400	500	600	mVp-p
17	Line Matching Impedance		Z_{INSP}	20,80		270		700	Ω
18	Speaker Amp. Gain		G_{SP}	30		5	8.5	12	dB

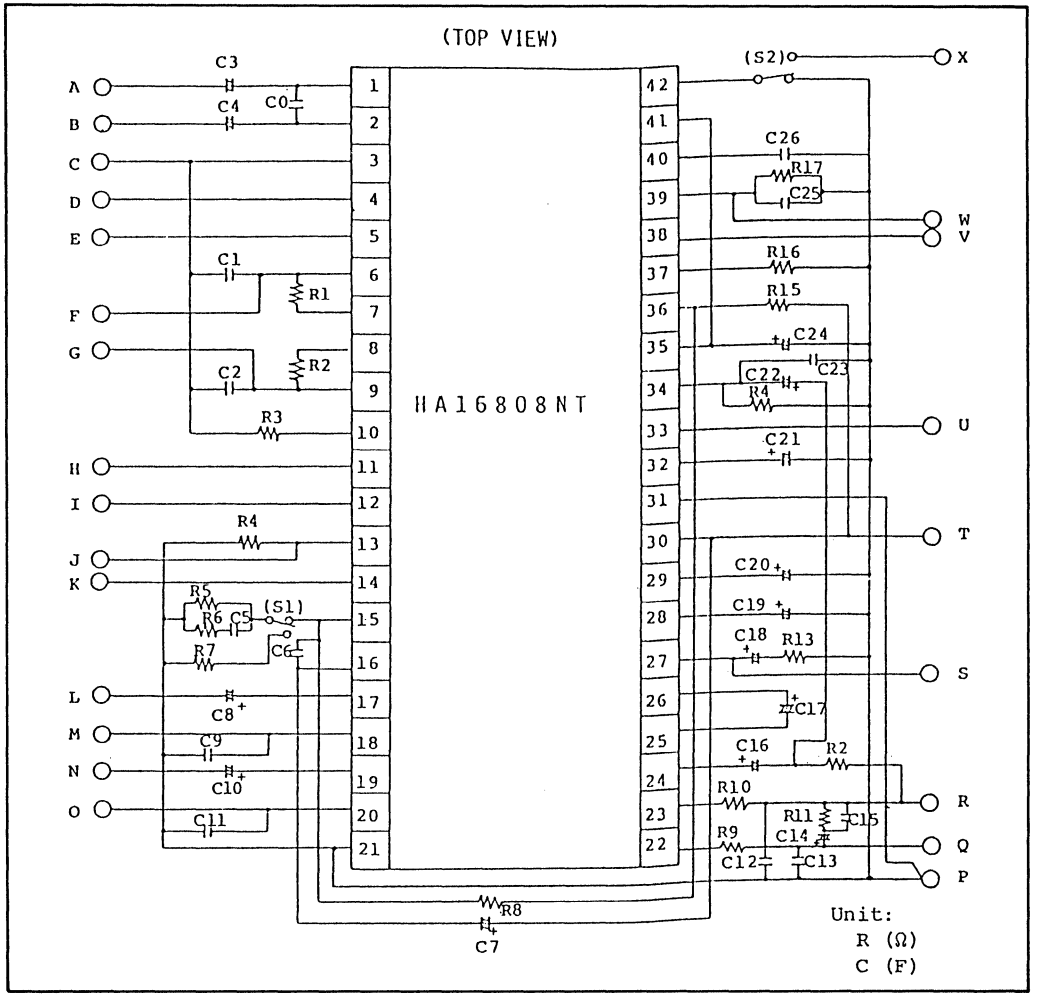
(melody mode)

NO	Item		Symbol	Test Conditions		Spec.			Unit
				I_L mA		min.	typ.	max.	
1	Supply Voltage		V_{LHD}	20		3.7	4.2	4.7	V
2				80		6		9	V
3	Melody IC	Voltage	V_{DH}	20		1.2	1.5	1.8	V
4	Supply	Current	I_{DH}	20		200			μA
5	Melody Sending Gain		G_{HD}	30	f=1kHz	23	25	27	dB
6				80		20	22	24	dB
7	Melody Sending Dynamic Range		DR_{HD}	30	f=1kHz	2.2			V_{p-p}
8	Melody Back	Receiver	BT_{HD}	30	f=1kHz	30	60	90	mV_{p-p}
9	tone	Speaker		30	Vin=70mV_{p-p}	350	450	550	mV_{p-p}

(tone ringer)

NO	Item	Symbol	Test Conditions	Spec.			Unit
				min.	typ.	max.	
1	Supply Initiation Voltage	V_{TH}		17	19	21	V
2	Supply Initiation Current	I_{TH}		0.7	1.5	3.0	mA
3	Sustaining Voltage	V_{SUS}		9	11		V
4	Sustaining Current	I_{SUS}	Vin=15V	0.5	1.0	2.0	mA
5	Output "H" Voltage	V_{OH}	Vin=24V, I_{OH}=−10mA	20.0	21.5	22.5	V
6	Output "L" Voltage	V_{OL}	Vin=24V, I_{OL}=10mA	0.7	1.0	2.0	V
7	Output Frequency	f_L	R_1=165kΩ, C_1=0.47μF	9	10	11	Hz
8		f_{H1}	R_2=190kΩ, C_2=6800pF	460	510	565	Hz
9		f_{H2}	Vin=24V	575	640	705	Hz

Test Circuit



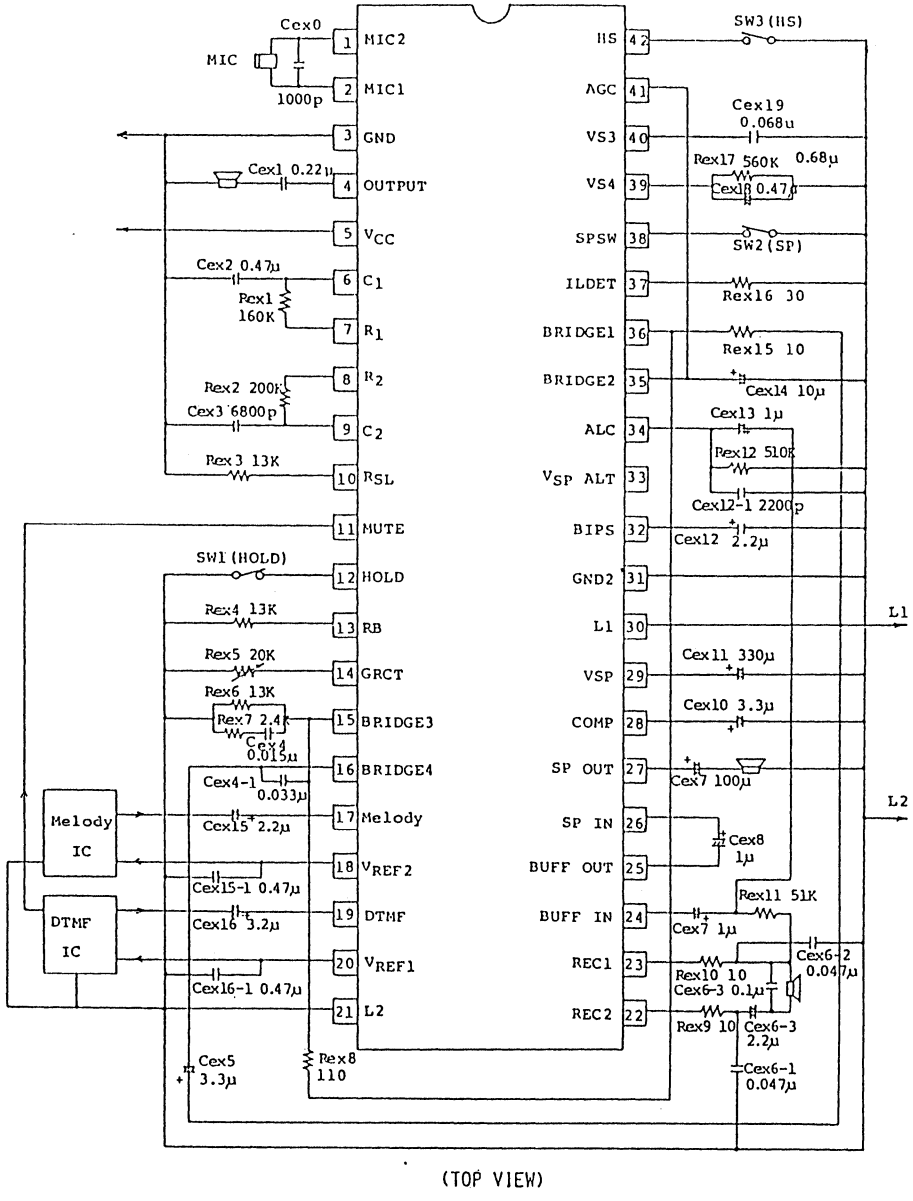
CNo.	C Value	CNo.	C Value	CNo.	C Value	CNo.	C Value	CNo.	C Value	CNo.	C Value
0	1000P	1	0.47μ	2	6800p	3	22μ	4	22μ	5	0.015μ
6	0.033μ	7	3.3μ	8	2.2μ	9	0.47μ	10	2.2μ	11	0.47μ
12	0.047μ	13	0.047μ	14	2.2μ	15	0.1μ	16	1μ	17	1μ
18	100μ	19	3.3μ	20	330μ	21	2.2μ	22	1μ	23	2200p
24	10μ	25	0.47μ	26	0.068μ						

RNo.	R Value	RNo.	R Value	RNo.	R Value	RNo.	R Value	RNo.	R Value	RNo.	R Value
1	165K	2	190K	3	13K	4	13K	5	13K	6	2.4K
7	12K	8	110	9	10	10	10	11	180	12	51K
13	8	14	510K	15	10	16	30	17	560K		

■ Pin Description

Pin No.	Symbol	Function	Function	Symbol	Pin No.
1	MIC 1	Mic. Input	Hook Switch	HS	42
2	MIC 2		AGC	AGC	41
3	GND	Tone Ringer	Voice SW Gain Adjustment	VS3	40
4	OUTPUT		(Noise Suppression)		
5	V_{CC}		Decide the Suppression	VS4	39
6	LOW FREQ.		Level (Speaker Receive)		
7	TIME		On at Speaker Mode	SP SW	38
	CONSTANT		Detect the Line Current	ILDET	37
8	HIGH FREQ.		Bridge part 1	BRG1	36
9	TIME		Bridge part 2	BRG2	35
	CONSTANT		Loss Pad on Speaker mode	ALC	34
10	R_{SL}		Variable V_{SP} Voltage	$V_{SP\ ALT}$	33
11	MUTE	DTMF Mode ($\geq$1.6V)	Regulating Capacitor	BIPS	32
12	HOLD	On at Melody Mode	Speaker Amp. GND	GND 2	31
			Line	L1	30
13	R_B	Decide IC Bias Current	Speaker Part Regulator	V_{SP}	29
14	GRCT	Adjust Receiving Gain	Compensating Capacitor	COMP1	28
15	BRG3	Receiving Signal Input	Speaker Amp. Output	SP OUT	27
16	BRG4	(one part of the bridge)	Speaker Amp. Input	SP IN	26
17	MEL	Melody Input	Buffer Amp. Output	BUFF OUT	25
18	V_{REF2}	Melody IC Supply	Buffer Amp. Input	BUFF IN	24
		Voltage	Receiver Output	REC1	23
19	DTMF	DTMF Input		REC2	22
20	V_{REF1}	DTMF IC Supply Voltage	Line (GND)	L2	21

■ Connection Arrangement and Application Circuit

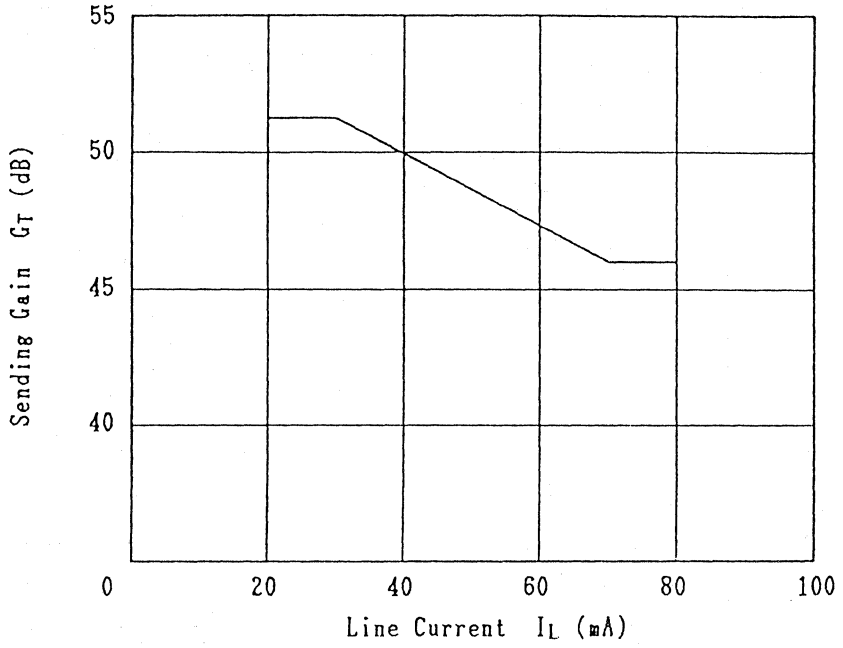


(TOP VIEW)

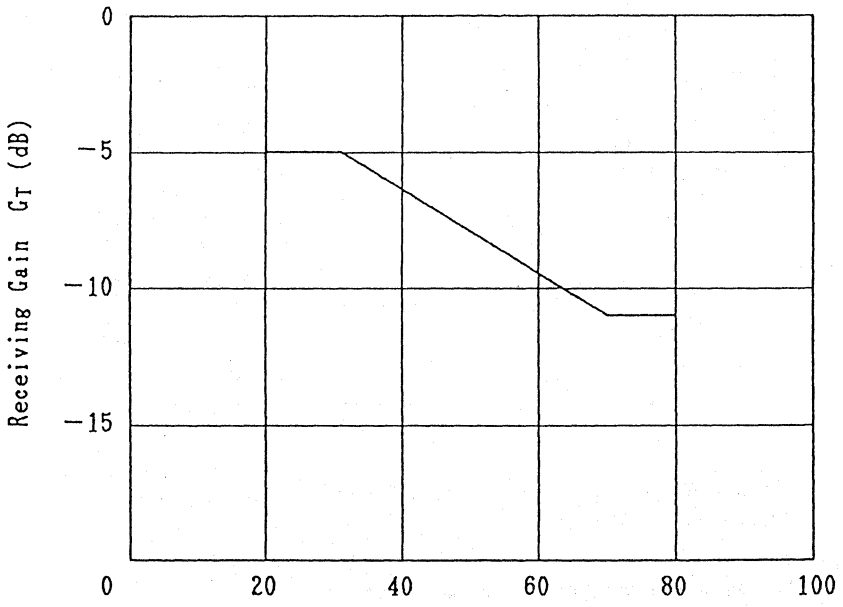
Note) ㉛ Pin must be separated with other GNDs and be connected to L2 directly. Unit (R (Ω), C (F))

External Components are only for reference.

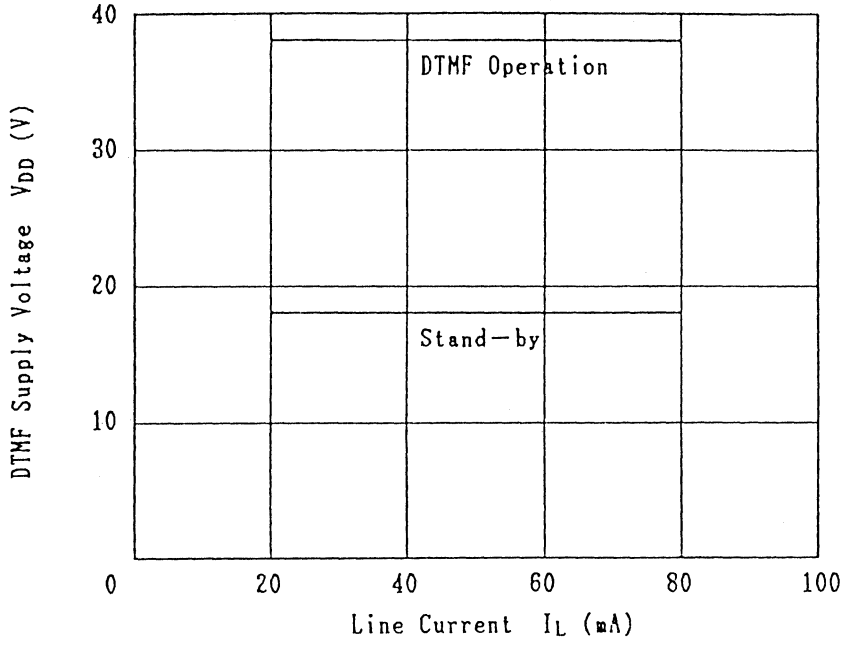
SENDING GAIN VS. LINE CURRENT



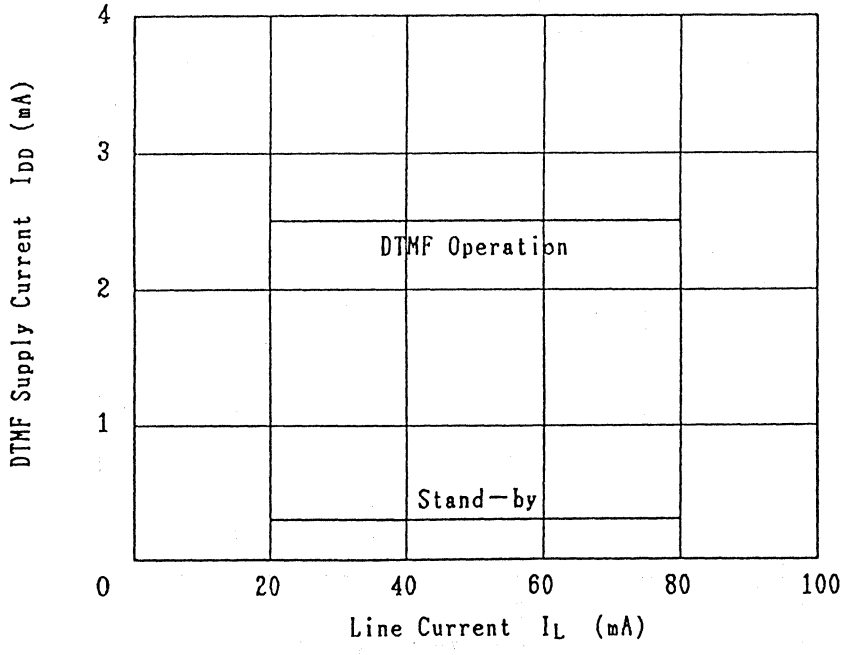
RECEIVING GAIN VS. LINE CURRENT



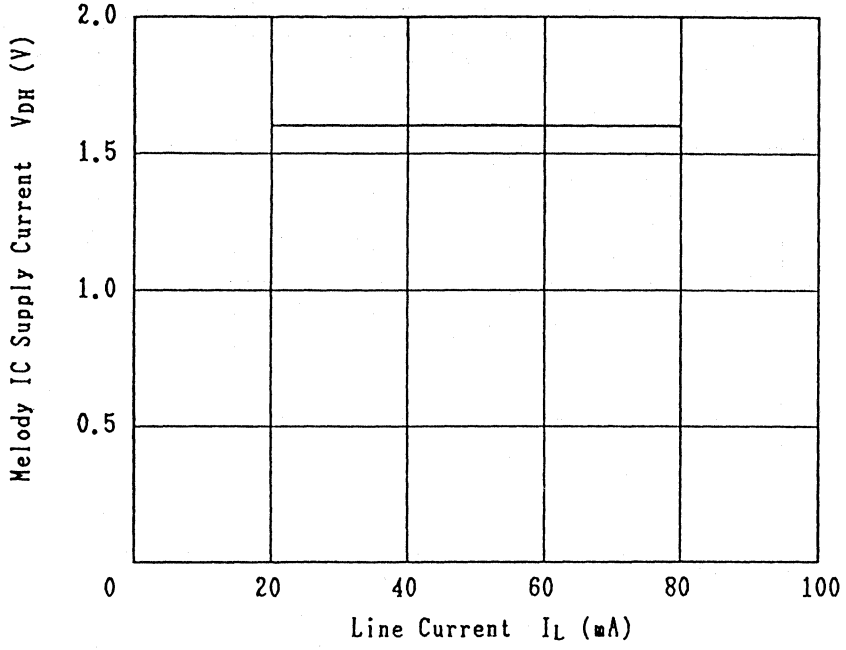
DTMF SUPPLY VOLTAGE VS. LINE CURRENT



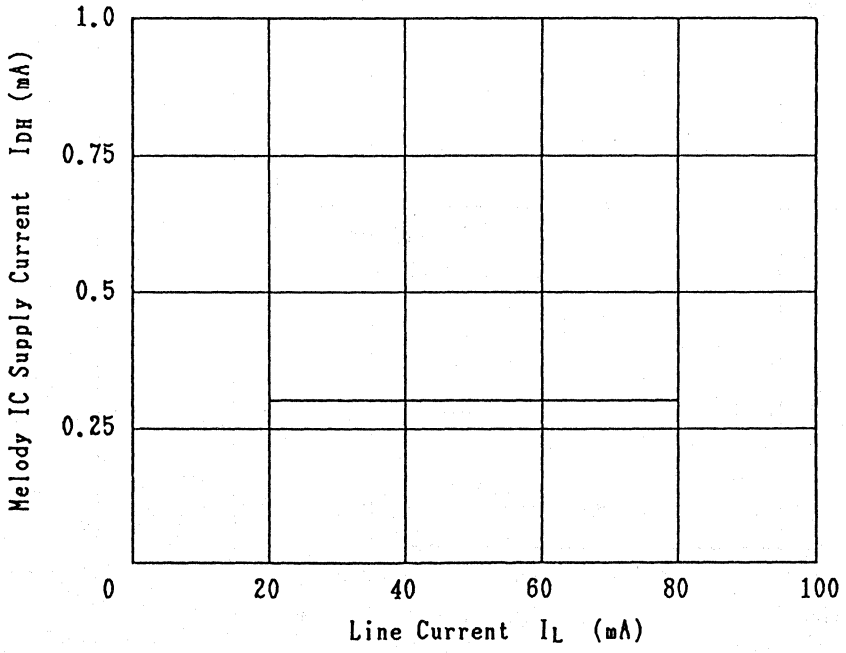
DTMF SUPPLY CURRENT VS. LINE CURRENT



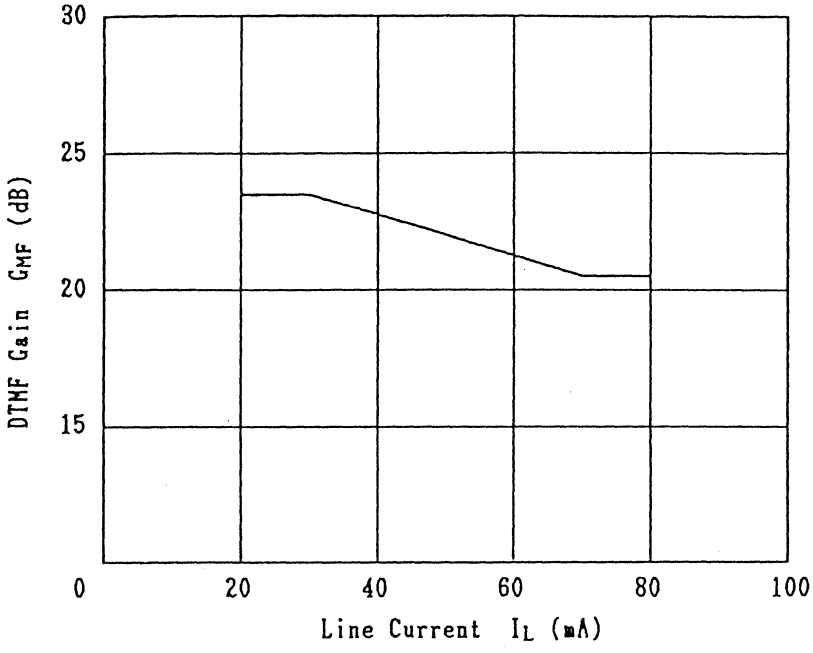
MELODY IC SUPPLY VOLTAGE VS. LINE CURRENT



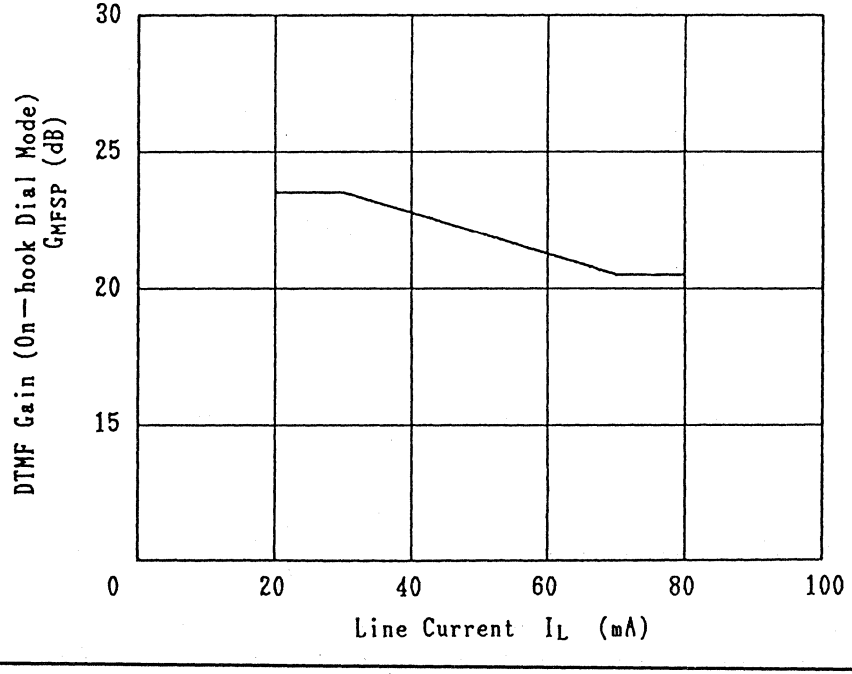
MELODY IC SUPPLY CURRENT VS. LINE CURRENT



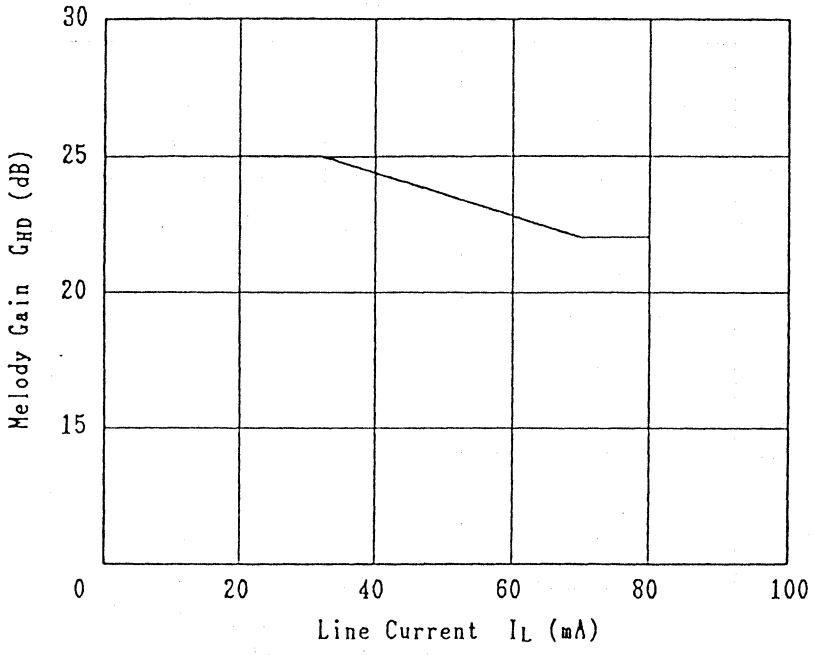
DTMF GAIN VS. LINE CURRENT



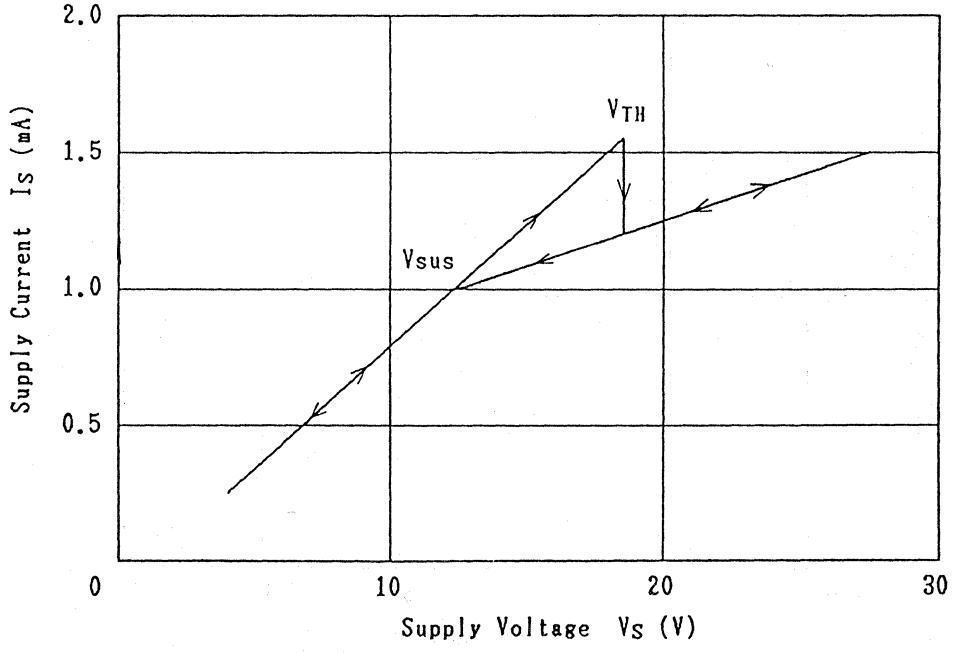
DTMF GAIN(ON-HOOK DIAL MODE) VS. LINE CURRENT



MELODY GAIN VS. LINE CURRENT



TONE RINGER SUPPLY VOLTAGE VS. SUPPLY CURRENT



Appendix III. Instruction Set of the HD6305 Family

Meanings of Symbols and abbreviations in the instruction set are as follows.

(1) Operation

( ) = Contents

$\rightarrow$ = Data transfer direction

+ = Addition

- = Subtraction

$\wedge$ = AND

$\vee$ = OR

$\oplus$ = Exclusive OR

$\overline{X}$ = NOT

(2) Register symbols in MCU/MPU

A = Accumulator A

X = Index register

SP = Stack pointer, 6 bits

CCR = Condition code register

(3) Contents of bits 0 through 4 of condition code register

C = Carry - borrow bit 0

Z = Zero bit 1

N = Negative bit 2

I = Interrupt mask bit 3

H = Half carry from bit 3 to bit 4 bit 4

(4) Memory and addressing codes

M = Stored address

M+1 = Stored address M plus 1

IMP = Implied addressing

Rel = Relative addressing

A = Accumulator addressing

X = Index register addressing

IMM = Immediate addressing

DIR = Direct addressing

EXT = Extended addressing

,X0 = Indexed addressing 0 byte offset

,X1 = Indexed addressing 1 byte offset

,X2 = Indexed addressing 2 byte offset

(5) Status of each bit before execution of instruction

Mn = Bit n of M (n=7, 6, 5, ..., 0)

(6) Symbols in instruction format

A = Accumulator addressing mode

X = Index register addressing mode

n = Bit n of memory (n=7, 6, 5, ..., 0)

(7) Status of interrupt pin

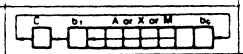
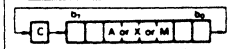
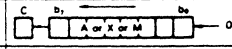
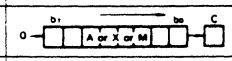
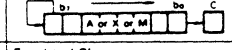
$\overline{INT}$ = Status of interrupt pin (high, low)

Register/Memory Instructions

Operations	Mnemonic	Addressing Modes																		Boolean/ Arithmetic Operation	Condition Code				
		Immediate			Direct			Extended			Indexed (No Offset)			Indexed (8-Bit Offset)			Indexed (16-Bit Offset)								
		OP	#	~	OP	#	~	OP	#	~	OP	#	~	OP	#	~	OP	#	~		H	I	N	Z	C
Load A from Memory	LDA	A6	2	2	B6	2	3	C6	3	4	F6	1	3	E6	2	4	D6	3	5	M→A	•	•	^	^	•
Load X from Memory	LDX	AE	2	2	BE	2	3	CE	3	4	FE	1	3	EE	2	4	DE	3	5	M→X	•	•	^	^	•
Store A in Memory	STA	-	-	-	B7	2	3	C7	3	4	F7	1	4	E7	2	4	D7	3	5	A→M	•	•	^	^	•
Store X in Memory	STX	-	-	-	BF	2	3	CF	3	4	FF	1	4	EF	2	4	DF	3	5	X→M	•	•	^	^	•
Add Memory to A	ADD	AB	2	2	BB	2	3	CB	3	4	FB	1	3	EB	2	4	DB	3	5	A+M→A	^	•	^	^	^
Add Memory and Carry to A	ADC	A9	2	2	B9	2	3	C9	3	4	F9	1	3	E9	2	4	D9	3	5	A+M+C→A	^	•	^	^	^
Subtract Memory	SUB	A0	2	2	B0	2	3	C0	3	4	F0	1	3	E0	2	4	D0	3	5	A−M→A	•	•	^	^	^
Subtract Memory from A with Borrow	SBC	A2	2	2	B2	2	3	C2	3	4	F2	1	3	E2	2	4	D2	3	5	A−M−C→A	•	•	^	^	^
AND Memory to A	AND	A4	2	2	B4	2	3	C4	3	4	F4	1	3	E4	2	4	D4	3	5	A∧M→A	•	•	^	^	•
OR Memory with A	ORA	AA	2	2	BA	2	3	CA	3	4	FA	1	3	EA	2	4	DA	3	5	A∨M→A	•	•	^	^	•
Exclusive OR Memory with A	EOR	A8	2	2	B8	2	3	C8	3	4	F8	1	3	E8	2	4	D8	3	5	A⊕M→A	•	•	^	^	•
Arithmetic Compare A with Memory	CMP	A1	2	2	B1	2	3	C1	3	4	F1	1	3	E1	2	4	D1	3	5	A−M	•	•	^	^	^
Arithmetic Compare X with Memory	CPX	A3	2	2	B3	2	3	C3	3	4	F3	1	3	E3	2	4	D3	3	5	X−M	•	•	^	^	^
Bit Test Memory with A (Logical Compare)	BIT	A5	2	2	B5	2	3	C5	3	4	F5	1	3	E5	2	4	D5	3	5	A∧M	•	•	^	^	•
Jump Unconditional	JMP				BC	2	2	CC	3	3	FC	1	2	EC	2	3	DC	3	4		•	•	•	•	•
Jump to Subroutine	JSR				BD	2	5	CD	3	6	FD	1	5	ED	2	5	DD	3	6		•	•	•	•	•

Symbols: Op = Operation
= Number of bytes
~ = Number of Cycles

Read/Modify/Write Instructions

Operations	Mnemonic	Addressing Modes: Implied(A) OP	#	~	Implied(X) OP	#	~	Direct OP	#	~	Indexed (No Offset) OP	#	~	Indexed (8-Bit Offset) OP	#	~	Boolean/Arithmetic Operation	Condition Code H	I	N	Z	C
Increment	INC	4C	1	2	5C	1	2	3C	2	5	7C	1	5	6C	2	6	A+1→A or X+1→X or M+1→M	•	•	^	^	•
Decrement	DEC	4A	1	2	5A	1	2	3A	2	5	7A	1	5	6A	2	6	A−1→A or X−1→X or M−1→M	•	•	^	^	•
Clear	CLR	4F	1	2	5F	1	2	3F	2	5	7F	1	5	6F	2	6	00→A or 00→X or 00→M	•	•	0	1	•
Complement	COM	43	1	2	53	1	2	33	2	5	73	1	5	63	2	6	$\overline{A}$→A or $\overline{X}$→X or $\overline{M}$→M	•	•	^	^	1
Negate (2's Complement)	NEG	40	1	2	50	1	2	30	2	5	70	1	5	60	2	6	00−A→A or 00−X→X or 00−M→M	?	•	^	^	^
Rotate Left Thru Carry	ROL	49	1	2	59	1	2	39	2	5	79	1	5	69	2	6		?	•	^	^	^
Rotate Right Thru Carry	ROR	46	1	2	56	1	2	36	2	5	76	1	5	66	2	6		•	•	^	^	^
Logical Shift Left	LSL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6		•	•	^	^	^
Logical Shift Right	LSR	44	1	2	54	1	2	34	2	5	74	1	5	64	2	6		•	•	0	^	^
Arithmetic Shift Right	ASR	47	1	2	57	1	2	37	2	5	77	1	5	67	2	6		•	•	^	^	^
Arithmetic Shift Left	ASL	48	1	2	58	1	2	38	2	5	78	1	5	68	2	6	Equal to LSL	•	•	^	^	^
Test for Negative or Zero	TST	4D	1	2	5D	1	2	3D	2	4	7D	1	4	6D	2	5	A−00 or X−00 or M−00	•	•	^	^	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Branch Instructions

Operations	Mnemonic	Addressing Modes: Relative OP	#	~	Branch Test	Condition Code H	I	N	Z	C
Branch Always	BRA	20	2	3	None	•	•	•	•	•
Branch Never	BRN	21	2	3	None	•	•	•	•	•
Branch IF Higher	BHI	22	2	3	C∨Z=0	•	•	•	•	•
Branch IF Lower or Same	BLS	23	2	3	C∨Z=1	•	•	•	•	•
Branch IF Carry Clear	BCC	24	2	3	C=0	•	•	•	•	•
(Branch IF Higher or Same)	(BHS)	24	2	3	C=0	•	•	•	•	•
Branch IF Carry Set	BCS	25	2	3	C=1	•	•	•	•	•
(Branch IF Lower)	(BLO)	25	2	3	C=1	•	•	•	•	•
Branch IF Not Equal	BNE	26	2	3	Z=0	•	•	•	•	•
Branch IF Equal	BEQ	27	2	3	Z=1	•	•	•	•	•
Branch IF Half Carry Clear	BHCC	28	2	3	H=0	•	•	•	•	•
Branch IF Half Carry Set	BHCS	29	2	3	H=1	•	•	•	•	•
Branch IF Plus	BPL	2A	2	3	N=0	•	•	•	•	•
Branch IF Minus	BMI	2B	2	3	N=1	•	•	•	•	•
Branch IF Interrupt Mask Bit is Clear	BMC	2C	2	3	I=0	•	•	•	•	•
Branch IF Interrupt Mask Bit is Set	BMS	2D	2	3	I=1	•	•	•	•	•
Branch IF Interrupt Line is Low	BIL	2E	2	3	INT=0	•	•	•	•	•
Branch IF Interrupt Line is High	BIH	2F	2	3	INT=1	•	•	•	•	•
Branch to Subroutine	BSR	AD	2	5	——	•	•	•	•	•

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

HITACHI

Bit Manipulation Instructions

Operations	Mnemonic	Addressing Modes						Boolean/ Arithmetic Operation	Branch Test	Condition Code				
		Bit Set/Clear			Bit Test and Branch					H	I	N	Z	C
		OP	#	~	OP	#	~							
Branch IF Bit n is set	BRSET n(n=0···7)	–	–	–	2·n	3	5	——	Mn=1	●	●	●	●	↕
Branch IF Bit n is clear	BRCLR n(n=0···7)	–	–	–	01+2·n	3	5	——	Mn=0	●	●	●	●	↕
Set Bit n	BSET n(n=0···7)	10+2·n	2	5	–	–	–	1→Mn	——	●	●	●	●	●
Clear Bit n	BCLR n(n=0···7)	11+2·n	2	5	–	–	–	0→Mn	——	●	●	●	●	●

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

Control Instructions

Operations	Mnemonic	Addressing Modes			Boolean Operation	Condition Code				
		Implied				H	I	N	Z	C
		OP	#	~						
Transfer A to X	TAX	97	1	2	A→X	●	●	●	●	●
Transfer X to A	TXA	9F	1	2	X→A	●	●	●	●	●
Set Carry Bit	SEC	99	1	1	1→C	●	●	●	●	1
Clear Carry Bit	CLC	98	1	1	0→C	●	●	●	●	0
Set Interrupt Mask Bit	SEI	9B	1	2	1→I	●	1	●	●	●
Clear Interrupt Mask Bit	CLI	9A	1	2	0→I	●	0	●	●	●
Software Interrupt	SWI	83	1	10		●	1	●	●	●
Return from Subroutine	RTS	81	1	5		●	●	●	●	●
Return from Interrupt	RTI	80	1	8		?	?	?	?	?
Reset Stack Pointer	RSP	9C	1	2	$FF→SP	●	●	●	●	●
No-Operation	NOP	9D	1	1	Advance Prog. Cntr. Only	●	●	●	●	●
Decimal Adjust A	DAA	8D	1	2	Converts binary add of BCD charcters into BCD format	●	●	↕	↕	↕*
Stop	STOP	8E	1	4		●	●	●	●	●
Wait	WAIT	8F	1	4		●	●	●	●	●

Symbols: Op = Operation
= Number of bytes
~ = Number of cycles

* Are BCD characters of upper byte 10 or more? (They are not cleared if set in advance.)

Operation Code Map

	Bit Manipulation		Branch	Read/Modify/Write					Control		Register/Memory						
	Test & Branch	Set/ Clear	Rel	DIR	A	X	,X1	,X0	IMP	IMP	IMM	DIR	EXT	,X2	,X1	,X0	
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	← HIGH
0	BRSET0	BSET0	BRA	NEG					RTI*	–	SUB						0
1	BRCLR0	BCLR0	BRN	–					RTS*	–	CMP						1
2	BRSET1	BSET1	BHI	–					–	–	SBC						2
3	BRCLR1	BCLR1	BLS	COM					SWI*	–	CPX						3
4	BRSET2	BSET2	BCC	LSR					–	–	AND						4
5	BRCLR2	BCLR2	BCS	–					–	–	BIT						5
6	BRSET3	BSET3	BNE	ROR					–	–	LDA						6
7	BRCLR3	BCLR3	BEQ	ASR					–	TAX*	–	STA				STA(+1)	7
8	BRSET4	BSET4	BHCC	LSL/ASL					–	CLC	EOR						8
9	BRCLR4	BCLR4	BHCS	ROL					–	SEC	ADC						9
A	BRSET5	BSET5	BPL	DEC					–	CLI*	ORA						A
B	BRCLR5	BCLR5	BMI	–					–	SEI*	ADD						B
C	BRSET6	BSET6	BMC	INC					–	RSP*	–	JMP(–1)					C
D	BRCLR6	BCLR6	BMS	TST(–1)	TST			TST(–1)	DAA*	NOP	BSR*	JSR(+2)		JSR(+1)		JSR(+2)	D
E	BRSET7	BSET7	BIL	–					STOP*	–	LDX						E
F	BRCLR7	BCLR7	BIH	CLR					WAIT*	TXA*	–	STX				STX(+1)	F
	3/5	2/5	2/3	2/5	1/2	1/2	2/6	1/5	1/*	1/1	2/2	2/3	3/4	3/5	2/4	1/3	

LOW

(NOTES) 1. "–" is an undefined operation code.

2. Lowermost numbers in each column represent a byte count and the number of cycles required (byte count/number of cycles).
Number of cycles for asterisked (*) mnemonics is as follows:

RTI	8	TAX	2
RTS	5	RSP	2
SWI	10	TXA	2
DAA	2	BSR	5
STOP	4		
WAIT	4		

3. Parenthesized numbers must be added to the cycle count of the particular instruction.

NOTES

NOTES

NOTES

NOTES

HITACHI AMERICA, LTD.
SEMICONDUCTOR AND IC DIVISION

U.S. SALES OFFICE

Hitachi America, Ltd.
Semiconductor and IC Division
2210 O'Toole Avenue
San Jose, CA 95131
Tel: 408-435-8300
Telex: 17-1581
Twx: 910-338-2103
Fax: 408-435-2748
Fax: 408-435-2749
Fax: 408-435-2782

REGIONAL OFFICES

MID-ATLANTIC REGION

Hitachi America, Ltd.
1700 Galloping Hill Rd.
Kenilworth, NJ 07033
201/245-6400

NORTHEAST REGION

Hitachi America, Ltd.
5 Burlington Woods Drive
Burlington, MA 01803
617/229-2150

SOUTH CENTRAL REGION

Hitachi America, Ltd.
Two Lincoln Centre, Suite 865
5420 LBJ Freeway
Dallas, TX 75240
214/991-4510

NORTH CENTRAL REGION

Hitachi America, Ltd.
500 Park Blvd., Suite 415
Itasca, IL 60143
312/773-4864

NORTHWEST REGION

Hitachi America, Ltd.
2210 O'Toole Avenue
San Jose, CA 95131
408/435-2200

SOUTHWEST REGION

Hitachi America, Ltd.
18300 Von Karman Avenue, Suite 730
Irvine, CA 92715
714/553-8500

SOUTHEAST REGION

Hitachi America, Ltd.
4901 N.W. 17th Way, Suite 302
Fort Lauderdale, FL 33309
305/491-6154

DISTRICT OFFICES

- Hitachi America, Ltd.
  3800 W. 80th Street, Suite 1050
  Bloomington, MN 55431
  612/896-3444

- Hitachi America, Ltd.
  80 Washington St., Suite 302
  Poughkeepsie, NY 12601
  914/485-3400

- Hitachi America, Ltd.
  6 Parklane Blvd., #558
  Dearborn, MI 48126
  313/271-4410

- Hitachi America, Ltd.
  6161 Savoy Dr., Suite 850
  Houston, TX 77036
  713/974-0534

- Hitachi (Canadian) Ltd.
  2625 Queensview Dr.
  Ottawa, Ontario, Canada K2A 3Y4
  613/596-2777

- Hitachi America, Ltd.
  401 Harrison Oaks Blvd., Suite #317
  Cary, NC 27513
  919/481-3908

HITACHI

We make things possible

Hitachi America, Ltd.
Semiconductor and IC Division
2210 O'Toole Avenue, San Jose, CA 95131
1-408-435-8300

7.5M

Printed in U.S.A.