

TMS7000 Family Data Manual

8-bit Microcomputer Family



TEXAS
INSTRUMENTS

TMS7000

Family Data

Manual

8-bit Microcomputer Family

IMPORTANT NOTICE

Texas Instruments reserves the right to make changes at any time in order to improve design and to supply the best product possible.

Texas Instruments assumes no responsibility for infringement of patents or rights of others based on Texas Instruments applications assistance or product specifications, since TI does not possess full access to data concerning the use or applications of customer's products. TI also assumes no responsibility for customer product designs.

Copyright © 1983 by Texas Instruments Incorporated

TABLE OF CONTENTS

SECTION	PAGE
1.	INTRODUCTION
1.1	General 1-1
1.2	Background And Design Philosophy 1-1
1.2.1	Strip Chip Architecture Topology (SCAT) 1-2
1.2.2	Microprogramming 1-3
1.3	Key Features Of The TMS7000 Family 1-4
1.4	Support 1-6
1.4.1	Development Tools 1-6
1.4.2	Hotline Assistance 1-7
1.4.3	Training Support 1-7
1.4.3.1	TDC-700-TMS7000 Family Systems Design 1-8
1.4.3.2	ATS-710-TMS7000 Family Microprogramming 1-8
1.4.4	Design Expertise 1-8
2.	TMS7000 FAMILY ARCHITECTURE
2.1	On-Chip RAM And Registers 2-2
2.1.1	Register File (RF) 2-2
2.1.2	Peripheral File (PF) 2-3
2.1.3	Stack Pointer (SP) 2-3
2.1.4	Status Register (ST) 2-3
2.1.5	Program Counter (PC) 2-4
2.2	On-Chip General Purpose I/O Ports 2-4
2.3	Memory Modes 2-6
2.3.1	Single-Chip Mode. 2-8
2.3.2	Peripheral Expansion Mode 2-13
2.3.3	Full Expansion Mode 2-14
2.3.4	Microprocessor Mode 2-14
2.3.5	System Emulator Mode 2-15
2.4	I/O Control Registers 2-16
2.5	Interrupt and Reset Clock Options 2-19
2.5.1	Interrupt Priority 2-19
2.5.2	Device Initialization 2-20
2.5.3	CPU Interface To Interrupt Logic 2-21
2.5.4	Interrupt Logic 2-22
2.6	Programmable Timer/Event Counters 2-24
2.6.1	Real Time Clock (RTC) 2-27
2.6.2	Event Counter (EC) 2-27
2.6.3	Timer And Prescaled Clock 2-27
2.6.4	Timer Interrupt Pulse 2-28
2.6.5	Timer 2 2-28
2.6.6	Pulse Width Measurement 2-29
2.6.7	Pulse Width Modulation (PWM) Theory Of Operation 2-29
2.6.8	Multi-Interrupt Pulse Width Modulation (PWM). 2-31
2.7	Serial Port (TMS70X1 Versions Only) 2-33
2.7.1	Description 2-33
2.7.2	Clock Sources And Serial Port Modes 2-35
2.7.2.1	Asynchronous Communication Mode 2-35
2.7.2.2	Isosynchronous Communication Mode 2-36
2.7.2.3	Serial I/O Communication Mode 2-37

2.7.3	Multiprocessor Communication	2-37
2.7.3.1	Motorola (MC6801) Protocol	2-38
2.7.3.2	Intel (I8051) Protocol	2-39
2.7.4	Timer 3	2-40
2.7.5	Serial Port Registers	2-42
2.7.5.1	Mode Register (SMODE)	2-42
2.7.5.2	Serial Control 0 Register (SCTLO)	2-44
2.7.5.3	Serial Port Status Register (SSTAT)	2-45
2.7.5.4	Serial Control 1 Register (SCTL1)	2-46
2.7.5.5	Timer 3 Data Register	2-48
2.7.5.6	Receiver Buffer	2-48
2.7.5.7	Transmitter Buffer	2-49
2.7.6	Serial Port Initialization	2-49
2.7.7	Serial Port Interrupt	2-49
2.8	Pin Description	2-50

3. STANDARD INSTRUCTION SET

3.1	Definitions	3-1
3.2	Addressing Modes	3-3
3.2.1	Direct Addressing Modes	3-3
3.2.1.1	Single Register Addressing Mode	3-3
3.2.1.2	Register File Addressing Mode	3-4
3.2.1.3	Peripheral File Addressing Mode	3-5
3.2.1.4	Immediate Addressing Mode	3-6
3.2.1.5	Program Counter Relative Addressing Mode	3-6
3.2.2	Extended Addressing Modes	3-7
3.2.2.1	Direct Memory Addressing	3-7
3.2.2.2	Register File Indirect Addressing Mode	3-7
3.2.2.3	Indexed Addressing Mode	3-8
3.3	Instructions	3-8
3.3.1	Implied Operand Instructions	3-8
3.3.2	Single Operand Instructions	3-9
3.3.3	Dual Operand Instructions	3-10
3.3.3.1	Register File Instruction Types	3-11
3.3.3.2	Peripheral File Instruction Types	3-11
3.3.4	Jump Instructions	3-12
3.3.4.1	Simple Relative Instruction Type	3-13
3.3.4.2	Single Relative Instruction Type	3-13
3.3.4.3	Dual Relative Instruction Type	3-13
3.3.4.4	Peripheral Relative Instruction Type	3-14
3.3.5	Extended Address Instructions	3-14
3.3.6	Miscellaneous Instructions	3-15
3.3.6.1	MOVD Instruction	3-16
3.3.6.2	TRAP Instructions	3-17
3.4	Custom Microcoding	3-17
3.5	Instruction Descriptions	3-20
3.5.1	ADC - Add With Carry	3-21
3.5.2	ADD - Add	3-22
3.5.3	AND - And	3-22
3.5.4	ANDP - And Peripheral Register	3-23
3.5.5	BTJO - Bit Test And Jump If One	3-23
3.5.6	BTJOP - Bit Test And Jump If One Peripheral	3-24
3.5.7	BTJZ - Bit Test And Jump If Zero	3-24

3.5.8	BTJZP - Bit Test And Jump If One Peripheral	3-25
3.5.9	BR - Branch	3-25
3.5.10	CALL - Call	3-26
3.5.11	CLR - Clear	3-26
3.5.12	CLRC - Clear The Carry Bit	3-27
3.5.13	CMP - Compare	3-27
3.5.14	CMPA - Compare Accumulator Extended	3-27
3.5.15	DAC - Decimal Add With Carry	3-28
3.5.16	DEC - Decrement	3-28
3.5.17	DECD - Decrement Double	3-29
3.5.18	DINT - Disable Interrupts	3-29
3.5.19	DJNZ - Decrement Register And Jump If Not Zero	3-30
3.5.20	DSB - Decimal Subtract With Borrow	3-30
3.5.21	EINT - Enable Interrupts	3-31
3.5.22	IDLE - Idle Until Interrupt	3-31
3.5.23	INC - Increment	3-32
3.5.24	INV - Invert	3-32
3.5.25	JMP - Jump Unconditional	3-33
3.5.26	J<cond> - Jump On Condition	3-33
3.5.27	LDA - Load A Register	3-34
3.5.28	LDSP - Load Stack Pointer	3-35
3.5.29	MOV - Move	3-35
3.5.30	MOVD - Move Double	3-36
3.5.31	MOVP - Move To/From Peripheral File	3-36
3.5.32	MPY - Multiply	3-37
3.5.33	NOP - No Operation	3-37
3.5.34	OR - Or	3-38
3.5.35	OPR - Or Peripheral File Register	3-38
3.5.36	POP - Pop From Stack	3-39
3.5.37	PUSH - Push On Stack	3-39
3.5.38	RETI - Return From Interrupt	3-40
3.5.39	RETS - Return From Subroutine	3-40
3.5.40	RL - Rotate Left	3-41
3.5.41	RLC - Rotate Left Through Carry	3-41
3.5.42	RR - Rotate Right	3-42
3.5.43	RRC - Rotate Right Through Carry	3-42
3.5.44	SBB - Subtract With Borrow	3-43
3.5.45	SETC - Set Carry	3-43
3.5.46	STA - Store A Register	3-44
3.5.47	STSP - Store Stack Pointer	3-44
3.5.48	SUB - Subtract	3-45
3.5.49	SWAP - Swap Nibbles	3-45
3.5.50	TRAP - Trap To Subroutine	3-46
3.5.51	TSTA - Test A Register	3-47
3.5.52	TSTB - Test B Register	3-47
3.5.53	XCHB - Exchange with B Register	3-47
3.5.54	XOR - Exclusive Or	3-48
3.5.55	XORP - Exclusive Or Peripheral File	3-48

4.	ELECTRICAL SPECIFICATIONS	
4.1	TMS7000/TMS7020/TMS7040/TMS70120/TMS7001/TMS7041	4-1
4.1.1	Description Of The TMS7000/TMS7020/TMS7040/TMS70120 TMS7001/TMS7041 Devices	4-1

4.1.2	Key Features	4-2
4.1.3	Absolute Maximum Ratings Over Operating Free-Air Temperature Range	4-3
4.1.4	Recommended Operating Conditions	4-3
4.1.5	Electrical Characteristics Over Full Range Of Operating Conditions	4-3
4.1.6	Recommended Crystal/Clockin Operating Conditions Over Full Operating Range	4-4
4.1.7	Memory Interface Timing At 10MHz Over Full Operating Free Air Temperature Range	4-5
4.1.8	Application Of Ceramic Resonator	4-7
4.1.9	Serial Port Timing	4-8
4.1.9.1	Internal Serial Clock	4-8
4.1.9.2	External Serial Clock	4-9
4.1.9.3	RX Signals In Communication Modes	4-10
4.1.9.4	TX Signals In Communication Modes	4-11
4.1.9.5	RX Signals In Serial I/O Mode	4-12
4.1.9.6	TX Signals In Serial I/O Mode	4-13
4.1.10	Pin Descriptions	4-14
4.1.10.1	Pin Descriptions Of The TMS7000/TMS7020/TMS7040 TMS70120	4-14
4.1.10.2	Pin Descriptions Of The TMS7001/TMS7041	4-15
4.2	TMS70C00/TMS70C20/TMS70C40	4-16
4.2.1	Description Of The TMS70C00/TMS70C20/TMS70C40	4-16
4.2.2	Key Features	4-17
4.2.3	Absolute Maximum Rating Over Operating Free-Air Temperature Range	4-18
4.2.4	Recommended Operating Conditions	4-18
4.2.5	Electrical Characteristics Over Full Range of Operating Conditions	4-18
4.2.6	AC Characteristics For Input/Output Ports	4-19
4.2.7	Recommended Crystal/Clockin Operating Conditions Over Full Operating Range	4-19
4.2.8	Memory Interface Timing At VDD = 5V, FOSC = 3MHz Over The Full Operating Free-Air Temperature Range	4-21
4.2.9	Pin Descriptions Of The TMS70C00/TMS70C20/TMS70C40	4-24
4.3	SE70P161	4-25
4.3.1	Description Of The SE70P161 Prototyping Component	4-25
4.3.2	Prototyping	4-25
4.3.2.1	TMS7041 Prototyping	4-25
4.3.2.2	TMS7020/TMS7040/TMS70120 Prototyping	4-25
4.3.3	Programming And Installing Eproms	4-26
4.3.4	Absolute Maximum Ratings Over Operating Free-Air Temperature Range	4-26
4.3.5	Recommended Operating Conditions	4-27
4.3.6	Electrical Characteristics Over Full Range Of Operating Conditions.	4-27
4.3.7	Recommended Crystal/Clockin Operating Conditions Over Full Operating Range.	4-27
4.3.8	Memory Interface Timing At 10MHz Over Full Operating Free-Air Temperature Range	4-28
4.3.9	Pin Description Of The SE70P161	4-30

5. MICROPROGRAMMING

5.1	TMS7000 Custom Microcoding Description	5-1
5.1.1	Typical Applications	5-1

5.1.2	Key Features	5-2
5.1.3	Microcoding Example	5-5
5.1.4	Tradeoffs Of Microcoding	5-5
5.1.5	Microcode Development Cycle	5-6
5.1.6	Available Support	5-8
5.1.6.1	TMS7000 Microassembler Software Package	5-8
5.1.6.2	TMS7000 AMPL Emulator System	5-8
5.1.6.3	TMS7000 Microcode Documentation Package	5-8
5.2	Microcoded Benchmarks	5-9
5.2.1	Benchmark Rules	5-9
5.2.2	Benchmark 1: 16 Bit Binary Addition	5-10
5.2.3	Benchmark 2: 16 Bit Binary Coded Decimal Addition	5-10
5.2.4	Benchmark 3: Block Move	5-11
5.2.5	Benchmark 4: Table Search	5-12
5.2.6	Benchmark 5: Binary To BCD Conversion	5-13
5.2.7	Benchmark 6: Bit I/O	5-14
5.3	Microarchitecture Description	5-15
5.3.1	TMS7000 Family Address Space	5-15
5.3.2	Basic TMS7000 Architecture	5-16
5.3.3	Microinstruction Format	5-18
5.3.3.1	Microinstruction Cycle Timing	5-20
5.3.3.2	Memory Cycle Timing	5-21
5.3.3.3	Short Memory References	5-21
5.3.3.4	Long Memory References	5-23
5.3.3.5	Interrupt Vector Reads	5-24
5.3.3.6	Memory Control Signals	5-25
5.3.4	Organization Of The TMS7000 CPU	5-26
5.3.4.1	P Bus	5-28
5.3.4.2	N Bus	5-28
5.3.4.3	AL Bus	5-29
5.3.4.4	AH Bus	5-29
5.3.4.5	O Bus	5-30
5.3.4.6	MD Bus	5-32
5.3.4.7	ALU Operation	5-33
5.3.4.8	Shifter Operation	5-35
5.3.4.9	IR Register	5-37
5.3.4.10	Status Register	5-38
5.3.4.10.1	(STC) Status Carry Bit	5-39
5.3.4.10.2	STSB - Status Sign Bit	5-39
5.3.4.10.3	STEZ - Status Equal To Zero Bit	5-39
5.3.4.10.4	STINT - Status Interrupt Enable Bit	5-40
5.3.4.11	BCD Constant Register	5-40
5.3.4.12	Other Registers	5-43
5.3.5	Microinstruction Sequence Control Overview	5-44
5.3.5.1	Dispatch Conditions	5-45
5.3.5.1.1	Unconditional Branching - JUNC	5-45
5.3.5.1.2	Function Dispatch - IRL	5-45
5.3.5.1.3	Test Sign Bit - JT7	5-46
5.3.5.1.4	Test If Zero - JUZ	5-47
5.3.5.1.5	Test If Interrupt - INT	5-47
5.3.5.1.6	Group Dispatch - IRH	5-48
5.3.5.1.7	Test If Carry - JC	5-49
5.3.5.1.8	Test Status Register - MJMP	5-50
5.3.6	Reset Operation	5-51

6.	DESIGN AIDS	
6.1	Interfacing The TMS7000 To Peripheral And Memory Devices	6-1
6.1.1	Introduction	6-1
6.1.2	Peripheral Expansion Mode Example	6-4
	6.1.2.1 Read Cycle Timing For The Peripheral Expansion Mode	6-4
	6.1.2.2 Write Cycle Timing For The Peripheral Expansion Mode	6-5
6.1.3	Microprocessor Mode Example	6-7
	6.1.3.1 Read Cycle Timing For The Microprocessor Mode	6-7
	6.1.3.2 Write Cycle Timing For The Microprocessor Mode	6-8
6.1.4	Software Considerations	6-10
6.2	Serial Communication With The TMS7000 Family	6-11
6.2.1	Communication Formats	6-11
6.2.2	Design Constraints For Software And Hardware UART	6-12
	6.2.2.1 Design Of A Software UART For The TMS7040	6-13
	6.2.2.2 Hardware UART (TMS7041)	6-25
	6.2.2.3 RS-232-C Interface	6-34
	6.2.2.4 Other Design Approach	6-36
6.3	Instruction Set Application Notes	6-48
6.3.1	The Status Register	6-48
	6.3.1.1 Compare And Jump Instructions	6-49
	6.3.1.2 Addition And Subtraction Instructions	6-51
	6.3.1.3 Swap And Rotation Instructions	6-54
6.3.2	Stack Operations	6-56
6.3.3	Subroutine Instructions	6-57
6.3.4	Multiply And Shifting	6-58
6.3.5	Branch Instructions	6-61
6.3.6	Interrupts	6-61
7.	DEVELOPMENT SUPPORT TOOLS	
7.1	Introduction	7-1
7.1.1	XDS Concept	7-2
7.1.2	Key Features	7-3
7.2	CrossWare	7-3
7.3	XDS Hardware	7-3
7.3.1	Model 22	7-4
7.3.2	Model 33	7-4
7.3.3	Differences And Similarities - Model 22/Model 33	7-6
7.3.4	XMPL	7-7
7.3.5	Breakpoint And Trace Functions	7-8
7.3.6	Multiprocessing	7-9
7.4	Evaluation Modules	7-9
7.4.1	TMS7000 EVM	7-10
	7.4.1.1 Operating System	7-10
7.5	Prototype Component	7-10
7.5.1	SE70P161 Description	7-11
	7.5.1.1 Prototyping	7-12
	7.5.1.2 TMS7041 Prototyping	7-12
	7.5.1.3 TMS7020/TMS7040/TMS70120 Prototyping	7-12
	7.5.1.4 SE70P161 Electrical Data	7-12
7.6	Physical And Ordering Information	7-12
7.6.1	CrossWare	7-12
7.6.2	XDS Hardware	7-12
	7.6.2.1 Physical Specifications	7-13

7.6.3	Evaluation Modules	7-13
7.6.4	Warranty And Subscription Services	7-13
8.	INDEPENDENT SUPPORT	
8.1	Introduction	8-1
8.2	Processor Innovations - Intel Based Support Tools	8-1
8.2.1	XI Workstation Device Support	8-1
8.2.2	Company To Contact	8-2
8.2.3	Product Offerings	8-3
	8.2.3.1 PIDS 1810-11	8-3
	8.2.3.2 PIDS 1810-12	8-3
	8.2.3.3 PIDS 1810-32	8-3
8.3	Allen Ashley - CP/M Based Support Tools	8-3
8.3.1	Company To Contact	8-4
8.3.2	Product Offerings	8-4
	8.3.2.1 CP/M Based Development Software For TMS7000 Family ..	8-4
8.4	SEQ: Self-Adaptive EEROM	8-4
8.4.1	Company To Contact	8-5
9.	QUALITY AND RELIABILITY	
9.1	Introduction	9-1
9.2	Average Outgoing Quality	9-1
9.3	New Product And Major Change Reliability Qualification Testing	9-2
9.4	Reliability Monitoring	9-2
9.5	TMS7020/TMS7040 Reliability Performance	9-3
10.	GENERAL INFORMATION	
10.1	TMS7000 Family Devices	10-1
10.1.1	Prototype And Production Flow	10-1
10.1.2	Device Prefix Designators	10-3
10.1.3	Clock Options	10-5
10.1.4	Reserved ROM Locations	10-6
10.1.5	Ordering Information	10-6
	10.1.5.1 TMS7000 Family Members With On-Chip ROM	10-6
	10.1.5.2 TMS7000 Family Members Without On-Chip ROM	10-7
10.1.6	Mechanical Data	10-8
10.2	Development Support Tools	10-10
10.2.1	CrossWare	10-10
10.2.2	XDS Hardware	10-10
10.2.3	Evaluation Modules	10-10
10.3	TMS7000-Family Documentation	10-10
10.4	Worldwide Regional Technology Centers (RTC)	10-11

APPENDICES

APPENDIX	PAGE
Appendix A Instruction Execution Times	A-1
Appendix B TMS7000 Bus Activity Chart	B-1
Appendix C TMS7500 Data Encryption Device	C-1
Appendix D References	D-1

LIST OF ILLUSTRATIONS

FIGURE	PAGE
1-1 TMS7020 Microcomputer Bar Plan	1-3
2-1 TMS7000 Internal Architecture	2-1
2-2 Example Of Stack Initialization In the Register File	2-3
2-3 Status Register (ST)	2-3
2-4 Bidirectional I/O Logic	2-5
2-5 I/O Ports: Single-Chip Mode	2-9
2-6 Interrupt Generation: System Emulator Mode	2-15
2-7 IOCNT0 - I/O Control Register 0	2-17
2-8 IOCNT1 - I/O Control Register 1	2-18
2-9 CPU Interface To Interrupt Logic	2-22
2-10 Interrupt Logic	2-23
2-11 Programmable Timer/Event Counter	2-25
2-12 Timers 1 & 2 Data And Control Registers	2-25
2-13 Pulse Width Measurement	2-29
2-14 Pulse Width Modulated Pulse Train	2-29
2-15 TMS7000 PWM INT3 Timing	2-30
2-16 Simultaneous Interrupts, INT2 Preceding	2-31
2-17 Simultaneous Interrupts, INT3 Preceding	2-32
2-18 Serial Port Functional Blocks	2-34
2-19 Serial Port I/O Logic	2-35
2-20 Asynchronous Communication Format	2-36
2-21 Isosynchronous Communication Format	2-36
2-22 Serial I/O Communication Format	2-37
2-23 Double Buffered WUT And TXSHF	2-39
2-24 Motorola Multiprocessor Communication Format	2-39
2-25 Intel Multiprocessor Communication Format	2-40
2-26 TIMER 3 Block Diagram	2-41
2-27 Serial Mode Register - SMODE	2-42
2-28 Serial Control 0 Register - SCTL0	2-44
2-29 Serial Port Status Register - SSTAT	2-45
2-30 Serial Control 1 Register - SCTL1	2-47
2-31 Timer 3 Data Register - T3DATA	2-48
2-32 Receiver Buffer - RXBUF	2-48
2-33 Transmitter Buffer - TXBUF	2-49
2-34 SC, PE, FE, And Microprocessor Pin Assignments	2-52
2-35 System Emulator Mode Pin Assignments	2-54
3-1 Single Register Addressing Mode Object Code	3-4
3-2 Register File Addressing Mode Object Code	3-5
3-3 Peripheral File Addressing Mode Object Code	3-5
3-4 Immediate Addressing Mode Object Code	3-6
3-5 Program Counter Relative Addressing Mode Object Code	3-6
3-6 Direct Memory Addressing Mode Object Code	3-7
3-7 Register File Indirect Addressing Mode Object Code	3-7
3-8 Indexed Addressing Mode Object Code	3-8
3-9 Trap Vector Table	3-17
4-1 Output Loading Circuit For Test	4-3
4-2 Measurement Points For Switching Characteristics	4-4
4-3 Clock Timing	4-4
4-4 Recommended Clock Connections	4-5

4-5	Read and Write Cycle Timing	4-6
4-6	Ceramic Resonator Circuit	4-7
4-7	SC, FE, PE, and Microprocessor Mode Pin Assignments (TMS7000)	4-14
4-8	SC, FE, PE, and Microprocessor Mode Pin Assignments (TMS7001)	4-15
4-9	Output Loading Circuit For Test	4-19
4-10	Clock Timing	4-20
4-11	Measurement Points For Switching Characteristics (VDD = 5V)	4-20
4-12	Read And Write Cycle Timing	4-22
4-13	Recommended Clock Connections	4-23
4-14	SC, FE, PE, and Microprocessor Mode Pin Assignments	4-24
4-15	Read And Write Cycle Timing	4-29
5-1	TMS7000 CPU Internal Block Diagram	5-4
5-2	Assembly Language Multiply Sequence	5-5
5-3	Non-Core Assembly Language Instructions	5-6
5-4	Microcode Development Flowchart	5-7
5-5	TMS7000 Family Address Space	5-16
5-6	TMS7000 Overall Block Diagram	5-17
5-7	Sample Of A Miasm Statement	5-20
5-8	Microinstruction Cycle Phases	5-20
5-9	On-Chip RAM Memory Cycle Timing	5-22
5-10	Long Memory Cycle Timing	5-23
5-11	Interrupt Vector Reads	5-24
5-12	Interrupt Vector References	5-25
5-13	Internal Organization Of The TMS7000 CPU	5-27
5-14	P Bus Sources	5-28
5-15	N Bus Sources	5-28
5-16	AL Bus Sources	5-29
5-17	AH Bus Sources	5-30
5-18	Lowwrite (1-0) Description	5-30
5-19	O Bus Destinations	5-31
5-20	MD Bus Destinations	5-32
5-21	ALU Block Diagram	5-33
5-22	ALU Functions	5-33
5-23	ALU Carry In Values	5-34
5-24	Microcode Example	5-35
5-25	SHIFT/ALU Carry-In Controls	5-36
5-26	Shifter Operation	5-37
5-27	IR Register Formats	5-38
5-28	Status Register	5-38
5-29	ST Register Sources	5-39
5-30	BCD Correction Constant Generation	5-41
5-31	BCD Arithmetic Operation Timing	5-42
5-32	MICASM Statement	5-43
5-33	Microinstruction Dispatch Example	5-44
5-34	Next MICRO Address Generation	5-45
5-35	IRL Dispatch	5-46
5-36	JT7 Dispatch	5-46
5-37	JUZ Dispatch	5-47
5-38	INT Dispatch	5-47
5-39	TMS7000 Group Numbers	5-48
5-40	IRH Dispatch	5-49
5-41	JC DISPATCH	5-50
5-42	Macro Jump Conditions	5-50

5-43	MJMP Dispatch	5-51
6-1	TMS70XX Read And Write Cycle Timing	6-3
6-2	Peripheral Expansion Mode Example	6-6
6-3	Memory Address Decode	6-7
6-4	Microprocessor Mode Example	6-9
6-5	Asynchronous Communication Format	6-11
6-6	I/O Interface	6-12
6-7	SWXMIT Routine Flowchart	6-17
6-8	SWCRVD Routine Flowchart	6-20
6-9	Delay Constants Calculation	6-21
6-10	Start Bit Detection	6-22
6-11	Interrupt 4 Service Routine	6-31
6-12	HWXMIT Routine Flowchart	6-32
6-13	HWRCVD Routine Flowchart	6-33
6-14	Status Register	6-48
6-15	Unsigned System With 8 Bits Of Magnitude: 0-255	6-53
6-16	Signed System With 7 Bits Of Magnitude: -127 TO + 127	6-53
6-17	SWAP And Rotation Operations	6-55
6-18	Example Of A Dispatch Table	6-56
6-19	Example Of A Subroutine Call	6-58
6-20	Example Of A 16-Bit By 16-Bit Multiplication Subroutine	6-60
7-1	Typical Microprocessor Development System	7-1
7-2	Typical XDS Configuration	7-2
7-3	The XDS Model 22	7-5
7-4	Memory Configuration	7-7
7-5	Levels Of XMPL Interface	7-8
7-6	The RTC/EVM 7000 Evaluation Module	7-9
10-1	Prototype And Production Flow	10-1
10-2	Development Flowchart	10-4
10-3	TI Standard Symbolization	10-7
10-4	TI Standard Symbolization With Customer Part Number	10-7
10-5	TI Standard Device Symbolization Without On-Chip ROM	10-8
10-6	40 Pin Plastic Package, 100 Mil Pin Spacing (N Package)	10-8
10-7	40 Pin Plastic Package, 70 Mil Pin Spacing (NS Package)	10-9
10-8	40 Pin Plastic Package, 70 Mil Pin Spacing (JD Package)	10-9

LIST OF TABLES

TABLE		PAGE
1-1	TMS7000 Family Members	1-5
1-2	TMS7000 Standard NMOS Product Family: TMS7000, TMS7020, TMS7040, TMS70120, TMS7001, TMS7041	1-5
1-3	TMS7000 Standard CMOS Product Family: TMS70C00, TMS70C20, TMS70C40	1-6
2-1	TMS7000 Family Summary	2-2
2-2	Mode-Select Conditions	2-6
2-3	70X0 Memory Map	2-7
2-4	70X1 Memory Map	2-7
2-5	TMS70X0 Peripheral Memory Map	2-10
2-6	TMS70X1 Peripheral Memory Map	2-11
2-7	Reset And Interrupt Vector Locations In ROM	2-22

2-8	Serial Port Control Registers	2-33
2-9	SC, PE, FE, AND Microprocessor Pin Assignments	2-51
2-10	System Emulator Mode Pin Assignments	2-53
3-1	TMS7000 Symbol Definitions	3-2
3-2	TMS7000 Addressing Modes	3-3
3-3	Implied Operand Instructions	3-9
3-4	Machine Instruction Format: Implied Operand Instructions	3-9
3-5	Single Operand Instructions	3-9
3-6	Machine Instruction Formats: Single Operand instructions	3-10
3-7	Dual Operand Instructions	3-10
3-8	Machine instruction Formats: Register File	3-11
3-9	Machine Instruction Formats: Peripheral File Instructions	3-12
3-10	Jump Instructions	3-12
3-11	Machine Instruction Formats: Simple Relative Instructions	3-13
3-12	Machine Instruction Formats: Single Relative Instructions	3-13
3-13	Machine Instruction Formats: Dual Relative Instructions	3-14
3-14	Machine Instruction Formats: Peripheral Relative Instruction	3-14
3-15	Extended Address Instructions	3-14
3-16	Machine Instruction Formats: Extended Address Instructions	3-15
3-17	Machine Instruction Formats: Miscellaneous Instructions	3-15
3-18	Machine Instruction Formats: MOVD Instruction	3-16
3-19	TMS7000 Core (Reserved) Instructions	3-18
3-20	TMS7000 Non-Core (Available For Microcode) Instructions	3-20
3-21	Conditional Jump Instructions	3-34
4-1	TMS70X0 and TMS70X1 Family Features	4-1
4-2	TMS70CX0 Family Features	4-16
4-3	Eprom Use	4-26
5-1	Benchmark 1-6 Comparison (2.5 MHz)	5-9
5-2	Microinstruction Word Format	5-19
5-3	Memory Controls	5-26
6-1	Timing Data For Sample Circuits	6-1
6-2	I/O Pin Assignments	6-13
6-3	Cycle Calculation	6-21
6-4	Half Bit Cycles Calculation	6-22
6-5	Crystal-Dependent Constants For The Software UART	6-23
6-6	Serial Port Registers	6-26
6-7	P And L Values In Hex	6-34
6-8	Classification Of Instructions According To Status Bits Affected	6-48
6-9	Compare Instructions Examples: Status Bit Values	6-49
6-10	Status Bit Values For Conditional Jump Instructions	6-50
6-11	Add And Subtract Instructions	6-51
6-12	Multi-Bit Right Or Left Shifts By Immediate Multiply	6-59
7-1	Hardware Configuration Difference Model 22 To Model 33	7-6
7-2	EPROM Use	7-11
9-1	Dynamic Life Test	9-3
9-2	Environmental Tests	9-4
10-1	Valid ROM Start Addresses	10-5



1. INTRODUCTION

1.1 GENERAL

This section of the manual introduces the TMS7000 family of single-chip microcomputers and presents the underlying design philosophy and information on family support tools and assistance.

The use of TMS7000 refers to all family members (TMS7000, TMS7020, TMS7040, TMS70120, TMS7001, TMS7041, TMS70C00, TMS70C20, TMS70C40, SE70P161, and all future family members) unless otherwise stated.

Sections 2 through 4 present in detail the TMS7000 family architecture, instruction set, and electrical specifications. These sections present the specifics required by the user to implement a TMS7000 solution in his application. Application examples for hardware interface and software algorithms are presented in Section 6 after the reader has acquired a thorough understanding of the standard instruction set.

Development support tools are an extremely important aspect of microcomputer selection and algorithm development, Sections 7 and 8 present the support tools and several development scenarios for the TMS7000 family.

The enormous technological advances in integrated circuits have enabled semiconductor manufacturers to offer single-chip microcomputers incorporating a central processing unit (CPU), read only memory (ROM), random access memory (RAM) and input/output (I/O) all on a single silicon chip. Texas Instruments' TMS1000 family was the original 4-bit microcomputer entry. The TMS1000 family's price, performance, and reliability have made it the industry leader in a broad range of applications including timers, electronic toys and games, appliance controls, vending machines, temperature controllers, automotive instruments, test instruments, and a variety of other controller applications. The TMS9900 family, the industry's first 16-bit microcomputer, continues to stand in the fore-front of single-chip microcomputer products. Recent TMS9900 family introductions include the TMS9995 and TMS99000 which expand the families use to very high performance applications. It was a logical progression then, for Texas Instruments to introduce the first fully programmable 8-bit microcomputer, the TMS7000 family.

The TMS7000 family capitalized on Texas Instruments' experience and leadership position in the microcomputer market, thereby introducing a true second generation 8-bit Microcomputer family. The second generation design approach is evident by the powerful instruction set, addressing modes, and I/O flexibility all centered around the basic register to register architecture. Flexibility, in hardware and software, was a basic design goal, therefore the TMS7000 family consists of a variety of RAM and ROM sizes, I/O functions, and instruction set definitions, in both NMOS and CMOS, to efficiently address the user's application requirements.

1.2 BACKGROUND AND DESIGN PHILOSOPHY

Originating from extremely low cost calculator-chip designs, early microcomputers necessarily implemented extremely simple CPU's, resulting in primitive instruction sets that made the simplest programming tasks at best difficult and at worst impossible. This seed of primitive, hard to program instruction sets continues today in many microcomputers.

The reason for this trend lies in economics, not engineering. Microcomputers are typically used in extremely high volume applications. The recurring costs of the system, i.e., the price of the

device far outweigh the one-time cost of the program development. So the emphasis is on building the least expensive device containing the most functionality. It is an established economic fact of VLSI design that the larger the silicon bar, the more expensive the device. Therefore a basic question in the design philosophy of microcomputers centers around the trade-off in bar size between the CPU complexity (which determines the power of the instruction set) and the amount of program memory, ROM and RAM.

The CPU, which implements the instruction set, is typically made up of: an accumulator and other registers, an arithmetic logic unit (ALU), a control programmed logic array (PLA), and a large number of data buses and control lines interconnecting the three. Traditional microcomputers built with PLA's and random logic implement the simplest possible CPU, to minimize bar area, resulting in instructions which may be simple to implement in the design of the bar, but extremely difficult to program. In these traditional microcomputers, the trade-off in maintaining minimum bar area through implementation of a simple CPU, is at the expense of larger ROM and RAM requirements to implement the user's algorithm with the resulting primitive instruction set. One example of this is the restriction among many first generation microcomputers limiting jumps to within the same page of ROM.

It is fact that the larger the bar, the more expensive the device, however, this does not imply that a more powerful CPU cannot be implemented on a single microcomputer chip without increasing the bar size and cost of the device. The issue lies in the traditional design and layout of microcomputers. Two significant design innovations have allowed the TMS7000 family to provide true second generation capabilities and still maintain an extremely small bar and low cost. These innovations in microcomputer design philosophy are:

- Strip Chip Architecture Topology (SCAT)
- Microprogramming

1.2.1 Strip Chip Architecture Topology (SCAT)

SCAT is Texas Instruments' term for the design philosophy which incorporates the non-memory elements of a microcomputer architecture (the registers, ALU and control logic) in a strip of vertical blocks in the logic design. Figure 1-1 shows the overall layout of the TMS7020, the 2K ROM version of the TMS7000 family. The row of blocks labeled "timer", "I/O control", etc., is called the "strip", and all of the logic is implemented in the early mask steps of the silicon bar itself. Most of the interconnection between the blocks (in the form of data and address buses) is implemented on a layer of metal over the silicon. As a result, valuable bar area is not wasted in providing the interconnect of the logic elements. This is the essence of SCAT, designing the structures of the entire bar before logic design begins, so that logic element and interconnect space requirements are minimized, thereby reducing the cost of the chip.

The modularity of SCAT inherently enable existing TMS7000 designs to be easily modified and additional features implemented to create new members of the TMS7000 family customized to the user's needs. The indirect benefit which SCAT offers to the user is a full featured product family with various ROM, RAM, and I/O configurations, as well as greatly reduced design cycle time and minimum bar size of all subsequent family additions. Examples of SCAT designs are the doubling of the ROM from 2K bytes (TMS7020) to 4K bytes (TMS7040), and the addition of the UART function to the 4K byte member (TMS7041).

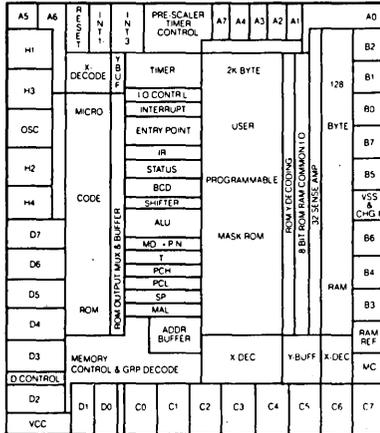


FIGURE 1-1 — TMS7020 MICROCOMPUTER BAR PLAN

1.2.2 Microprogramming

Another important feature of the TMS7000 family is the Microcode Control Rom (CROM) and the internal control of the TMS7000 by microprogramming. Most other microcomputers implement their internal control by a programmed logic array (PLA). Each instruction execution is divided into a number of "states" and on each state the PLA outputs both the current control signals and the next state number. The PLA is a very compact logic structure, but it still leaves the problem of routing the relevant control signals throughout the rest of the bar for decoding and control.

With a microprogramming Control ROM, all of the necessary control signals are contained in a single microinstruction. The outputs of the microcode CROM are made available lengthwise down the microcode CROM. Like any other ROM, each microinstruction has its own address, and when it is read, it immediately supplies the control signals horizontally across to the strip, right where they are needed. No complex routing or combinational logic is required. The block of logic called "entry point" in the strip calculates the next address to feed to the microcode CROM, and the "micro state" is entered. Because a ROM is more compact than a PLA, more control transistors can be built in the form of a microcode CROM than in a PLA, therefore a more powerful TMS7000 family standard instruction set was implemented in the microcode CROM than in an equivalent bar area for a PLA and control decode approach. The benefits to the user of the microcode CROM with the standard instruction set are smaller bar size, thereby reducing the cost of the device, and the implementation of a more powerful instruction set since all CPU control is provided directly by the microcode CROM.

Another direct benefit for the user is the ability of the TMS7000 family microcode CROM to be re-microprogrammed by the user, modifying the standard instruction set to optimize the TMS7000 in the user's application. A user defined instruction set provides the advantages of faster throughput, more efficient utilization of program ROM memory, and improved system security through unique software algorithms. The ability to re-microcode the TMS7000 family also provides an alternate solution for designs initially using the standard instruction set, but encountering a critical timing loop or macro code ROM space limitations, thereby avoiding system redesign through re-microprogramming of the TMS7000.

Microcoding of the TMS7000 family can be performed by the user, an independent consultant, the Regional Technology Center (RTC), or the factory. Full support in the form of documentation, microassemblers, and in-circuit emulators are described in Section 5.

1.3 KEY FEATURES OF THE TMS7000 FAMILY

- Microprogrammable instruction set
- Strip Chip Architecture Topology (SCAT) for rapid family expansion
- Register-to-register architecture
- Family members with 2K, 4K, and 12K bytes of on-chip ROM and ROMless versions
- On-chip 8-bit timer/event counter with:
 - Programmable 5-bit prescale
 - Internal interrupt with automatic reloading
 - Capture latch
- 128-byte RAM register file
- Full-feature data/program stack
- 32 Individual I/O pins:
 - 16 bi-directional pins
 - 8 output pins
 - 8 high-impedance input pins
 - Memory-mapped ports for easy addressing
- 256-byte peripheral file
- Memory expansion capability:
 - 64K byte address space
- 8-bit instruction word
- Eight powerful addressing formats including:
 - Register-to-register arithmetic
 - Indirect addressing on any register pair
 - Indexed and indirect branches and calls
- Two's complement arithmetic
- Single-instruction binary coded decimal (BCD) add and subtract
- Two external maskable interrupts
- Flexible interrupt handling:
 - Priority servicing of simultaneous interrupts
 - Software execution of hardware interrupts
 - Precise timing of interrupts with the capture latch
 - Software monitoring of interrupt status
- Accurate pulse width measurement and modulation
- Silicon gate NMOS and CMOS, 5-volt power supply
- 40-pin, 600 mil, dual in-line package
- 100-mil or 70-mil pin-to-pin spacing packages

Tables 1-1, 1-2, and 1-3 present the features and benefits of the TMS 7000 family in addressing the user's application requirements.

TABLE 1-1 – TMS7000 FAMILY MEMBERS

FEATURES		7000	7020	7040	70120	7001	7041	70P161	70C00	70C20	70C40
ON-CHIP ROM (BYTES)		NONE	2K	4K	12K	NONE	4K	16K EPROM PIGGY BACK	NONE	2K	4K
ON-CHIP RAM (BYTES)		128	128	128	128	128	128	128	128	128	128
INTERRUPT LEVELS		4	4	4	4	6	6	6	4	4	4
TIMERS	13-BIT	1	1	1	1	2	2	2	1	1	1
	10-BIT	0	0	0	0	1	1	1	0	0	0
I/O LINES: BI-DIRECTIONAL INPUT ONLY		16 8	16 8	16 8	16 8	22 2	22 2	22 2	16 8	16 8	16 8
OUTPUT ONLY		8	8	8	8	8	8	8	8	8	8
ADDITIONAL I/O		-	-	-	-	SERIAL PORT	SERIAL PORT	SERIAL PORT	-	-	-
PROCESS TECHNOLOGY		NMOS	NMOS	NMOS	NMOS	NMOS	NMOS	NMOS	CMOS	CMOS	CMOS

TABLE 1-2 – TMS7000 STANDARD NMOS PRODUCT FAMILY TMS7000, TMS7020, TMS7040, TMS70120, TMS7001, TMS7041

CUSTOMER NEED	FEATURES	BENEFITS
<ul style="list-style-type: none"> SATISFY COMPLEX APPLICATIONS 	<ul style="list-style-type: none"> SECOND GENERATION 8 BIT MICROCOMPUTER 	<ul style="list-style-type: none"> ADDRESSES HIGH PERFORMANCE PRODUCTS
<ul style="list-style-type: none"> PRODUCT UPGRADE WITH NO SOFTWARE REDESIGN 	<ul style="list-style-type: none"> WIDE SPECTRUM OF FAMILY MEMBERS NOW. ALL FUTURE MEMBERS SOFTWARE COMPATIBLE 	<ul style="list-style-type: none"> INCREASE CAPABILITY WITHOUT HARDWARE CHANGE; BUILDS ON PRIOR SOFTWARE EFFORTS
<ul style="list-style-type: none"> LARGE MEMORY FOR DATA, HIGH-LEVEL LANGUAGES, VOCABULARIES 	<ul style="list-style-type: none"> UP TO 12K BYTES OF ON CHIP ROM 	<ul style="list-style-type: none"> 3 DEVICES FOR THE PRICE OF 1.5.
<ul style="list-style-type: none"> FEWER EXTERNAL I/O CHIPS 	<ul style="list-style-type: none"> 16 I/O PINS (INDIVIDUALLY DIRECTION PROGRAMMABLE), 8 INPUT ONLY (I/O ON 7001/7041), 8 OUTPUT ONLY 	<ul style="list-style-type: none"> MINIMUM SYSTEM COST THROUGH FLEXIBLE I/O STRUCTURE
<ul style="list-style-type: none"> HIGH THROUGHPUT AND CODE DENSITY, MINIMUM PROGRAMMING TIME 	<ul style="list-style-type: none"> 8X8 MULTIPLY, BCD/ BINARY ADD/SUBTRACT, SINGLE AND DOUBLE PRECISION, S/W TRAPS, I/O INSTRUCTIONS 9 ADDRESSING MODES 	<ul style="list-style-type: none"> FLEXIBLE AND EASY TO USE INSTRUCTION SET
<ul style="list-style-type: none"> REAL-TIME CONTROL 	<ul style="list-style-type: none"> ON CHIP TIMERS 	<ul style="list-style-type: none"> ELIMINATES EXTERNAL PARTS
<ul style="list-style-type: none"> COMMUNICATIONS LINK 	<ul style="list-style-type: none"> ON-CHIP UART (SERIAL PORT ON 7001/7041) 	<ul style="list-style-type: none"> ELIMINATES NEED FOR EXTERNAL UART PARTS; LOWER SYSTEM COST

TABLE 1-3 — TMS7000 STANDARD CMOS PRODUCT FAMILY TMS70C00, TMS70C20, TMS70C40*

CUSTOMER NEED	FEATURES	BENEFITS
<ul style="list-style-type: none"> BATTERY POWER OPERATION 	<ul style="list-style-type: none"> CMOS TECHNOLOGY, 6 MA TYPICAL SUPPLY CURRENT 	<ul style="list-style-type: none"> USEABLE IN PORTABLE APPLICATIONS, LOW COST POWER SUPPLY OR BATTERY
<ul style="list-style-type: none"> LESS POWER CONSUMPTION DURING STANDBY 	<ul style="list-style-type: none"> WAKE-UP MODE = 500 UA, HALT MODE = 250 UA 	<ul style="list-style-type: none"> BATTERY LONGEVITY
<ul style="list-style-type: none"> INEXPENSIVE POWER SUPPLY, OPERATES ON LOW BATTERIES 	<ul style="list-style-type: none"> 3V - 6V POWER REQUIREMENT 	<ul style="list-style-type: none"> TOLERANT POWER SUPPLY VOLTAGE
<ul style="list-style-type: none"> OPERATION IN AN ELECTRICALLY NOISY ENVIRONMENT 	<ul style="list-style-type: none"> INCREASED NOISE MARGIN WITH CMOS INPUTS 	<ul style="list-style-type: none"> GREATER IMMUNITY TO ELECTRICAL NOISE
<ul style="list-style-type: none"> ADDED SYSTEM FUNCTIONS WITH EXISTING POWER SUPPLY 	<ul style="list-style-type: none"> LOWER OPERATION POWER 	<ul style="list-style-type: none"> EXTENDED PRODUCT LIFE

* CMOS FEATURES INCLUDE MICROPROGRAMMABILITY, SCAT AND S/W COMPATIBILITY WITH NMOS VERSIONS

1.4 SUPPORT

TI offers extended development support that consists of the following facets:

- Development Tools
- Hotline Assistance
- Training Support
- Application Expertise

1.4.1 Development Tools

A microcomputer product, being complex and mask ROM programmed, must be supported by high level development tools to facilitate ease of application development and verification, and increase development productivity. The TMS7000 family of 8-bit microcomputers has available a complete spectrum of development tools from single board systems to full scale development systems. Each provides in-circuit emulation, with various levels of development and debug capability. The XDS (Extended Development Support) concept provides host independent in-circuit emulation and debug. When coupled with the transportable crossware (cross support software package), which operates on the system already familiar to the user, the TMS7000 family will provide the user with a cost effective approach to full scale microcomputer development. The AMPL-7000 Development System provides for standard macrocode development and emulation as well as microcode support for those applications utilizing this capability. The single board evaluation module (EVM) has been developed for evaluation and basic in-circuit emulation of the TMS7000 family in an extremely cost effective manner. The SE70P161 prototyping component is provided to support form factor emulation in the user's application. In addition to these development tools to support system

development through in-circuit emulation, the TMS7000 family is supported by software development tools through several third party independent sources. This wide range of development tools provides the user with options to select the appropriate level of support required for his application development. Development support tools and independent support are described in Sections 7 and 8.

1.4.2 Hotline Assistance

Customers may call into one of the worldwide Regional Technology Centers (RTC) for assistance on TMS7000 family development. Whether it be an elaboration of the basic instruction set or a question regarding the microcomputer architecture, the RTC's have the expertise and tools to provide the answer. Please consult the following list and contact the closest RTC if assistance is needed.

Atlanta
Texas Instruments, Inc.
3300 N.E. Expressway
Building 8
Atlanta, GA 30341
(404) 452-4682
(404) 452-4686 Hotline

Boston
Texas Instruments, Inc.
400-2 Totten Pond Rd.
Waltham, MA 02154
(617) 890-6671
(617) 890-4271 Hotline

Chicago
Texas Instruments, Inc.
515 W. Algonquin Rd.
Arlington Heights, IL
(312) 640-2909
(312) 228-6008 Hotline

Northern California
Texas Instruments, Inc.
5353 Betsy Ross Drive
Santa Clara, CA 95054
(408) 748-2220
(408) 980-0305 Hotline

Southern California
Texas Instruments, Inc.
17981 Cartwright Rd.
Irvine, CA 92714
(714) 660-8140
(714) 660-8164 Hotline

Dallas
Texas Instruments, Inc.
10001 E. Campbell Road
Richardson, TX 75081
(214) 680-5066
(214) 680-5096 Hotline

Bedford, England
Texas Instruments, LTD
Manton Lane
Bedford, MK41 7PA
0234 223000

Freising, West Germany
Texas Instruments Deutschland GmbH
Haggertystr. 1
8050 Freising
08161 800

Tokyo, Japan
Texas Instruments Japan
Aoyama Fuji Bldg.
6-12, Kita Aoyama 3 Chome
03-498-2111

Hannover, West Germany
Texas Instruments Deutschland GmbH
Kirchhorsterstr Str 2
3000 Hannover 51
0511/648021

1.4.3 Training Support

The Regional Technology Centers (RTC's) offer courses for the benefit of customers requiring engineering details on Texas Instruments' parts for design or evaluation purposes. Information (description, schedules, entry instructions) regarding any of the RTC seminars may be obtained by contacting the local RTC.

All courses are offered on a regularly scheduled basis in the RTC, but can also be presented at the customer's location when more than four to five students request training.

Two courses are offered for the TMS7000 family of Microcomputers:

- TDC-700-TMS7000 Family System Design
- ATS-710-TMS7000 Family Microprogramming

1.4.3.1 *TDC-700-TMS7000 Family Systems Design*

The TMS7000 family Systems Design course is an introduction to the TMS7000 family of single-chip microprocessors. Leading off with a description of the chip architecture, the course gives an understanding of instruction set usage in example situations. The labs give hands-on experience with the TMS7000 and its development systems. Experience in assembly language programming and microprocessor/microcomputer hardware design is a prerequisite.

1.4.3.2 *ATS-710-TMS7000 Family Microprogramming*

The TMS7000 family Microprogramming course is intended for engineers who need to customize the standard microcoded instruction set to better suit their needs. It starts with an introduction to microprogramming in general and leads into the specifics of microprogramming the TMS7000 family.

Through examples, students learn the operation of the standard instruction set and how to customize it to efficiently implement new instructions through microcoding. Testing considerations are discussed and hands-on lab sessions allow the student to gain experience in the use of development software. Experience in assembly language programming and a basic familiarity with the TMS7000 instruction set and architecture is a prerequisite.

1.4.4 **Design Expertise**

Texas Instruments can provide in-depth technical design assistance through consultations with contract design services. This assistance can take many forms that encompass the application hints in this document to the application groups in the factory and the design assistance teams in the RTC. Contact your local Field Sales Engineer for current information.

2. TMS7000 FAMILY ARCHITECTURE

Throughout this manual the term TMS7000 family or TMS7000 will include all of the members of the group. The term 70X1 refers to those devices containing a serial port (7001, 7041 and 70P161). The term 70X0 refers to those devices which do not contain a serial port (7000, 7020, 7040, 70120, 70C00, 70C20, 70C40). The major components of the TMS7000 family internal architecture are shown in Figure 2-1. For a more detailed description consult the TMS7000 FAMILY MICROARCHITECTURE USER'S GUIDE (MP061). The main features of the TMS7000 family devices are summarized in Table 2-1.

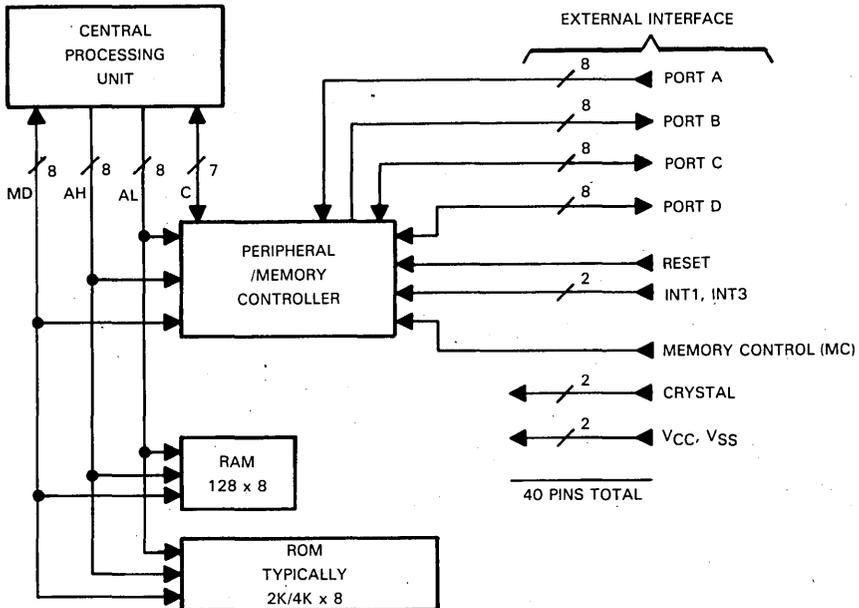


FIGURE 2-1 — TMS7000 INTERNAL ARCHITECTURE

TABLE 2-1 — TMS7000 FAMILY SUMMARY

GROUP	70X0							70X1		
	7000 7020 7040 70120				-70CX0-			7001	7041	70P161
DEVICE	7000	7020	7040	70120	70C00	70C20	70C40	7001	7041	70P161
ROM	0	2K	4K	12K	0	2K	4K	0	4K	16K External EPROM
RAM	128				128			128		
TYPE	NMOS				CMOS			NMOS		
TIMERS	1				1			3		
INT. CLOCK	2.5 MHz				1.75 MHz			2.5 MHz		
INTERRUPTS	3 + RESET				3 + RESET			5 + RESET		
INT TYPE	3 LATCHED and LEVEL				1 LATCHED 2 LATCHED and LEVEL			3 LATCHED and LEVEL		
SERIAL PORT	NO				NO			YES		
GENERAL PURPOSE										
INPUT PINS	8				8			2		
GENERAL PURPOSE										
OUTPUT PINS	8				8			8		
GENERAL PURPOSE										
I/O PINS	16				16			22		
CLOCK OPTNS	/2,/4	/2,/4	/2,/4	/2 only	/2 only			/2,/4		
VOLTAGE	5V				3V-6V			5V		
OTHER					LOW POWER HALT MODE WAKE-UP MODE			ASYNCH AND SYNCHR SERIAL PORT, MULTI- PROCESSOR COMMUN. CASCADEABLE TIMERS		

2.1 ON-CHIP RAM AND REGISTERS

The TMS7000 family has a maximum memory address space of 64K bytes. On-chip and Off-chip memory address spaces vary according to the particular TMS7000 family member used (see Tables 2-3 or 2-4) and the operating mode selected (see Section 2.3). In the sections that follow, the Register File (RF) and the Peripheral File (PF) are described along with three important registers in the CPU: the Stack Pointer (SP), the Status Register (ST), and the Program Counter (PC).

2.1.1 Register File (RF)

The 128-byte on-chip RAM resides in locations >0000 to >007F ('>' means hex) of the TMS7000's address space and is called the Register File (RF). The RAM is treated as registers by much of the instruction set and is numbered R0 - R127. The first two registers, R0 and R1, are also called the A and B registers respectively. Several instructions specify A or B as either the source or destination register, e.g., STSP stores the contents of the Stack Pointer (SP) in the B register. Except where stated otherwise, any register in the Register File can be addressed as an 8-bit source or destination register.

The stack is also located in the Register File. Refer to Section 2.1.3 for information regarding the initialization of the Stack Pointer (SP) and stack definition in the Register File.

2.1.2 Peripheral File (PF)

The Peripheral File (PF) resides in locations >0100 to >01FF of the TMS7000's address space. Some of the TMS7000 instructions are optimized for efficient access to and from registers that reside in the peripheral file. Peripheral File locations are numbered P0 - P255. The PF registers are used for memory expansion, interrupt control, parallel I/O ports, timer control, and serial port control (if available).

2.1.3 Stack Pointer (SP)

The Stack Pointer (SP) is an 8-bit register in the CPU that is typically used to hold a pointer in RAM (the Register File). However, the SP can also be used as temporary data storage if a stack is not implemented, or if the SP contents are not needed. When a stack is implemented, the SP points to the last or top entry on the stack. The SP is automatically incremented just before data is pushed onto the stack and automatically decremented immediately after data is popped from the stack. Upon assertion of the RESET function (see Section 2.5), >01 is loaded into the SP. The size of the stack can be changed from the 126-level stack at RESET to a smaller stack by executing a stack initialization program as illustrated in Figure 2-2. This feature allows the stack to be located anywhere in the Register File. The SP is initialized through the B register (R1).

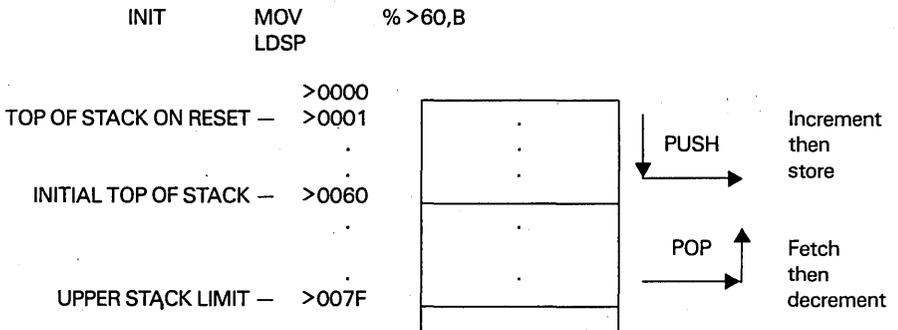
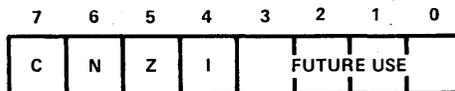


FIGURE 2-2 — EXAMPLE OF STACK INITIALIZATION IN THE REGISTER FILE

2.1.4 Status Register (ST)

The Status Register (ST) is an 8-bit register in the CPU that contains three conditional status bits; Carry (C), Sign (N), Zero (Z), and a global Interrupt Enable bit (I) as shown in Figure 2-3.



- C — CARRY OUT
- N — SIGN
- Z — ZERO
- I — INTERRUPT ENABLE

FIGURE 2-3 — STATUS REGISTER (ST)

The C, N, and Z bits are used mostly for arithmetic operations, bit rotating, and conditional branching. The Carry (C) bit is used as the carry-in and the carry-out for most of the rotate and arithmetic instructions. The Sign (N) bit contains the most significant bit of the destination operand contents after instruction execution. The Zero (Z) bit contains a one when all bits of the destination operand are equal to zero after instruction execution. The C, N, and Z status bits also have jump-on-condition instructions associated with them. The global Interrupt Enable (I) bit must be set to one by the EINT instruction in order for any of the individual interrupts (INTn) to be recognized by the CPU. The Interrupt Enable (I) bit can be cleared by the DINT instruction or by executing a device RESET (see Section 2.5.2). A detailed description of the condition of these bits for each instruction is described in the TMS7000 ASSEMBLY LANGUAGE PROGRAMMER'S GUIDE (MP916).

2.1.5 Program Counter (PC)

The TMS7000's 16-bit Program Counter (PC) consists of two 8-bit registers in the CPU which contain the MSB and the LSB respectively of a 16-bit address: the Program Counter High (PCH) and Low (PCL). The PC acts as the 16-bit address pointer of the opcodes and operands in memory of the currently executing instruction. Upon assertion of the RESET function, the MSB and the LSB of the PC are loaded into the A and B registers of the Register File (see Section 2.5.2).

2.2 ON-CHIP GENERAL PURPOSE I/O PORTS

The TMS7000 family members have 32 I/O pins organized as four 8-bit parallel ports labelled Ports A, B, C, D. Each port is mapped into an 8-bit data value register in the Peripheral File (PF) depending upon the memory mode configuration of the device. The data value registers are usually called APORT, BPORT, CPORT, and DPORT in a program. Ports C and D are implemented as bidirectional I/O ports on all TMS7000 family devices. In addition, Port A is also partially implemented as a bidirectional port on all TMS70X1 devices. Each bidirectional 8-bit port has a corresponding 8-bit Data Direction Register (DDR) that programs each I/O pin as an input or output. A bit set to one in the DDR will cause the corresponding pin to be an output, while a zero in the DDR will cause the pin to be a high impedance input. Upon RESET, the DDR flip-flop registers are set to zero by the on-chip circuitry, forcing them to become inputs. Likewise, the output DATA flip-flop registers are set to one by on-chip circuitry upon RESET. After RESET, if '1's are written to the DDR register sometime before the output data register is changed then the corresponding I/O pins will output a '1'. For this reason, it is good practice that Ports A, C and D output data registers be loaded with the desired value before any bits are configured as outputs. The logic for each bidirectional I/O line is shown in Figure 2-4.

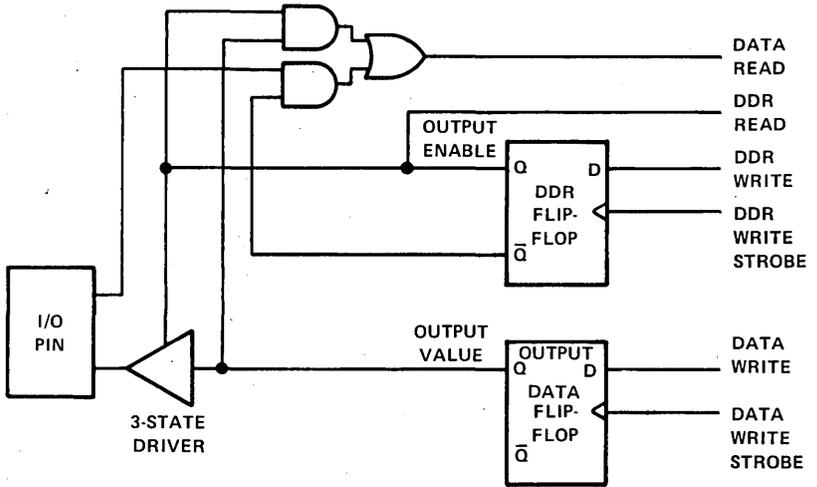


FIGURE 2-4 – BIDIRECTIONAL I/O LOGIC

If a port is bidirectional, i.e., if it is Port C or D on the 70X0 devices, or Port A, C, or D on the 70X1 parts then a single pin in the port may be used for both input and output by modifying its Data Direction Register bit. As shown in Figure 2-4, the output value in the DATA flip-flop register is not changed when the DDR flip-flop bit is switched into the input mode.

The characteristics of the four Ports A, B, C, D can be summarized as follows:

- Port A:** On the 70X0 parts, Port A is an 8-bit high impedance input only port, providing eight general-purpose input lines. Pin A7 may also be used to clock the on-chip timer/event counter. On the 70X1 parts, bits 0-4 and bit 7 of Port A are bidirectional I/O lines. Port A pins A5 and A6 are input only pins that also have other functions when using the serial port. Pin A5 is RXD which receives incoming serial data and pin A6 is the serial clock output or the serial clock input. Pin A6 and A7 may also be used to clock the on-chip timer/event counters, Timer 2 and Timer 1 respectively, of the 70X1 devices.
- Port B:** When in the single chip mode, Port B is an eight bit general-purpose output port. In all other modes, Port B is split into two parts with the lower nibble (pins B0-B3) being general-purpose output only pins and the most significant nibble (pins B4-B7) are the bus control signals: ALATCH, R/W, ENABLE, and CLOCK OUT. On 70X1 devices, pin B3 is also the serial output line (TXD) for the serial port.
- Port C:** In Single-Chip Mode, Port C is an 8-bit bidirectional I/O port where any of its eight pins may be individually programmed as an input or output line under software control. In any other mode, Port C becomes a multiplexed address/data port for the off-chip memory bus; in this case, the least significant byte of a 16-bit address is provided followed by 8-bits of read or write data.

Port D: In Single-Chip or Peripheral Expansion Mode, Port D is an 8-bit bidirectional I/O port where any of its eight pins may be individually programmed as an input or output line under software control. In Full Expansion and Microprocessor Modes, Port D provides the most significant byte of the 16-bit address.

Further details of I/O and memory operations are contained in the memory mode sections in Section 2.3.

2.3 MEMORY MODES

The TMS7000 can be reconfigured to reference up to 64K bytes of ROM and RAM. Five memory modes can be selected by a combination of software and hardware: the Single-Chip, Peripheral Expansion, Full Expansion, Microprocessor, and System Emulator modes. The Mode Control (MC) input pin, if at a logic one, will force the TMS7000 into the Microprocessor Mode. If the MC pin is held at +14 volts, the TMS7000 will enter the System Emulator mode. If the MC pin is held at logic zero, the remaining memory modes are selected by the two MSBs of the I/O Control Register (IOCNT0). i.e., bits 6 and 7, as shown in Table 2-2.

TABLE 2-2 — MODE SELECT CONDITIONS

MODE	MODE SELECT CONDITIONS	
	MODE CNTL	I/O CONTROL
	PIN	REG. BIT 7, 6
Single-Chip	0 V	0 0
Peripheral Expansion	0 V	0 1
Full Expansion	0 V	1 0
Microprocessor	V _{CC}	X X
System Emulator	+14 V	X X

NOTE: X = Don't Care

Upon RESET, the IOCNT0 Register is set to zeros. Refer to Section 2.5.2 for a detailed description of RESET and the recommended initialization procedure for the IOCNT0 Register. The five memory modes are summarized in Table 2-3 and 2-4 and described in the following paragraphs.

TABLE 2-3 – 70X0 MEMORY MAP

	SINGLE CHIP	PERIPHERAL EXP	FULL EXPANSION	MICRO.	EMULATOR
>0000-	REGISTER FILE				
>007F-	REGISTER FILE				
>0080-	RESERVED FOR FUTURE EXPANSION				
>00FF-	RESERVED FOR FUTURE EXPANSION				
>0100-	ON CHIP I/O (TIMERS, INTERRUPTS, I/O PORTS)				
>010B-	ON CHIP I/O (TIMERS, INTERRUPTS, I/O PORTS)				
>010C-	PERIPHERAL EXPANSION				
>0200-	NOT AVAILABLE				
>C000-	NOT AVAILABLE				
>D000-	70120			MEMORY EXPANSION	
>F000-	12K ROM	7040 ROM 70C40	7020 ROM 70C20 2K ROM	12K ROM	7040 ROM 70C40
>F800-		7040 ROM 70C40	7020 ROM 70C20 2K ROM	12K ROM	7040 ROM 70C40
>FFFF-					7020 ROM 70C20 2K ROM
	SINGLE CHIP	PERIPHERAL EXP	FULL EXPANSION	MICRO-PROCESSOR	EMULATOR

TMS7000 USES THESE MODES ONLY (No on-chip ROM)

TABLE 2-4 – 70X1 MEMORY MAP

	SINGLE CHIP	PERIPHERAL EXP	FULL EXPANSION	MICRO.	EMULATOR
>0000-	REGISTER FILE				
>007F-	REGISTER FILE				
>0080-	RESERVED FOR FUTURE EXPANSION				
>00FF-	RESERVED FOR FUTURE EXPANSION				
>0100-	ON CHIP I/O (TIMERS, INTERRUPTS, I/O PORTS, SERIAL PORT)				
>0117-	ON CHIP I/O (TIMERS, INTERRUPTS, I/O PORTS, SERIAL PORT)				
>0118-	PERIPHERAL EXPANSION				
>01FF-	NOT AVAILABLE				
>0200-	NOT AVAILABLE				
>F000-	7041 ON-CHIP PROGRAM ROM, 4K BYTES				7001 USES ONLY THESE MODES.
>FFFF-	7041 ON-CHIP PROGRAM ROM, 4K BYTES				7001 USES ONLY THESE MODES.
	SINGLE CHIP	PERIPHERAL EXP	FULL EXPANSION	MICRO-PROCESSOR	EMULATOR

SINGLE-CHIP MODE:

In the Single-Chip mode, all 32 I/O pins are used for input/output, and no off-chip memory bus is implemented. All programs and data reside in the on-chip ROM and RAM.

PERIPHERAL EXPANSION MODE:

In the Peripheral Expansion mode, the Peripheral File addresses are available externally. 20 of the 32 general purpose I/O lines are still used as general purpose I/O and 12 pins implement a multiplexed 8-bit address/data bus and a 4-bit control bus. Out of the total 256 addresses in the Peripheral File, 246 of these are memory mapped externally on the 70X0 devices and 238 are mapped externally on the 70X1 devices. This expansion mode may be used to address ROM, RAM, or peripheral devices.

FULL EXPANSION MODE:

In the Full Expansion mode, 12 of the 32 I/O pins are used for general purpose I/O. The remaining 20 I/O pins are then used to implement a 8-bit most significant address bus, a multiplexed 8-bit least significant address and 8-bit data bus, and a 4-bit control bus, to external memory. The on-chip ROM is still used, but additional off-chip memory for program or data storage may also be referenced.

MICROPROCESSOR MODE:

In the Microprocessor mode, the 32 I/O pins are in the same configuration as in the Full Expansion mode. However, the addresses for the on-chip ROM are located off-chip, allowing the user's program to be prototyped in EPROM. Since the TMS7000 and TMS7001 have no on-chip ROM, this mode and the emulator mode are usually the only modes in which they can operate.

SYSTEM EMULATOR MODE: The System Emulator mode is provided for self-emulation and system development. No on-chip I/O is implemented. In addition, the on-chip timer and interrupt controls are disabled.

2.3.1 Single-Chip Mode

In the Single-Chip mode, the TMS7000 functions as a standalone microcomputer with no off-chip memory expansion bus. The 32 I/O lines may be used for various purposes, such as scanning keyboards, driving displays, and controlling other mechanisms. The four ports are configured as shown in Figure 2-5.

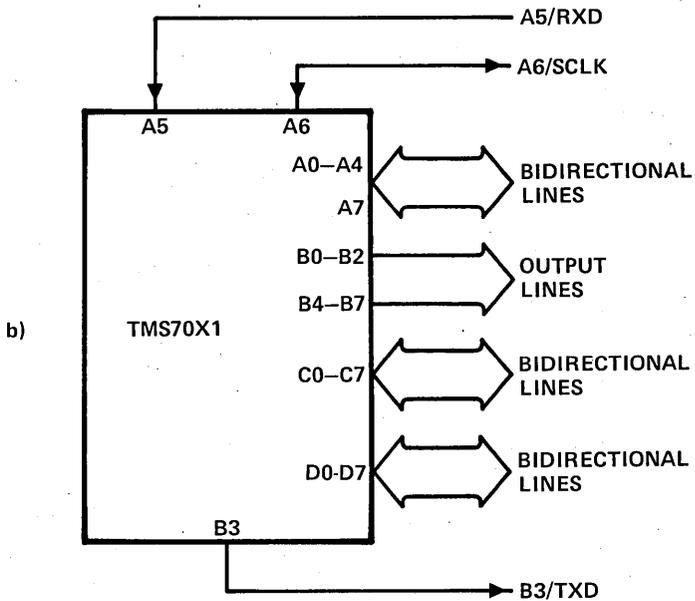
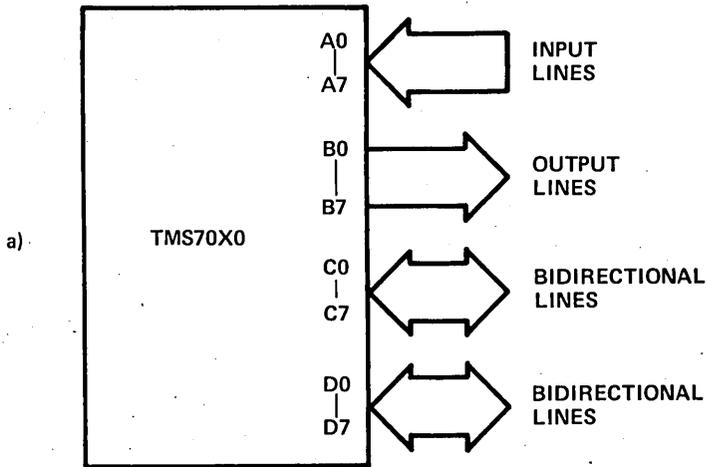


FIGURE 2-5 - I/O PORTS: SINGLE-CHIP MODE

Table 2-3 and 2-4 illustrate the Single-Chip mode memory map. The unused Peripheral File (PF) locations and off-chip memory addresses are not available. When reading from unavailable addresses, an undefined value is returned. Writing to these addresses has no effect. Peripheral File registers, P0-P11, are used to reference the I/O ports and other on-chip functions. Table 2-5 and 2-6 list the Peripheral File (PF) registers that are available in the Single-Chip mode configuration.

TABLE 2-5 — TMS70X0 PERIPHERAL MEMORY MAP

	SINGLE CHIP	PERIPHERAL EXP.	FULL EXPAND.	MICROPROCESSOR
>0100-	I/O CONTROL REGISTER (IOCNT0)			P0
>0101-	RESERVED			P1
>0102-	TIMER DATA (TIDATA)			P2
>0103-	TIMER CONTROL (TICNTL)			P3
>0104-	PORT A DATA VALUE (APORT)			P4
>0105-	RESERVED			P5
>0106-	— BITS 0-3 —	PORT B DATA VALUE (BPORT)		P6
>0106-	PORT B DATA	PERIPHERAL EXPANSION	— BITS 4-7 —	P6
>0107-	RESERVED			P7
>0108-	PORT C DATA	PERIPHERAL EXPANSION		P8
>0109	C PORT DATA DIRECTION (CDDR)			P9
>010A	PORT D DATA VALUE (DPORT)			P10
>010B-	D PORT DATA DIRECTION REGISTER (DDDR)			P11
>010C-	UNUSEABLE	PERIPHERAL EXPANSION		P12
>01FF-				P255
	SINGLE CHIP	PERIPHERAL EXP	FULL EXPANSION	MICROPROCESSOR

NOTE: There are no on-chip peripheral registers in the Emulator mode.

TABLE 2-6 – TMS70X1 PERIPHERAL MEMORY MAP

	SINGLE CHIP	PERIPHERAL EXP	FULL EXP.	MICROPROCESSOR
>0100-	I/O CONTROL REGISTER 0 (IOCNT0)			P0
>0101-	RESERVED			P1
>0102-	TIMER DATA (T1DATA)			P2
>0103-	TIMER CONTROL (T1CNTL)			P3
>0104-	PORT A DATA VALUE (APORT)			P4
>0105-	PORT A DATA DIRECTION REG. (ADDR)			P5
>0106-	– BITS 0-3 –	PORT B DATA VALUE (BPORT)		P6
>0106-	PORT B DATA	PERIPHERAL EXPANSION – BITS 4-7 –		P6
>0107-	RESERVED			P7
>0108-	PORT C DATA	PERIPHERAL EXPANSION		P8
>0109-	C PORT DATA DIRECTION (CDDR)			P9
>010A-	PORT D DATA VALUE (DPORT)		PERIPHERAL EXPANSION	P10
>010B-	D PORT DATA DIRECTION REG.			P11
>010C-	UNUSEABLE	PERIPHERAL EXPANSION		P12-
>010F-				P15
>0110-	I/O CONTROL REGISTER 1 (IOCNT1)			P16
>0111-	FIRST WRITE SERIAL MODE (SMODE) WRITE SERIAL CONTROL 0 (SCTL0) READ STATUS REGISTER (SSTAT)			P17
>0112-	TIMER 2 DATA (T2DATA)			P18
>0113-	TIMER 2 CONTROL (T2CNTL)			P19
>0114-	TIMER 3 DATA (T3DATA)			P20
>0115-	SERIAL CONTROL (SCTL1)			P21
>0116-	RECEIVER BUFFER (RXBUF)			P22
>0117-	TRANSMITTER BUFFER (TXBUF)			P23
>0118-	UNUSEABLE	PERIPHERAL EXPANSION		P24
>01FF-				P255
	SINGLE CHIP	PERIPHERAL EXP	FULL EXP.	MICROPROCESSOR

Port A is referenced as PF Register P4 (APORT). When P4 is read, such as with a move from PF (MOV_P) instruction, the value on the Port A input pins is returned. The input data is read approximately two machine cycles before the completion of the instruction.

Bit A0 is the LSB, and bit A7 is the MSB of Port A. When the on-chip Timer/Event Counter is placed in the External Event Counter Mode, bit A7 serves as the external clock input, triggering the Event Counter on every positive-going transition.

On the 70X1 parts, pins A0-A4 and pin A7 are bidirectional I/O pins. Each of these pins can become either an output or an input pin depending upon the value in the A port Data Direction Register (ADDR) P5. If a '1' is put in the bit position of P5 then the corresponding pin of the A port is an output. If a '0' is written there, then the A port pin becomes a high impedance input pin. Refer to Figure 2-4 for a diagram of the bidirectional I/O logic. On the 70X1 parts, A5 and A6 have multiple functions. Normally they are both input only pins like the 70X0 parts, but A5 also can be the serial data receiver (RXD). Pin A6 can also be the serial clock I/O pin (SCLK) for the serial port. It can be either the serial clock output or it can drive the on-chip serial clock when connected to an external clock. See the serial port section for more information, Section 2.7.2. Pin A6 can also be the external clock input for Timer 2.

The Port B pins always assert the value of the Port B data value register, which is PF Register P6 (BPORT). Writing to P6 loads the Port B register and hence modifies the Port B output pins. Positive logic is used. While RESET is active, Port B register contents are forced to ones by the on-chip circuitry.

The C and D ports (CPORT and DPORT) are bidirectional I/O pins and are located at P8 and P10 of the Peripheral File. Each of these pins can become either an output or an input pin depending upon the value in the C and D port Data Direction Register (CDDR and DDDR), P9 and P11. If a '1' is put in a bit position of the DDR then the corresponding pin of the port is an output. If a '0' is written there, then the port pin becomes a high impedance input pin. Writing to CPORT or DPORT modifies the programmed output pins but has no effect on the input pins. Reading CPORT and DPORT provides the input values for input pins and the current output value for output pins. Refer to Figure 2-4 for a diagram of the bidirectional I/O logic.

Reading from an output pin (or a bidirectional pin in the output mode) provides the current value being output on that pin. Peripheral File instructions ANDP, ORP, and XORP perform a read/modify/write cycle to PF registers so that when applied to a port data register, these instructions can clear, set, and complement the output pins on the port. The following program fragment illustrates the use of the I/O lines in the Single-Chip mode:

IOCNT0	EQU	P0	
APOINT	EQU	P4	
BPOINT	EQU	P6	
CPOINT	EQU	P8	
CDDR	EQU	P9	
DPOINT	EQU	P10	
DDDR	EQU	P11	
RESET	MOVP	% >3F,IOCNT0	Set Single-Chip Mode, enable all interrupts, clear all pulse flip-flop
L1	MOVP	% >02,DPOINT	Load Port D with 0000 0010
L2	MOVP	% >00,CPOINT	Load Port C with 0000 0000
	MOVP	% >F0,CDDR	Config C7-C4 outputs, C3-C0 inputs
	MOVP	% >0F,DDDR	Config D3-D0 outputs, D7-D4 inputs
	ORP	% >04,DPOINT	Set D2
	ANDP	% >7F,CPOINT	Clear C7
	BTJZP	% >08,CPOINT,L1	Jump if C3 is '0'
	MOVP	% >55,BPOINT	Set Port B to 0101 0101
	XORP	% 1,BPOINT	Toggle bit B0
	BTJOP	% >41,APOINT,L2	Jump if either A6 or A0 is a '1'

NOTE

The percent sign (%) indicates the Immediate Addressing Mode (see Section 3.1).
The instruction set is described in Section 3.2.

2.3.2 Peripheral Expansion Mode

The Peripheral Expansion mode incorporates features of both the I/O-intensive single-chip mode and the memory-intensive Full Expansion mode. Table 2-5 and 2-6 show the memory maps for the Peripheral Expansion mode. References to addresses in the Peripheral File (locations >0100 to >01FF) not corresponding to on-chip registers, result in off-chip memory cycles. During peripheral file instructions, a peripheral file port is read, even if the value is not needed such as in a MOVP A,P6. If this read is undesirable because of hardware configuration, a STA (store A) instruction with the memory-mapped address of the peripheral register can be used.

The ability to reference off-chip addresses permits the TMS7000 to be directly connected to most of the popular peripheral devices developed for 8-bit microprocessors. The TMS7000's Peripheral File (PF) instructions can be used to reference these off-chip peripherals just as easily as the on-chip PF registers are accessed. In Peripheral Expansion Mode, Port A functions the same as in Single-Chip Mode.

Port B is divided into two sections: B3-B0 function as individual output pins, the same as in Single-Chip Mode; pins B7-B4, however, function as external memory bus controls as follows:

- Pin B4 (ALATCH) is strobed to logic one while Port C asserts the memory address.
- Pin B5 (R/W) is driven to logic one for a read cycle and to logic zero for a write cycle.
- Pin B6 (ENABLE) is asserted at logic zero whenever an external memory cycle is in progress.
- Pin B7 (CLOCKOUT) is an output clock intended for general memory control timing.

Exact signal timing is described in Section 4.

References to the PF register corresponding to Port B are handled in a special manner. When a write is done to the Port B data value register, B3-B0 output their new value. An external memory write cycle, writing the full 8-bit Port B value to address >0106, is performed as well. When a read is done from the Port B data value register, the least significant nibble is provided by the current value on pins B3-B0. The most significant nibble, however, is obtained from an external memory read cycle, reading from address >0106. The least significant nibble from the external memory read cycle is discarded.

Port C functions as a multiplexed address/data port for the memory expansion bus. In normal configurations, Port C is attached to the input of an 8-bit latch such as an SN74LS373. Signal B4 (ALATCH) drives the G input of the latch, so that: the outputs follow the inputs while ALATCH is high, and latch when ALATCH falls. After ALATCH falls, Port C either becomes a high-impedance input for read cycles or it asserts the output data for write cycles. Port D functions identically to a bit-programmable bidirectional I/O port, as in the Single-Chip Mode.

NOTE

Because ALATCH and Port C are active for both external and internal (ROM and RAM) memory cycles, it is recommended that $\overline{\text{ENABLE}}$ be gated with the chip select input of all external memory devices.

2.3.3 Full Expansion Mode

The Full Expansion Mode may be used to extend the memory addressing capability of the TMS7000 to its full 64K byte limit. External memory may be accessed with instructions using the Direct, Register File Indirect, and Indexed Addressing modes of the instruction set. This capability allows a variety of application requirements to be met by expanding the external program or data storage.

Full Expansion Mode input/output is identical to the Peripheral Expansion mode except that Port D is used to output the most significant byte (MSB) of the 16-bit address and is not available as an I/O port. The I/O memory assignments for the Full Expansion mode are shown in Table 2-5 and Table 2-6.

As in the Peripheral Expansion mode, addresses to the Peripheral File (locations >0100 to >01FF) which are not directly implemented as on-chip registers, result in off-chip memory cycles. The on-chip Peripheral File registers are listed in Table 2-5 and Table 2-6. Note that the Port D data value register (DPORT) and the Port D Data Direction Register (DDDR) are implemented as off-chip addresses in the Full Expansion mode.

2.3.4 Microprocessor Mode

The Microprocessor mode is intended for applications not justifying the use of on-chip ROM. The port pins are configured exactly as in the Full Expansion mode (see Table 2-2). However, unlike the Full Expansion mode, no on-chip ROM is referenced in the Microprocessor mode as shown in Table 2-3. The MC pin must be held at +5 volts to place the device into the Microprocessor Mode.

2.3.5 System Emulator Mode

The System Emulator mode is a special purpose mode designed to support system development and self-emulation. The TMS7000 is placed in the System Emulator mode by applying a +14 volt level to the Mode Control (MC) input pin. This disables all internal ROM and I/O. In addition, the internal structure for handling interrupts is disabled.

NOTE

The last 48 bytes (>FFD0->FFFF) of off-chip memory may be assigned to Traps 0-23.

The usefulness of System Emulator Mode is predicated on its flexible interrupt structure. Up to 128 interrupts may be implemented by wire-ORing them to either the maskable interrupt input (INT) or to the non-maskable interrupt input (NMI).

Both interrupt lines are level-activated in System Emulator Mode. They do not have the pulsed interrupt latch, as described in Section 2.5.

The processor acknowledges interrupts in the System Emulator mode by asserting an Interrupt Acknowledge (INTA) output on pin B3 of Port B. This is comparable to the INTA signal sent from the CPU to internal interrupt logic, described in Section 2.5.3. When INTA is asserted, external circuitry must apply an 8-bit interrupt code into Port C, which is then used by the CPU to generate the address of the interrupt vector. The vector address is computed by adding the interrupt code input to >FF00 and then rotating the result left one bit. This is the address of the LSB of the vector: the MSB is in the preceding address.

The interrupt vector is the same as the TRAP instruction opcode; for example, a Level 2 interrupt code is >FD, which is the same as the TRAP 2 opcode. Interrupt vector generation is illustrated in Figure 2-6.

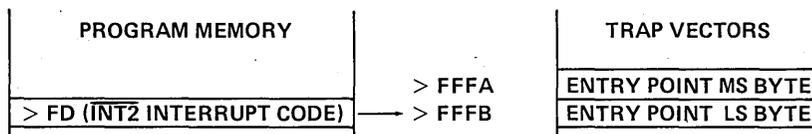


FIGURE 2-6 – INTERRUPT GENERATION: SYSTEM EMULATOR MODE

As with all interrupts, the processor pushes the contents of the Status Register and the Program Counter onto the stack before branching to the address specified by the interrupt vector.

2.4 I/O CONTROL REGISTERS

The I/O control registers are located in the Peripheral File and are responsible for memory mode definition and interrupt control. All TMS7000 family members contain the I/O Control 0 (IOCNT0) register; however, the I/O Control 1 (IOCNT1) register is available only in the 70X1 members. The I/O control registers are mapped into locations P0 (IOCNT0) and P16 (IOCNT1) of the Peripheral File as shown in Figures 2-7 and 2-8. The memory expansion modes and individual interrupt masks and resets are controlled through these registers. The interrupt sources may also be individually tested by reading the interrupt flags. The interrupt flag values are independent of the interrupt enable values. Section 2.3 describes how bits 7 and 6 of the IOCNT0, together with the Mode Control (MC) pin, determine in which memory expansion mode the TMS7000 is functioning. See Table 2-2.

The INTn FLAG values are independent of the INTn ENABLE values. Writing a '1' to the INTn CLEAR bit will clear the corresponding INTn FLAG, but writing a '0' to the INTn CLEAR bit has no effect on the bit. If INTn is to be recognized by the CPU, three conditions must be met:

- 1) A one must be written to the INTn ENABLE bit in the IOCNT0 or IOCNT1 Register.
- 2) The global INTERRUPT ENABLE bit, i.e., bit 4 or I in the Status Register (see Section 2.1.4), must be set to one by the EINT instruction.
- 3) INTn must be the highest priority interrupt asserted within an instruction boundary (see Section 2.5).

All of the TMS7000's interrupts may be tested in software, independent of whether the interrupt is enabled or disabled. For example, the following program fragment waits for the rising edge of the interrupt input on the INT1 pin by testing INT1 FLG:

```
WAIT   BTJOP   %>02,P0,WAIT   Wait for INT1.
```

This allows the interrupt pins to be polled as 'latching' inputs when the interrupt action is not desired. Refer to Section 2.5 for a detailed description of the TMS7000's interrupt logic and operation.

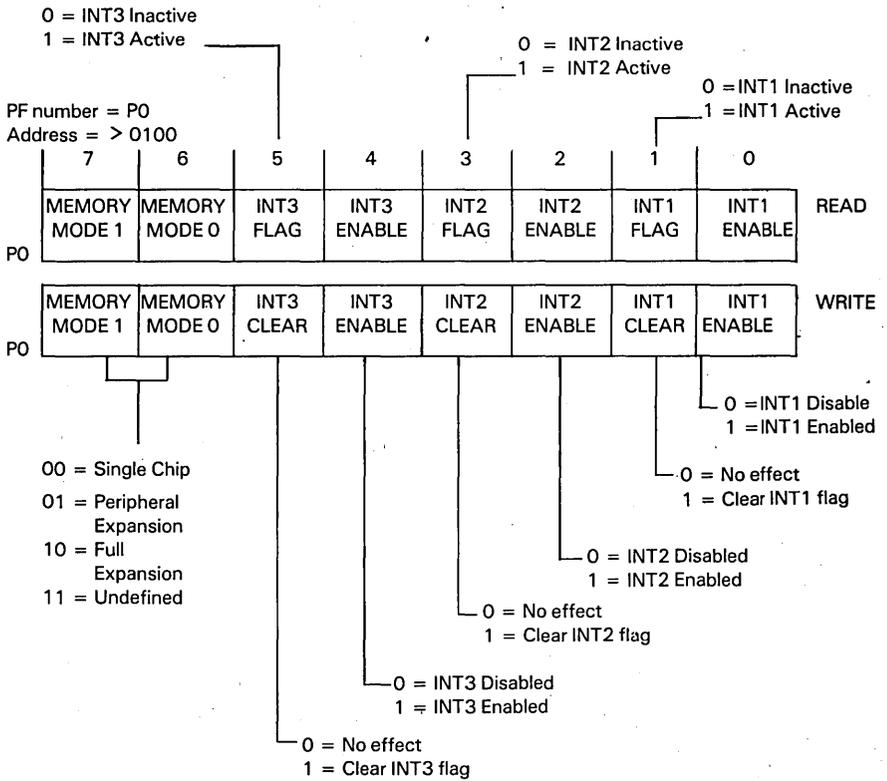


FIGURE 2-7 — IOCNT0 - I/O CONTROL REGISTER 0

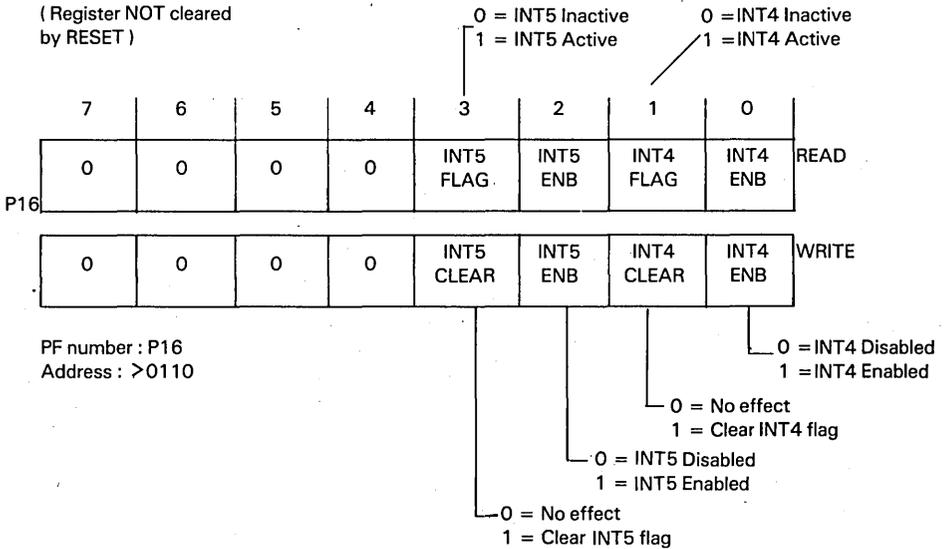


FIGURE 2-8 – IOCNT1 - I/O CONTROL REGISTER 1

Due to the read/modify/write nature of the bit manipulation instructions (ANDP, ORP, and XORP), it is possible that a pulsed interrupt could occur during the operation of these instructions on the IOCNT0 and IOCNT1 and be missed. These instructions could also cause the other interrupt flags to be unintentionally cleared or set. For example, there is no problem if an XORP instruction is used to enable INT1 and not alter the condition of the INT1 flag (XORP % >03,PO), as long as the flag flip-flop does not change state during instruction execution. However, if a short INT1 pulse occurs during execution, a 0 may be read and a 1 would be written to reclear the INT1 flag. In this case, the INT1 pulse would be undetected by the processor. This same instruction would also affect the INT2 and INT3 flags as they are also located in IOCNT0. To avoid these occurrences, use the MOVP and STA instructions when writing data to IOCNT0 and IOCNT1.

The following code segment is an example of how the user can regulate the memory mode bits and individual interrupt masks and resets through program control:

```

IOCNT0 EQU      PO
*      MOV      % >3B,IOCNT0      SINGLE-CHIP MODE, CLEAR ALL INT FLAGS
                                     ENABLE INT1 AND INT3
      BTJOP    % >08,IOCNT0,LABEL TEST IF INT2 SET, IF SET THEN JUMP
      ANDP    % >E5,IOCNT0      CLEAR AND DISABLE INT3
LABEL  EQU      $

```

NOTE

This example is one of the few situations where use of the ANDP instruction on the IOCNT0 register is possible.

On RESET, the IOCNT0 register is written with all 0s. This disables INT1, INT2, and INT3 individually and configures the TMS7000 in Single-Chip mode. In the 70X1 devices, the IOCNT1 register is not written to during RESET. In order to ensure that INT4 and INT5 are also individually disabled, it is recommended that all '0's be written to the IOCNT1 register immediately after RESET. Note that following RESET, all interrupts are globally disabled because the Interrupt (I) bit in the status register is reset to 0.

Because the state of the interrupt flag flip-flops (INTn FLG) are undetermined after RESET, it is recommended that the flags be cleared by writing a 1 to bit positions 1, 3, and 5 in PO (IOCNT0) and positions 1 and 3 in P16 (IOCNT1).

2.5 INTERRUPTS AND RESET CLOCK OPTIONS

The internal machine cycle frequency, called Phi (ϕ), is derived from either a crystal or an external clock source. There are two options available for converting the external frequency to ϕ and they are called the divide by two (/2) or the divide by four (/4) clock options. These are mask options which means the option is placed on a manufacturing template, a mask, which copies the actual circuit onto the silicon device. This means the clock option is finalized at the start of manufacture and is NOT changeable by software or hardware. If the /2 clock option is chosen, the external frequency divided by 2 is the internal machine cycle. A 5 MHz crystal would give an internal cycle of 2.5 MHz with the divide by 2 option. If the /4 clock option is used, the external clock is divided by 4 so that the same 5 MHz crystal would result in a ϕ of 1.25 MHz. In order to get a 2.5 MHz internal cycle a 10 MHz crystal would be used.

The /2 option is recommended for use with crystals and the /4 option can use either crystals or another external source. It is not recommended to use an external source to drive a /2 device. If a crystal is used it is connected between pins XTAL1 and XTAL2. To improve the crystal waveform, 15 pF capacitors are connected between XTAL1 and ground and between XTAL2 and ground. If an external clock source is used it is connected to CLKIN, also called XTAL2, and XTAL1 is left floating.

2.5.1 Interrupt Priority

The TMS70X0 has priority servicing of three interrupt levels and reset, the TMS70X1 has five interrupt levels plus reset. These levels are defined as follows:

- 1) Level 0 is the highest priority and is reserved for the RESET function.
- 2) Level 1 is the second highest priority and is a user-defined external interrupt (INT1).
- 3) Level 2 is the third highest priority and is reserved for the on-chip hardware Timer 1 (INT2).

- 4) Level 3 is the fourth highest priority and is a user-defined external interrupt (INT3).
- 5) Level 4 is the fifth highest priority and is available only on the 70X1 devices. This interrupt is used when the serial port is ready for data transfer, or it can be used by Timer 3 (INT4).
- 6) Level 5 is the lowest priority and is available only on the 70X1 devices. This interrupt is reserved for the on-chip hardware Timer 2 (INT5).

All external interrupts and RESET have Schmitt trigger inputs. The external interrupt interface consists of three discrete active low input lines which require no external synchronization: RESET, INT1, and INT3. The INT1 and INT3 inputs are both latch and level triggered on all TMS7000 devices, with some exceptions on CMOS parts. The INT1 input is only latch triggered on the TMS70C00, TMS70C20 and TMS70C40. Interrupt Level 2 (INT2) is asserted upon rollover of the programmable timer (see Section 2.6).

Each interrupt (INT_n) is associated with an INT_n ENABLE and FLAG bit in the IOCNT0 and IOCNT1 Registers (see Section 2.4). The INT_n ENABLE bit must be set before INT_n can be recognized by the interrupt logic. In addition, there is a global INTERRUPT ENABLE bit (I) in the Status Register which must be set by the EINT instruction in order for an interrupt to be recognized by the CPU.

The TMS7000's reset function, CPU/interrupt interface, and interrupt logic are described in the sections that follow.

2.5.2 Device Initialization

Interrupt Level 0 (RESET) cannot be masked and will be recognized immediately, even in the middle of an instruction. To execute the Level 0 interrupt, the RESET pin must be held low for a minimum of 1.25 internal clock cycles (ϕ) to guarantee recognition by the device. During assertion of the RESET pin, the Data Direction Registers CDDR and DDDR registers (and ADDR on 70X1 devices) are cleared to all '0's and the OUTPUT DATA flip-flops of Ports B, C, and D (and Port A on 70X1 devices) are set to all ones (see I/O logic, Figure 2-4). This causes Ports C and D (and Port A on 70X1 devices) to be placed in the high impedance input mode and Port B to output all ones (>FF) regardless of the state of the internal machine clock. When RESET is removed, the following operations are performed prior to the first instruction acquisition.

- 1) All zeros are written to the IOCNT0 Register and the Status Register. This disables INT1, INT2, and INT3 and leaves the INT_n FLAG bits unchanged. Note that the IOCNT1 Register in 70X1 devices is not written to.
- 2) The MSB and LSB values of the Program Counter just before RESET are stored in R0 and R1 (A and B registers) respectively.
- 3) The Stack Pointer is initialized to >01.
- 4) The MSB and LSB of the reset vector are fetched from locations >FFFE and >FFFF respectively (see Table 2-10) and loaded into the Program Counter.
- 5) Program execution begins from the address placed in the Program Counter.

As stated above, the reset function does not change the INT_n FLAG bits in the IOCNT0 register (since all zeros are written) and does not write at all to the IOCNT1 register. Also, the OUTPUT DATA flip-flops of the A, C, and D Ports are set to all '1's. If any of the bits in a DDR register is

set to a '1' ; the corresponding port pin would become an output, producing a '1' level. It is generally good practice to initialize the OUTPUT DATA flip-flop with the desired output value (by writing to the port data value register) before writing to the DDR flip-flop to make the corresponding pin an output. The following sequence of code is an example of what a typical initialization routine could be after a RESET.

RESET	MOVP	% >2E,P0	Clear INT1, INT2, and INT3 FLAGS and place device in Single-Chip mode. Enable INT2.
	MOVP	% >0A,P16	Clear INT4, INT5 FLAGS (70X1 only). Disable INT4 and INT5
	MOVP	% VALU1,P8	Load Port C data value register (CPORT).
	MOVP	% MASK1,P9	Load Port C data direction register (CDDR).
	MOVP	% VALU2,P10	Load Port D data value register (DPORT).
	MOVP	% MASK2,P11	Load Port D data direction register (DDDR).
	MOVP	% VALU3,P2	Load Timer 1 Latch (TL).
	MOVP	% VALU4,P3	Load timer source, internal prescaler latch and start timer.
	EINT		Set global interrupt enable bit to allow interrupts.

The Stack Pointer can also be reinitialized in the Register File following reset by executing a program similar to the one below.

STACK	MOV LDSP	% VALUE,B	Load B with the stack starting point Put this value into the stack pointer
-------	-------------	-----------	---

2.5.3 CPU Interface To Interrupt Logic

Once an interrupt has been asserted (the INTn pin goes low), it becomes active if its ENABLE bits are set to one, and the global Status Register INTERRUPT ENABLE bit (I) is set to one. An active interrupt is one which is capable of being recognized by the CPU but has not yet been acknowledged.

As shown in Figure 2-9, the TMS7000's on-chip logic recognizes an active interrupt and sends an INT ACTIVE signal to the CPU. When the currently executing instruction is completed, the CPU selects the highest priority active interrupt and routes INTA back to the INTn ACK (interrupt acknowledge) line of the recognized interrupt. In the case of more than one interrupt active within the same instruction boundary, i.e., simultaneous interrupts, then the interrupts will be acknowledged by the CPU according to the priority levels described at the beginning of Section 2.5. For example, if both INT2 and INT3 occur within the same instruction boundary, INT2 will always be serviced first. Refer to Section 2.6.8 for an application of this example.

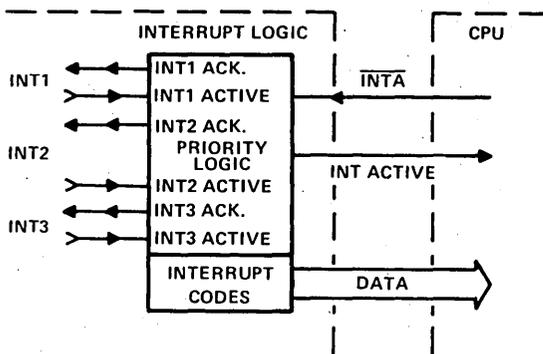


FIGURE 2-9 — CPU INTERFACE TO INTERRUPT LOGIC

Once INT_n has been acknowledged by the CPU, the INT_n ACK line, as shown in Figure 2-10, clears the corresponding INT_n FLAG flip-flop. The CPU then pushes the contents of the Status Register and the Program Counter (MSB and LSB) onto the stack, and zeros the Status Register, including the global INTERRUPT ENABLE (I) bit. The CPU reads an interrupt code from the interrupt logic and branches to the address contained in the corresponding interrupt vector location in memory. The addresses of the trap vector locations for each interrupt level are shown in Table 2-7. There are 19 internal clock cycles (ϕ) required between the end of an instruction in the interrupted program and the start of the first instruction of the interrupt routine. Interrupting out of the IDLE state requires 17 machine cycles.

TABLE 2-7 — RESET AND INTERRUPT VECTOR LOCATIONS IN ROM

VECTOR MSB	VECTOR LSB	DESCRIPTION	SERVICE ORDER
>FFFE	>FFFF	RESET	Immediate
>FFFC	>FFFD	INT1 External	1
>FFFA	>FFFB	INT2 Timer 1	2
>FFF8	>FFF9	INT3 External	3
		70X1 only below	
>FFF6	>FFF7	INT4 Serial port	4
>FFF4	>FFF5	INT5 Timer 2	5

The interrupt service routine can explicitly enable nested interrupts by executing the EINT instruction to directly set the I bit in the status register to a one, thus permitting nested interrupts to be recognized. When the nested interrupt service routine completes, it returns to the previous interrupt service routine by executing the RETI instruction.

2.5.4 Interrupt Logic

The internal interrupt logic for each the three maskable interrupts for the 70X0 devices and five maskable interrupts for the 70X1 devices is shown in Figure 2-10.

The logic is slightly different for INT1 on the 70CX0 devices so that this interrupt logic will only detect the Q1 output of the Pulse flip-flop and not INTn. On the CMOS parts, INT1 is a latched interrupt and not a latched and level as on the other interrupts.

To even further conserve the already low power requirements of the CMOS devices, two low power modes are provided. These modes are called Halt and Wake-up and are entered by executing a IDLE instruction. Either an external interrupt or the timer interrupt will release the device from the low power modes depending on whether it is in the Halt or Wake-up mode. See Section 4 for a complete description of the modes and interrupts.

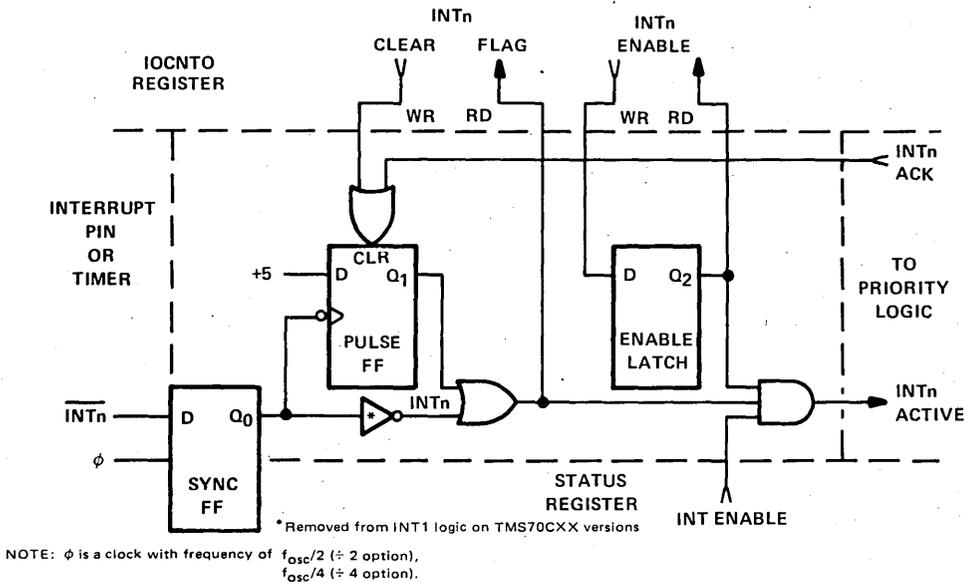


FIGURE 2-10 – INTERRUPT LOGIC

When an external interrupt is first asserted, its level is gated into the Sync flip-flop by the Phi (ϕ) clock signal, which has a frequency of $f_{OSC}/2$ for the $/2$ clock option and $f_{OSC}/4$ for the $/4$ clock option. In order for a pulse interrupt signal to be detected, the pulse width must be a minimum of 1.25 Phi (ϕ) frequency periods. The output of the Sync flip-flop clocks a 1 into the Pulse flip-flop. This is the only time a 1 is loaded into the Pulse flip-flop. The Pulse flip-flop will be set within 1.25 machine cycles (ϕ) of the assertion of the interrupt. If INTn is removed before the interrupt is recognized, its occurrence is latched in by the INTn Pulse flip-flop (Q1). The INTn ENABLE bit is used separately to individually mask interrupt levels. This bit must be 1 for the interrupt to be recognized.

As previously stated, all interrupt control bits are implemented in the IOCNT0 and IOCNT1 registers in the Peripheral File. I/O instructions may simply read from and write to each INTn ENABLE bit (Q2).

The INTn FLAG is handled differently. When the INTn FLAG bit is read, the logical OR of the Pulse flip-flop output (Q1) and $\overline{\text{INTn}}$ (inverted INTn pin) is returned. As long as the INTn pin is low, the INTn FLAG bit will be read as a 1, regardless of the state of the pulse flip-flop. This makes the external interrupts both latch and level sensitive. This is different on INT1 of the 70CX0 devices however. When the INT1 FLAG is read, the pulse flip-flop output (Q1) is the only return. This makes INT1 of the TMS70CX0 a latched interrupt only and not a level interrupt. When a 1 is written to the INTn CLEAR bit (See Section 2.5.3), the pulse flip-flop is cleared. Writing a 0 to INTn CLEAR has no effect.

The pulse flip-flop allows short pulsed external interrupt signals to be recognized by the CPU. A pulsed interrupt signal must have a minimum pulse width of 1.25Φ frequency periods in order to be gated into the pulse flip-flop. The pulse flip-flop will retain the signal until the interrupt is recognized. When the interrupt is acknowledged by the CPU, the pulse flip-flop is cleared automatically. To make sure the pulsed interrupt is not interpreted as a level signal, the maximum pulse (time low) of a pulsed interrupt cannot exceed the following:

$$(16 + N) / \Phi$$

where N equals the number of machine cycles in the interrupt service routine, up to and including the EINT or RETI instruction and Φ is the internal machine clock frequency.

This ensures that the INTn FLAG is cleared prior to the first possible instruction boundary in which the interrupt could be rescheduled. Note that this is not of any concern to INT1 on the TMS70CX0 devices since INT1 is not level sensitive.

The interrupt structure of the TMS7000 also permits wire-ANDing of multiple interrupt sources onto a single INTn pin, by allowing level-sensitive interrupt detection in addition to pulse-sensitive detection. A high-to-low transition on the INTn pin sets the pulse flip-flop, as previously described, and this, as well as the low level of the INTn pin, sets the INTn FLAG in the active state. When the interrupt is accepted, the pulse flip-flop is cleared and will not be set again until after the next high-to-low transition of the INTn pin. If the INTn pin remains at a low level, the corresponding INTn FLAG will remain active, and the interrupt will be recognized again.

This structure allows multiple interrupts to be wire-ANDed onto one interrupt, since the interrupt will be repeatedly recognized as long as the interrupt pin is low. An application program could determine which of several interrupts are requesting service and set its own priority structure.

Interrupt inputs can be tested, using the interrupt FLAG bits (See Section 2.4) without actually recognizing the interrupt, thus permitting flexible multi-device control. Under program control, each interrupt routine can retain complete control of the processor or allow nested interrupts, as described in Section 2.4.

2.6 PROGRAMMABLE TIMER/EVENT COUNTERS

The programmable timer/event counters are 8-bit counters with a programmable prescaled clock source as shown in Figure 2-11. The TMS70X0 devices contain one timer/event counter and the TMS70X1 devices contain two timer/event counters. Timer 1, with its 8-bit capture latch, is available in all TMS7000 family members and is accessed at P2 and P3 of the peripheral file. Timer 2 is available only in the TMS70X1 family members and is accessed at P18 and P19 of the peripheral file (see Figure 2-12).

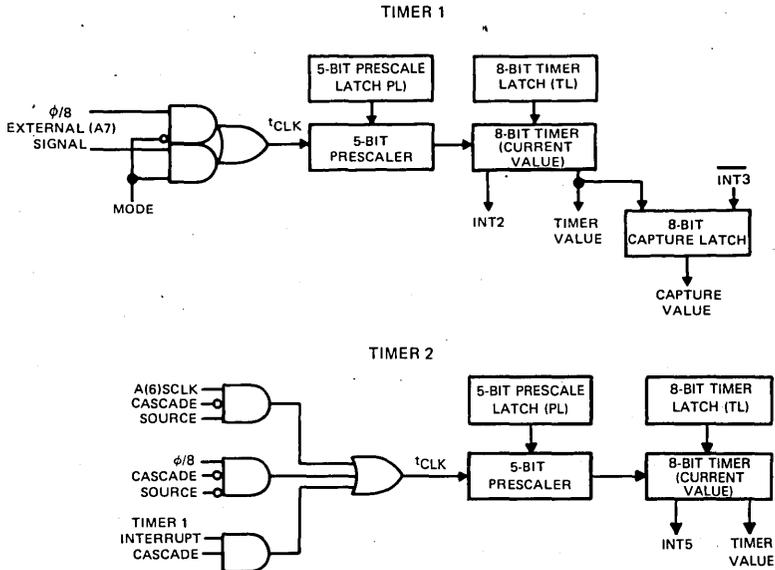


FIGURE 2-11 — PROGRAMMABLE TIMER/EVENT COUNTER

TIMER 1 DATA REGISTER - T1DATA

	7	6	5	4	3	2	1	0	
PF number: P2 Address: >0102	MSB							LSB	READ
	CURRENT TIMER VALUE								
	MSB							LSB	WRITE
TIMER LATCH VALUE (TL)									

TIMER 1 CONTROL REGISTER - T1CTRL

	7	6	5	4	3	2	1	0	
PF number: P3 Address: >0103	MSB							LSB	READ
	CAPTURE LATCH VALUE (CL)								
	START	SOURCE	IDLE	PRESCALE LATCH VALUE (PL)					WRITE

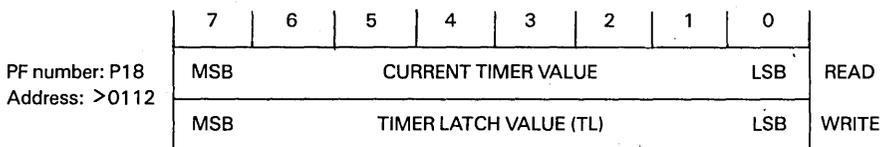
0 for all NMOS devices
 0 = Wake-up low power mode, 70CX0 only
 1 = Halt low power mode, 70CX0 only

1 = External clock source from pin A7
 0 = Internal clock source = $\phi/8$

1 = Start timer
 0 = Stop timer

FIGURE 2-12 — TIMERS 1 AND 2 DATA AND CONTROL REGISTERS

TIMER 2 DATA REGISTER - T2DATA



TIMER 2 CONTROL REGISTER - T2CTRL

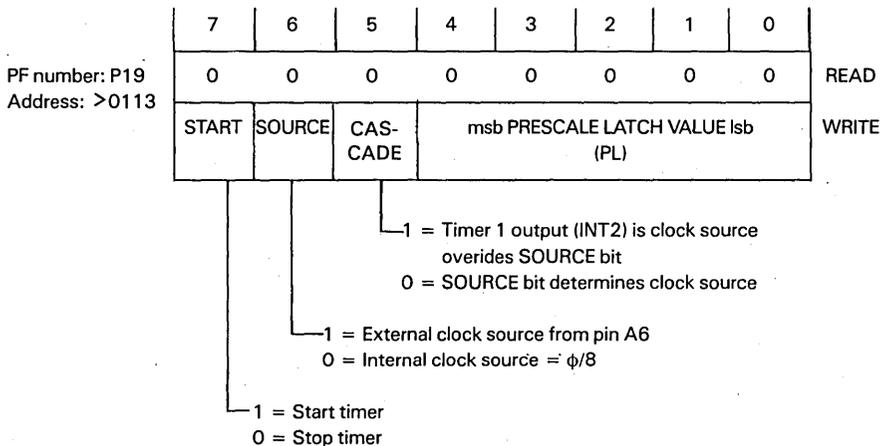


FIGURE 2-12 – TIMERS 1 AND 2 DATA AND CONTROL REGISTERS (CONTINUED)

The clock source and prescaling value of both timers are determined by the timer control registers (T1CTRL/T2CTRL). These control bits are write-only and therefore restrict timer control register manipulations to the following instructions:

```

MOV P    % >XX,Pn          STA  @>01XX
MOV P    A,Pn              STA  *Rn
MOV P    B,Pn              STA  @>01XX(B)
  
```

Where:

- >XX = Immediate 8-bit data value in hex
- >01XX = 16-Bit peripheral file address in hex
- A = A register
- B = B register
- Rn = General purpose register pair number
- Pn = Peripheral file register number

The same instructions are required for writing to the timer data registers.

The clock source of Timer 1 and Timer 2 is determined by bit 6 (SOURCE) of T1CTRL and T2CTRL respectively. A SOURCE bit of 0 selects the internally generated $\phi/8$ ($f_{OSC}/32$, /4 option or $f_{OSC}/16$, /2 option) clock and places the Timer/Event Counter in the Real Time Clock (RTC) mode. A SOURCE bit of 1 selects the external clock source and places the Timer/Event Counter in the Event Counter mode. In the external mode, the clock sources for Timers 1 and 2 are input on the two Most Significant Bits of I/O port A (A7) and (A6) respectively.

Bit 7 of the timer control registers is the START bit for the respective programmable timers. When a 0 is written to the START bit, the timer chain is disabled or frozen at the current count value. When a 1 is written to the START bit, regardless of whether it was a 0 or a 1 before, the prescaler and counter decremeters are loaded with the corresponding latch values, and the Timer/Event Counter operation begins. When the prescaler and counter decrement through zero together, an interrupt flag is set and the prescaler and counter decremeters are immediately and automatically reloaded with the corresponding latch values. The interrupt levels generated by the timers are INT2 for Timer 1 and INT5 for Timer 2. Timer 1 has a Capture Latch (CL) associated with it which "captures" the current value of the counter whenever INT3 is triggered. The capture latch will store the timer value even when INT3 is disabled. On the CMOS parts, the capture latch is disabled during the IDLE instruction.

2.6.1 Real Time Clock (RTC)

In the RTC mode, the internally generated $\phi/8$ ($f_{OSC}/32$, /4 option or $f_{OSC}/16$, /2 option) is the decremter clock source. Each positive pulse transition of the $\phi/8$ period signal decrements the count chain.

The RTC mode allows a program to periodically call a service routine, such as a display refresh, by simply setting the prescale latch value and the timer latch value so the routine is called at the desired frequency.

2.6.2 Event Counter (EC)

When Timer 1 or Timer 2 is in the EC mode, the counter functions as in the RTC mode except pin A7 and A6 of Port A are the decremter clock sources for Timer 1 and Timer 2 respectively. A positive edge transition on these external pins decrements the count chain. Note that this will allow INT2 and INT5 to function as a positive edge-triggered external interrupt by loading a start value of '0' into both the prescaler and timer latches. A positive transition on A7 or A6 will decrement the corresponding timer through zero and generate an INT2 or INT5. The EC mode can also be used as an externally provided RTC if the external clock is input to I/O pin A7. The maximum clock frequency on A7 or A6 in the EC mode must not be greater than $\phi/8$; or $f_{OSC}/32$, assuming the /4 clock option and $f_{OSC}/16$, assuming the /2 clock option. The minimum pulse width must not be less than 1.25 machine cycles ($1.25 \times \phi$) as shown in Section 4.

2.6.3 Timer and Prescaled Clock

The timer clock, whether internal or external, is prescaled by a 5-bit modulo-N counter. The prescaling value is determined by the least significant five bits of the timer control register. The actual prescaling value is equal to the timer control latch value plus one. Thus, a value of >88 , ($>80 + >8$ where >80 is the start bit and >8 is the prescale value) in the timer control latch would result in a $f_{OSC}/160$ clock output from the prescaler, assuming a /4 clock option.

An INT2 interrupt for Timer 1 or an INT5 interrupt for Timer 2 is momentarily pulsed when both the prescaler and counter decrement past the zero value together. This sets the INT2 or INT5 flag flip-flop, as described in Section 2.5.4. The prescaler and counter are then immediately

reloaded with the contents of the prescale latch (PL) and the timer latch (TL) and the timer will start decrementing with the new PL and TL value. The TL is loaded through the Timer 1 data register (T1DATA) for Timer 1 and the Timer 2 data register (T2DATA) is loaded into Timer 2. This value is write-only. When read, the timer data register contains the current value of the counter. The PL is loaded through the Timer 1 control register (T1CTRL) for Timer 1 and the Timer 2 control register (T2CTRL) loads into Timer 2. When read, the T1CTRL contains the Capture latch (CL) value and the T2CTRL contains all zeros.

2.6.4 Timer Interrupt Pulses

The period of the timer INT2 and INT5 interrupt pulses may be calculated by the following formula:

$$t_{INT} = t_{CLK} * (PL + 1) * (TL + 1)$$

where:

t_{INT} = period of timer interrupts

t_{CLK} = $8/\phi$ ($32/f_{OSC}$ on /4 option or $16/f_{OSC}$ on /2 option) for internal RTC mode or the period of input external EC mode

PL = Prescaler Latch value

TL = Timer Latch value

At the falling edge of the INT3 input, the Timer 1 value is loaded into the Capture Latch (CL). When read, the Timer 1 control register contains the CL value. This feature provides the capability to determine when an external event occurred relative to the internal timer.

NOTE

During the HALT mode of the CMOS version, the capture latch may not be loaded by INT3.

2.6.5 Timer 2

Timer 2 is only available on the TMS70X1 family devices (i.e. TMS7001, TMS7041, SE70P161). Timer 2 is similar to Timer 1 except that there is no Capture Latch associated with Timer 2, and INT5 is generated by Timer 2. In addition, T2CTRL also contains the CASCADE bit (bit 5). This bit is used in conjunction with T2CTRL SOURCE (bit 6) to determine the decrementing source of Timer 2.

A CASCADE bit of 1 selects the interrupt generated by Timer 1 (INT2) as the decrementing input to the prescaler of Timer 2. The CASCADE bit overrides the SOURCE bit, i.e., if the CASCADE bit is set to 1 the SOURCE bit of Timer 2 has no effect.

As with Timer 1, a SOURCE bit of 0 selects the internally generated $\phi/8$ ($f_{OSC}/32$, /4 option or $f_{OSC}/16$, /2 option), and places the the timer in the Real Time Clock (RTC) mode. A SOURCE bit of 1 selects the external clock source and places the Timer/Event Counter in the Event Counter (EC) mode.

The external EC input for Timer 2 is general purpose I/O pin A6/SCLK of Port A. A6/SCLK is also the I/O line (depending on mode of operation) for the baud rate generator clock (SCLK). Section 2.7.2 describes the SCLK signal.

Driving the external EC line for Timer 2 with the A6/SCLK produces the following modes:

- 1) With both SCLK and T2 external, the input signal drives the baud rate timer (T3) and Timer 2 (T2).
- 2) With SCLK external and T2 internal, the I/O bit (A6/SCLK) drives the baud rate timer (T3) and $\phi/8$ drives Timer 2's prescaler.
- 3) With SCLK and T2 internal, the A6/SCLK pin is the 1x baud rate output signal from T3 and the T2 source is $\phi/8$.
- 4) With SCLK internal and T2 external, A6/SCLK is the 1x baud rate signal from T3 and drives T2. In this mode, the baud rate timer and Timer 2 are cascaded, with the baud rate timer driving Timer 2. This is done by setting the CASCADE bit to 0 and the Timer 2 SOURCE bit to 1. Timer 2 can then be cascaded with either Timer 1 or the baud rate timer.

2.6.6 Pulse Width Measurement

Through the use of the Capture Latch (CL) the Timer/Event Counter can work with pulse width measurement applications. A simple exclusive OR-gate is all that is needed to set up the TMS7000 to handle a pulse width modulated input as shown in Figure 2-13. In software, the user outputs the inverted input pulse train through one of the output lines (BO in this case). This line is exclusive-ORed with the input data line resulting in an input to the INT3 pin. This causes the Capture Latch to be loaded with the current value of the timer at each transition of the input pulse train. The user program can then compare these values to determine width values.

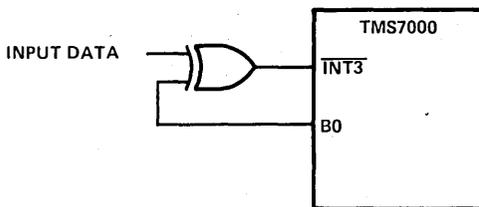


FIGURE 2-13 – PULSE WIDTH MEASUREMENT

2.6.7 Pulse Width Modulation (PWM) Theory of Operation

Pulse Width Modulation (PWM) involves the encoding of information in the width of a pulse. Information can be contained in the widths of these pulses when these pulses occur at a base frequency as shown in Figure 2-14.

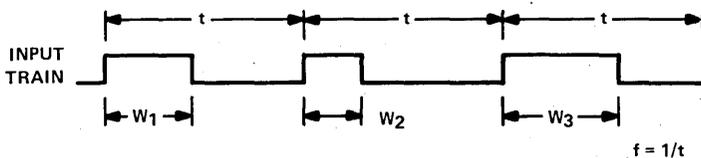


FIGURE 2-14 – PULSE WIDTH MODULATED PULSE TRAIN

Since the interrupts are only latched on a low level, a technique to give a low level at the beginning and end of a pulse is shown in Figure 2-15 which allows a simple timing program to measure the pulse width. This technique can be extended from PWM to any interval measurement application:

The TMS7000 is equipped to perform pulse measurement with the addition of a single exclusive OR-gate.

The edges of the PWM measurement are driven off of INT3 while the onboard counter times the event. The TMS7000 interrupt is structured so that the current value of the timer is captured at the CL (P3) on receipt of INT3. The actual time between events can then be derived from this captured value. The additional output BO is used to disable INT3 between successive edges of input train (Figure 2-15).

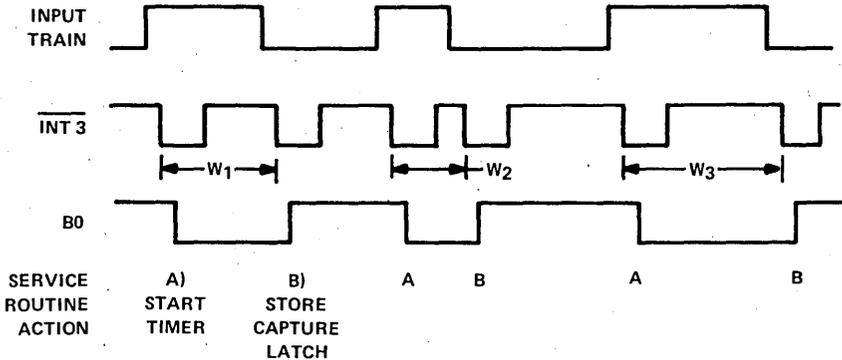


FIGURE 2-15 – TMS7000 PWM INT3 TIMING

The decoded data, now encoded in the interval between INT3s, is available on alternate interrupts at the Capture Latch (P3). A sample INT3 service routine is:

```

INT3   XORP   % >01,P6          TOGGLE B0
       DEC    R2                MARK YOUR PLACE
       .....
       BTJO   % >01,R2,RSTRT    JUMP OFF OF MARKER
       MOVPP P3,B              SAVE CAPTURE LATCH DATA
       RETI

RSTRT  MOVPP  % >80,P3          RESTART TIMER
       RETI
    
```

In this sample, R2 is used to keep the interval measurement on the proper portion of the pulse, and to flag the interrupt to the mainline program. Pin BO saves the Capture Latch data for the mainline program to interpret.

For long pulse widths, the prescale value can be adjusted to prevent the timer from rolling over before receiving an INT3. An alternate solution is to maintain a zero value of prescale, but use INT2 (the timer interrupt) to drive a software counter. A sample code is:

```

INT2 → ORP  % >08,PO          CLEAR INT2 FLAG
      INC   R4                INCREMENT UPPER STAGE
      RETI                   COUNTER

```

NOTE:

This sample code involves using the TMS7000 in a multi-interrupt environment. Care must be taken to ensure that a correct sequence of interrupts is performed. Multi-interrupt Pulse Width Modulation is described in the following paragraphs.

2.6.8 Multi-Interrupt Pulse Width Modulation (PWM)

A simultaneous interrupt occurs when the INT3 service routine is delayed due to the receipt of a higher priority INT2 at the same time.

For example, when the user is operating the INT2 timer at high resolution (low value of prescale) to time intervals between successive INT3 events, the INT2 service routine increments a software controlled RAM byte. This byte serves as an upperstage byte for the timer, so the high resolution offered by a low value of prescale can be maintained.

However, when both interrupts occur within an instruction cycle, one of the two sequences shown in Figures 2-16 and 2-17 has occurred.

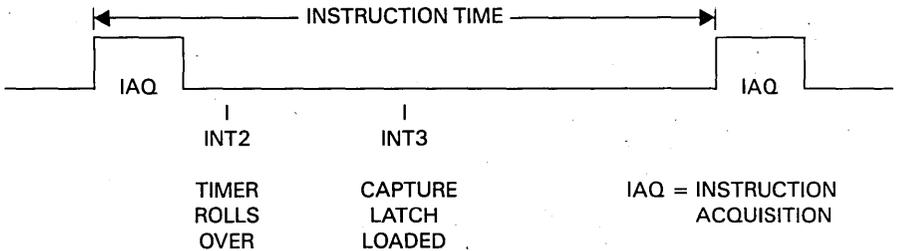


FIGURE 2-16 — SIMULTANEOUS INTERRUPTS, INT2 PRECEDING

In the first sequence, if INT2 precedes INT3 within an instruction boundary, the receipt of INT2 implies that the timer has rolled over and its latch value (>FF) is reloaded into the current timer register. The current timer value was captured upon receipt of the interrupt (3). The INT2 service routine increments the software (RAM) counter and exits. The INT3 is then immediately serviced as the current timer value was captured upon receipt of the interrupt (3). The service routine reads the capture latch value, and a correct interval may be deduced from this capture value and the software upperstage counter value.

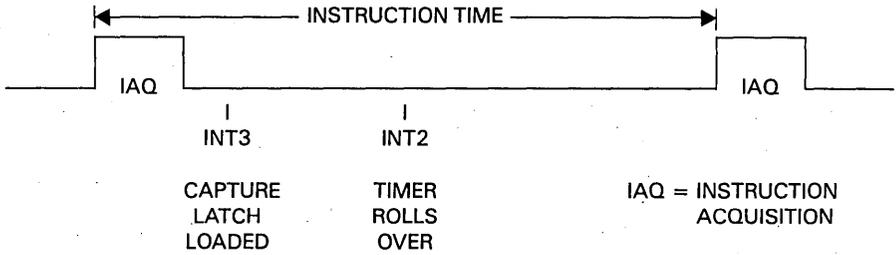


FIGURE 2-17 — SIMULTANEOUS INTERRUPTS, INT3 PRECEDING

The second sequence that can occur is when INT3 precedes INT2 within an instruction boundary. As in the previous case, INT2 is serviced first. However, the current timer value is captured by hardware when INT3 comes in, before actual servicing of INT3. INT2 has not yet occurred and the hardware has therefore captured a timer value that has not rolled over. This timer value is likely to be near or at >00. The INT2 service routine, if it does not check for this condition (by testing the most significant bit (MSB) of the timer for rollover) will increment the upper stage of software by default and will cause an incorrect value to be assumed for the interval. This condition occurs because the timer (implemented in hardware) and the program's upperstage counter (software driven) are out of sync.

The following code will correct the situation.

```

INT2   BTJZP   % >20,P0,OKAY   CHECK FOR PENDING INT2
       BTJO    % >80,P3,OKAY   CHECK CAPTURED VALUE
       JMP     RET3
OKAY   INC
       .....
RET3   RETI

```

INT3 then becomes:

```

INT3   XORP    % >01,P6       TOGGLE BO
       DEC     R2             MARK YOUR PLACE
       .....
       BJTO   % >01,R2,RSTRT  JUMP OFF OF MARKER
       MOVP   P3,B           SAVE CAPTURE LATCH DATA
       RETI
RSTRT  MOVP    % >80,P3       RESTART TIMER
       CLR    R4             RESET SOFTWARE UPPER STAGE
       RETI

```

2.7 SERIAL PORT (TMS70X1 VERSIONS ONLY)

2.7.1 Description

The TMS70X1 contains a serial port which greatly enhances its I/O and communication capability. It is not available in the TMS70X0 versions of the TMS7000 family. The serial port can operate in several modes which let the TMS70X1 interface with Universal Asynchronous Receiver/Transmitter (UART) peripheral devices, as well as multiple microcomputers (TMS70X1, MC6801, I8051). These serial links are implemented using standard asynchronous protocols. These multiprocessor protocols, described in Section 2.7.3, are compatible with those used by the Motorola MC6801 and Intel I8051.

A second mode, isosynchronous, permits very high transmission rates.*

The third mode, a serial I/O mode, can be used to expand I/O lines using external shift registers, and to communicate with peripheral devices requiring a non-UART serial input (e.g. display drivers).

Including a hardware serial port on-chip saves ROM code and allows much higher transmission rates than could be achieved in software. The full-duplex serial port has a double buffered transmitter and receiver.

The serial port consists of a receiver (RX), transmitter (TX), and Timer 3 (T3). The complete functional definition of the serial port is programmed by the TMS70X1 software. A set of control words must first be sent out to the serial port to initialize it, so that it supports the desired communications format. These control words will determine the baud rate, character length, even/odd/off parity, number of stop bits, etc.

The serial port is controlled and accessed through registers in the peripheral file. The registers associated with the serial port are:

TABLE 2-8 — SERIAL PORT CONTROL REGISTERS

REGISTER	NAME	TYPE	FUNCTION
P17	SMODE	FIRST WRITE	Serial Port Mode
P17	SCTLO	WRITE	Serial Port Control-0
P17	SSTAT	READ	Serial Port Status
P20	T3DATA	R/W	Timer 3 Data
P21	SCTL1	R/W	Serial Port Control-1
P22	RXBUF	READ	Receiver Buffer
P23	TXBUF	WRITE	Transmission Buffer

The SMODE register is the receive/transmit (RX/TX) write-only control register. The SCTLO and SSTAT are the RX/TX write-only control register and read-only status register, respectively. These registers are all accessed through P17. The first write after a hardware or serial port reset accesses SMODE (See Section 2.7.5.1). All subsequent writes access SCTLO. These registers are common to both RX and TX, and both RX and TX will have the same mode and frame format.

* Isosynchronous is the term given to this second communication mode of the serial port. This mode has the same frame format as the asynchronous mode, but uses only one serial clock (SCLK) cycle per data bit as opposed to 16 SCLKs per data bit for the asynchronous mode. This allows transmission rates 16 times those of the asynchronous mode.

The T3DATA register accesses the Timer 3 8-bit timer. It is similar to T1DATA and T2DATA. The SCTL1 register is a read/write control register for the RX/TX and Timer 3.

The RXBUF is a read-only register containing data from the RX. The RXBUF is double buffered with the internal shift register (RXSHF) so that the TMS70X1 CPU has at least a full frame to read the received data before the RX may overwrite it with new data.

The TXBUF is a write-only register from which the TX takes the data it transmits. It is double buffered with the TX shift register (TXSHF), so that the TMS70X1 CPU has a full frame to write new data before TXBUF becomes empty.

Figure 2-18 is a block diagram of the serial port registers and functional blocks. Figure 2-19 illustrates serial port I/O logic. Section 2.7.5 describes serial port registers in detail.

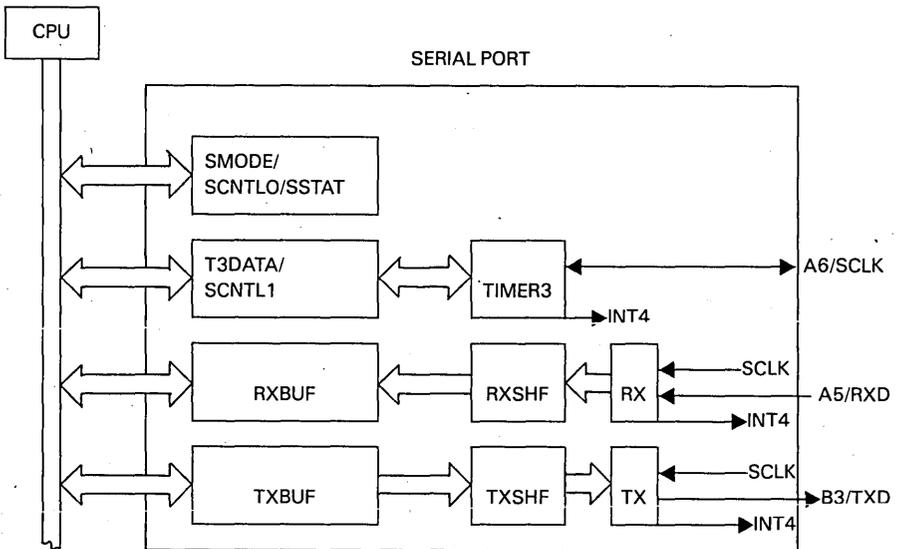
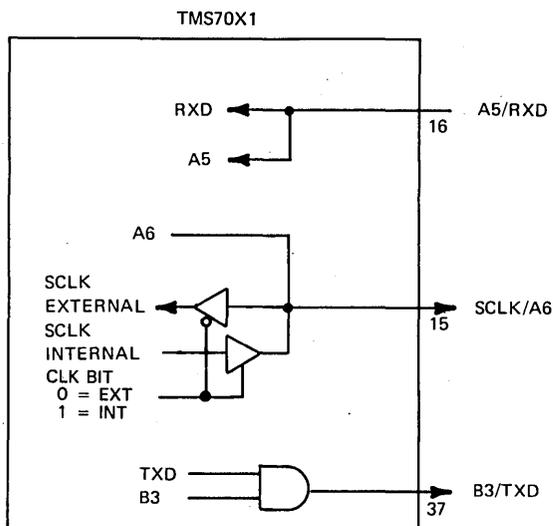


FIGURE 2-18 — SERIAL PORT FUNCTIONAL BLOCKS



The TXD and RXD lines use I/O lines B3 and A5 respectively. This configuration allows the TXD and RXD pins to be used as I/O pins if desired. If serial port transmission is disabled, then TXD follows B3. If reception is disabled, then no receiver interrupts occur and A5 is an input bit.

2.7.2 Clock Sources and Serial Port Modes

The serial port can be driven by an internal (Timer 3) or external baud rate generator. The source of the serial clock (SCLK) is determined by the clock (CLK) bit, SCTL1(6) (See Section 2.7.5). An external clock source is input on the high impedance A6/SCLK line. An internal clock source is output on the low impedance A6/SCLK line, being derived from Timer 3 via a $\phi/2$ clock ($f_{osc}/8$ for /4 option, $f_{osc}/4$ for /2 option) as shown in Figure 2-19. The internally generated SCLK has a 50% duty cycle. The current value of SCLK (internal or external) can be determined by reading A6/SCLK. The RX receives data on the rising SCLK edges and the TX transmits data on the falling SCLK edges.

The RX/TX has three communication modes: asynchronous, isosynchronous, and serial I/O. The serial I/O mode is used to link the serial port to shift registers for simple I/O expansion. The isosynchronous and asynchronous communication modes are used to link to other synchronous and asynchronous devices. These two mode also have extra features for two formats of multiprocessor communication. In all modes I/O is NRZ (non-return to zero) format, i.e. data value 1 = high level, and data value 0 = low level.

2.7.2.1 Asynchronous Communication Mode

When the serial port is operating in the asynchronous communication mode, the frame format consists of a start bit, five to eight data bits, even/odd/no parity, and one or two stop bits. The bit period is 16 times the SCLK period.

RX operation is initiated by reception of a valid start bit, which consists of a negative edge (1 and then 0 in adjacent SCLK periods) followed by taking a majority vote of three samples where 2 of the samples must be zero. These samples occur seven, eight, and nine SCLK periods after the negative edge. This sequence provides false start bit rejection and also locates the center of bits in the frame, where the bits will be read on a majority basis. Figure 2-20 illustrates the asynchronous communication format, with a start bit showing how edges are found and majority vote taken.

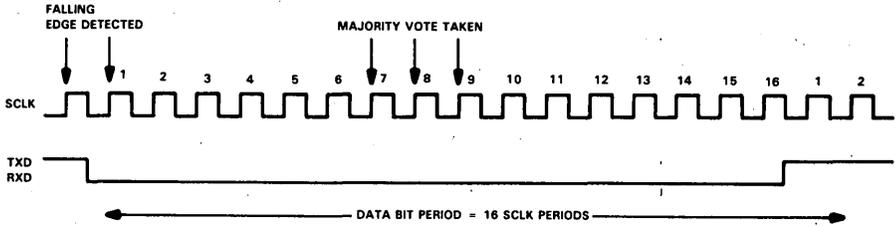


FIGURE 2-20 — ASYNCHRONOUS COMMUNICATION FORMAT

Since the RX synchronizes itself to frames, the external transmitting and receiving devices do not have to use the same SCLK; it may be generated locally. If the internal SCLK is used it will be output continuously on pin A6.

2.7.2.2 Isosynchronous Communication Mode

In this mode, the frame format consists of a start bit, five to eight data bits, even/odd/no parity, and one or two stop bits. The bit period equals the SCLK period. RX operation is initiated by reception of a valid start bit, which consists of a negative edge. Bits are read on a single value basis. Since the RX does not synchronize itself to data bits the transmitter and receiver must be supplied with a common SCLK. If the internal SCLK is used it is output continuously on pin A6/SCLK. Figure 2-21 illustrates the isosynchronous communication format, with a complete frame consisting of a start bit, six data bits, even parity, and two stop bits.

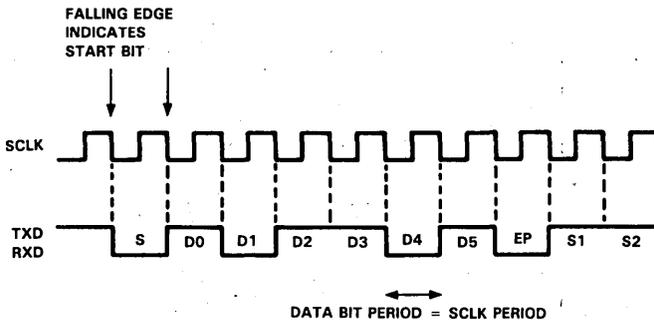


FIGURE 2-21 — ISOSYNCHRONOUS COMMUNICATION FORMAT

In both the asynchronous and isosynchronous communication modes, when a frame is fully received, RXBUF is loaded from RXSHF, RXRDY and INT4 FLG are set to 1, and the error status bits are set accordingly. RXRDY is reset to 0 when the CPU reads RXBUF.

Transmission is initiated after the CPU writes to TXBUF. This sets TXRDY to 0. Once TXSHF is empty, it is loaded from TXBUF, setting TXRDY and INT4 FLG to 1. Upon completion of the transmission, TXSHF will reload if TXBUF is full; if not the TX will idle and TXE will be 1 until TXBUF is written to.

2.7.2.3 Serial I/O Communication Mode

When the serial I/O mode is in operation, the frame format is five to eight data bits and one stop bit, with no corresponding clock edge for the stop bit. The clock does not send pulses during the stop bits. The bit period is equal to the SCLK period. TX operation is initiated by writing to TXBUF, when TXRDY equals 1. RX operation is initiated by writing a 1 to the RXEN bit. Figure 2-22 illustrates the serial I/O format for two back to back frames, each containing five data bits.

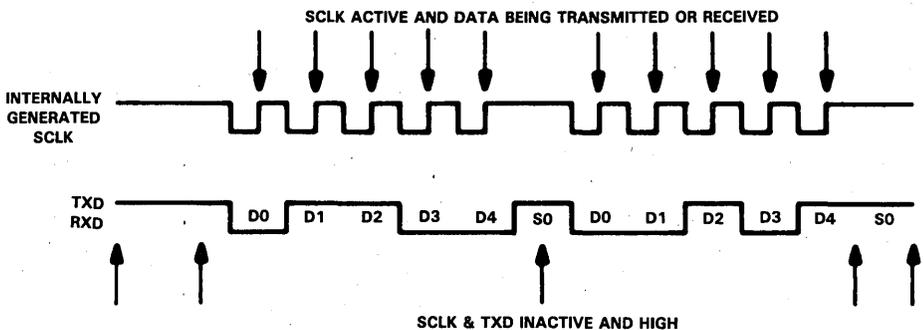


FIGURE 2-22 – SERIAL I/O COMMUNICATION FORMAT

An internal SCLK source will be output on pin A6/SCLK. In the serial I/O mode, SCLK is gated on pin A6/SCLK and will only be active when data is being transmitted or received; otherwise, pin A6/SCLK will have a one value. An external SCLK may be selected and will drive the serial port. However, this clock mode will be useless since there is no on-chip method to generate a gated SCLK to drive the external shift registers.

2.7.3 Multiprocessor Communication

When the serial port is in either the asynchronous or isosynchronous communications mode, the multiprocessor communication formats are available. These formats are used to transfer information between many microcomputers on the same serial link. Information is transferred as a block of frames from a particular source to some destination(s). The TMS70X1 has features to identify the start of blocks, and suppress interrupts and status information from the RX until a block start is identified.

In both multiprocessor modes the sequence is as follows: the serial port wakes up at the start of a block and the TMS70X1 CPU reads the first few frames (containing a destination address). If the block is addressed to the microcomputer the CPU reads the rest of the block; if not it puts the serial port to sleep again and therefore will not receive serial port interrupts until the next block start.

In order to provide more flexibility, the TMS70X1 implements two multiprocessor protocols, one supported by Motorola and the other by Intel. These protocols are described in the following paragraphs. The Motorola protocol is compatible with the Motorola MC6801 processor mode and the Intel protocol is compatible with the Intel protocol for the 8051. The mode of TMS70X1 multiprocessor protocol is software selectable via the MULTI bit in the SMODE register (see section 2.7.5). Both formats use the WU and SLEEP flags to control the TX and RX features of these modes.

Because the Intel multiprocessor mode contains an extra address/data bit, it is not as efficient as the Motorola mode in handling large blocks (over 10 bytes) of data. The Intel mode on the other hand, is more efficient in handling many small blocks of data because it does not have to wait in between blocks of data as does the Motorola mode.

2.7.3.1 *Motorola (MC6801) Protocol*

In this protocol, blocks are distinguished by having a longer idle time between the blocks than between frames in the blocks. An idle time 10 bits or more after a frame indicates the start of a new block.

In the Motorola mode of multiprocessor communications, the processor wakes up (serial port resets the SLEEP bit to 0) after the block start signal. The processor will now recognize the next serial port interrupt. The user's service routine then receives the address sent out by the transmitter and compares this address to its own. If the CPU is addressed, the service routine will not set the SLEEP bit, and receive the rest of the block. If the CPU is not addressed, the service routine sets the SLEEP bit (in software) to a 1. This lets the CPU continue to execute its main program without being interrupted by the serial port. The serial port will set the SLEEP bit to 0 whenever it detects a block start signal.

There are two ways to send a block start signal. The first is to deliberately leave an idle time of 10 bits or more by delaying the time between the transmission of the last frame of data in the previous block and the address frame of the new block. In the second way, the TMS70X1 implements a more efficient method of sending a block start signal. Using the wake up (WU) bit, an idle time of exactly one frame (timed by the serial port) can be sent. The serial communications line is therefore not idle any longer than necessary.

Associated with the WU bit is the wake up temporary (WUT) flag. WUT is an internal flag, double buffered with WU. When TXSHF is loaded from TXBUF, WUT is loaded from WU and WU is reset to 0. This configuration is shown in Figure 2-23.

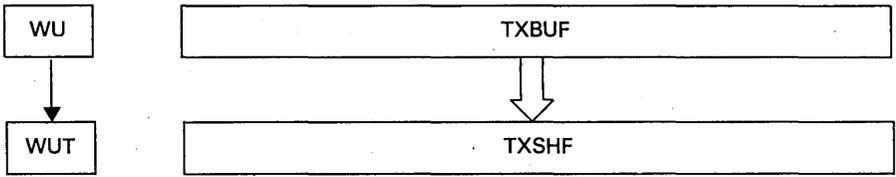


FIGURE 2-23 — DOUBLE BUFFERED WUT AND TXSHF

Sending out a block start signal of exactly one frame time is accomplished as follows:

A 1 must first be written to the WU bit. Then a data word (don't care) must be written to the TXBUF. When the TXSHF is free again, the contents of the TXBUF are shifted to the TXSHF, and the WU value is shifted to WUT. If the WU bit had been set to a 1, the start, data, and parity bits will be suppressed and an idle period of one frame, timed by the serial port, will be transmitted. The next data word, shifted out of the serial port after the block start signal, will be the second data word written to the TXBUF after writing a 1 to the WU bit. The first data word written is suppressed while the block start signal is sent out, and ignored after that.

However, writing the first don't care data word to the TXBUF is necessary so the WU bit value can be shifted to WUT. After the don't care data word is shifted to the TXSHF, the TXBUF (and WU if necessary) may be written to again, since WUT and TXSHF are both double buffered.

Although the RX still operates when the SLEEP bit is 1, it will not set RXRDY, INT4 FLG, or the error status bits to 1. The RX will set the SLEEP bit to 0 if it times an appropriate 10 bit idle time on RXD. The Motorola multiprocessor communication format is shown in Figure 2-24.

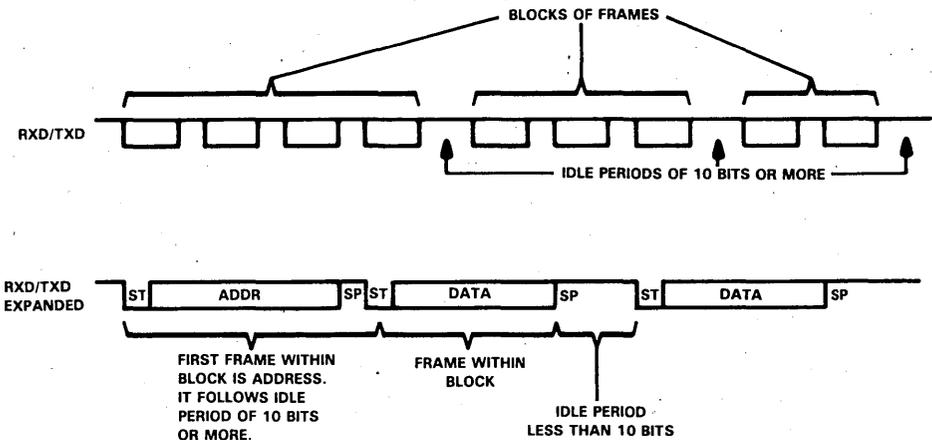


FIGURE 2-24 — MOTOROLA MULTIPROCESSOR COMMUNICATION FORMAT

2.7.3.2 Intel (I8051) Protocol

In the Intel protocol, the frame has an extra or address bit just before the parity bit. Blocks are distinguished by the first frame(s) in the block with the address bit set to 1, and all other frames with the address bit set to 0. The idle period timing is irrelevant.

The WU bit is used to set the address bit. In the TX, when the TXBUF and WU are loaded into the TXSHF and WUT, WU is reset to 0 and WUT is the value of the address bit of the current frame. Thus, to send an address, the WU bit must be set to a 1, and the appropriate address value should then be written to the TXBUF. When this address value is transferred to the TXSHF and shifted out, its address bit will be sent as a 1, which flags the other processors on the serial link to read the address. Since the TXSHF and WUT are both double buffered, the TXBUF and WU may be written to immediately after TXSHF and WUT are loaded. To transmit non-address frames in the block, the WU bit must be left at 0.

On the serial link, all processors set their SLEEP bit to 1 so that they will only be interrupted when the address bit in the data stream is a 1. When the processors receive the address of the current block, they compare it to their own addresses and those processors which are addressed set their SLEEP bit to a 0, so that they will read the rest of the block.

Though the RX still operates when the SLEEP bit is 1, it will not set RXRDY, INT4-FLG, or the error status bits to 1 unless the address bit in the received frame is a 1. The RX does not alter the SLEEP bit: this must be done in software. The Intel multiprocessor communication format is shown in Figure 2-25.

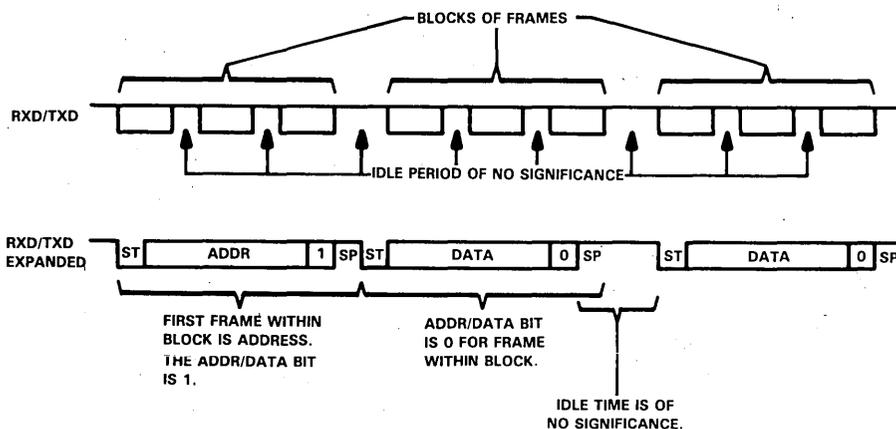


FIGURE 2-25 — INTEL MULTIPROCESSOR COMMUNICATION FORMAT

2.7.4 Timer 3

Timer 3 is a simplified version of Timer 1 and 2 and, like Timer 2, is only available on the TMS70X1 versions of the TMS7000 family. Figure 2-26 is a block diagram of Timer 3.

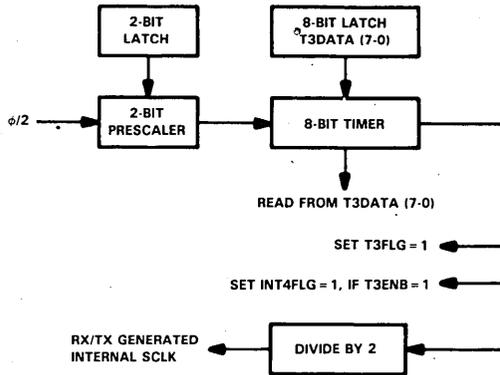


FIGURE 2-26 – TIMER 3 BLOCK DIAGRAM

Timer 3 is accessed through T3DATA (similar to T1DATA and T2DATA), and SCTL1 (shared with RX/TX functions). The clock source for Timer 3 is internal only, and has a frequency of $\phi/2$. Timer 3 is a free running counter and is updated with new timer values when it decrements through zero.

Timer 3 consists of a 2-bit prescaler and an 8-bit timer. Both the prescaler and the timer are reloaded from 2-bit and 8-bit latches respectively, when they decrement through zero. The latches are write only, but the 8-bit counter can be read.

The Timer 3 output goes to the serial port via a divide by two circuit, producing an equal mark-space ratio internal SCLK. The baud rate generated by Timer 3 is user programmable and is determined by the value of the 2-bit prescaler and the 8-bit timer latch. The equations for determining the baud rates for both the asynchronous and isosynchronous modes are as follows:

$$\text{Asynchronous Baud Rate} = \frac{\phi}{64 \times (\text{PL} + 1) \times (\text{TL} + 1)}$$

$$\text{Isosynchronous Baud Rate} = \frac{\phi}{4 \times (\text{PL} + 1) \times (\text{TL} + 1)}$$

where:

- f_{osc} = crystal frequency
- ϕ = internal machine clock frequency
(either 1/4 or 1/2 of f_{osc} depending on clock choice)
- PL = Timer 3 prescale latch value
- TL = Timer 3 latch value

For example, to program the serial port to operate at 300 baud in the asynchronous mode (with $\phi = 2.5$ MHz), the prescaler value is set to 0 and the latch value set to 129 decimal, or >81.

The Timer 3 output always sets T3FLG to 1, and sets INT4 FLG to 1 if T3ENB is a 1 when the timer and prescaler decrement through 0. This allows Timer 3 to be used as a utility timer if it is not used by the serial port. Timer 3 and its flags are not affected by the serial port software reset, UR. Therefore, Timer 3 may be used independently of the serial port.

2.7.5 Serial Port Registers

2.7.5.1 Mode Register (SMODE)

SMODE (see Figure 2-27) is a write-only register and is accessed through P17 in the peripheral file. It describes the character format and type of communications mode (asynchronous or isosynchronous). SMODE is only accessible after a hardware reset or after resetting the UART through the UR bit. It must be the first register written to in the serial port immediately following a reset. After writing to the SMODE register, it cannot be accessed without first performing a reset operation. The first operation to location P17 in the peripheral file, immediately following a reset, will access the SMODE register. All subsequent writes to P17 will access the control register (SCTL0).

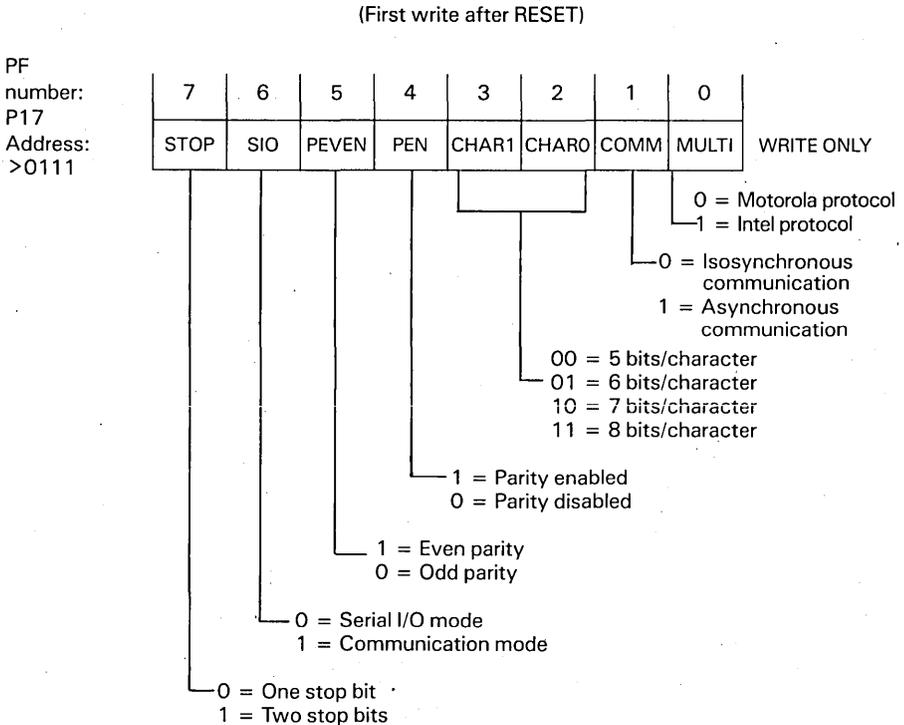


FIGURE 2-27 – SERIAL MODE REGISTER - SMODE

NOTE

If the serial port is configured so that some features are irrelevant, then the corresponding flags are don't care. For example, when configured in the serial I/O mode, bits 7, 4, 1, and 0 are Don't Cares.

MULTIPROCESSOR MODE (MULTI) BIT 0:

There are two possible multiprocessor protocols, the Motorola and the Intel. Both are described in Section 2.7.3. Setting this bit to a 0 selects the Motorola protocol; setting it to a 1 selects the Intel protocol. The multiprocessor communication is different from the other communication modes in that the multiprocessor mode uses the Wake-Up and the Sleep functions.

COMMUNICATIONS MODE (COMM) BIT 1:

This bit determines the serial port mode of communication. Setting the bit to 1 selects the asynchronous mode. In this mode the bit period is 16 times the SCLK period and bits are read on a two out of three vote basis. Setting the bit to 0 selects the isosynchronous mode. In this mode, the bit period is equal to the SCLK period and bits are read on a single value basis. These modes of operation are described in section 2.7.2.

NUMBER OF BITS PER CHARACTER (CHAR1 ,CHAR0) BITS 2,3:

Characters are programmable to 5, 6, 7 or 8 bits. Characters of less than 8 bits are right-justified in RXBUF and TXBUF. Characters of less than 8 bits are padded with leading zeros in the RXBUF. The unused leading bits in the TXBUF may be written as don't care values. The RXBUF and TXBUF register formats are given in sections 2.7.5.6 and 2.7.5.7.

PARITY ENABLE (PEN) BIT 4:

If parity is disabled then no parity bit is generated during transmission or expected during reception. A received parity bit is not transferred to the RXBUF with the received data as it is not considered one of the data bits when programming the character field.

PARITY EVEN (PEVEN) BIT 5:

If PEN is set, then this bit defines odd or even parity according to the number of odd or even 1 bits in both transmitted and received characters.

SERIAL I/O OR COMMUNICATION MODE (SIO) BIT 6:

This bit determines whether the serial port operates in the serial I/O mode or one of the communication modes. Setting this bit to a 0 sets the serial port in the serial I/O mode. Deletion of the start and stop bits, in conjunction with an internal 1x clock, allows ease of I/O expansion by use of external shift registers. Setting this bit to a 1 selects the communication mode. When this bit is set to 1 the COMM bit determines whether the serial port is in the asynchronous or isosynchronous mode.

NUMBER OF STOP BITS (STOP) BIT 7:

This bit determines the number of stop bits sent when the serial port is in one of the communication modes. Setting this bit to a 0 selects one stop bit, and setting it to a 1 selects two stop bits. The receiver checks for one stop bit only.

2.7.5.2 Serial Control 0 Register (SCTLO)

SCTLO (see Figure 2-28) is a write-only register, and is accessed through P17 of the peripheral file. The SCTLO register is used to control the serial port functions, such as transmit and receive enable, clearing of error flags and software reset. After a hardware or software reset, the SMODE register must be written to before accessing the SCTLO register, since the SMODE and SCTLO registers are accessed through the same location. Any subsequent writes to this register location (P17) will load the SCTLO register. SCTLO is cleared by a reset (hardware or software).

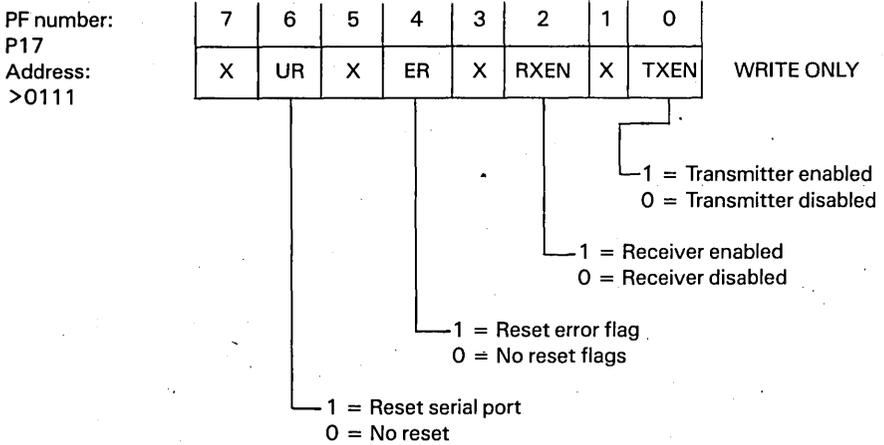


FIGURE 2-28 – SERIAL CONTROL 0 REGISTER - SCTLO

TRANSMIT ENABLE (TXEN) BIT 0:

Data transmission through TXD cannot take place unless this bit is set to a 1. When resetting to a 0, the transmission is not halted until all the data previously written to TXBUF has been sent. TXEN is set to 0 by a reset (hardware or software).

RECEIVE ENABLE (RXEN) BIT 2:

In the communication modes (asynchronous and isosynchronous) setting the RXEN bit to 1 allows RX to set INT4 FLG, and enable RXRDY. When reset to 0, this bit prevents received characters from being transferred into the receiver buffer, and no RXRDY interrupt is generated. However, the receiver shift register continues to assemble characters. Thus, if RXEN is set partially through reception of a character, it will be transferred complete into RXBUF. In the serial I/O mode writing a 1 to RXEN initiates RX operation. If an internally generated SCLK is used, a gated SCLK at pin A6 is enabled. When the entire frame is received, RX disables SCLK and sets RXRDY and INT4 FLG to a 1, and RXEN to 0. RXEN has no direct effect on RXRDY or INT4 FLG in this mode. RXEN is set to 0 by UR.

ERROR RESET (ER) BIT 4:

Writing a 1 to this bit clears all three error flags in the SSTAT register (PE, OE, FE). Writing a 0 has no effect.

SOFTWARE UART RESET (UR) BIT 6:

Writing a 1 to this bit puts the serial port in the reset condition, and enables the SMODE register for initialization. A6/SCLK is put in the high impedance state (input), the TXD signal is held at 1, so that the B3 pin may be used as a general purpose output line (see Figure 2-19). Until a 0 is written to UR, all affected logic is held in the reset state. UR must be set to 0 before the TMS70X1 CPU can write a 1 to CLK and output SCLK on Port A. UR is set to 1 by reset (hardware). The UART Reset affects only the items above and it is not a general device reset like the RESET pin.

2.7.5.3 Serial Port Status Register (SSTAT)

This status register (see Figure 2-29) is a read-only register and is accessed through P17 of the Peripheral File. It is used for determining the status of the serial port. Bits 0, 1, and 6 of this register are cleared by a reset (hardware or software).

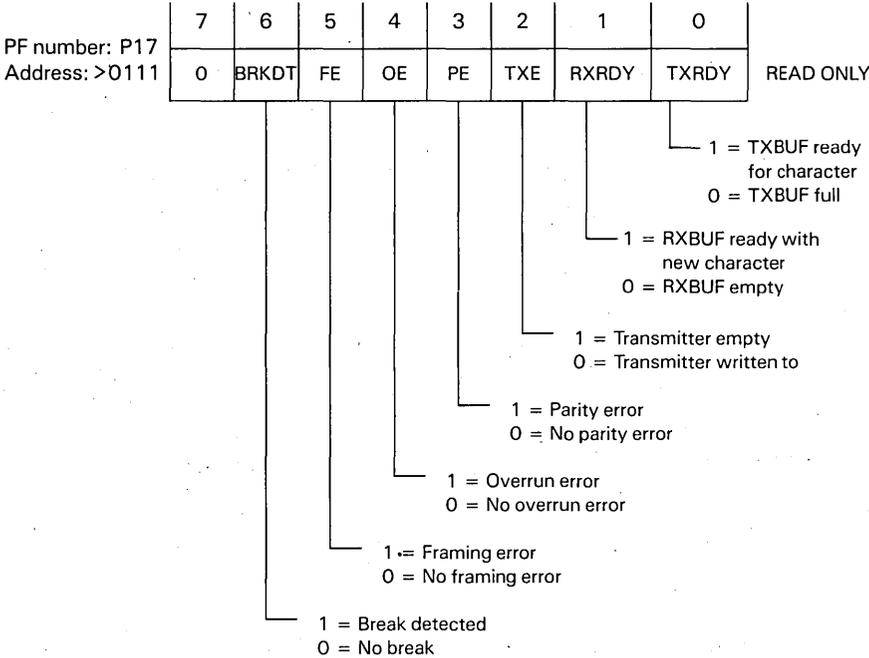


FIGURE 2-29 — SERIAL PORT STATUS REGISTER - SSTAT

TRANSMITTER READY (TXRDY) BIT 0:

The TXRDY bit is set by the transmitter to indicate that TXBUF is ready to receive another character, and is automatically reset when a character is loaded. The serial port interrupt (INT4) is issued at the same time (if enabled) the TXRDY bit is set. This bit is set to 1 by UR.

RECEIVER READY (RXRDY) BIT 1:

This bit is set by the receiver to indicate that RXBUF is ready with a new character, and is automatically reset when the character is read out. The serial port interrupt (if enabled) is issued at the same time the RXRDY bit is set. RXRDY is set to 0 by UR.

TRANSMITTER EMPTY (TXE) BIT 2:

The TXE bit is set to 1 when the transmitter shift register and TXBUF are empty, and reset to 0 when the TXBUF is written to. This bit is set to 1 by UR.

PARITY ERROR (PE) BIT 3:

PE is set when a character is received with a mismatch between the number of 1s and its parity bit. This bit is reset by the ER bit in SCTL0.

OVERRUN ERROR (OE) BIT 4:

The overrun error bit is set when a character is transferred into RXBUF before the previous character has been read out. The previous character is overwritten and lost. OE is reset by the ER bit in SCTL0.

BREAK DETECT (BRKDT) BIT 6:

The BRKDT bit will show that a break condition has occurred. BRKDT is set if the RXD line remains continuously low for 10 bits or more, starting from the end of a frame (stop bit). When the break ends, BRKDT is set to a 0 immediately. In the serial I/O mode BRKDT remains a 0. This bit is reset to 0 by UR. A break is generated by setting Port B bit 3 low. Setting B port bit 3 high again resumes operation of the TXD line.

Figure 2-16, Serial Port I/O Logic, shows how the TXD and RXD lines are multiplexed on I/O lines B3 and A5 respectively. This configuration allows the TXD and RXD pins to be used as I/O pins if desired. If transmission is disabled, then TXD follows B3. If reception is disabled, then no receiver interrupts occur and A5 is an input bit.

FRAMING ERROR (FE) BIT 5:

The FE bit is set when a character is received with a 0 stop bit. The stop bit indicates that synchronization with the start bit has been lost and the character is incorrectly framed. FE is reset by the ER bit in SCTL0.

2.7.5.4 Serial Control 1 Register (SCTL1)

The SCTL1 (see Figure 2-30) is a read/write register and is accessed through P21 in the peripheral file. This register is used to control the source of SCLK, multiprocessor communications, Timer 3 interrupt, and the Timer 3 prescaler value.

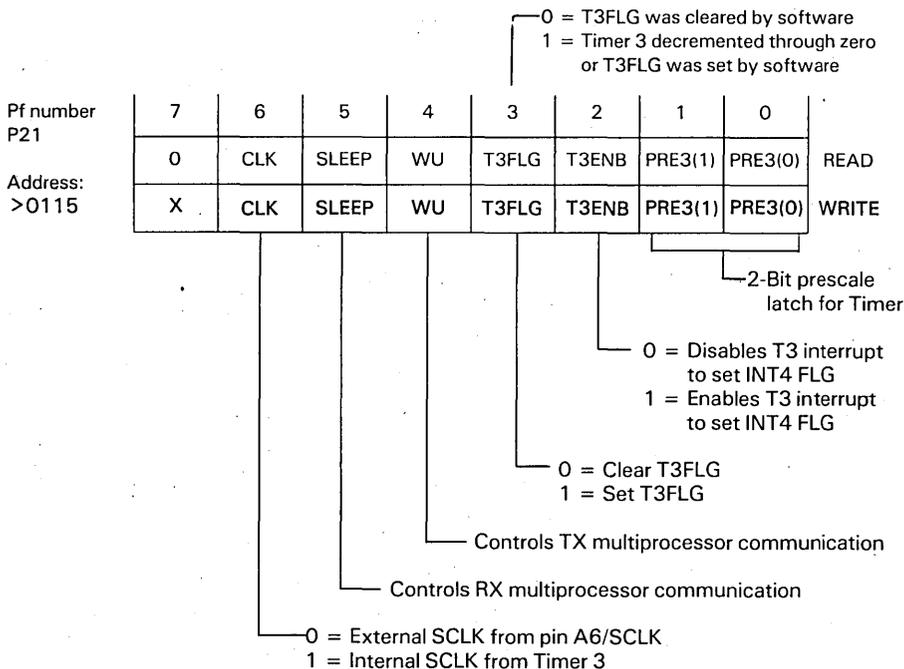


FIGURE 2-30 – SERIAL CONTROL 1 REGISTER - SCTL1

TIMER 3 PRESCALE LATCH (PRE3(1), PRE3(0)) BITS 0,1:

These bits act as the prescale bits for Timer 3. The internal clock input to the Timer 3 is either $f_{osc}/4$, $/8$, $/16$, or $/32$ ($/2$ option) or $f_{osc}/8$, $/16$, $/32$, or $/64$ ($/4$ option) depending on the setting of these bits. The output of timer 3 divided by 2 is the actual baud rate for the isosynchronous mode or divided by 32 for the asynchronous mode.

TIMER 3 INTERRUPT ENABLE (T3ENB) BIT 2:

When T3ENB is set to a 1, Timer 3 will set INT4FLG to a 1 when it sets T3FLG to a 1. T3ENB is reset to 0 by a hardware reset, but not by UR. This allows Timer 3 to operate independently of the serial port.

TIMER 3 INTERRUPT FLAG (T3FLG) BIT 3:

The T3FLG bit is set to a 1 when both the Timer 3 prescaler and Timer 3 decrement through zero together. T3FLG indicates that Timer 3 was the source of the serial port interrupt. T3FLG must be cleared by software in the T3 interrupt service routine, since it is not cleared when the INT4 vector is fetched by the CPU. This bit is also reset to 0 by a hardware reset, but not by UR. This allows Timer 3 to operate independently of the serial port.

WAKE UP (WU) BIT 4:

The WU bit controls the TX features of the multiprocessor modes (Section 2.7.3). WU is reset to 0 by UR and cannot be set again until UR is cleared.

SLEEP (SLEEP) BIT 5:

The SLEEP bit is used to control the RX features of the multiprocessor modes (Section 2.7.3). This bit is reset to 0 by UR.

SERIAL CLOCK SOURCE (CLK) BIT 6:

The CLK determines the source of SCLK. Setting this bit to a 0 selects an external SCLK, which is input on the high impedance A6/SCLK line. Setting it to a 1 selects an internal SCLK, derived from Timer 3. This signal is output on the low impedance A6/SCLK line. The CLK bit is reset to 0 by UR and cannot be set again until UR is cleared.

2.7.5.5 Timer 3 Data Register

The Timer 3 Data register - T3DATA (see Figure 2-31) is a read/write register and is accessed through P20 in the Peripheral File.

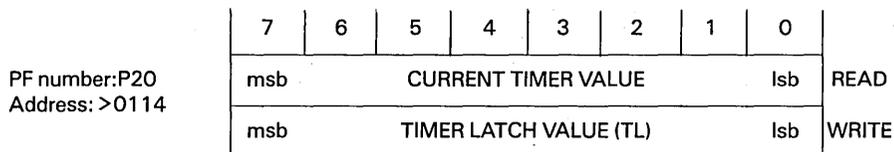


FIGURE 2-31 – TIMER 3 DATA REGISTER - T3DATA

2.7.5.6 Receiver Buffer

The receiver buffer - RXBUF (see Figure 2-32) is a read-only register and is accessed through P22 in the Peripheral File. This register contains the current data from the RX. Writing has no direct effect on this register. Note that the read/write sequence of the MOVP instruction (as well as ORP, XORP, ANDP) performs a read before a write. This action will perform a spurious clearing of the RXBUF, and will set RXRDY to 0. Data in the RXBUF is right justified with padded 0s.

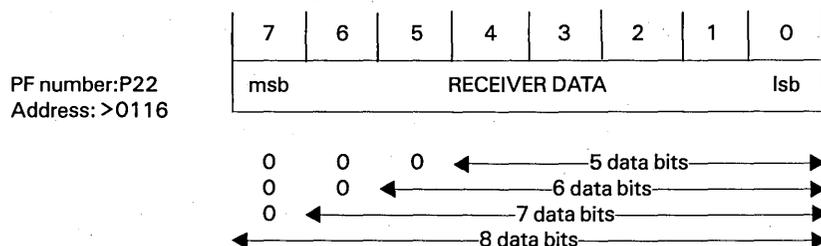


FIGURE 2-32 – RECEIVER BUFFER - RXBUF

2.7.5.7 Transmitter Buffer

The transmitter buffer - TXBUF (see Figure 2-33) is a write-only register and is accessed through P23 in the Peripheral File. This register contains the data to be transmitted by the TX. Reading P23 returns >00. Data written to the TXBUF must be right justified since the left-most bits will be ignored for characters less than eight bits in length.

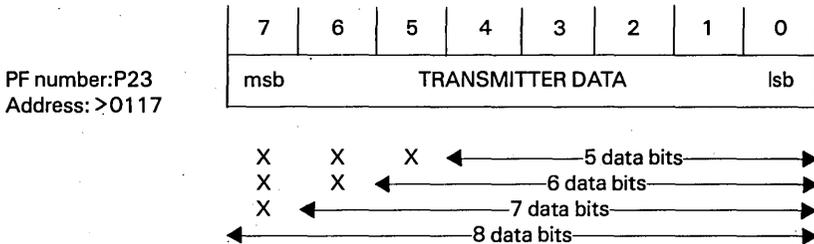


FIGURE 2-33 – TRANSMITTER BUFFER - TXBUF

2.7.6 Serial Port Initialization

To use the serial port on the TMS70X1, the user must first initialize it. After initialization, the serial port is operated by simply reading and writing to Peripheral File registers. Initialize the serial port as follows:

- 1) Set B3 Data value to 1. This allows the TXD line to transmit.
- 2) Write to the SMODE register (P17). This sets the character format and the type of communication mode.
- 3) Write to the SCTL0 register (second write to P17) to set the UR bit to 0. This same write can also enable the transmitter, receiver, or both.

Once the serial port is initialized it can be operated continuously in the selected operational mode. If the mode needs to be changed, the serial port must be reset, and then reinitialized for the desired mode. The serial port can be reset in two ways; hardware reset (via RESET pin), or software reset (via UR bit in SCTL0).

2.7.7 Serial Port Interrupts

INT4 is dedicated to the serial port. Three sources can generate an interrupt through INT4: the transmitter (TX), the receiver (RX), and Timer 3 (T3). Setting TXEN to a 1 allows data loaded into the TXBUF to be shifted into the TXSHF. The TX sets TXRDY and INT4 FLG to 1 when TXSHF is loaded from TXBUF.

In the communication modes, if RXEN is set to 1 the RX sets RXRDY and INT4 FLG to a 1 when RXBUF is loaded from RXSHF. If RXEN is 0, the RXSHF still receives frames and shifts them into the RXBUF, but RXRDY and INT4 FLG are held to 0. If a character is in RXBUF, and RXEN is then set to a 1, RXRDY and INT4 FLG will be set to 1.

In the serial I/O mode the RXEN is set to initiate the reception of a frame. When the last bit of the frame is received RXEN is reset to 0.

However, RXRDY and INT4 FLG are still set to 1 when the character is shifted from RXSHF to RXBUF. RXRDY and INT4 FLG bits are not masked by RXEN.

Timer 3 sets T3FLG, and INT4 FLG if T3ENB = 1, when its prescaler and timer decrement through 0 together.

Thus when INT4 is acknowledged by the CPU; RXRDY, TXRDY, and T3FLG are the flags to indicate its source. The INT4 service routine must determine which of these sources caused INT4 in the specific application. For example, if all three are likely sources, the INT4 service routine must check for the following possible situations:

- 1) RXRDY only
- 2) TXRDY only
- 3) T3 only
- 4) RXRDY, TXRDY, T3
- 5) RXRDY, TXRDY
- 6) RXRDY, T3
- 7) TXRDY, T3
- 8) None

The last situation check is necessary because RXRDY, TXRDY, or T3FLG can set INT4 FLG. Therefore it is possible that one or more interrupts may occur between CPU acknowledgement of INT4 and INT4 service routine testing of RXRDY, TXRDY, and T3FLG. The INT4 FLG bit is cleared by the CPU when it acknowledges INT4. If a second source of INT4 is set in the time between this clearing and the software testing, the second or third interrupts will be serviced by current INT4 service routine. Thus when INT4 is again acknowledged (INT4 FLG was set again by the second interrupt) RXRDY, TXRDY, and T3FLG will all be set to 0.

2.8

PIN DESCRIPTION

Table 2-9 and Table 2-10 defines the pin assignments and describes the function of each pin for the Single-Chip, Peripheral Expansion, Full Expansion, Microprocessor and Emulator modes for the TMS70X0 and TMS70X1. All the TMS7000 family devices discussed in this manual are pin compatible. Some pins on 70X1 devices have extra functions and CMOS devices have different electrical specifications (see Section 4).

TABLE 2-9 — SC, PE, FE, AND MICROPROCESSOR PIN ASSIGNMENTS

SIGNATURE	I/O	DESCRIPTION	APPLICABLE SECTIONS	
A0	I/O	A0-A4 and A7 are general purpose bi-directional pins and A5,A6 are input-only general purpose pins for the 70X1 only. A0-A7 are general purpose input pins for 70X0 devices. Serial port receiver Serial port clock, input or output Real Time Clock used to decrement Timer 1	2.2	
A1	I/O		2.3	
A2	I/O			
A3	I/O			
A4	I/O			
A5/RXD	IN		2.7.1	
A6/SCLK	I/O		2.7.2	
A7	I/O		2.6.1	
B0	OUT	B0-B3 Output only pins	2.2	
B1	OUT	B4-B7 Output only pins in single chip mode	2.3	
B2	OUT	B4-B7 Memory interface in all other modes		
B3/TXD	OUT	Serial port transmitter in 70X1 devices only	2.7.1	
B4/ALATCH	OUT	Memory interface Address Latch strobe		
B5/RW	OUT	Memory interface Read or Write signal		
B6/ENABLE	OUT	Memory interface Enable strobe		
B7/CLOCKOUT	OUT	Internal clock out		
C0	I/O	General purpose bi-direction pins in single chip mode Multiplexed low address and data bus in all other modes	2.2	
C1	I/O		2.3	
C2	I/O			
C3	I/O			
C4	I/O			
C5	I/O			
C6	I/O			
C7	I/O			
D0	I/O	General purpose Bi-direction pins in single chip and peripheral expansion modes	2.2	
D1	I/O		2.3	
D2	I/O	High address bus in Full Expansion and Microprocessor modes		
D3	I/O			
D4	I/O			
D5	I/O			
D6	I/O			
D7	I/O			
<u>INT1</u>	IN		Maskable interrupt of higher priority	2.4
<u>INT3</u>	IN		Maskable interrupt of lower priority	2.4
<u>RESET</u>	IN	Device reset	2.5.2	
MC	IN	Mode control	2.3	
XTAL2/CLKIN	IN	Crystal input for control of internal oscill. or input pin for external oscill.	2.5	
XTAL1	IN	Crystal input for control of internal oscill. leave open for external oscill.	2.5	
VCC	IN	Supply voltage (+5V NMOS, 3 to 6V for CMOS) ground reference		
VSS	IN			

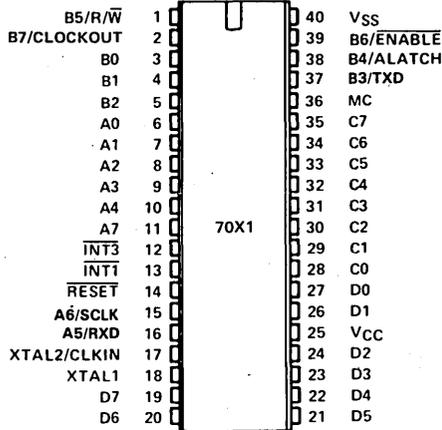
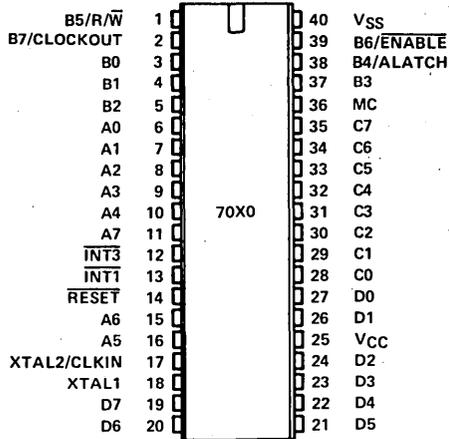


FIGURE 2-34 — SC, FE, PE, AND MICROPROCESSOR MODE PIN ASSIGNMENTS

TABLE 2-10 – SYSTEM EMULATOR MODE PIN ASSIGNMENTS

SIGNATURE	I/O	DESCRIPTION	APPLICABLE SECTIONS
A0	I/O	Not Connected	2.3.5
A1	I/O	NC	
A2	I/O	NC	
A3	I/O	NC	
A4	I/O	NC	
A5/RXD	IN	NC	
A6/SCLK	I/O	NC	
A7	I/O	NC	
B0	OUT	NC	
B1	OUT	NC	
B2	OUT	NC	
B3/TXD	OUT	Interrupt Acknowledge	2.5.3
B4/ALATCH	OUT	Memory interface Address latch	2.3.5
B5/RW	OUT	Memory interface Read or Write	2.3
B6/ENABLE	OUT	Memory interface Memory Enable	
B7/CLOCKOUT	OUT	Internal clock out	
C0	I/O	ADDR0	
C1	I/O	ADDR1	2.3
C2	I/O	ADDR2 Multiplexed low address and data bus	
C3	I/O	ADDR3	
C4	I/O	ADDR4	
C5	I/O	ADDR5	
C6	I/O	ADDR6	
C7	I/O	ADDR7	
D0	I/O	ADDR8	2.2
D1	I/O	ADDR9	2.3
D2	I/O	ADDR10	
D3	I/O	ADDR11 High order address byte	
D4	I/O	ADDR12	
D5	I/O	ADDR13	
D6	I/O	ADDR14	
D7	I/O	ADDR15	
<u>NMI</u>	IN	Non-Maskable interrupt of higher priority	2.3.5
<u>INT</u>	IN	Maskable interrupt of lower priority Note: This pin is NC on the CMOS version	2.3.5
<u>RESET</u>	IN	Device reset	2.5.2
<u>MC</u>	IN	Mode control: must be held at + 14 Volts	2.3
<u>XTAL2/CLKIN</u>	IN	Crystal input for control of internal oscill. or input pin for external oscill.	2.5
<u>XTAL1</u>	IN	Crystal input for control of internal oscill. leave open for external oscill.	2.5
<u>VCC</u>	IN	Supply voltage (+ 5 V NMOS, 3V to 6V for CMOS)	
<u>VSS</u>	IN	Ground reference	

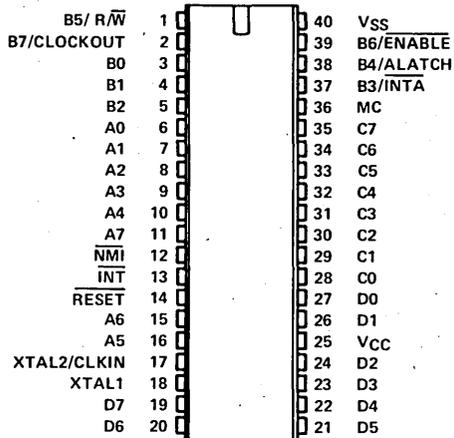


FIGURE 2-35 — SYSTEM EMULATOR MODE PIN ASSIGNMENTS

3. STANDARD INSTRUCTION SET

The TMS7000 instruction set is composed of 61 instructions that provide for input, output, manipulation, and comparison of data. The instruction set is divided into eight functional categories:

ARITHMETIC INSTRUCTIONS

BRANCH AND JUMP INSTRUCTIONS

COMPARE INSTRUCTIONS

CONTROL INSTRUCTIONS

LOAD AND MOVE INSTRUCTIONS

LOGICAL INSTRUCTIONS

SHIFT INSTRUCTIONS

I/O INSTRUCTIONS

Refer to the TMS7000 ASSEMBLY LANGUAGE PROGRAMMER'S GUIDE (MP 916) for a detailed description of the instruction set, machine formats, addressing modes, and other information relevant to the execution of a TMS7000 assembly language program. The sections that follow summarize the key features of the TMS7000 Assembler.

3.1 DEFINITIONS

The symbols used in the instructions are listed and defined in Table 3-1.

TABLE 3-1 – TMS7000 SYMBOL DEFINITIONS

SYMBOL	DEFINITION
\$	Current value of Program Counter
A	Register A or R0 in Register File
B	Register B or R1 in Register File
b	Bit number as in b7 ($0 \leq b \leq 7$)
Rn	Register n of Register File ($0 \leq n \leq 127$)
Rn-1	Register File number n-1 ($0 \leq n \leq 127$)
Pn	Port n of Peripheral File ($0 \leq n \leq 255$)
PC	Program Counter
IPC	Interprete Program Counter
ST	Status Register
SP	Stack Pointer
s	Source operand (either a reg or an immed 8-bit operand)
Rs	Source register in Register File ($0 \leq s \leq 127$)
d	Destination operand (always a register)
Rd	Destination register in Register File ($0 \leq d \leq 127$)
Pd	Destination in peripheral file
Rd-1	Register File number d-1 ($0 \leq d \leq 127$)
iop	Immediate operand
ra	Relative Address ($ra = ta - pcn$)
ta	Target Address ($ta = ra + pcn$)
pcn	Location of the next instruction
cnd	Condition
@	Indicates an address or label
%	Indicates immediate operand
*	Indicates Indirect Register File Addressing Mode
>	Hexidecimal number
MSB	Most significant byte or bit
LSB	Least significant byte or bit

3.2 ADDRESSING MODES

The TMS7000 Assembly Language supports eight addressing modes. Five of these modes specify 8-bit operands and are classified as Direct Addressing Modes. The remaining three addressing modes generate a 16-bit address and are classified as Extended Addressing Modes. Table 3-2 summarizes both classifications.

TABLE 3-2 – TMS7000 ADDRESSING MODES

CLASS	ADDRESSING MODE	EXAMPLE	SEE SECTION
DIRECT	SINGLE REGISTER	LABEL DEC B INC R45 CLR R23	3.2.1.1
	REGISTER FILE	LABEL MOV B,A ADD A,R17 CMP R32,R73	
	PERIPHERAL FILE	LABEL XORP A,P17 MOV P42,B	3.2.1.3
	IMMEDIATE	LABEL AND % >C5,R55 ANDP % VALUE,P32 BTJO % >D6,R80,LABEL	3.2.1.4
	PROGRAM COUNTER RELATIVE	LABEL1 JMP LABEL DJNZ A,LABEL BTJO % > 16,R12,LABEL BTJOP B,P7,LABEL	3.2.1.5
EXTENDED	DIRECT MEMORY	LABEL LDA @>F3D4 CMPA @LABEL	3.2.2.1
	REGISTER FILE INDIRECT	LABEL STA *R43	3.2.2.2
	INDEXED	LABEL2 BR @LABEL(B)	3.2.2.3

3.2.1 Direct Addressing Modes

The five Direct Addressing modes specify 8-bit operands. Each is described in the following sections.

3.2.1.1 Single Register Addressing Mode

The Single Register Addressing mode specifies a single register in the Register File as containing the 8-bit operand. The register can be specified as Rn or n (See Table 3-2), where n is the Register File number and 0 is less than or equal to n which is less than or equal to 127. When specifying either the A or B register, A or B can be substituted for R0 or R1 respectively in the operand field of the assembly language statement. As is explained in Section 3.3.1, instructions using the Single Register Addressing mode are also called implied operand instructions if either the A or B register is specified. Instructions using the Single Register Addressing Mode and specifying Rn, where 2 is less than or equal to n which is less than or

equal to 127, are also called single operand instructions and are described in Section 3.3.2. Figure 3-1 illustrates the object code generated by a Single Register instruction for the following cases:

Case 1: <inst> A
 <inst> B

Case 2: <inst> Rn (where 0 less than or equal to n which is less than or equal to 127)

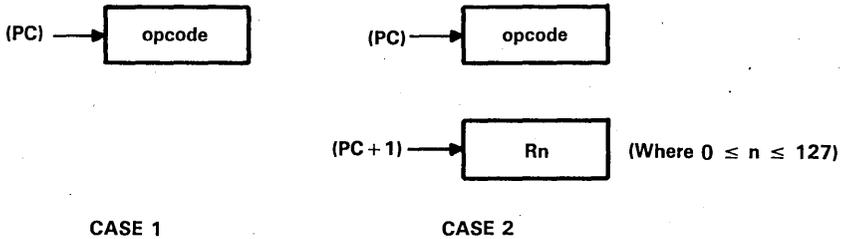


FIGURE 3-1 – SINGLE REGISTER ADDRESSING MODE OBJECT CODE

3.2.1.2 Register File Addressing Mode

The Register File Addressing mode specifies a source and a destination register in the Register File as containing the 8-bit operands. As illustrated in Table 3-2, the assembly language statement specifies the source register before the destination register. Figure 3-2 illustrates the object code generated by an instruction using the Register File Addressing mode for the following cases:

Case 1: <inst> B,A

Case 2: <inst> A,B
 <inst> Rs,A
 <inst> Rs,B

Case 3: <inst> A,Rd
 <inst> B,Rd
 <inst> Rs,Rd

NOTE: The MOV instruction is uniquely defined for Register File Addressing mode. Refer to Table 3-8 for definition.

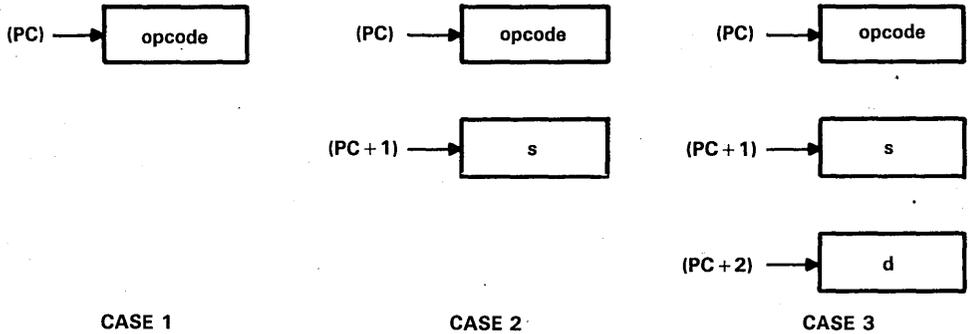
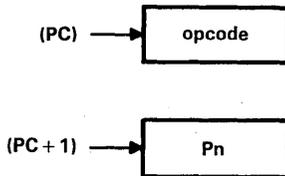


FIGURE 3-2 – REGISTER FILE ADDRESSING MODE OBJECT CODE

3.2.1.3 Peripheral File Addressing Mode

The Peripheral File Addressing mode is used to perform I/O tasks. Each PF register is an 8-bit port which can be referred to as Pn or n, as shown in Table 3-2. There are four instructions that use the Peripheral File Addressing mode: MOVP, ANDP, ORP, and XORP. BTJOP and BTJZP are also peripheral instructions but they have a different format which is discussed in Section 3.3.4.3. All four instructions may be executed using either the A or B register as the source register and Pn as the destination register. However, only the MOVP instruction may also be executed using the Pn as the source register and either A or B as the destination register. Figure 3-3 illustrates the object code generated by an instruction using the Peripheral File Addressing mode for the following cases:

- Case 1: <inst > A,Pn
 <inst > B,Pn
- Case 2: MOVP Pn,A
 MOVP Pn,B



CASES 1 AND 2

FIGURE 3-3 – PERIPHERAL FILE ADDRESSING MODE OBJECT CODE

3.2.1.4 Immediate Addressing Mode

The Immediate Addressing mode uses the contents of the byte following the opcode byte as an 8-bit operand. As shown in Table 3-2, the immediate operand (*iop*) can be a hex constant or a label, and is indicated by a percent sign preceding the expression. Immediate operands can be used by RF, PF, and Jump instructions. Refer to Tables 3-8, 3-9, 3-13, and 3-14 for an illustration of the particular machine instruction formats. In addition, the MOVD instruction uses immediate operands in two special formats (See Table 3-18). Figure 3-4 illustrates the simplest case of an instruction using the Immediate Addressing mode.

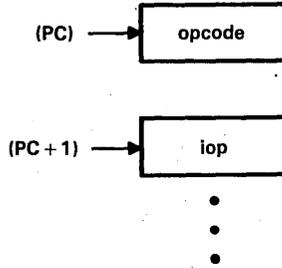
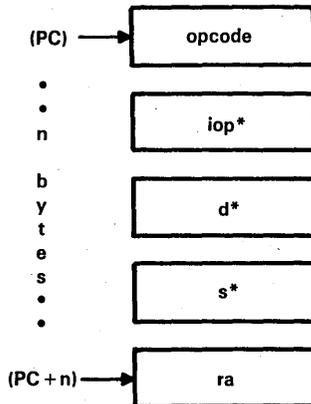


FIGURE 3-4 – IMMEDIATE ADDRESSING MODE OBJECT CODE

3.2.1.5 Program Counter Relative Addressing Mode

The Program Counter Relative Addressing mode is used by all jump instructions. As shown in Table 3-2, the assembly language statement for a jump instruction always includes a target address (*ta*) in the form of a label. During assembly, the target address is used by the microcomputer to calculate a relative address (*ra*) as follows: $ra = ta - pcn$, where *pcn* is the location of the next instruction and -128 is less than or equal to *ra* which is less than or equal to 127 . Note that the relative address is also referred to as the offset. The machine instruction formats for the various types of jump instructions are given in Tables 3-11, 3-12, 3-13, and 3-14. Figure 3-5 illustrates the object code generated by a jump instruction.



*n optional bytes, depending upon the particular jump instruction

FIGURE 3-5 – PROGRAM COUNTER RELATIVE ADDRESSING MODE OBJECT CODE

3.2.2 Extended Addressing Modes

The three Extended Addressing modes generate 16-bit addresses to memory. The 16-bit address space includes the Register File, the Peripheral File, on-chip program memory, and off-chip memory. Each of the Extended Addressing modes is described in the sections that follow.

3.2.2.1 Direct Memory Addressing Mode

Direct Addressing Mode specifies a 16-bit address that contains the operand. As shown in Table 3-2, the 16-bit address is preceded by an @ sign and can be written as a hex constant or as a label. Figure 3-6 shows how the object code produced by an instruction using the Direct Memory Addressing mode is used to generate a 16-bit effective address.

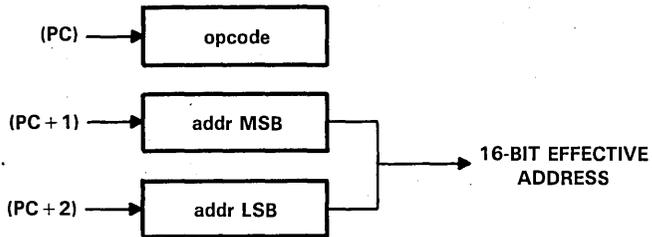


FIGURE 3-6 — DIRECT MEMORY ADDRESSING MODE OBJECT CODE

3.2.2.2 Register File Indirect Addressing Mode

The Register File Indirect Addressing mode uses the contents of a register pair as a 16-bit effective address. As shown in Table 3-2, the indirect register file address is written as a register number (Rn) preceded by an asterisk (*), i.e.: *Rn. The LSB of the address is contained in Rn, and the MSB of the address is contained in the previous register (Rn-1). Note that R0 cannot be specified. Figure 3-7 shows how the object code produced by an instruction using the Register File Indirect Addressing mode is used to generate a 16-bit effective address.

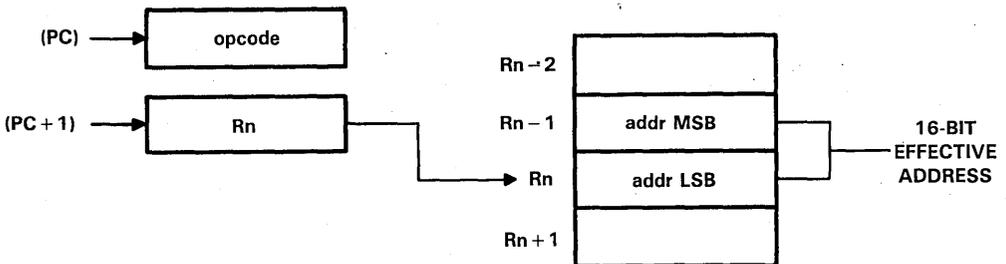


FIGURE 3-7 — REGISTER FILE INDIRECT ADDRESSING MODE OBJECT CODE

3.2.2.3 Indexed Addressing Mode

The Indexed Addressing mode generates a 16-bit address by summing the contents of the B register with a 16-bit direct memory address. As shown in Table 3-2, the assembly language statement for the Indexed Addressing mode contains the direct memory address written as a label preceded by an @ sign, followed by a B in parentheses, i.e.: @LABEL(B). The summing operation automatically transfers any carries into the MSB. Figure 3-8 illustrates how the object code produced by an instruction using the Indexed Addressing mode is used to generate a 16-bit effective address. This mode should not be confused with the move double (MOVD) instruction's %VALUE(B) addressing mode; see Section 3.3.6.

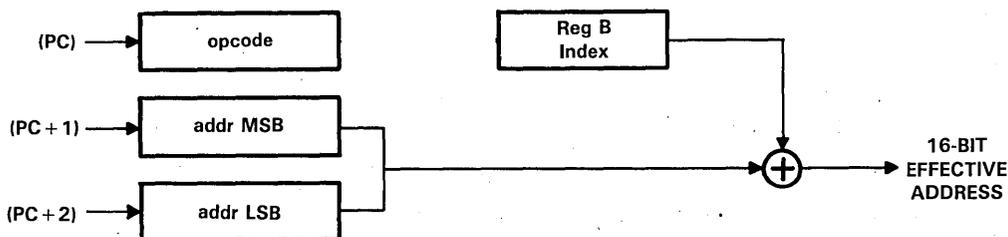


FIGURE 3-8 — INDEXED ADDRESSING MODE OBJECT CODE

3.3 INSTRUCTIONS

The instruction set is divided into the following types of instructions: Implied Operand, Dual Operand, Jump, Extended Address, and Miscellaneous instructions. Each instruction type is defined in the sections that follow. For additional details, refer to the TMS7000 ASSEMBLY LANGUAGE PROGRAMMER'S GUIDE (MP 916).

3.3.1 Implied Operand Instructions

Implied Operand instructions are one-byte instructions whose operands, if any, are implied by the opcode itself. Table 3-3 lists the implied operand instructions in alphabetical order, along with a brief functional description of each instruction. Table 3-4 shows the machine instruction format for all Implied Operand instructions.

TABLE 3-3 – IMPLIED OPERAND INSTRUCTIONS

MNEMONIC	MEANING	STATUS BITS AFFECTED	DESCRIPTION
CLRC	Clear Carry Bit	C,N,Z	0 – C,N,Z, set from A register
DINT	Disable Interrupts	C,N,Z,I	0 – I, 0 – C, 0 – n, 0 – Z
EINT	Enable Interrupts	C,N,Z,I	1 – I, 0 – C, 0 – n, 0 – Z
IDLE	Idle until Interrupt	none	Suspend until interrupt
LDSP	Load Stack Pointer	none	B register – SP
NOP	No operation	none	PC + 1 – PC
POP ST	Pop Status from Stack	none	Top of Stack → ST; SP – 1 – SP
PUSH ST	Push Status onto Stack	none	SP + 1 → SP; ST → Top of stack
SETC	Set Carry	C,N,Z	1 – C, 0 – N, 1 – Z
STSP	Store Stack Pointer	none	SP – B register
RETI	Return from Interrupt	loaded from stack	Operand address – PC Stack → PC LSB byte, SP – 1 → SP Stack → PC MSB byte, SP – 1 → SP Stack → ST, SP – 1 → SP
RETS	Return from Subroutine	none	Stack → PC LSB byte, SP – 1 → SP Stack → PC MSB byte, SP – 1 → SP

TABLE 3-4 – MACHINE INSTRUCTION FORMAT: IMPLIED OPERAND INSTRUCTION

ASSEMBLY LANGUAGE STATEMENT	MACHINE INSTRUCTION FORMAT (BYTE 1)
<inst>	opcode

3.3.2 Single Operand Instructions

Single Operand instructions are either one- or two-byte instructions that use the Single Register Addressing mode exclusively. Table 3-5 lists the Single Operand instructions in alphabetical order, along with a brief functional description of each. Table 3-6 shows the machine instruction formats for all single operand instructions.

TABLE 3-5 – SINGLE OPERAND INSTRUCTIONS

MNEMONIC	MEANING	STATUS BITS AFFECTED	DESCRIPTION
CLR	Clear Operand	C,N,Z	0 – dest
DEC	Decrement	C,N,Z	Dest – 1 – dest
DECD	Decrement Double	C,N,Z	Register pr – 1 – register pr
INC	Increment	C,N,Z	Dest + 1 – dest
INV	Invert	C,N,Z	Inverted dest – dest
POP	Pop from Stack	C,N,Z	Top of Stack – dest, SP – 1 → SP
PUSH	Push on Stack	C,N,Z	SP + 1 → SP, Dest – top of stack
RL	Rotate Left	C,N,Z	bn – bn + 1, b7 – b0, C
RLC	Rotate Left through carry	C,N,Z	bn – bn + 1, C – b0, b7 – C
RR	Rotate Right	C,N,Z	bn + 1 – bn, b0 – b7, C
RRC	Rotate Right through carry	C,N,Z	bn + 1 – bn, C – b7, b0 – C
SWAP	Swap Nibbles	C,N,Z	b7-b4 ↔ b3-b0
XCHB	Exchange with Register B	C,N,Z	B ↔ dest, N,Z set on Dest contents

TABLE 3-6 – MACHINE INSTRUCTION FORMATS: SINGLE OPERAND INSTRUCTIONS

ASSEMBLY LANGUAGE STATEMENT	MACHINE INSTRUCTION FORMAT	
	BYTE 1	BYTE 2
<inst> A <inst> B	opcode	
<inst> Rd	opcode	d

3.3.3 Dual Operand Instructions

Dual Operand instructions are one-, two-, or three-byte instructions that specify one of the following:

- Both a source and destination register
- An immediate operand and a destination register

Table 3-7 lists the Dual Operand instructions in alphabetical order, along with a brief description of each.

TABLE 3-7 – DUAL OPERAND INSTRUCTIONS

MNEMONIC	MEANING	STATUS BITS AFFECTED	DESCRIPTION
ADC	Add with Carry	C,N,Z	Source + dest + carry → dest
ADD	Add Bytes	C,N,Z	Source + dest → dest
AND	AND bytes	C,N,Z	Source logically ANDed with dest → dest
ANDP	AND Peripheral File	C,N,Z	Source logically ANDed with PF → PF
CMP	Compare	C,N,Z	Dest - source computed but not stored
DAC	Decimal Add w/Carry	C,N,Z	Source + dest + carry → dest
DSB	Decimal Subtract w/Borrow	C,N,Z	Dest - source - 1 + carry → dest
MOV	Move	C,N,Z	Source → dest
MOVP	Move to/from PF	C,N,Z	Read or write data from/to Pf
MPY	Multiply	C,N,Z	Source x Dest → A, B
OR	OR	C,N,Z	Source logically ORed with dest → dest
ORP	OR Peripheral File	C,N,Z	Source logically ORed with PF → PF
SBB	Subtract with Borrow	C,N,Z	Dest - source - 1 + carry → dest
SUB	Subtract Bytes	C,N,Z	Dest - source → dest
XOR	Exclusive OR	C,N,Z	Source exclusively ORed with dest → dest
XORP	Exclusive OR PF	C,N,Z	Source exclusively ORed with PF → PF

3.3.3.1 Register File Instruction Types

Table 3-8 lists the machine instruction formats for the Dual Operand instructions which address the Register File. The instructions which use these formats are:

ADC	ADD	AND	CMP	DAC	DSB
MOV	MPY	OR	SBB	SUB	XOR

These instructions use either the Register File Addressing mode or a combination of the Register File and Immediate Addressing modes. Note that the MOV instruction is specifically illustrated in Table 3-8, because its formats are uniquely defined.

TABLE 3-8 — MACHINE INSTRUCTION FORMATS: REGISTER FILE INSTRUCTIONS

ASSEMBLY LANGUAGE STATE	MACHINE INSTRUCTION FORMAT		
	BYTE 1	BYTE 2	BYTE 3
<inst> B,A	opcode		
<inst> A,B <inst> Rs,A <inst> Rs,B	opcode	s	
<inst> A,Rd <inst> B,Rd <inst> Rs,Rd	opcode	s	d
<inst> %<iop>,A <inst> %<iop>,B	opcode	iop	
<inst> %<iop>,Rd	opcode	iop	d
MOV A,B MOV B,A	opcode		
MOV A,Rd MOV B,Rd	opcode	d	
MOV Rs,A MOV Rs,B	opcode	s	

3.3.3.2 Peripheral File Instruction Type

Table 3-9 shows the machine instruction formats for the Dual Operand instructions that address the Peripheral File. The instructions which use these formats are:

ANDP	MOVP	ORP	XORP
------	------	-----	------

These instructions use either the Peripheral File Addressing mode or a combination of the Peripheral File and Immediate Addressing modes. Note that the MOVP instruction is specifically illustrated in Table 3-9 because its formats are uniquely defined.

TABLE 3-9 – MACHINE INSTRUCTION FORMATS: PERIPHERAL FILE INSTRUCTIONS

ASSEMBLY LANGUAGE STATE	MACHINE INSTRUCTION FORMAT		
	BYTE 1	BYTE 2	BYTE 3
<inst> A, Pn <inst> B, Pn	opcode	n	
<inst> %<iop>, Pn	opcode	iop	n
MOV Pn, A MOV Pn, B MOV A, Pn MOV B, Pn	opcode	n	

3.3.4 Jump Instructions

Jump instructions are two-, three-, and four-byte instructions that use the Program Counter Relative Addressing mode. These instructions are divided into four format types: Simple Relative, Single Relative, Dual Relative, and Peripheral Relative. All jump instructions must specify a target address (ta) in the form of a label in the assembly language statement, so that a relative address (ra) can be calculated according to the following formula:

$$ra = ta - pcn.$$

where pcn is the location of the next instruction and -128 is less than or equal to ra which is less than or equal to 127 (See Section 3.2.1.5). Table 3-10 lists all jump instructions in alphabetical order, along with a brief description of each instruction.

TABLE 3-10 – JUMP INSTRUCTIONS

MNEMONIC	MEANING	STATUS BITS AFFECTED	DESCRIPTION
BTJO	Bit Test Jump if One	C,N,Z	If source ANDed with dest \neq 0, jump
BTJOP	Bit Test Jump if One PF	C,N,Z	If source ANDed with PF \neq 0, jump
BTJZ	Bit Test Jump if Zero	C,N,Z	If source ANDed with inverted dest \neq 0, jump
BTJZP	Bit Test Jump if Zero PF	C,N,Z	If source ANDed with inverted PF \neq 0, jump
DJNZ	Dec.Reg.Jump Non-Zero	none	Dest - 1 - dest, if dest \neq 0, jump
JMP	Jump Unconditional	none	PC + offset - PC
JC/JHS	Jump if Carry Set/ Jump if Higher or Same	none	If C = 1, PC + offset - PC
JN	Jump if Negative	none	If N = 1, PC + offset - PC
JNC/JL	Jump if No carry/ Jump if Lower	none	If C = 0, PC + offset - PC
JNZ/JNE	Jump if Not Zero/ Jump if Not Equal	none	If Z = 0, PC + offset - PC
JP	Jump if Positive	none	If N = 0, Z = 0, PC + offset - PC
JPZ	Jump if Pos. or Zero	none	If N = 0, PC + offset - PC
JZ/JEQ	Jump if Zero/ Jump if Equal to	none	If Z = 1, PC + offset - PC

NOTE: Some conditional jump instructions have two names: one indicating the condition of the Status Register bits that are tested and one indicating the result of a CMP (compare) instruction.

3.3.4.1 Simple Relative Instruction Type

Table 3-11 shows the machine instruction format for the Simple Relative Instruction type. This format requires only the target address (label) in the operand field of the assembly language statement. The Simple Relative Jump instructions are:

JMP	Jump (Unconditional)
JC/JHS	Jump If Carry Set/Jump If Higher Or Same
JN	Jump If Negative
JNC/JL	Jump If No Carry/Jump If Lower
JNZ/JNE	Jump If Not Zero/Jump If Not Equal
JP	Jump If Positive
JPZ	Jump If Positive Or Zero
JZ/JEQ	Jump If Zero/Jump If Equal To

TABLE 3-11 – MACHINE INSTRUCTION FORMAT: SIMPLE RELATIVE INSTRUCTIONS

ASSEMBLY LANGUAGE STATEMENT	MACHINE INSTRUCTION FORMAT	
	BYTE 1	BYTE 2
<inst> <ta>	opcode	ra

3.3.4.2 Single Relative Instruction Type

Table 3-12 shows the machine instruction formats for the Single Relative instruction type. These formats require a Register File number and a target address (label) in the operand field of the assembly language statement. DJNZ is the only Single Relative jump instruction.

TABLE 3-12 – MACHINE INSTRUCTION FORMATS: SINGLE RELATIVE INSTRUCTIONS

ASSEMBLY LANGUAGE STATE	MACHINE INSTRUCTION FORMAT		
	BYTE 1	BYTE 2	BYTE 3
<inst> A, <ta>	opcode	ra	
<inst> B, <ta>			
<inst> Rn, <ta>	opcode	n	ra

3.3.4.3 Dual Relative Instruction Type

Table 3-13 shows the machine instruction formats for the Dual Relative instruction type. These formats require a target address (label) and either a Register File number or an immediate operand in the operand field of the assembly language statement. BTJO and BTJZ are the Dual Relative Jump instructions.

TABLE 3-13 – MACHINE INSTRUCTION FORMATS DUAL RELATIVE INSTRUCTIONS

ASSEMBLY LANGUAGE STATE	MACHINE INSTRUCTION FORMAT			
	BYTE 1	BYTE 2	BYTE 3	BYTE 4
<inst> B, A, <ta>	opcode	ra		
<inst> Rs, A, <ta>	opcode	s	ra	
<inst> Rs, B, <ta>				
<inst> Rs, Rd, <ta>	opcode	s	d	ra
<inst> %<iop>, A, <ta>	opcode	iop	ra	
<inst> %<iop>, B, <ta>				
<inst> %<iop>, Rd, ta	opcode	iop	d	ra

3.3.4.4 Peripheral Relative Instruction Type

Table 3-14 shows the machine instruction formats for the Peripheral Relative instruction type. These formats require a target address (label), a Peripheral File register number, and either an immediate operand or one of two possible Register File Registers (the A or B register) in the operand field of the assembly language statement. BTJOP and BTJZP are the Peripheral Relative jump instructions.

TABLE 3-14 – MACHINE INSTRUCTION FORMATS: PERIPHERAL RELATIVE INSTRUCTIONS

ASSEMBLY LANGUAGE STATE	MACHINE INSTRUCTION FORMAT			
	BYTE 1	BYTE 2	BYTE 3	BYTE 4
<inst> A, Pn, <ta>	opcode	n	ra	
<inst> B, Pn, <ta>				
<inst> %<iop>, Pd, <ta>	opcode	iop	n	ra

3.3.5 Extended Address Instructions

Extended Address instructions are two- or three-byte instructions that reference a 16-bit address in memory. Table 3-15 lists the Extended Address instructions in alphabetical order, along with a brief description of each instruction.

TABLE 3-15 – EXTENDED ADDRESS INSTRUCTIONS

MNEMONIC	MEANING	STATUS BITS	DESCRIPTION
		AFFECTED	
BR	Unconditional Branch	none	Dest – PC
CALL	Call Subroutine	none	SP + 1 → SP, PCMS byte → stack SP + 1 → SP, PCLS byte → stack
CMPA	Compare to A Register	C,N,Z	A – Source computed but not stored
LDA	Load A Register	C,N,Z	Source → A
STA	Store A Register	C,N,Z	A → dest

Table 3-16 shows the machine instruction formats for the three addressing modes available to Extended Address instructions: Direct, Register File Indirect, and Indexed Addressing modes.

TABLE 3-16 – MACHINE INSTRUCTION FORMATS: EXTENDED ADDRESS INSTRUCTIONS

ASSEMBLY LANGUAGE STATE	MACHINE INSTRUCTION FORMAT		
	BYTE 1	BYTE 2	BYTE 3
<inst> @<addr>	opcode	addr MSB	addr LSB
<inst> *Rd	opcode	d	
<inst> @<addr>(B)	opcode	addr MSB	addr LSB

3.3.6 Miscellaneous Instructions

The MOVD and the twenty-four TRAP instructions are special instructions that do not belong in any of the previously described categories of instruction types or addressing modes. These instructions are shown in Table 3-17 and are discussed in the sections that follow.

TABLE 3-17 – MACHINE INSTRUCTIONS FORMATS: MISCELLANEOUS INSTRUCTIONS

MNEMONIC	MEANING	STATUS BITS AFFECTED	DESCRIPTION
MOVD	Move Double	C,N,Z	a. iop → register pr b. indexed iop → register pr c. register pr → register pr
TRAP 0 △ △ △	Trap to Subroutine	none	SP + 1 → SP, PCMS byte → stack SP + 1 → SP, PCLS byte → stack Entry vector → PC
TRAP 23			

3.3.6.1 *MOVD Instruction*

The **MOVD** instruction moves a two-byte value into a register pair in the Register File. This destination register pair is specified by a single register number; *Rd*, which indicates that the MSB is contained in *Rd-1* and the LSB is contained in *Rd*. As shown in Table 3-18, the two-byte value may be a 16-bit immediate operand, a 16-bit indexed immediate operand, or the contents of a register pair in the Register File. These formats are useful for the following tasks:

- MOVD** *%iop,Rd* Register pair initialization with an immediate value before executing an instruction in the Register File Indirect Addressing mode.

- MOVD** *%iop(B),Rd* Register pair initialization with an indexed immediate value before executing an instruction in the Register File Indirect Addressing mode.

- MOVD** *Rs,Rd* Register pair to register pair transfer in the Register File.

The C, N, and Z status bits are affected by the execution of the **MOVD** instruction as follows:

C — Set to zero

N — Set to one if MSB is negative; set to zero if MSB is positive or zero

Z — Set to one if MSB is zero; set to zero if MSB is nonzero

Refer to Section 3.4.2 for more details on the status bits.

TABLE 3-18 – MACHINE INSTRUCTION FORMATS: MOVD INSTRUCTION

ASSEMBLY LANGUAGE STATE	MACHINE INSTRUCTION FORMAT			
	BYTE 1	BYTE 2	BYTE 3	BYTE 4
MOVD <i>%iop, Rd</i>	opcode	<i>iop</i> MSB	<i>iop</i> LSB	<i>d</i>
MOVD <i>%iop, (B), Rd</i>	opcode	<i>iop</i> MSB	<i>iop</i> LSB	<i>d</i>
MOVD <i>Rs, Rd</i>	opcode	<i>s</i>	<i>d</i>	

3.3.6.2 TRAP Instructions

The TRAP instructions branch to a two-byte location in a reserved section of memory called the Trap Vector Table. As shown in Figure 3-9, each trap location stores a 16-bit address which references either the reset function (TRAPO), one of the three interrupt service routines (TRAP1-INT1, TRAP2-INT2, TRAP3-INT3), or a subroutine (TRAP4-23).

>FFD0	TRAP23 Address	MSB
>FFD1	TRAP23 Address	LSB
..
..
>FFE0	TRAP15 Address	MSB
>FFE1	TRAP15 Address	LSB
..
..
>FFFA	TRAP2 Address	MSB
>FFFB	TRAP2 Address	LSB
>FFFC	TRAP1 Address	MSB
>FFFD	TRAP1 Address	LSB
>FFFE	TRAPO Address	MSB
>FFFF	TRAPO Address	LSB

FIGURE 3-9 – THE TRAP VECTOR TABLE

The TRAP instructions are all single-byte instructions, i.e., the machine instruction format requires only the opcode byte. No status bits are affected by the execution of these instructions.

TRAPs 0-23 push the contents of the Program Counter onto the stack (PC MSB followed by PC LSB) before executing the subroutine stored at the address in the Trap Vector Table. See Section 3.5.50 and Section 6.3.3 for more information.

3.4 CUSTOM MICROCODING

For applications requiring unusually high performance, or for customers wishing to tailor the instruction set to their application program, the TMS7000 instruction set is implemented with 160 micro-instructions of 45 bits each with which Texas Instruments is prepared to support limited customer re-microcoding. More details of custom microcoding can be found in Section 5 of this book.

Certain instructions in the instruction set may be removed and replaced with a unique customer-defined instruction, others may not. The instructions which may not be altered comprise the core instruction set; those which may be altered or removed are classified as non-core instructions. A listing of the core (reserved) and non-core (available for microcoding) instructions is provided in Tables 3-19 and 3-20 respectively.

TABLE 3-19 — TMS7000 CORE (RESERVED) INSTRUCTIONS

MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE
NOP	00	OR Rn,A	14	BTJO Rn,A	16
IDLE	01	OR %n,A	24	BTJO %n,A	26
MOV Rn,A	12	OR Rn,B	34	BTJO Rn,B	36
MOV %n,A	22	OR Rn,Rn	44	BTJO Rn,Rn	46
MOV Rn,B	32	OR %n,B	54	BTJO %n,B	56
MOV Rn,Rn	42	OR B,A	64	BTJO B,A	66
MOV %n,B	52	OR %n,Rn	74	BTJO %n,Rn	76
MOV B,A	62				
MOV %n,Rn	72	XOR Rn,A	15	BTJZ Rn,A	17
MOV A,B	C0	XOR %n,A	25	BTJZ %n,A	27
MOV A,Rn	D0	XOR Rn,B	35	BTJZ Rn,B	37
MOV B,Rn	D1	XOR Rn,Rn	45	BTJZ Rn,Rn	47
		XOR %n,B	55	BTJZ %n,B	57
AND Rn,A	13	XOR B,A	65	BTJZ B,A	67
AND %n,A	23	XOR %n,Rn	75	BTJZ %n,Rn	77
AND Rn,B	33				
AND Rn,Rn	43	TSTA/CLRC	B0	POPST	08
AND %n,B	53	TSTB	C1	PUSHST	0E
AND B,A	63	SETC	07	LDSP	0D
AND %n,Rn	73	RETS	0A	STSP	09
DINT	06	RETI	0B	EINT	05
ADD Rn,A	18	ADC Rn,A	19	SUB Rn,A	1A
ADD %n,A	28	ADC %n,A	29	SUB %n,A	2A
ADD Rn,B	38	ADC Rn,B	39	SUB Rn,B	3A
ADD Rn,Rn	48	ADC Rn,Rn	49	SUB Rn,Rn	4A
ADD %n,B	58	ADC %n,B	59	SUB %n,B	5A
ADD B,A	68	ADC B,A	69	SUB B,A	6A
ADD %n,Rn	78	ADC %n,Rn	79	SUB %n,Rn	7A
SBB Rn,A	1B	LDA @n	8A	STA @n	8B
SBB %n,A	2B	LDA *Rn	9A	STA *Rn	9B
SBB Rn,B	3B	LDA @n(B)	AA	STA @n(B)	AB
SBB Rn,Rn	4B				
SBB %n,B	5B	BR @n	8C	CALL @n	8E
SBB B,A	6B	BR *Rn	9C	CALL *Rn	9E
SBB %n,Rn	7B	BR @n(B)	AC	CALL @n(B)	AE
CMP Rn,A	1D	DEC A	B2	INC A	B3
CMP %n,A	2D	DEC B	C2	INC B	C3
CMP Rn,B	3D	DEC Rn	D2	INC Rn	D3
CMP Rn,Rn	4D				
CMP %n,B	5D	INV A	B4	CLR A	B5
CMP B,A	6D	INV B	C4	CLR B	C5
CMP %n,Rn	7D	INV Rn	D4	CLR Rn	D5

TABLE 3-19 — TMS7000 CORE (RESERVED) INSTRUCTIONS (CONTINUED)

MNEMONIC	OP CODE	MNEMONIC	OP CODE	MNEMONIC	OP CODE
PUSH A	B8	POP A	B9	DJNZ A	BA
PUSH B	C8	POP B	C9	DJNZ B	CA
PUSH Rn	D8	POP Rn	D9	DJNZ Rn	DA
RR A	BC	RRC A	BD	RL A	BE
RR B	CC	RRC B	CD	RL B	CE
RR Rn	DC	RRC Rn	DD	RL Rn	DE
TRAP 7	F8	RLC A	BF	JMP	E0
TRAP 6	F9	RLC B	CF	JN/JLT	E1
TRAP 5	FA	RLC Rn	DF	JZ/JEQ	E2
TRAP 4	FB			JC/JHS	E3
TRAP 3	FC			JP/JGT	E4
TRAP 2	FD			JPZ/JGE	E5
TRAP 1	FE			JNZ/JNE	E6
TRAP 0	FF			JNC/JL	E7

TABLE 3-20 — TMS7000 NON-CORE (AVAILABLE FOR MICROCODE) INSTRUCTIONS

MNEMONIC	OP CODE	MNEMONIC	OP CODE
MPY Rn,A	1C	TRAP 23	E8
MPY %n,A	2C	TRAP 22	E9
MPY Rn,B	3C	TRAP 21	EA
MPY Rn,Rn	4C	TRAP 20	EB
MPY %n,B	5C	TRAP 19	EC
MPY B,A	6C	TRAP 18	ED
MPY %n,Rn	7C	TRAP 17	EE
		TRAP 16	FF
DAC Rn,A	1E	TRAP 15	F0
DAC %n,A	2E	TRAP 14	F1
DAC Rn,B	3E	TRAP 13	F2
DAC Rn,Rn	4E	TRAP 12	F3
DAC %n,B	5E	TRAP 11	F4
DAC B,A	6E	TRAP 10	F5
DAC %n,Rn	7E	TRAP 9	F6
		TRAP 8	F7
DSB Rn,A	1F	ANDP A,Pn	83
DSB %n,A	2F	ANDP B,Pn	93
DSB Rn,B	3F	ANDP %n,Pn	A3
DSB Rn,Rn	4F		
DSB %n,B	5F	ORP A,Pn	84
DSB B,A	6F	ORP B,Pn	94
DSB %n,Rn	7F	ORP %n,Pn	A4
MOVD %n,Rn	88	XORP A,Pn	85
MOVD Rn,Rn	98	XORP B,Pn	95
MOVD %n(B),Rn	A8	XORP %n,Pn	A5
DECD A	BB	BTJOP A,Pn	86
DECD B	CB	BTJOP B,Pn	96
DECD Rn	DB	BTJOP %n,Pn	A6
SWAP A	B7	BTJZP A,Pn	87
SWAP B	C7	BTJZP B,Pn	97
SWAP Rn	D7	BTJZP %n,Pn	A7
CMPA @n	8D	MOVP A,Pn	82
CMPA *Rn	9D	MOVP B,Pn	92
CMPA @n(B)	AD	MOVP %n,Pn	A2
XCHB A	B6	MOVP Pn,A	80
XCHB B	C6		
XCHB Rn	D6	MOVP Pn,B	91

3.5 INSTRUCTION DESCRIPTIONS

The assembler for the TMS7000 family will accept these instructions in the indicated Assembly Language format. The byte count for each instruction may be determined from its instruction type and its operands. Refer to Appendix A for specification for opcode assignment and instruction timing information.

The instruction descriptions are presented in alphabetic order. The discussion of each instruction includes mnemonic, syntax, instruction type, example, status bits affected, and some useful notes. All instructions may have optional labels before the mnemonic and comments after the operands. Label, mnemonics, operand field and comments must be separated by a space.

All of the logical instructions, AND, OR, XOR, ORP ... follow the tables below. These functions operate on the eight bits in the source and their corresponding bits in the destination. The AND function is useful in clearing bits, the OR function can set bits to '1' and the XOR function can toggle the bits from '1' to '0' or from '0' to '1'.

LOGICAL FUNCTIONS

— AND —			— OR —			— XOR —		
Src	Dst	Rst	Src	Dst	Rst	Src	Dst	Rst
0	0	0	0	0	0	0	0	0
0	1	0	0	1	1	0	1	1
1	0	0	1	0	1	1	0	1
1	1	1	1	1	1	1	1	0

Src = source bit
 Dst = destination bit
 Rst = result bit

3.5.1 ADC Add with Carry ADC

SYNTAX: ADC <s>, <d>

EXECUTION RESULTS: (s) + (d) + C → (d)

EXAMPLE: LABEL ADC R66,R117 Adds the contents of register 66 to register 117 plus the carry.

TYPE: Dual Register

STATUS C — set to '1' on carry-out of (s) + (d) + C

BITS: Z — set on result

 N — set on result

ADC with an immediate operand of zero value is equivalent to a conditional increment of the destination operand. ADC may also be used to implement multi-precision addition of signed or unsigned integers. For example, the 16-bit integer in register pair (R2,R3) may be added to the 16-bit integer in (A,B) as follows:

ADD	R3,B	Low order bytes added
ADC	R2,A	High order bytes added

3.5.2

ADD**Add****ADD**

SYNTAX: **ADD** <s>, <d>

EXECUTION RESULTS: (s) + (d) \rightarrow (d)

EXAMPLE: **LABEL** **ADD** A, B Adds the contents of the A and B register and stores the results in B

TYPE: Dual Register

STATUS C — '1' on carry-out of (s) + (d)

BITS: Z — set on result

 N — set on result

ADD is used to add two bytes, and may be used for signed two's complement or unsigned addition.

3.5.3

AND**And****AND**

SYNTAX: **AND** <s>, <d>

EXECUTION RESULTS: (s) .AND. (d) \rightarrow (d)

EXAMPLE: **AND** % > 1, R12 Clear all bits in R12 except bit 0

TYPE: Dual Register

STATUS C \leftarrow 0

BITS: N — set on result

 Z — set on result

AND is used to perform a logical AND of the two operands. Each bit of the 8-bit result follows the truth table which is at the beginning of this section. AND is useful in clearing or resetting bits. If a bit needs to be cleared in the destination, then a '0' is put at that bit location in the source. A '1' in the source will cause the bit in the destination to remain the same.

3.5.4**ANDP****And Peripheral Register****ANDP**

SYNTAX: ANDP <s>, <d>

EXECUTION RESULTS: (s).AND.(p) → (p)

EXAMPLE: CLRBIT ANDP %>DF,P6 Clear bit 5 of the B port

TYPE: Dual Peripheral File

STATUS C ← 0

BITS: N — set on result

Z — set on result

ANDP may be used to clear one or more bits in the peripheral file. Thus, it may be used to reset an individual output line to zero. This may be done with an ANDP instruction where the source is an immediate operand that serves as a mask field. The example above shows how bit 5 of the B Port (P6) is cleared. The only valid source operands are A, B and %iop.

3.5.5**BTJO****Bit Test and Jump if One****BTJO**

SYNTAX: BTJO <s>, <d>, <offset>

EXECUTION RESULTS: If (s).AND.(D) <> 0, then PC + (offset) → PC

EXAMPLE: BITSET BTJO %>14,R4,ISSET Jump if R4 (bit 2) or
R4 (bit 4) is a '1'

TYPE: Dual Relative

STATUS C ← 0

BITS: N — set on (s).AND.(d)

Z — set on (s).AND.(d)

Use the BTJO instruction to test for at least one bit which has a corresponding '1' bit in each operand. For example, the source operand can be used as a bit mask to test for one or more '1' bits in the destination address. The operands are not changed by this instruction.

3.5.6 BTJOP Bit Test and Jump if One Peripheral BTJOP

SYNTAX: BTJOP <s>, <p>, <offset>

EXECUTION RESULTS: If (s).AND.(p) <> 0, then PC + (offset) → PC

EXAMPLE: LABEL BTJOP % >81,P4,THERE Jump if Port A(bit 0) or Port A(bit 7) is '1'

.TYPE: Peripheral-Relative

STATUS C ← 0

BITS: N — set on (s).AND.(p)
Z — set on (s).AND.(p)

Use the BTJOP instruction to test for at least one bit position which has a corresponding '1' in each operand. For example, the source operand can be used as a bit mask to test for at least one '1' bit in the destination peripheral file register. The example above tests bit 0 and bit 7 of the input A port, and jumps if either is a '1'.

3.5.7 BTJZ Bit Test and Jump if Zero BTJZ

SYNTAX: BTJZ <s>, <d>, <offset>

EXECUTION RESULTS: if (s).AND.(NOT d) <> 0, then PC + (offset) → PC

EXAMPLE: ISZERO BTJZ A,R23,ZERO If any '1' bits in A correspond to '0' bits in R23 then jump

.TYPE: Dual Relative

STATUS C ← 0

BITS: N — set on (s).AND.(NOT d)
Z — set on (s).AND.(NOT d)

Use the BTJZ instruction to test for at least one 0 bit in the destination operand which has a corresponding '1' bit in the source operand. The operands are not changed by the instruction.

3.5.8 BTJZP Bit Test and Jump if Zero Peripheral BTJZP

SYNTAX: BTJZP <s>, <d>, <offset>

EXECUTION RESULTS: if (s).AND.(NOT d) <> 0, then PC + (offset) → PC

EXAMPLE: LABEL BTJZP % > 21, P4, THERE Jump if P4(bit 0) or P4(bit 5) is '0'

TYPE: Peripheral Relative

STATUS C ← 0

BITS: N — set on (s).AND.(NOT d)
Z — set on (s).AND.(NOT d)

Use the BTJZP instruction to test for at least one bit position which has a '1' in the source and an '0' in the peripheral file register. For example, the source operand can be used as a bit mask to test for zero bits in the destination peripheral file register. The example above tests bit 0 and bit 5 of the input A port, and jumps if either is a '0'. The jump is calculated starting from the opcode of the instruction just after the BTJZP. The operands are unchanged by this instruction.

3.5.9 BR Branch BR

SYNTAX: BR <d>

EXECUTION RESULTS: (d) → PC

EXAMPLES: LABEL BR @THERE Direct addressing
BR @TABLE(B) Indexed addressing
BR *R14 Indirect addressing

TYPE: Extended Address

STATUS

BITS: Not changed

BR may be used to branch to ANY location in the the 64K memory space including the Register space. This extended address type instruction supports three different modes. The powerful concept of computed GOTO's is supported by the BR *Rn instruction. An indexed branch instruction of the form BR @TABLE(B) is an extremely efficient way of executing one of several actions on the basis of some control input. This is similar to the CASE statement of Pascal and other high-level languages. For example, suppose register R3 contains a control value. The program can branch to label ACTION0 if R3 = 0, ACTION1 if R3 = 1, etc, for up to 128 different actions. This technique may also be used to transfer control on character inputs, error codes, etc. See section 6.3.5 for examples.

3.5.10

CALL

Call

CALL

SYNTAX: CALL <d>

EXECUTION RESULTS:

SP + 1	→	SP
PC MS Byte	→	stack
SP + 1	→	SP
PC LS Byte	→	stack
operand address	→	PC

EXAMPLES: LABEL1 CALL @LABEL4 Direct addressing
 CALL @LABEL5(B) Indexed addressing
 CALL *R12 Indirect addressing

TYPE: Extended Address

STATUS

BITS: Not changed

CALL is used to invoke a subroutine. The PUSH and POP instructions can be used to save, pass, or restore status or register values. The extended addressing modes of the CALL instruction allow powerful transfer of control functions.

3.5.11

CLR

Clear

CLR

SYNTAX: CLR <d>

EXECUTION RESULTS: 0 → (d)

EXAMPLE: ZEROIT CLR B

TYPE: Single Register

STATUS C ← 0

BITS: N ← 0

Z ← 1

CLR is used to clear or initialize any file register including the A and B registers.

3.5.12 CLRC Clear the Carry bit CLRC

SYNTAX: CLRC
EXECUTION RESULTS: status bits set

EXAMPLE: LABEL CLRC

TYPE: Implied Operand

STATUS C ← 0
BITS: N — set on value of A register
Z — set on value of A register

CLRC is used to clear the carry flag if required before an arithmetic or rotate instruction. Note that the logical and move instructions typically clear the Status carry bit. The CLRC opcode is equivalent to the TSTA opcode.

3.5.13 CMP Compare CMP

SYNTAX: CMP <s>, <d>

EXECUTION RESULTS: (d) - (s) computed

EXAMPLE: LABEL CMP R13, R89

TYPE: Dual Register

STATUS C — '1' if (d) is logically greater than
or equal to (s)
BITS: N — Sign of result
Z — '1' if (d) is equal to (s)

CMP is used to compare the destination operand to the source operand. For a complete discussion of this instruction see 6.3.1.1.

3.5.14 CMPA Compare Accumulator Extended CMPA

SYNTAX: CMPA <s>

EXECUTION RESULTS: (A) - (s) computed but not stored

EXAMPLE: LABEL CMPA @TABLE2 Direct addressing
CMPA @TABLE(B) Indexed
CMPA *R123 Indirect

TYPE: Extended Address

STATUS C — '1' if (A) is logically greater than or
equal to (s)
BITS: N — '1' if (A) is arithmetically less than (s)
Z — '1' if (A) is equal to (s)

CMPA may be used to compare a long-addressed operand (e.g., via direct, indirect, or indexed addressing modes) to the A register. It is especially useful in table lookup programs in which the table is stored either in extended memory or in program ROM. The status bits are set exactly as if register A were the destination (d) and the addressed byte the source (s).

3.5.15 DAC**Decimal Add With Carry****DAC**

SYNTAX: DAC <s>, <d>

EXECUTION RESULTS: (s) + (d) + C → (d) Decimal

EXAMPLE: LABEL DAC %>24,A Add the packed BCD value 24 to Accum.

TYPE: Dual Register

STATUS C — '1' if value of (s) + (d) + C ≥ 100

BITS: N — set on result

Z — set on result

DAC is used to add bytes in binary-coded decimal (BCD) form. Each byte is assumed to contain two BCD digits. Operation of DAC is undefined for non-BCD operands. DAC with an immediate operand of zero value is equivalent to a conditional increment of the destination operand. The DAC instruction automatically performs a decimal adjust on the binary sum of (s) + (d) + C. The carry bit is added to facilitate adding multi-byte BCD strings, and so the carry bit must be cleared before execution of the first DAC instruction.

3.5.16 DEC**Decrement****DEC**

SYNTAX: DEC <d>

EXECUTION RESULTS: (d) - 1 → (d)

EXAMPLE: LABEL DEC R102

TYPE: Single Register

STATUS C — '0' if (d) decrements from >00 to >FF;

BITS: '1' otherwise.

N — set on result

Z — set on result

The DEC instruction is used to subtract a value of '1' from any addressable operand. The DEC instruction is also useful in counting and addressing byte arrays.

3.5.17**DECD****Decrement Double****DECD**

SYNTAX: DECD <rp>

TYPE: Single Register

EXAMPLE: LABEL DECD R51 Decrement (R50,R51) register pair
R51 = LSB

EXECUTION RESULTS: (rp) - 1 → (rp)

STATUS C ← '0' if most significant byte decrements from
BITS: >00 to >FF. Otherwise, C = '1'.
 N — set on most significant byte of result
 Z — set on most significant byte of result

DECD may be used to decrement 16-bit indirect addresses stored in the register file. Tables longer than 256 bytes may be scanned using this instruction.

3.5.18**DINT****Disable Interrupts****DINT**

SYNTAX: DINT

EXECUTION RESULTS: 0 → interrupt enable status bit

EXAMPLE: LABEL DINT

TYPE: Implied Operand

STATUS I ← 0
BITS: C ← 0
 N ← 0
 Z ← 0

DINT is used to turn off all interrupts simultaneously. Since the interrupt enable flag is stored in the status register, the POP ST; RETI or LDSP instructions may reenable interrupts even though a DINT instruction has been executed. During the interrupt service, the interrupt enable bit is automatically cleared after the old status register value has been pushed onto the stack.

3.5.19 DJNZ Decrement Register And Jump If Not-Zero DJNZ

SYNTAX: DJNZ <d>, <offset>

EXECUTION RESULTS: (d)-1→(d); if (d) <> 0, then PC + (offset) → PC

EXAMPLE: LABEL DJNZ R15, THERE

TYPE: Single-Relative

STATUS

BITS: Not changed

The DJNZ instruction is used for looping control. Combines the DEC and the JNZ instruction together to give a more compact and faster instruction. This instruction does not change any of the status bits.

3.5.20 DSB Decimal Subtract With Borrow DSB

SYNTAX: DSB <s>, <d>

EXECUTION RESULTS: (d) - (s) - 1 + C → (d) Decimal

EXAMPLE: LABEL DSB R15, R76

TYPE: Dual Register

STATUS C — '1' no borrow required, '0' if borrow required

BITS: N — set on result

Z — set on result

DSB is used for multiprecision decimal BCD subtraction. A DSB instruction with an immediate operand of zero value is equivalent to a conditional decrement of the destination operand. The carry status bit functions as a borrow bit, so if no borrow in is required, the carry bit should be set to '1'. This can be accomplished by executing the SETC instruction.

3.5.21**EINT****Enable Interrupts****EINT**

SYNTAX: EINT

EXECUTION RESULTS: 1 → interrupt enable

EXAMPLE: LABEL EINT

TYPE: Implied Operand

STATUS I ← 1
 BITS: C ← 1
 N ← 1
 Z ← 1

EINT is used to turn on all enabled interrupts simultaneously. Since the interrupt enable flag is stored in the status register, the POP ST, LDST, and RETI instructions may disable interrupts even though a EINT instruction has been executed. During the interrupt service, the interrupt enable bit is automatically cleared after the old status register value has been pushed onto the stack. Thus, the EINT instruction must be included inside the interrupt service routine to permit nested or multilevel interrupts.

3.5.22**IDLE****Idle until Interrupt****IDLE**

SYNTAX: IDLE

EXECUTION RESULTS: pc → pc until interrupt
 pc + 1 → pc after return from interrupt

EXAMPLE: LABEL IDLE

TYPE: Implied Operand

STATUS
 BITS: Not changed

IDLE is used to allow the program to suspend operation until either an interrupt or reset occurs. It is the programmer's responsibility to assure that the interrupt enable status bit (and individual interrupt enable bits in the I/O control register) are set before executing the IDLE instruction. Upon return from an interrupt, control passes to the instruction following the IDLE instruction.

3.5.23

INC

Increment

INC

SYNTAX: INC <d>

EXECUTION RESULTS: $(d) + 1 \rightarrow (d)$

EXAMPLE: LABEL INC A

TYPE: Single Register

STATUS C ← '1' if (d) incremented from >FF to >00;

BITS: '0' otherwise.

N — set on result

Z — set on result

INC is used to increment the value of any register. It is useful in incrementing counters into tables.

3.5.24

INV

Invert

INV

SYNTAX: INV <d>

EXECUTION RESULTS: NOT (d) → (d)

EXAMPLE: LABEL INV A

TYPE: Single Register

STATUS C ← 0

BITS: N — set on result

Z — set on result

INV performs a logical or one's complement of the operand. A two's complement of the operand can be made by following the INV instruction with an increment (INC). A one's complement reverses the value of every bit in the destination.

3.5.25**JMP****Jump unconditional****JMP**

SYNTAX: **JMP** <offset>

EXECUTION RESULTS: PC + (offset) → PC The PC is taken from the instruction after the JMP

EXAMPLE: **LABEL** **JMP** **THERE**

TYPE: Simple Relative

STATUS

BITS: Not changed

Jump unconditionally to the address specified in the operand. The second byte of the JMP instruction is loaded with the 8-bit relative address of the operand. The operand address must therefore be within - 128 to + 127 bytes of the location of the instruction following the JMP instruction. The assembler will indicate an error if the target address is beyond - 128 to + 127 bytes from the next instruction. For a longer jump the BR (branch) instruction can be used.

3.5.26**J<cond>****Jump On Condition****J<cond>**

SYNTAX: **J<cond>** <offset>

EXECUTION RESULTS: If tested condition is true, PC + offset → PC

EXAMPLES: **LABEL** **JNC** **THERE**
 LABEL **JP** **HERE**

TYPE: Simple Relative

STATUS

BITS: Not affected

TABLE 3-21 — CONDITIONAL JUMP INSTRUCTIONS

INSTRUCTION	MNEMONIC	CONDITION FOR JUMP (STATUS BIT VALUES)		
		CARRY	NEGATIVE	ZERO
Jump If Carry	JC	1	X	X
Jump If Equal	JEQ	X	X	1
Jump If Higher Or Same	JHS	1	X	X
Jump If Lower	JL	0	X	X
Jump If Negative	JN	X	1	X
Jump If No Carry	JNC	0	X	X
Jump If Not Equal	JNE	X	X	0
Jump If Non-zero	JNZ	X	X	0
Jump If Positive	JP	X	0	0
Jump If Positive Or Zero	JPZ	X	0	X
Jump If Zero	JZ	X	X	1

The J <cond> instructions may be used after a CMP instruction to branch according to the relative values of the operands tested. After MOV, MOV_P, LDA, or STA operations, a JZ or JNZ may be used to test if the value moved was equal to zero. JN and JPZ may be used in this case to test the sign bit of the value moved. For a more complete description of the Jump instructions see 6.3.1.1.

3.5.27

LDA

Load A register

LDA

SYNTAX: LDA <s>

EXECUTION RESULTS: (s) → Addr

EXAMPLES: LABEL1 LDA @LABEL4 Direct
 LABEL2 LDA @LABEL5(B) Indexed
 LABEL3 LDA *R13 Indirect

TYPE: Extended Address

STATUS C ← 0

BITS: N — set on value loaded
 Z — set on value loaded

The LDA instruction is used to read values stored anywhere in the full 64K memory space. The direct addressing mode provides an efficient means of directly accessing a variable in memory. Indexed addressing gives an efficient table look-up capability for most applications. Indirect addressing allows the use of very large look-up tables and the use of multiple memory pointers since any pair of registers can be used as the pointer. The DJNZ (Decrement and Jump if Non-Zero) instruction can be used with either indexed or indirect addressing to create fast and efficient program loops or table searches.

3.5.28**LDSP****Load Stack Pointer****LDSP**

SYNTAX: LDSP

EXECUTION RESULTS: (B) → SP

EXAMPLE: LABEL LDSP

TYPE: Implied Operand

STATUS

BITS: Not changed

Copy the contents of the B register to the stack pointer register. LDSP is used to initialize the stack pointer.

3.5.29**MOV****Move****MOV**

SYNTAX: MOV <s>, <d>

TYPE: Dual Register

EXECUTION RESULTS: (s) → (d)

EXAMPLES:	LABEL1	MOV	A,B	Move the contents of A reg. to B reg.
	LABEL2	MOV	R32,R105	Move the contents of R32 to R105
	LABEL3	MOV	%10,R3	Move the value 10 to R3

STATUS C ← 0

BITS: N — set on value loaded

Z — set on value loaded

MOV is used to transfer values within the register space. Immediate values may be loaded into registers directly from the instruction. The fact that the A or B register is an operand is implied in the MOV opcode, resulting in shorter and quicker moves from the A or B register.

3.5.30

MOVD**Move Double****MOVD**SYNTAX: **MOVD** <s>, <d>

EXECUTION RESULTS: (rp) → (rp)

EXAMPLE:	LABEL	MOVD	% >1234,R3	Load Register Pair R2,R3 with >1234
		MOVD	R5,R3	Copy R4,R5 to R2,R3 R5,R3 = LSB
		MOVD	%TAB(B),R3	Copy indexed address to R2,R3

TYPE: Miscellaneous

STATUS **C** ← 0BITS: **N** — set on most significant byte moved**Z** — set on most significant byte moved

MOVD moves a two-byte value to the register pair indicated by the destination register number. The destination is the second operand of the instruction and it points to the LSB of the destination register pair. The source may be a 16-bit constant, another register pair, or an indexed address. For the latter case, the source must be of the form "%ADDR(B)" where ADDR is a 16-bit constant or address. This 16-bit value is added (via 16-bit addition) to the contents of the B register, and the result placed in the destination register pair. This will store an indexed address into a register pair, for use later in indirect addressing mode.

3.5.31

MOVP**Move To/From Peripheral File****MOVP**SYNTAX: **MOVP** <s>, <d>

EXECUTION RESULTS: <s> → <d>

EXAMPLE:	SETIMR	MOVP	A,P2	Setup timer value
	RDPORT	MOVP	P4,B	Read Port A data

TYPE: Peripheral File

STATUS **C** ← 0BITS: **N** — set on value moved**Z** — set on value moved

MOVP is used to transfer values to and from the peripheral file. This may be used to input or output 8-bit quantities on the I/O ports. The peripheral file also contains control registers for the interrupt lines, the I/O ports, and the timer controls. The operands supported by this instruction are A, B and %iop.

During peripheral file instructions, a peripheral file port is read. The read can include output operations each as 'MOV A,P6'. If this read is undesirable because of hardware configuration, a STA (Store A) instruction with the memory-mapped address of the peripheral register can be used.

3.5.32 MPY Multiply MPY

SYNTAX: MPY <s>, <d>

EXECUTION RESULTS: (s) X (d) → (A,B) Result always stored in A,B

EXAMPLE: LABEL MPY R3,A Multiply R3 and A
 LABEL2 MPY %32,B Shift B register 5 places left

TYPE: Dual Register

STATUS C ← 0

BITS: N — set on most significant byte of results (A register)

Z — set on most significant byte of results (A register)

MPY performs an 8-bit multiply for a general source and destination operand. The 16-bit result is placed in the 'A,B' register pair with the most significant byte in A. Multiplying by a power of two is a convenient means of performing double-byte shifts. If a double byte shift is three places or less, then it may be faster to use RLC or RRC instead of multiply. If a single byte needs shifting then it is almost always faster to use RLC or RRC.

3.5.33 NOP No Operation NOP

SYNTAX: NOP

EXECUTION RESULTS: PC + 1 → PC

EXAMPLE: LABEL NOP

TYPE: Implied Operand

STATUS

BITS: Not changed

NOP is useful as a pad instruction during program development, to "patch out" unwanted or erroneous instructions or to leave room for code changes during development. It is also useful in software timing loops.

3.5.34

OR

Or

OR

SYNTAX: OR <s>, <d>

EXECUTION RESULTS: (s) .OR. (d) → (d)

EXAMPLE: LABEL OR A,R12 Or the A register with R12, Store in R12
 SETBIT OR %>0FA Set lower nibble of A to '1's, leave
 upper nibble unchanged

TYPE: Dual Register

STATUS C ← 0

BITS: N — set on result

Z — set on result

OR is used to perform a logical OR of the two operands. Each bit of the 8-bit result follows the truth table at the beginning of this section. The OR operation is used to set bits in a register. If a register needs a '1' in the destination then a '1' is placed in the corresponding bit location in the source operand.

3.5.35

ORP

OR Peripheral File Register

ORP

SYNTAX: ORP <s>, <d>

EXECUTION RESULTS: (s) .OR. (d) → (d)

EXAMPLE: LABEL ORP A,P12

TYPE: Peripheral File

STATUS C ← 0

BITS: N — set on result

Z — set on result

ORP is used to perform a logical OR of the source operand with a peripheral file location, and write the result back to the peripheral file. This may be used to set an individual I/O bit of a peripheral register. Since the peripheral file is read before it is ORed, it may not work with some peripheral locations which have different function when reading then when writing.

3.5.36 POP**POP From Stack****POP**

SYNTAX: POP <d>

EXECUTION RESULTS: Stack top → (d) Move value then decrement SP
 SP - 1 → SP

EXAMPLES: GETIT POP R32
 PUTBCK POP ST

TYPE: Single Register
 "POP ST" Special, see below

STATUS C ← 0
 BITS: N — set on value POPed
 Z — set on value POPed

The data stack can be used to save or to pass values, especially during subroutines and interrupt service routines. The POP instruction pulls a value from the stack. The status register may be replaced with the contents on the stack by the statement: POP ST. This one-byte instruction is usually executed in conjunction with a previously performed "PUSH ST" instruction.

3.5.37 PUSH**Push On Stack****PUSH**

SYNTAX: PUSH <d>

EXECUTION RESULTS: SP + 1 → SP; Increment SP then move value
 (d) → (stack top)

EXAMPLES: STORE1 PUSH A
 SAVEST PUSH ST

TYPE: Single Register
 "PUSH ST" Special, see below

STATUS C ← 0
 BITS: N — set on value PUSHed
 Z — set on value PUSHed

The data stack is used to save or pass values, especially during subroutines and interrupt service routines. The PUSH instruction places a value on the stack. The Status register may be pushed on the stack with the statement: PUSH ST. This one-byte instruction is usually executed in conjunction with a subsequently performed "POP ST" instruction. The status register is unaffected.

3.5.38**RETI****Return From Interrupt****RETI**

SYNTAX: RETI

EXECUTION RESULTS: Stack → PC LS Byte
 SP - 1 → SP
 Stack → PC MS Byte
 SP - 1 → SP
 Stack → ST
 SP - 1 → SP

EXAMPLE: LABEL RETI

TYPE: Implied Operand

STATUS Status Register
 BITS: is loaded from
 the stack

RETI is typically the last instruction in an interrupt service routine. RETI restores the status register to its state immediately before the interrupt occurred and branches back to the program at the instruction boundary where the interrupt occurred. The A and B registers, if used, must be restored to original values before the RETI instruction.

3.5.39**RETS****Return From Subroutine****RETS**

SYNTAX: RETS

EXECUTION RESULTS: Stack → PC LS Byte
 SP - 1 → SP
 Stack → PC MS Byte
 SP - 1 → SP

EXAMPLE: LABEL RETS

TYPE: Implied Operand

STATUS
 BITS: Not changed

RETS is typically the last instruction in a subroutine. RETS results in a branch to the location immediately following the subroutine call instruction. In the called subroutine there must be an equal number of POPs and PUSHes so that the stack is pointing to the return address and not some other data.

3.5.40

RL

Rotate Left

RL

SYNTAX: RL <d>

EXECUTION RESULTS: Bit(n) → Bit(n + 1)
 Bit(7) → Bit(0) and Carry

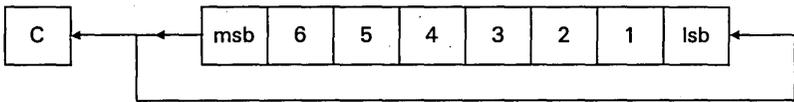
EXAMPLE: LABEL RL R102

TYPE: Single Register

STATUS C – set to bit 7 of the original operand

BITS: N – set on result

Z – set on result



An example of the RL instruction is: If the B register contains the value >93, then the RL instruction changes the contents of B to >27 and sets the carry status bit.

3.5.41

RLC

Rotate Left Through Carry

RLC

SYNTAX: RLC <d>

EXECUTION RESULTS: Bit(n) → Bit(n + 1)
 Carry → Bit(0)
 Bit(7) → Carry

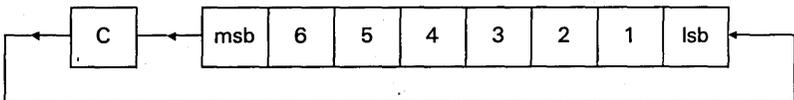
EXAMPLE: LABEL RLC R72

TYPE: Single Register

STATUS C set to bit 7 of the original operand

BITS: N – set on result

Z – set on result



An example of the RLC instruction is: if the B register contains the value >93 and the carry status bit is a zero, then the RLC instruction changes the operand value to >26 and carry to one. Rotating left effectively multiplies the value by 2. Using multiple rotates, any power of 2 (2, 4, 8, 16...) can be achieved. This type of multiply is usually faster than the MPY (multiply) instruction. This instruction is also useful in rotates where a value is contained in more than one byte such as an address or in multiplying a large multibyte number by 2. Care must be taken to assure that the carry is at the proper value. The SETC or CLRC instructions may be use to setup the correct value.

3.5.42 RR

Rotate Right

RR

SYNTAX: RR <d>

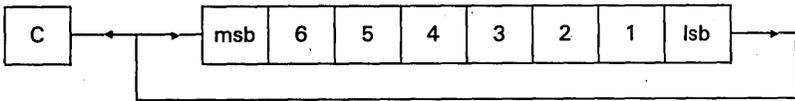
EXECUTION RESULTS: + 1 → Bit(n)
 Bit(0) → Bit (7) and carry

EXAMPLE: LABEL RR A

TYPE: Single Register

STATUS C – set to bit 0 of the original value

BITS N – set on result
 Z – set on result



An example of the RR instruction is: If the B register contains the value >93, then the "RR B" instruction changes the contents of B to >C9 and sets the carry status bit.

3.5.43 RRC

Rotate Right Through Carry

RRC

SYNTAX: RRC <d>

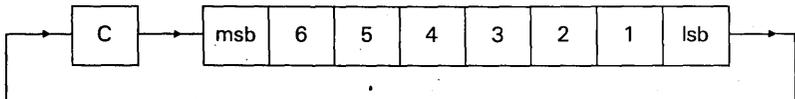
EXECUTION RESULTS: Bit(n + 1) → Bit(n)
 Carry → Bit(7)
 Bit(0) → Carry

EXAMPLE: LABEL RRC R32

TYPE: Single Register

STATUS C – set to bit 0 of the original value

BITS N – set on result
 Z – set on result



An example of the RRC instruction is: If the B register contains the value >93 and the carry status bit is zero, then the 'RRC B' instruction changes the operand value to >49 and sets the carry status bit. When the carry is '0' this instruction effectively divides the value by 2. A value of >80 becomes >40. By using this instruction more once, the value can be divided by any power of 2. Care must be taken to assure the correct value in the carry bit.

3.5.44**SBB****Subtract With Borrow****SBB**

SYNTAX: SBB <s>, <d>

EXECUTION RESULTS: $(d) - (s) - 1 + C \rightarrow (d)$

EXAMPLE: LABEL SBB %23,B Subtract 23 from B register

TYPE: Dual Register

STATUS C — set to '1' if no borrow; '0' otherwise

BITS: N — set on result.

Z — set on result.

SBB is used for multiprecision two's complement subtraction. An SBB instruction with an immediate operand of zero value is equivalent to a conditional decrement of the destination operand. With $(s) = 0$, and $C = '0'$, then (d) is decremented, otherwise it is unchanged. A borrow occurs if the result is negative. In this case, the carry bit is set to '0'.

3.5.45**SETC****Set Carry****SETC**

SYNTAX: SETC

EXECUTION RESULTS: $1 \rightarrow \text{carry}$

EXAMPLE: LABEL SETC

TYPE: Implied Operand

STATUS C ← 1

BITS: N ← 0

Z ← 1

SETC is used to set the carry flag if required before an arithmetic or rotate instruction.

3.5.46

STA**Store A Register****STA**

SYNTAX: STA <d>

EXECUTION RESULTS: (A) → (d)

EXAMPLES: LABEL1 STA @LABEL4 Direct addressing
 LABEL2 STA @LABEL5(B) Indexed
 LABEL3 STA *R13 Indirect

TYPE: Extended Address

STATUS C ← 0

BITS: N — set on value loaded
 Z — set on value loaded

The STA instruction is used to store values anywhere in the 64K memory address space. The direct addressing provides an efficient means of directly accessing a variable in general memory. The indexed addressing provides an efficient table look-up capability. Indirect addressing allows the use of very large look-up tables and the use of multiple memory pointers since any pair of registers can be used as the pointer. The Decrement Register and Jump if Non-Zero instruction (DJNZ) can be used with either indexed or indirect addressing to create fast and efficient program loops or table searches.

3.5.47

STSP**Store Stack Pointer****STSP**

SYNTAX: STSP

EXECUTION RESULTS: (SP) → (B)

EXAMPLE: LABEL STSP

TYPE: Implied Operand

STATUS

BITS: Not changed

STSP is used to make a copy of the SP if required. This instruction can be used to test the stack size. The indexed addressing mode may be used to reference operands on the stack. Ex: STSP then LDA @>0000(B) will put the present value on top of the stack into A register.

3.5.48 SUB Subtract SUB

SYNTAX: SUB <s>, <d>

EXECUTION RESULTS: (d) - (s) → (d)

EXAMPLE: LABEL SUB R19,B

TYPE: Dual Register

STATUS C — set to '1' if result >= 0, '0' otherwise

BITS: N — set on result

Z — set on result

SUB is used for two's complement subtraction. The carry bit is set to '0' if a borrow is required. The carry bit could be renamed a "No-Borrow" bit in this case.

3.5.49 SWAP Swap Nibbles SWAP

SYNTAX: SWAP <d>

EXECUTION RESULTS: bits(7,6,5,4, 3,2,1,0) → bits(3,2,1,0, 7,6,5,4)

EXAMPLE: LABEL SWAP R45

TYPE: Single Register

STATUS C — set to bit 0 of the result or bit 4 of the original

BITS: N — set on results

Z — set on results

SWAP exchanges the first four bits with the second four bits. This instruction is equivalent to 4 consecutive RL (rotate left) instructions. It is used to manipulate four bit operands, especially during packed BCD operations.

3.5.50

TRAP

Trap To Subroutine

TRAP

SYNTAX: TRAP <n> n = 0 - 23

EXECUTION RESULTS:

SP + 1	→	SP
PC MS Byte	→	stack
SP + 1	→	SP
PC LS Byte	→	stack
Entry vector	→	PC

EXAMPLE: LABEL TRAP 15

TYPE: Miscellaneous

STATUS

BITS: not changed

The operand <n> is a trap number which identifies a location in the Trap Vector Table, addresses >FFD0 to >FFFF in memory. The contents of the two-byte vector location form a 16-bit trap vector to which a subroutine call is performed. TRAP is an efficient way to invoke a subroutine. The highest block of memory is the Trap Vector Table, and contains as many subroutine addresses as available traps for the TMS7000 family member. The subroutine addresses are stored like all other addresses in memory, with the least significant byte in the higher-addressed location, as shown below.

TRAP VECTOR TABLE

>FFD0	Trap 23 address msb
>FFD1	Trap 23 address lsb
...	...
>FFE0	Trap 15 address msb
>FFE1	Trap 15 address lsb
...	...
>FFFA	Trap 2 address msb
>FFFB	" lsb
>FFFC	Trap 1 address msb
>FFFD	" lsb
>FFFE	Trap 0 address msb
>FFFF	Trap 0 address lsb

Note that TRAPs 0, 1, 2 and 3 correspond to the hardware-invoked interrupts 0, 1, 2, and 3 respectively. The hardware-invoked interrupts, however, push the program counter and the status register before branching to the interrupt routine, while the TRAP instruction pushes only the program counter. TRAP 0 will branch to the same code executed for a system reset but will not set or clear all the registers like the hardware RESET. For more information see Section 6.3.3.

3.5.51 TSTA Test A Register TSTA

SYNTAX: TSTA

EXECUTION RESULTS: C,N,Z bits set

EXAMPLE: LABEL TSTA Test A register

TYPE: Implied Operand

STATUS C ← 0

BITS: N — set on value in A register

Z — set on value in A register

This instruction can be used to set the status bits according to the value in the A register. This instruction is equivalent to the CLRC (Clear Carry) instruction.

3.5.52 TSTB Test B Register TSTB

SYNTAX: TSTB

EXECUTION RESULTS: C,N,Z bits set

EXAMPLE: LABEL TSTB Test B Register

TYPE: Implied Operand

STATUS C ← 0

BITS: N — set on value in B register

Z — set on value in B register

This instruction can be used to set the status bits according to the value in the B register. It may be used to clear the carry bit. This instruction is equivalent to the XCHB B (exchange B with B) instruction.

3.5.53 XCHB Exchange With B Register XCHB

SYNTAX: XCHB <d>

EXECUTION RESULTS: (B) ↔ (d)

EXAMPLE: LABEL XCHB A exchange B register with A register
XCHB R3 exchange B register with R3

TYPE: Single Register

STATUS C ← 0

BITS: N — set on original contents of B

Z — set on original contents of B

XCHB is used to exchange a register with the B register without going through an intermediate location. The XCHB instruction with the B register as the operand is equivalent to the TSTB instruction.

3.5.54

XOR**Exclusive Or****XOR**

SYNTAX: XOR <s>, <d>

EXECUTION RESULTS: (s) .XOR. (d) → (d)

EXAMPLE: LABEL XOR R98, R125
 XOR %1, R20 Toggle bit 0 in R20

TYPE: Dual Register

STATUS C ← 0

BITS: N — set on result
 Z — set on result

XOR is used to perform a bit-wise exclusive OR operation on the operands. The XOR instruction can be used to complement bits in the destination operand. Each bit of the 8-bit result follows the truth table shown at the beginning of this section. This operation can also toggle a bit in a register. If the bit value in the destination needs to be the opposite from what it currently is, then the source should contain a '1' in that bit location.

3.5.55

XORP**Exclusive Or Peripheral File****XORP**

SYNTAX: XORP <s>, <d>

EXECUTION RESULTS: (s) .XOR. (d) → (d)

EXAMPLE: LABEL XORP % >01, P9 Reverse direction of pin C(0)

TYPE: Peripheral File

STATUS C ← 0

BITS: N — set on result
 Z — set on result

XORP is used to perform a bit-wise exclusive OR operation on the operands. The XORP instruction can be used to complement bits in the destination PF register. The example above inverts bit 0 of P9, which is the port C data direction register, thus reversing the direction of the pin.

4. ELECTRICAL SPECIFICATIONS

4.1 TMS7000/TMS7020/TMS7040/TMS70120/TMS7001/TMS7041

4.1.1 Description Of The TMS7000/TMS7020/TMS7040/TMS70120/TMS7001/TMS7041

The TMS70X0 devices (TMS7000, TMS7020, TMS7040, and TMS70120) are single chip 8-bit microcomputers containing a CPU, timer, I/O, RAM, and various amounts of on-chip ROM. The TMS7020 contains the CPU, RAM, timer, and I/O on-chip, and also provides 2K bytes of on-chip ROM. The TMS7040 offers the same features as the TMS7020 and has an increased on-chip ROM size of 4K bytes. The TMS70120 offers the same features as the general family and efficiently handles large programs with 12K bytes of on-chip ROM. The TMS7000 family member contains the same features of the TMS7020 except it contains no on-chip ROM.

The TMS70X1 devices (TMS7001 and TMS7041) contain a flexible on-chip serial port in addition the CPU, timer, I/O, and on-chip RAM and ROM. The TMS7041 contains 4K bytes of on-chip ROM, while the TMS7001 has no on-chip ROM.

Each member in the TMS70X0 and TMS70X1 families have 128 bytes of on-chip RAM, and all have the capability through memory expansion modes, to access up to 64K bytes of address space. For additional information on the TMS7000 family architecture, refer to Section 2.

Table 4-1 depicts the TMS70X0 and TMS70X1 family features.

TABLE 4-1 - TMS70X0 AND TMX70X1 FAMILY FEATURES

FEATURES	FAMILY MEMBERS					
	7000	7020	7040	70120	7001	7041
ON-CHIP ROM (BYTES)	NONE	2K	4K	12K	NONE	4K
ON-CHIP RAM (BYTES)	128	128	128	128	128	128
INTERRUPT LEVELS	4	4	4	4	6	6
TIMERS	13-BIT	1	1	1	2	2
	10-BIT	0	0	0	1	1
I/O LINES: BI-DIRECTIONAL INPUT ONLY OUTPUT ONLY	16	16	16	16	22	22
	8	8	8	8	2	2
	8	8	8	8	8	8
ADDITIONAL I/O	-	-	-	-	SERIAL PORT	SERIAL PORT
PROCESS TECHNOLOGY	NMOS	NMOS	NMOS	NMOS	NMOS	NMOS

Unless otherwise indicated the following specifications for the TMS7000 apply to the TMS7020, TMS7040, TMS70120, TMS7001, and TMS7041.

4.1.2 Key Features

- Microprogrammable instruction set
- Strip Chip Architecture Topology (SCAT) for rapid family expansion
- Register-to-register architecture
- Family members with 2K, 4K, and 12K bytes of on-chip ROM and ROMless versions
- On-chip 8-bit timer/event counter with 5-bit prescale:
 - Internal interrupt with automatic reload
 - Capture latch
 - Second 8-bit timer/event counter with 5-bit prescale and cascade capability (TMS7001 and TMS7041 only)
- Flexible on-chip serial port (TMS7001 and TMS7041 only)
 - Fully software programmable
 - Internal or external baud rate generator
 - Separate baud rate timer usable as a third timer
 - Asynchronous, isosynchronous, or serial modes
 - Two multiprocessor communication formats
- 128-byte RAM register file
- Full-feature data/program stack
- 32 TTL-compatible I/O pins:
 - 16 bi-directional pins (22 bi-directional pins on TMS7001 and TMS7041)
 - 8 output pins
 - 8 high-impedance input pins (2 input pins on TMS7001 and TMS7041)
- Memory-mapped ports for easy addressing
- 256-byte peripheral file
- Memory expansion capability:
 - 64K byte address space
- 8-bit instruction word
- Eight powerful addressing formats including:
 - Register-to-register arithmetic
 - Indirect addressing on any register pair
 - Indexed and indirect branches and calls
- Two's complement arithmetic
- Single-instruction binary coded decimal (BCD) add and subtract
- Two external maskable interrupts
- Flexible interrupt handling:
 - Priority servicing of simultaneous interrupts
 - Software execution of hardware interrupts
 - Precise timing of interrupts with the capture latch
 - Software monitoring of interrupt status
- Accurate pulse width measurement and modulation
- N-channel silicon gate MOS, 5-volt power supply
- 40-pin, 600-mil, dual-in-line package
- 100-mil or 70-mil pin-to-pin spacing packages

4.1.3 Absolute Maximum Ratings Over Operating Free-Air Temperature Range (Unless Otherwise Noted)[†]

Supply voltage, V_{CC} (See Note 1)	−0.3 V to 7 V
All input voltages	−0.3 V to 20 V
All output voltages	−0.3 V to 7 V
Continuous power dissipation	1 W
Operating free-air temperature range	0°C to 70°C
Storage temperature range	−55°C to 150°C

[†] Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the “Recommended Operating Conditions” section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: Unless otherwise noted, all voltages are with respect to V_{SS} .

4.1.4 Recommended Operating Conditions

PARAMETER		MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}		4.5	5	5.5	V
High-level input voltage, V_{IH}	CLOCKIN	2.6		$V_{CC} + 0.5$	V
	MC	14			V
	All others	2		$V_{CC} + 0.5$	V
Low-level input voltage, V_{IL}	CLOCKIN			0.6	V
	All others			0.8	V
High-level output current, I_{OH}				−400	μ A
Low-level output current, I_{OL}				10	mA
Operating free-air temperature, T_A		0		70	°C

4.1.5 Electrical Characteristics Over Full Range of Operating Conditions

PARAMETER	TEST CONDITIONS	MIN	TYP [†]	MAX	UNIT	
I_I	Input current, Port A INPUT-only pins		± 2	± 10	μ A	
I_I	Input current, I/O pins		± 10	± 100	μ A	
C_I	Input capacitance		2		pF	
V_{OH}	High-level output voltage		2.4	2.8	V	
V_{OL}	Low-level output voltage		0.2	0.4	V	
$t_{r(O)}$	Output rise time [‡]		9	50	ns	
$t_{f(O)}$	Output fall time [‡]		10	60	ns	
I_{CC}	Supply current	All outputs open		80	150	mA
$P_{D(av)}$	Average power dissipation	All outputs open		400	825	mW

[†] All typical values are at $V_{CC} = 5$ V, $T_A = 25^\circ\text{C}$.

[‡] Rise and fall times are measured between the maximum low level and the minimum high level using the 10% and 90% points (see Figure 4-2). Outputs have 100-pF loads to V_{SS} .

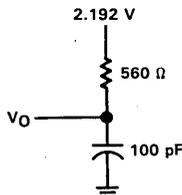


FIGURE 4-1 — OUTPUT LOADING CIRCUIT FOR TEST

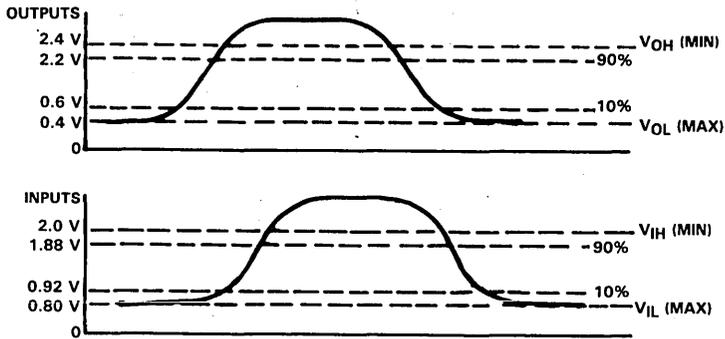


FIGURE 4-2 – MEASUREMENT POINTS FOR SWITCHING CHARACTERISTICS

4.1.6 Recommended CRYSTAL/CLOCKIN Operating Conditions Over Full Operating Range

PARAMETER		MIN	TYP	MAX	UNIT
f_{osc}	CRYSTAL/CLOCKIN frequency (divide-by-4 option)	2.0	10.1		MHz
f_{osc}	CRYSTAL frequency (divide-by-2 option) (see Note 1)	1.0	5.05		MHz
$t_c(P)$	CRYSTAL/CLOCKIN cycle time (divide-by-4 option)	99	500		ns
$t_c(P)$	CRYSTAL cycle time (divide-by-2 option)	198	1000		ns
$t_c(S)$	Internal state cycle time	396	2000		ns
$t_w(PH)$	CLOCKIN pulse width high	45			ns
$t_w(PL)$	CLOCKIN pulse width low	45			ns
t_r	CLOCKIN rise time [†]			30	ns
t_f	CLOCKIN fall time [†]			30	ns
$t_d(PH-CL)$	CLOCKIN rise to CLOCKOUT rise delay		125	200	ns

[†] Rise and fall times are measured between the maximum low level and the minimum high level using the 10% and 90% points (see Figure 4-2). Outputs have 100-pF loads to V_{SS} .

NOTE 1: Divide-by-4 option recommended with external clock drive.

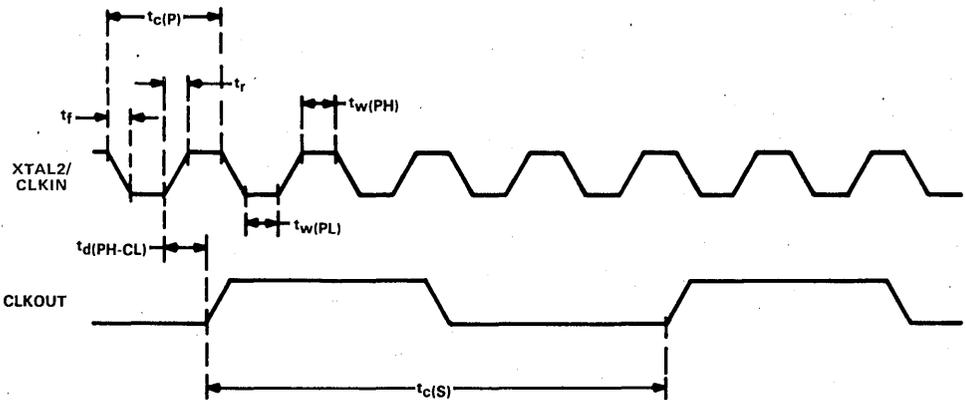
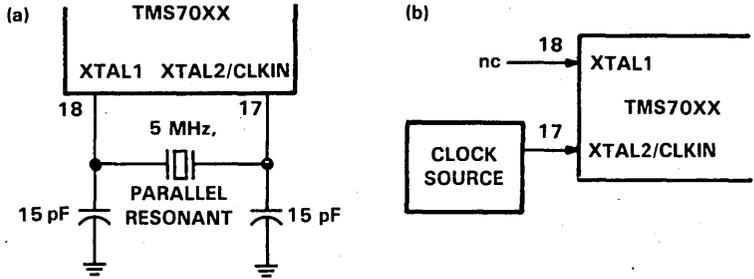


FIGURE 4-3 – CLOCK TIMING



NOTES: The divide-by-2 input can be used with XTAL only. Divide-by-4 can be used with XTAL or CLKIN inputs. Alternative use of ceramic resonators is illustrated in Section 4.1.8.

FIGURE 4-4 – RECOMMENDED CLOCK CONNECTIONS

4.1.7 Memory Interface Timing At 10 MHz Over Full Operating Free-Air Temperature Range

PARAMETER		MIN	TYP	MAX	UNIT
$t_c(C)$	CLOCKOUT cycle time (see Note)	400		2000	ns
$t_w(CH)$	CLOCKOUT high pulse width	130	170	200	ns
$t_w(CL)$	CLOCKOUT low pulse width	150	190	240	ns
$t_d(CH-JL)$	CLOCKOUT rising to ALATCH falling edge	260	300	340	ns
$t_d(CH-EL)$	CLOCKOUT rising to ENA falling	-10	15	50	ns
$t_w(JH)$	ALATCH high pulse width	150	190	230	ns
$t_d(AH-JL)$	High address valid before ALATCH fall	50	170	220	ns
$t_d(AL-JL)$	Low address valid before ALATCH fall	50	150	220	ns
$t_h(JL-AL)$	Low address hold after ALATCH fall	30	45	80	ns
$t_d(RW-JL)$	RD/WR valid before LATCH fall	50	140	200	ns
$t_h(EH-RW)$	RD/WR hold after ENA rise	40	100		ns
$t_h(EH-AH)$	High address hold after ENA rise	30	40		ns
$t_h(EH-Q)$	Data out hold after ENA rise	65	80		ns
$t_d(Q-EH)$	Data out valid before ENA rise	230	290		ns
$t_d(AF-EL)$	ENA fall after low address HI-Z	0	30	120	ns
$t_d(EH-AF)$	ENA rising to next address drive	60	85		ns
$t_d(EL-D)$	Data in after ENA falling	155	190		ns
$t_h(EH-D)$	Data in hold after ENA rise	0			ns
$t_d(A-D)$	Access time, data in from valid address	400	470		ns
$t_d(A-EH)$	ENA high after address valid	580		730	ns

NOTE: $t_c(C)$ is defined to be $4/f_{osc}$ (or $2/f_{osc}$ if the divide-by-2 option is selected) and may be referred to as a machine state or simply a state.

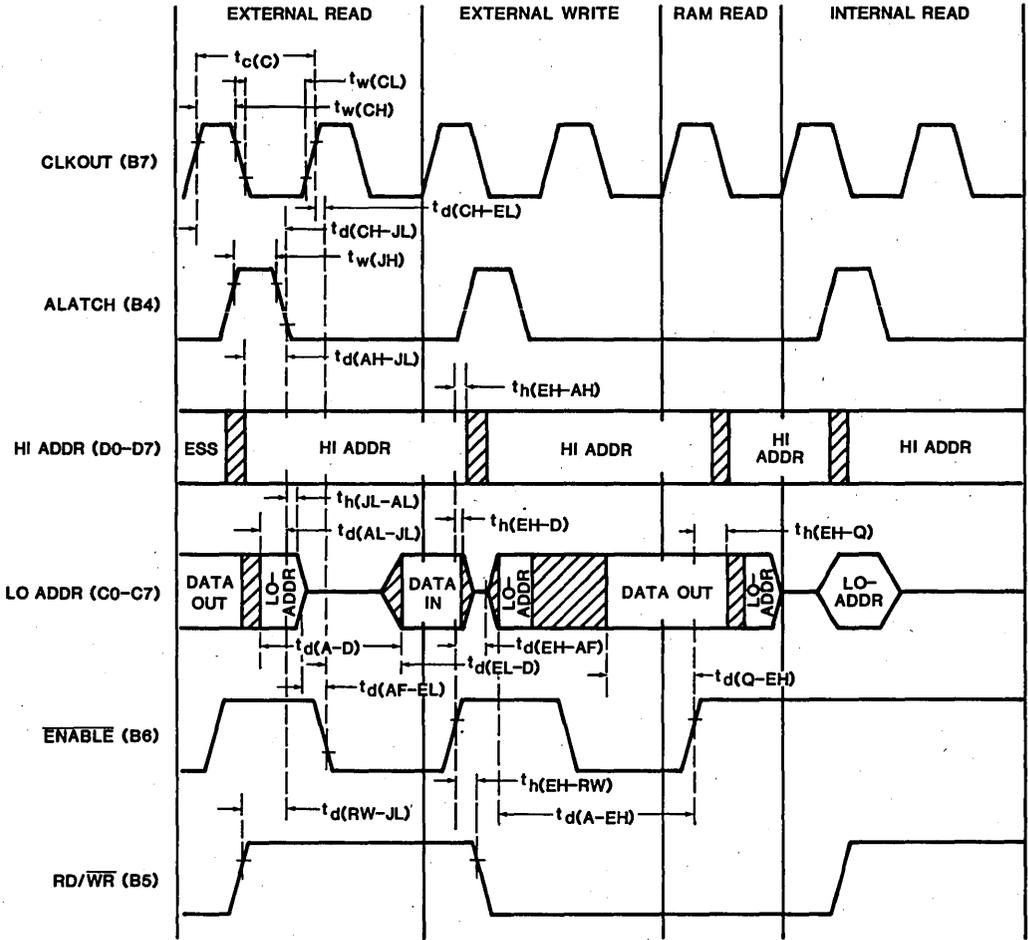


FIGURE 4-5 — READ AND WRITE CYCLE TIMING

4.1.8 Application of Ceramic Resonator

The resonant circuit shown in Figure 4-6 provides an economical alternative to quartz crystals where frequency tolerance is not a major concern. Frequency tolerance over temperature is about 1%.

Ceramic resonator suppliers.

MURATA CORPORATION OF AMERICA
 1148 Franklin Rd. SE.
 Marietta, GA. 30067
 404/952-9777
 Telex: 0542329 Murata ATL

For 5 MHz operation
 Resonator ceralock CSA5.00MT
 Resistor 1 MΩ 10%
 Capacitors (both) 30 pF

NGK SPARK PLUGS (USA) INC.
 20608 Madrona Ave.
 Torrance, CA 90503
 213/328-6882
 Telex: 664290

For 5 MHz operation
 Resonator R5.0M
 Resistor 1 MΩ 10%
 Capacitors 68 pF ± 10%

KYOCERA INTERNATIONAL
 8611 Balboa Ave.
 San Diego, CA 92123
 714/279-8310
 Telex: 697929

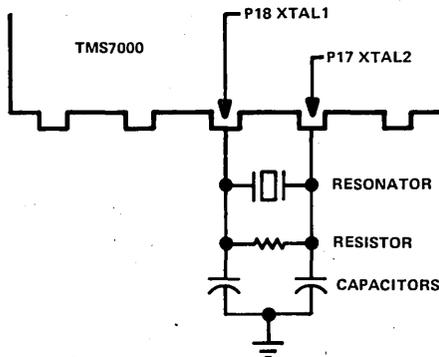
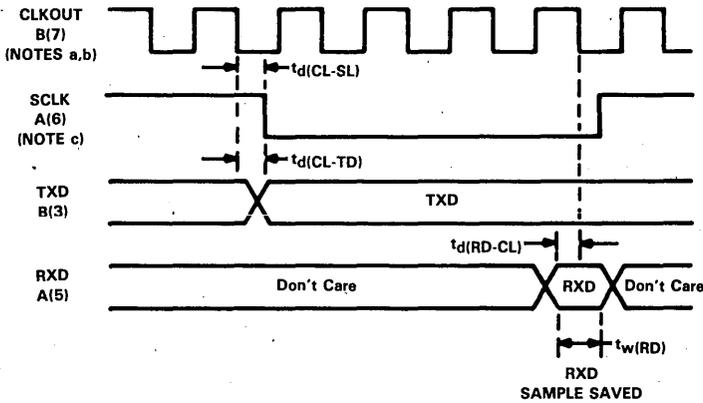


FIGURE 4-6 — CERAMIC RESONATOR CIRCUIT

4.1.9 Serial Port Timing (TMS7001, TMS7041, And SE70P161 Only)

4.1.9.1 Internal Serial Clock

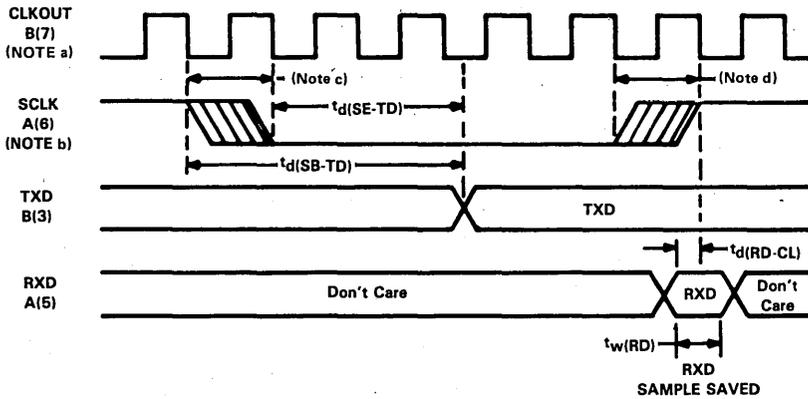


NOTES:

- The CLKOUT signal is not available in Single-Chip mode.
- $CLKOUT = t_c(C) = \emptyset$
- Example shows $SCLK = \emptyset/8$.

PARAMETER		TYP	UNIT
$t_d(CL-SL)$	CLKOUT low to SCLK low	$1/4 t_c(C)$	ns
$t_d(CL-TD)$	CLKOUT low to new TXD data	$1/4 t_c(C)$	
$t_d(RD-CL)$	RXD data valid before CLKOUT low	$1/4 t_c(C)$	
$t_w(RD)$	RXD data valid time	$1/2 t_c(C)$	

4.1.9.2 External Serial Clock

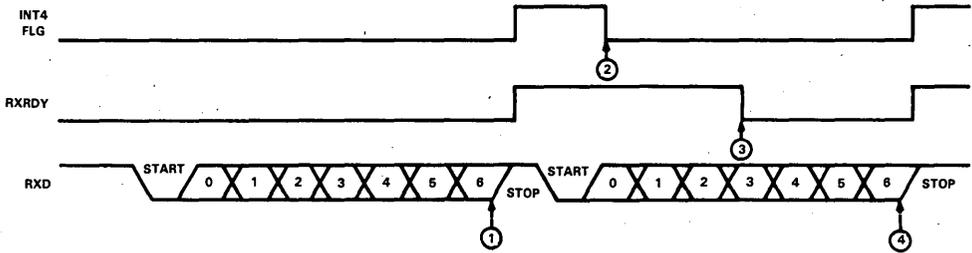


NOTES:

- The CLKOUT signal is not available in Single-Chip mode.
CLKOUT = $t_c(C) = \emptyset$
- Example shows SCLK = $\emptyset/10$.
- SCLK sampled; if 1 then 0, fall transition found.
- SCLK sampled; if 0 then 1, rise transition found.

PARAMETER	TYP	UNIT
$t_d(RD-CL)$	RXD data valid before CLKOUT low	$1/4 t_c(C)$
$t_w(RD)$	RXD data valid time	$1/2 t_c(C)$
$t_d(SB-TD)$	Start of SCLK sample to new TXD data	$3/4 t_c(C)$
$t_d(SE-TD)$	End of SCLK sample to new TXD data	$2/4 t_c(C)$

4.1.9.3 Rx Signals In Communication Modes



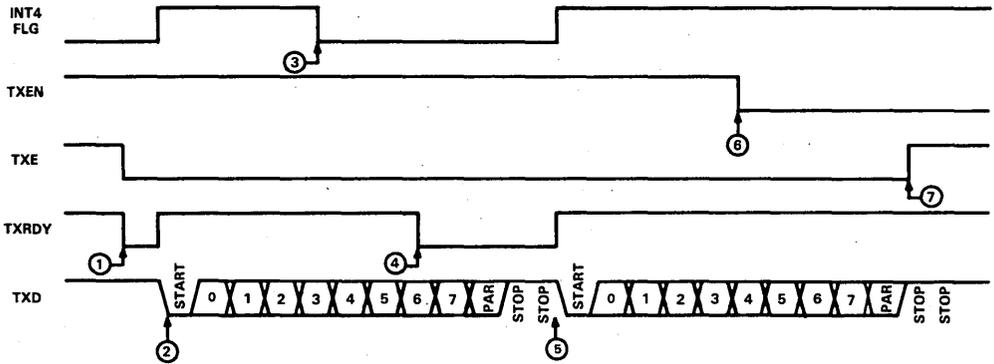
NOTES:

- a) Format shown is start bit + seven data bits + stop bits.
- b) SCLK is continuous, external or internal.
- c) User means user software executed by CPU.
- d) If $RXEN = 0$, RXSHF still receives data from RXD. However, the data is not transferred to RXBUF and RXRDY and INT4 FLG are not set.

SEQUENCE OF EVENTS

- 1) RXSHF data is transferred to RXBUF. Error status bits are set if an error is detected.
- 2) { User writes to INT4 CLR to clear INT4 FLG. If not, CPU clears.
- 3) { INT4 FLG on entry to Level 4 interrupt routine.
- 4) User reads RXBUF.

4.1.9.4 Tx Signals In Communication Modes



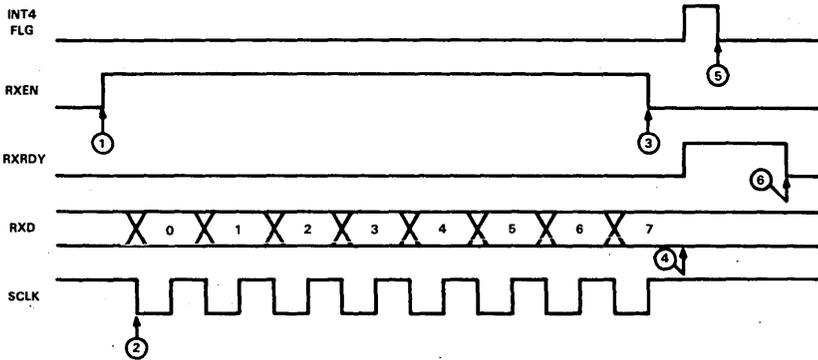
NOTES:

- a) Format shown is start plus eight data parity bits plus two stop bits.
- b) SCLK is continuous whether internal or external.
- c) User means user software executed by CPU.

SEQUENCE OF EVENTS

- 1) } User writes to TXBUF.
- 4) } TXBUF and WU data is transferred to TXSHF and WUT and
- 2) } INT4 FLG and TXRDY are set.
- 5) } User resets TXEN; current frame will finish and transmission will stop whether TXBUF is full or empty.
- 6) } TXE is set if TXBUF and TXSFT are empty.
- 7) } User writes to INT4 CLR to clear INT4 FLG or CPU clears INT4 FLG on entry to level 4 interrupt routine.
- 3) }

4.1.9.5 Rx Signals in Serial I/O Modes



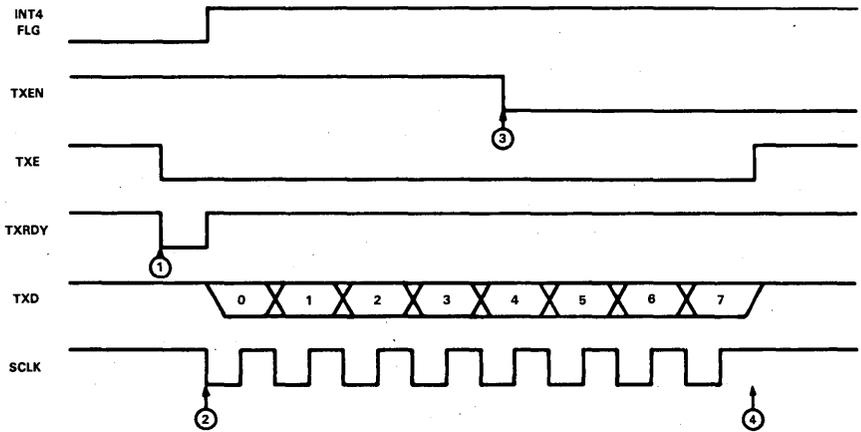
NOTES:

- a) RXEN has no effect on INT4 FLG or RXRDY in serial I/O mode.
- b) RXD is sampled on SCLK rise; external shift registers should be clocked on SCLK fall.
- c) The SCLK source should be internal as it is gated by internal circuitry.

SEQUENCE OF EVENTS

- 1) User starts receiving by setting RXEN.
- 2) Gated SCLK starts and data is received.
- 3) RXEN is automatically cleared in last data bit.
- 4) RXSHF data is transferred to RXBUF and RXRDY and INT4 are set.
- 5) User writes to INT4 CLR to clear INT4 FLG; if not CPU clears INT4 FLG on entry to level 4 interrupt routine.
- 6) User reads RXBUF.

4.1.9.6 Tx Signals in Serial I/O Modes



NOTES:

- a) Format shown is eight data bits.
- b) The SCLK source should be internal as it is gated by internal circuitry.

SEQUENCE OF EVENTS

- 1) User writes to TXBUF.
- 2) TXBUF data is transferred to TXSFT; INT4 FLG and TXRDY are set and SCLK starts.
- 3) User resets TXEN, current frame will finish and transmission will halt whether TXBUF is full or empty.
- 4) Frame ends and SCLK stops because TXEN = 0.

4.1.10 Pin Descriptions

4.1.10.1 Pin Description of The TMS7000/TMS7020/TMS7040/TMS70120

Figure 4-7 defines the pin assignments and describes the function of each pin for the Single-Chip (SC), Peripheral Expansion (PE), Full Expansion (FE), and Microprocessor Modes for the TMS70X0 family (TMS7000, TMS7020, TMS7040, TMS70120).

SIGNATURE	PIN	I/O	DESCRIPTION
A0 (LSB)	6	IN	I/O Port A: Input lines
A1	7	IN	(Specific I/O configuration for;
A2	8	IN	Single Chip Mode — see Section 2.3.1,
A3	9	IN	Peripheral Expansion Mode — see
A4	10	IN	Section 2.3.2, Full Expansion
A5	16	IN	Mode — see Section 2.3.3, Micro-
A6	15	IN	processor Mode — see Section 2.3.4)
A7 (MSB)	11	IN	
B0 (LSB)	3	OUT	I/O Port B: Output lines
B1	4	OUT	(Specific I/O configuration for;
B2	5	OUT	Single Chip Mode — see Section 2.3.1,
B3	37	OUT	Peripheral Expansion Mode — see
B4/ALATCH	38	OUT	Section 2.3.2, Full Expansion
B5/R/W	1	OUT	Mode — see Section 2.3.3,
B6/ENABLE	39	OUT	Microprocessor Mode — see Section
B7/CLOCKOUT	2	OUT	2.3.4)
C0 (LSB)	28	I/O	I/O Port C: General purpose bidirectional
C1	29	I/O	lines (Specific I/O configuration for; Single
C2	30	I/O	Chip Mode — see Section 2.3.1, Peripheral
C3	31	I/O	Expansion Mode — see Section 2.3.2, Full
C4	32	I/O	Expansion Mode — see Section 2.3.3,
C5	33	I/O	
C6	34	I/O	Microprocessor Mode — see Section 2.3.4)
C7 (MSB)	35	I/O	
D0 (LSB)	27	I/O	I/O Port D: General purpose
D1	26	I/O	bidirectional lines (Specific
D2	24	I/O	I/O Configurations for; Single
D3	23	I/O	Chip Mode — see Section 2.3.1,
D4	22	I/O	Peripheral Expansion Mode — see
D5	21	I/O	Section 2.3.2, Full Expansion Mode —
D6	20	I/O	see Section 2.3.3, Microprocessor
D7 (MSB)	19	I/O	Mode — see Section 2.3.4)
INT1	13	IN	Maskable Interrupt
INT3	12	IN	Maskable Interrupt
RESET	14	IN	RESET
MC	36	IN	Mode Control
XTAL2/CLKIN	17	IN	Crystal input for control of internal OSC.; input pin for external OSC. or LRC networks
XTAL1	18	IN	Crystal input for control of internal OSC.; leave open for external OSC.
VCC	25	IN	Supply voltage (+5V)
VSS	40	IN	Ground reference

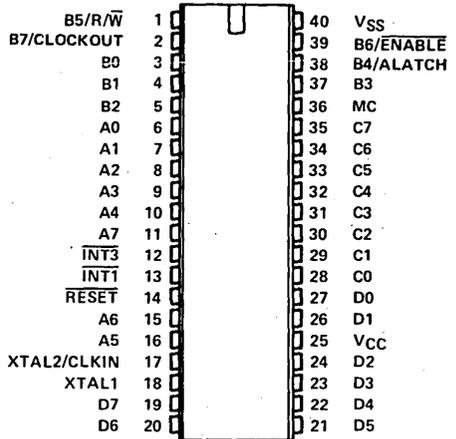


FIGURE 4-7 — SC, FE, PE, AND MICROPROCESSOR MODE PIN ASSIGNMENTS

4.1.10.2 Pin Description Of The TMS7001/TMS7041

Figure 4-8 defines the pin assignments and describes the function of each pin for the Single-Chip (SC), Peripheral Expansion (PE), Full Expansion (FE), and Microprocessor Modes for the TMS70X1 family (TMS7001 and TMS7041)

SIGNATURE	PIN	I/O	DESCRIPTION
A0 (LSB)	6	I/O	I/O Port A: General Purpose Bidirectional lines
A1	7	I/O	(Specific I/O configuration for:
A2	8	I/O	Single Chip Mode — see Section 2.3.1,
A3	9	I/O	Peripheral Expansion Mode — see
A4	10	I/O	Section 2.3.2, Full Expansion
A5/RXD	16	IN	Mode — see Section 2.3.3, Micro-
A6/SCLK	15	I/O	processor Mode — see Section 2.3.4)
A7	11	IN	
B0 (LSB)	3	OUT	I/O Port B: General purpose Output lines
B1	4	OUT	(Specific I/O configuration for:
B2	5	OUT	Single Chip Mode — see Section 2.3.1,
B3/TXD	37	OUT	Peripheral Expansion Mode — see
B4/ALATCH	38	OUT	Section 2.3.2, Full Expansion
B5/R/W	1	OUT	Mode — see Section 2.3.3,
B6/ENABLE	39	OUT	Microprocessor Mode — see Section
B7/CLOCKOUT	2	OUT	2.3.4)
C0 (LSB)	28	I/O	I/O Port C: General purpose bidirectional
C1	29	I/O	lines (Specific I/O configuration for: Single
C2	30	I/O	Chip Mode — see Section 2.3.1, Peripheral
C3	31	I/O	Expansion Mode — see Section 2.3.2, Full
C4	32	I/O	Expansion Mode — see Section 2.3.3,
C5	33	I/O	
C6	34	I/O	Microprocessor Mode — see Section 2.3.4).
C7 (MSB)	35	I/O	
D0 (LSB)	27	I/O	I/O Port D: General purpose
D1	26	I/O	bidirectional lines (Specific
D2	24	I/O	I/O Configuration for: Single
D3	23	I/O	Chip Mode — see Section 2.3.1,
D4	22	I/O	Peripheral Expansion Mode — see
D5	21	I/O	Section 2.3.2, Full Expansion Mode —
D6	20	I/O	see Section 2.3.3, Microprocessor
D7 (MSB)	19	I/O	Mode — see Section 2.3.4).
INT1	13	IN	Maskable Interrupt
INT3	12	IN	Maskable Interrupt
RESET	14	IN	RESET
MC	36	IN	Mode Control
XTAL2/CLKIN	17	IN	Crystal input for control of internal OSC.; input pin for external OSC. or LRC networks
XTAL1	18	IN	Crystal input for control of internal OSC.; leave open for external OSC.
VCC	25	IN	Supply voltage (+5 V)
VSS	40	IN	Ground reference

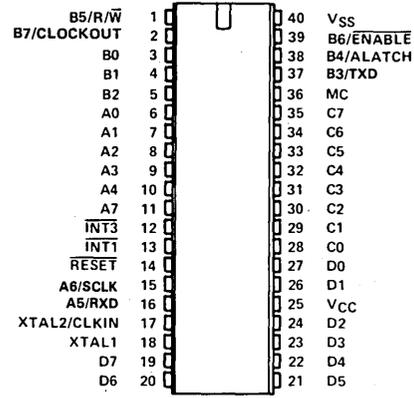


FIGURE 4-8 — SC, FE, PE, AND MICROPROCESSOR MODE PIN ASSIGNMENTS

4.2 TMS70C00/TMS70C20/TMS70C40

4.2.1 DESCRIPTION OF THE TMS70C00/TMS70C20/TMS70C40

The TMS70C00, TMS70C20, and TMS70C40 devices extend the TMS7000 family line into low power CMOS applications. They are single chip 8-bit microcomputers containing CPU, timers, I/O, and on-chip RAM and ROM. Table 4-2 presents the basic features of the present TMS70CXX family members.

The TMS70CXX family (TMS70C00, TMS70C20, and TMS70C40 devices) are fully software and pin compatible with their TMS70XX NMOS counterparts. They differ in the areas of interrupt operation, power down modes, input/output levels, operating voltage, and frequency range.

The TMS70CXX family maintains the four hardware interrupt levels of the TMS70XX family (RESET, TNT1, TNT2, and TNT3). The TMS70CXX family implements TNT1 as only a latched interrupt, not a latched and level interrupt as on the TMS70XX NMOS devices. The TMS70CXX family implements RESET, TNT2, and TNT3 in exactly the same manner as in the TMS70XX family (i.e., TNT3 is both latch and level sensitive). Refer to Section 2.5 for additional information on interrupt operation.

The TMS70CXX family supports two low power modes, the WAKE-UP mode and the HALT modes. Both of these modes are entered via execution of the IDLE instruction. The selection of the power down mode is determined by bit 5 of the timer 1 control register (T1CTRL) and then executing the IDLE instruction. The device is released from both power down modes through activation of RESET or acknowledgement of an enabled interrupt. Note that interrupts must be enabled in the status register and the I/O control register (IOCNT0) before the power down mode is entered for TNT1, TNT2 (timer), or TNT3 to be acknowledged. It is important that both power down modes provide RAM data retention.

Unless otherwise indicated, the following specifications for the TMS70C00 apply to the TMS70C20 and TMS70C40 as well.

TABLE 4-2 — TMS70CX0 FAMILY FEATURES

FEATURES	FAMILY MEMBER		
	70C00	70C20	70C40
ON-CHIP ROM (BYTES)	NONE	2K	4K
ON-CHIP RAM (BYTES)	128	128	128
INTERRUPT LEVELS	4	4	4
GENERAL PURPOSE INTERNAL REGISTERS	128	128	128
TIMERS	13-BIT	13-BIT	13-BIT
I/O LINES:			
BI-DIRECTIONAL	16	16	16
INPUT ONLY	8	8	8
OUTPUT ONLY	8	8	8
ADDITIONAL I/O	—	—	—
PROCESS TECHNOLOGY	CMOS	CMOS	CMOS

4.2.2 Key Features

- Microprogrammable instruction set
- Strip Chip Architecture Topology (SCAT) for rapid family expansion
- Register-to-register architecture
- Family members with 2K and 4K bytes of on-chip ROM and a ROMless version
- On-chip 8-bit timer/event counter with:
 - Programmable 5-bit prescale
 - Internal interrupt with automatic reloading
 - Capture latch
- 128-byte RAM register file
- Full-feature data/program stack
- 32 CMOS-compatible I/O pins:
 - 16 bi-directional pins
 - 8 output pins
 - 8 high-impedance input pins
- Memory-mapped ports for easy addressing
- Wide voltage operating range, frequency range
 - 3 V - 1 MHz typical
 - 5 V - 3.3 MHz typical
- Two software selectable low-power modes
- 256-byte peripheral file
- Memory expansion capability:
 - 64K byte address space
- 8-bit instruction word
- Eight powerful addressing formats including:
 - Register-to-register arithmetic
 - Indirect addressing on any register pair
 - Indexed and indirect branches and calls
- Two's complement arithmetic
- Single-instruction binary coded decimal (BCD) add and subtract
- Two external maskable interrupts
- Flexible interrupt handling:
 - Priority servicing of simultaneous interrupts
 - Software execution of hardware interrupts
 - Precise timing of interrupts with the capture latch
 - Software monitoring of interrupt status
- Accurate pulse width measurement and modulation
- Complementary silicon gate MOS
- 40-pin, 600-mil, dual-in-line package
- 100-mil or 70-mil pin-to-pin spacing packages

4.2.3 Absolute Maximum Rating Over Operating Free-Air Temperature Range (Unless Otherwise Noted)†

Supply voltage, V_{DD} (See Note 1)	−0.3 V to 7 V
All input voltages	−0.3 V to $V_{DD} + 0.3$ V
All output voltages	−0.3 V to $V_{DD} + 0.3$ V
Input current	+ 10 mA
Continuous power dissipation	0.5 W
Operating free-air temperature range	0°C to 70°C
Storage temperature range	−55°C to 150°C

† Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the “Recommended Operating Conditions” section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: Unless otherwise noted, all voltages are with respect to V_{SS} .

4.2.4 Recommended Operating Conditions

PARAMETER		MIN	NOM	MAX	UNIT
Supply voltage, V_{DD}		3		5.5	V
High-level input voltage, V_{IH}	$V_{DD} = 5$ V	$V_{DD} - 1$			V
	$V_{DD} = 4$ V	$V_{DD} - 0.7$			V
	$V_{DD} = 3$ V	$V_{DD} - 0.5$			V
Low-level input voltage, V_{IL}	$V_{DD} = 5$ V			1	V
	$V_{DD} = 4$ V			0.7	V
	$V_{DD} = 3$ V			0.5	V
Operating temperature range, T_A †		0		70	°C

† Plans are underway to extend the operating temperature range from −40°C to 85°C.

4.2.5 Electrical Characteristics Over Full Range Of Operating Conditions

PARAMETER		TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	$I_{OH} = -1$ mA, $V_{DD} = 5$ V		$V_{DD} - 2.5$	$V_{DD} - 0.5$		V
		$I_{OH} = -0.4$ mA, $V_{DD} = 5$ V		$V_{DD} - 0.5$	$V_{DD} - 0.2$		
V_{OL}	Low-level output voltage	$I_{OL} = 1.7$ mA, $V_{DD} = 5$ V			0.3	0.4	V
I_I	Input leakage current	$V_I = V_{DD}$, $V_{DD} = 5$ V				5	μA
I_{OH}	Source current	$V_{OH} = V_{DD} - 0.5$ V, $V_{DD} = 5$ V		−0.3	−1.2		mA
		$V_{OH} = V_{DD} - 0.5$ V, $V_{DD} = 4$ V		−0.2	−0.8		
		$V_{OH} = V_{DD} - 0.5$ V, $V_{DD} = 3$ V		−0.1	−0.5		
		$V_{OH} = 2.5$ V, $V_{DD} = 5$ V		−1	−4.5		
I_{OL}	Sink current	$V_{OL} = 0.4$ V, $V_{DD} = 5$ V		1.7	2.4		mA
		$V_{OL} = 0.4$ V, $V_{DD} = 4$ V		1.2	1.8		
		$V_{OL} = 0.4$ V, $V_{DD} = 3$ V		0.7	1		
I_{DD}	Supply current	Operating, $f_{osc} = 3$ MHz, $V_{DD} = 5$ V			5.5	8	mA
		Wake-up mode, $f_{osc} = 3$ MHz, $V_{DD} = 5$ V			500	800	
		Halt mode, $f_{osc} = 3$ MHz, $V_{DD} = 5$ V			250	550	
		Halt mode, XTAL/CLKIN = GND, all input = V_{DD} or GND, $V_{DD} = 5$ V			2	10	

† All typical values are at $V_{DD} = 5$ V, $T_A = 25$ °C.

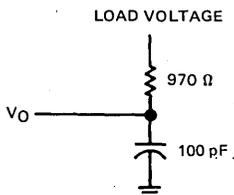


FIGURE 4-9 — OUTPUT LOADING CIRCUIT FOR TEST

4.2.6 AC Characteristics For Input/Output Ports

PARAMETER	TEST CONDITIONS	MIN	TYP †	MAX	UNIT
$t_{r(I/O)}$ I/O port output rise time †	$C_L = 15 \text{ pF}, V_{DD} = 5 \text{ V}$		50		ns
	$C_L = 50 \text{ pF}, V_{DD} = 5 \text{ V}$	70	110	150	
$t_{f(I/O)}$ I/O port output full time †	$C_L = 15 \text{ pF}, V_{DD} = 5 \text{ V}$		20		ns
	$C_L = 50 \text{ pF}, V_{DD} = 5 \text{ V}$	25	50	70	
$t_{t(I/O)}$ I/O port input rise/fall time †	$V_{DD} = 5 \text{ V}$			70	ns

† All typical values are at $V_{DD} = 5 \text{ V}, T_A = 25^\circ\text{C}$.

‡ Rise and fall times are measured between the maximum low level and the minimum high level using the 10% and 90% points (see Figure 4-11).

4.2.7 Recommended CRYSTAL/CLOCKIN Operating Conditions Over Full Operating Range

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
f_{osc} CRYSTAL frequency (see note 1)	$V_{DD} = 5 \text{ V}$	0.5		3.6	MHz
	$V_{DD} = 4 \text{ V}$	0.5		2.7	MHz
	$V_{DD} = 3 \text{ V}$	0.5		1.3	MHz
$t_{c(P)}$ CRYSTAL cycle time	$V_{DD} = 5 \text{ V}$	277		2000	ns
	$V_{DD} = 4 \text{ V}$	370		2000	ns
	$V_{DD} = 3 \text{ V}$	769		2000	ns
$t_{c(S)}$ Internal state cycle time	$V_{DD} = 5 \text{ V}$	554		4000	ns
	$V_{DD} = 4 \text{ V}$	740		4000	ns
	$V_{DD} = 3 \text{ V}$	1538		4000	ns
t_r CRYSTAL rise time †				30	ns
t_f CRYSTAL fall time †				30	ns
d_{osc} CRYSTAL duty cycle		45	50	55	%
$t_d(PL-CL)$ CRYSTAL fall to CLOCKOUT rise delay			100	200	ns

† Rise and fall times are measured between the maximum low level and the minimum high level using the 10% and 90% points.

NOTE 1: TMS70CXX family members currently use only the divide-by-two option as the INPUT CLOCK option.

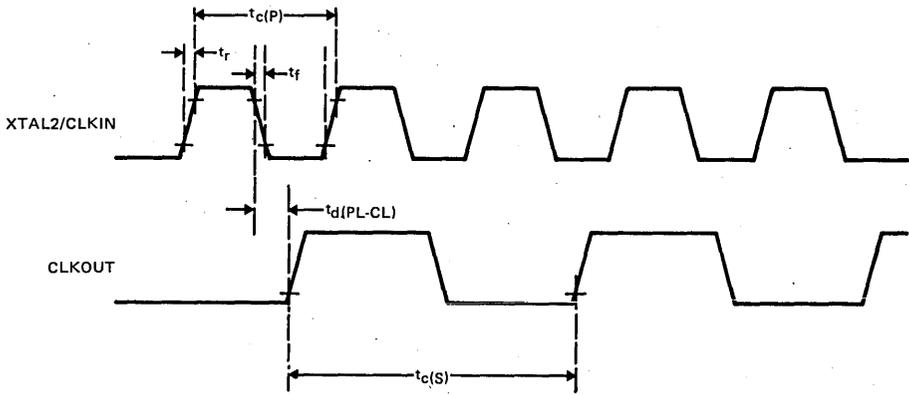


FIGURE 4-10 – CLOCK TIMING

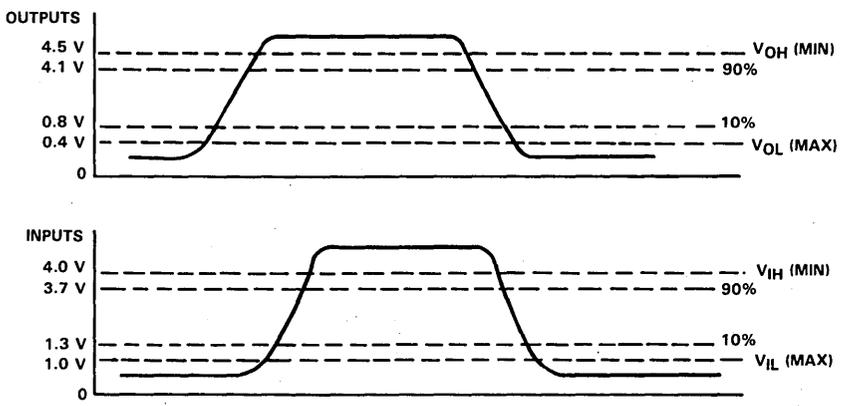


FIGURE 4-11 – MEASUREMENT POINTS FOR SWITCHING CHARACTERISTICS ($V_{DD} = 5V$)

4.2.8 Memory Interface Timing At $V_{DD} = 5\text{ V}$, $f_{osc} = 3\text{ MHz}$ Over The Full Operating Free-Air Temperature Range

PARAMETER		MIN	TYP	MAX	UNIT
$t_{c(C)}$	CLOCKOUT cycle time (see note)		665		ns
$t_{w(CH)}$	CLOCKOUT high pulse duration	260	340	470	ns
$t_{w(CL)}$	CLOCKOUT low pulse duration	190	270	360	ns
$t_d(CH-JL)$	CLOCKOUT rising to ALATCH falling edge	400	580		ns
$t_d(CH-EL)$	CLOCKOUT rising to $\overline{\text{ENABLE}}$ falling	30	60		ns
$t_{w(JH)}$	ALATCH high pulse duration	260	370		ns
$t_d(AH-JL)$	High address valid before ALATCH fall	230	330		ns
$t_d(AL-JL)$	Low address valid before ALATCH fall	220	320		ns
$t_d(JL-AL)$	Low address hold after ALATCH fall	110	160		ns
$t_d(RW-JL)$	RD/ $\overline{\text{WR}}$ valid before ALATCH fall	220	320		ns
$t_h(EH-RW)$	RD/ $\overline{\text{WR}}$ hold after $\overline{\text{ENABLE}}$ rise		170		ns
$t_h(EH-AH)$	High address hold after $\overline{\text{ENABLE}}$ rise		165		ns
$t_h(EH-Q)$	Data out hold after $\overline{\text{ENABLE}}$ rise	130	190		ns
$t_d(Q-EH)$	Data out valid before $\overline{\text{ENABLE}}$ rise	330	480		ns
$t_d(AF-EL)$	$\overline{\text{ENABLE}}$ fall after low address HI-Z	0	0	20	ns
$t_d(EH-AF)$	$\overline{\text{ENABLE}}$ rising to next address drive		130		ns
$t_d(EL-D)$	Data in after $\overline{\text{ENABLE}}$ falling			290	ns
$t_h(EH-D)$	Data in hold after $\overline{\text{ENABLE}}$ rise	0			ns
$t_d(A-D)$	Access time, data in from valid address			770	ns
$t_d(A-EH)$	$\overline{\text{ENA}}$ high after address valid	800	1150		ns

NOTE: TMS70CXX family members use a cycle time, $t_{c(C)}$, that is equal to $2/f_{osc}$ and is referred to as a machine state or simply a state.

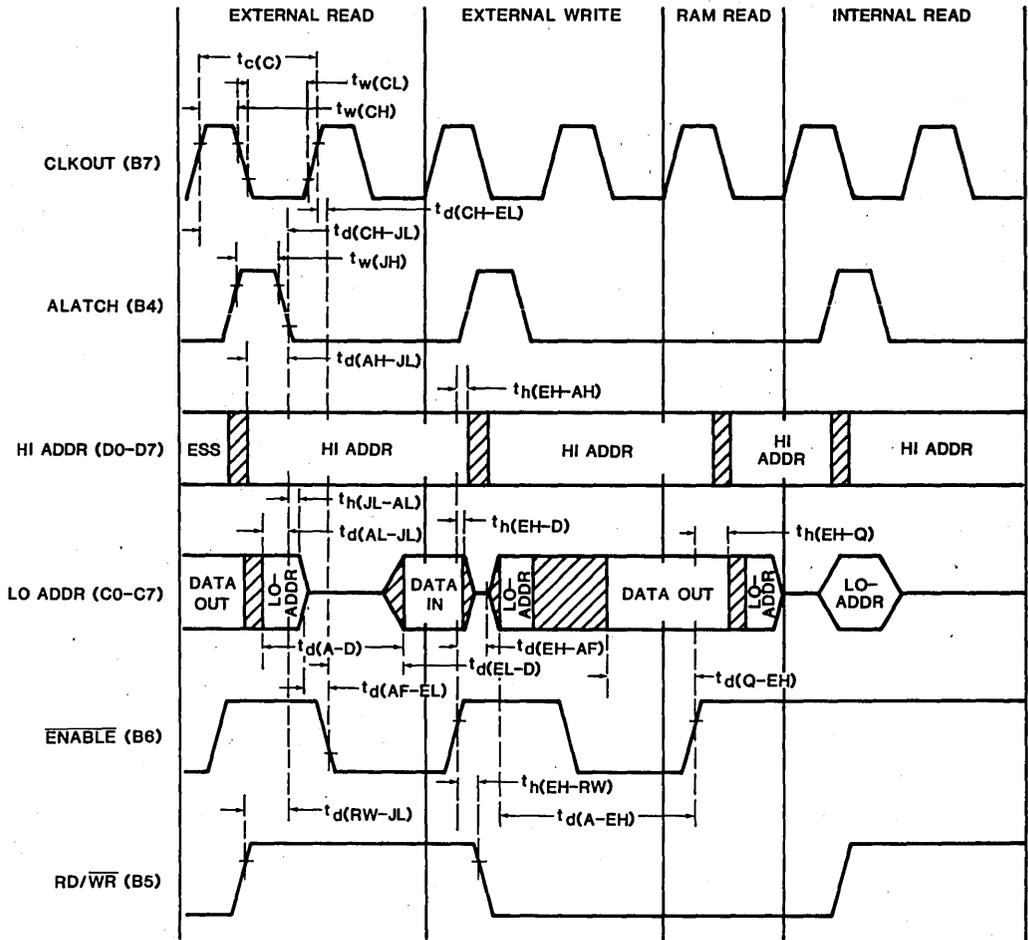
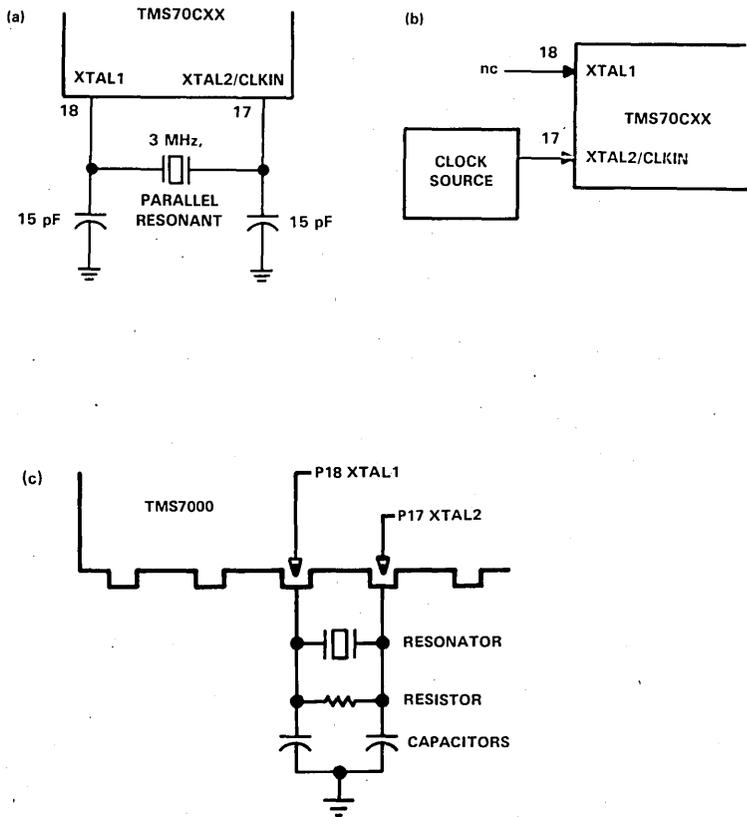


FIGURE 4-12 — READ AND WRITE CYCLE TIMING



NOTE: The TMS70CXX family currently uses only the divide-by-two option as the input clock options. Sources of ceramic resonators are given in Section 4.1.8.

FIGURE 4-13 — RECOMMENDED CLOCK CONNECTIONS

4.2.9 Pin Description Of The TMS70C00/TMS70C20/TMS70C40

Figure 4-14 defines the pin assignments and describes the function of each pin for the Single-Chip (SC), Peripheral Expansion (PE), Full Expansion (FE), and Microprocessor modes for the TMS70CX0 family (TMS70C00, TMS70C20, TMS70C40).

SIGNATURE	PIN	I/O	DESCRIPTION
A0 (LSB)	6	IN	I/O Port A: Input lines
A1	7	IN	(Specific I/O configuration for;
A2	8	IN	Single Chip Mode — see Section 2.3.1,
A3	9	IN	Peripheral Expansion Mode — see
A4	10	IN	Section 2.3.2, Full Expansion
A5	16	IN	Mode — see Section 2.3.3, Micro-
A6	15	IN	processor Mode — see Section 2.3.4)
A7 (MSB)	11	IN	
B0 (LSB)	3	OUT	I/O Port B: Output lines
B1	4	OUT	(Specific I/O configuration for;
B2	5	OUT	Single Chip Mode — see Section 2.3.1,
B3	37	OUT	Peripheral Expansion Mode — see
B4/ALATCH	38	OUT	Section 2.3.2, Full Expansion
B5/R/W	1	OUT	Mode — see Section 2.3.3,
B6/ENABLE	39	OUT	Microprocessor Mode — see Section
B7/CLOCKOUT	2	OUT	2.3.4)
C0 (LSB)	28	I/O	I/O Port C: General purpose bidirectional
C1	29	I/O	lines (Specific I/O configuration for; Single
C2	30	I/O	Chip Mode — see Section 2.3.1, Peripheral
C3	31	I/O	Expansion Mode — see Section 2.3.2, Full
C4	32	I/O	Expansion Mode — see Section 2.3.3,
C5	33	I/O	
C6	34	I/O	Microprocessor Mode — see Section 2.3.4)
C7 (MSB)	35	I/O	
D0 (LSB)	27	I/O	I/O Port D: General purpose
D1	26	I/O	bidirectional lines (Specific
D2	24	I/O	I/O Configurations for; Single
D3	23	I/O	Chip Mode — see Section 2.3.1,
D4	22	I/O	Peripheral Expansion Mode — see
D5	21	I/O	Section 2.3.2, Full Expansion Mode —
D6	20	I/O	see Section 2.3.3, Microprocessor
D7 (MSB)	19	I/O	Mode — see Section 2.3.4)
INT1	13	IN	Maskable Interrupt
INT3	12	IN	Maskable Interrupt
RESET	14	IN	RESET
MC	36	IN	Mode Control
XTAL2/CLKIN	17	IN	Crystal input for control of internal OSC.; input pin for external OSC. or LRC networks
XTAL1	18	IN	Crystal input for control of internal OSC.; leave open for external OSC.
VCC	25	IN	Supply voltage (+5V)
VSS	40	IN	Ground reference

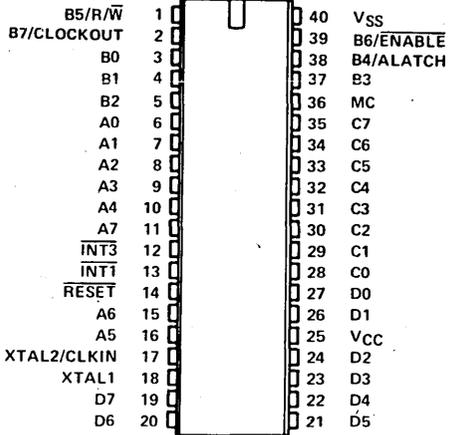


FIGURE 4-14 — SC, FE, PE, AND MICROPROCESSOR MODE PIN ASSIGNMENTS

4.3 SE70P161

4.3.1 Description Of The SE70P161 Prototyping Component

The SE70P161 prototyping component is another member of the TMS7000 family of single-chip 8-bit microcomputers. The SE70P161 is pin compatible with the TMS7020, TMS7040, TMS70120, TMS7041, and has the same instruction set described in Section 3 of this data manual.

The SE70P161 can also be used to emulate CMOS members of the TMS7000 family, with the following limitations. Because the SE70P161 is an NMOS device, its logic levels are not CMOS compatible. Also, this device does not support the low-power modes of the CMOS devices such as HALT or wake-up. Finally, INT1 on the SE70P161 is both latched and level triggered as in the NMOS devices, not just latched, as in the CMOS devices. Further details of these differences are provided in the sections which discuss the function.

The SE70P161 serves as a prototyping component for the TMS7000 devices and provides the ability to verify in real-time software written for all TMS7000 family members mentioned in the preceding paragraphs. This device uses standard TMS2764 or TMS27128 EPROMs. The EPROMs are located in a socket on top of a 40-pin dual-in-line package.

The SE70P161 is packaged so that an EPROM device can be plugged into the top of the package (piggy back). This two chip unit acts as an emulator of the TMS7020 (2K bytes of internal ROM space), the TMS7040/7041 (4K bytes of internal ROM space) and the TMS70120 (12K bytes of internal ROM space). The SE70P161 can also be used as an emulator of any future members derived from the TMS7040/7041 with up to 16K bytes of internal ROM space.

4.3.2 Prototyping

NOTE

System emulators and development tools are only to be used in a prototype environment. Texas Instruments does not warrant their use in customer's applications.

4.3.2.1 TMS7041 Prototyping

The SE70P161 uses either 2764 or 27128 EPROMs with 250 nanoseconds access time or better. The SE70P161 is identical to the TMS7041 except the supply current is a maximum of 150 mA higher because of the EPROM.

4.3.2.2 TMS7020/7040/70120 Prototyping

The SE70P161 system emulator can be used as a TMS7020/TMS7040/TMS70120 prototype. In this mode, internal peripheral port 16 must be cleared by adding MOV_P% >00, P16 to the initialization routine.

In any expansion mode, peripheral ports 13 through 23 are used internally and are not accessible to external peripherals in this memory space. In addition, in the full expansion mode, memory locations C000 through FFFF are reserved for an EPROM and are not externally available.

4.3.3 Programming And Installing EPROMS

All EPROM access times are not more than 250 nanoseconds. Pin 1 is identified by a nearby L-shaped gold trace; socket 1 for the EPROM is located in the same corner. Table 4-3 shows the use of the EPROMS.

TABLE 4-3 — EPROM USE

EPROM TYPE	70XX ROM SPACE	70XX + START ADDRESS	27XX START ADDRESS
27128	16K Bytes	>C006	>0006
2764	8K Bytes	>E006	>0006
2764	4K Bytes	>F006	>1006
2764	2K Bytes	>F806	>1806

†NOTE: Texas Instruments reserves the first 6 bytes of ROM. Addresses in this range may not be defined by the user program.

The SE70P161 is fabricated in two versions. Both versions have fixed internal ROM space of 16K bytes (C000-FFFF), one with a divide-by-two clock generator and the other with a divide-by-four. Note that on the SE70P161, none of the 16K EPROM address space can be mapped as external addresses except in microprocessor mode.

4.3.4 Absolute Maximum Ratings Over Operating Free-Air Temperature Range (Unless Otherwise Noted)†

Supply voltage, VCC (See Note 1)	—0.3 V to 7 V
All input voltage	—0.3 V to 20 V
All output voltages	— 0.3 V to 7 V
Continuous power dissipation	1 W
Operating free-air temperature range	0°C to 55°C
Storage temperature range	0°C to 100°C

† Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: Unless otherwise noted, all voltages are with respect to V_{SS}.

4.3.5 Recommended Operating Conditions

PARAMETER		MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}		4.5	5	5.5	V
High-level input voltage, V_{IH}	CLOCKIN	2.6		$V_{CC}+0.5$	V
	All others	2		$V_{CC}+0.5$	
Low-level input voltage, V_{IL}	CLOCKIN			0.6	V
	All others			0.8	
High-level output current, I_{OH}				-400	μ A
Low-level output current, I_{OL}				10	mA
Operating free-air temperature, T_A		0		55	$^{\circ}$ C

4.3.6 Electrical Characteristics Over Full Range Of Recommended Operating Conditions

PARAMETER		TEST CONDITIONS		MIN	TYP [†]	MAX	UNIT
V_{OH}	High-level output voltage	$I_{OH} = -0.4$ mA		2.4			V
V_{OL}	Low-level output voltage	$I_{OL} = 2$ mA				0.4	V
I_I	Input current	$V_I = V_{SS}$ to V_{CC}			10		μ A
I_{CC}	Average supply current [‡]	All outputs open			80	150	mA

[†] All typical values are at $V_{CC} = 5$ V, $T_A = 25^{\circ}$ C.

[‡] Average supply current without piggyback EPROM device installed.

4.3.7 Recommended CRYSTAL/CLOCKIN Operating Conditions Over Full Operating Range

PARAMETER		MIN	TYP	MAX	UNIT
f_{osc}	CRYSTAL/CLOCKIN frequency (divide-by-4 option)	2.0		10.1	MHz
f_{osc}	CRYSTAL frequency (divide-by-2 option) (see Note 1)	1.0		5.05	MHz
$t_{c(P)}$	CRYSTAL/CLOCKIN cycle time (divide-by-4 option)	99		500	ns
$t_{c(P)}$	CRYSTAL cycle time (divide-by-2 option)	198		1000	ns
$t_{c(S)}$	Internal state cycle time	396		2000	ns
$t_w(PH)$	CLOCKIN pulse width high	45			ns
$t_w(PL)$	CLOCKIN pulse width low	45			ns
t_r	CLOCKIN rise time [‡]			30	ns
t_f	CLOCKIN fall time [‡]			30	ns
$t_d(PH-CL)$	CLOCKIN rise to CLOCKOUT rise delay		125	200	ns

[‡] Rise and fall times are measured between the maximum low level and the minimum high level using the 10% and 90% points (see Figure 4-3). Outputs have 100-pF loads to V_{SS} .

NOTE 1: Divide-by-4 option recommended with external clock drive.

4.3.8 Memory Interface Timing At 10 MHz Over Full Operating Free-Air Temperature Range

PARAMETER		MIN	NOM	MAX	UNIT
$t_{c(C)}$	CLOCKOUT cycle time (see Note)	400		2000	ns
$t_w(CH)$	CLOCKOUT high pulse width	130	170	200	ns
$t_w(CL)$	CLOCKOUT low pulse width	150	190	240	ns
$t_d(CH-JL)$	CLOCKOUT rising to ALATCH falling edge	260	300	340	ns
$t_d(CH-EL)$	CLOCKOUT rising to ENABLE falling	-10	15	50	ns
$t_w(JH)$	ALATCH high pulse width	150	190	230	ns
$t_d(AH-JL)$	High address valid before ALATCH fall	50	170	220	ns
$t_d(AL-JL)$	Low address valid before ALATCH fall	50	150	220	ns
$t_h(JL-AL)$	Low address hold after ALATCH fall	30	45	80	ns
$t_d(RW-JL)$	RD/ \overline{WR} valid before ALATCH fall	50	140	200	ns
$t_h(EH-RW)$	RD/ \overline{WR} hold after \overline{ENABLE} rise	40	100		ns
$t_h(EH-AH)$	High address hold after \overline{ENABLE} rise	30	40		ns
$t_h(EH-Q)$	Data out hold after \overline{ENABLE} rise	65	80		ns
$t_d(Q-EH)$	Data out valid before \overline{ENABLE} rise	230	290		ns
$t_d(AF-EL)$	\overline{ENABLE} fall after low address HI-Z	0	30	120	ns
$t_d(EH-AF)$	\overline{ENABLE} rising to next address drive	60	85		ns
$t_d(EL-D)$	Data in after \overline{ENABLE} falling	155	190		ns
$t_h(EH-D)$	Data in hold after \overline{ENABLE} rise	0			ns
$t_d(A-D)$	Access time, data in from valid address	400	470		ns
$t_d(A-EH)$	\overline{ENA} high after address valid	580		730	ns

NOTE: $t_{c(C)}$ is defined to be $4/f_{osc}$ (or $2/f_{osc}$ if the divide-by-2 option is selected) and may be referred to as a machine state or simply a state.

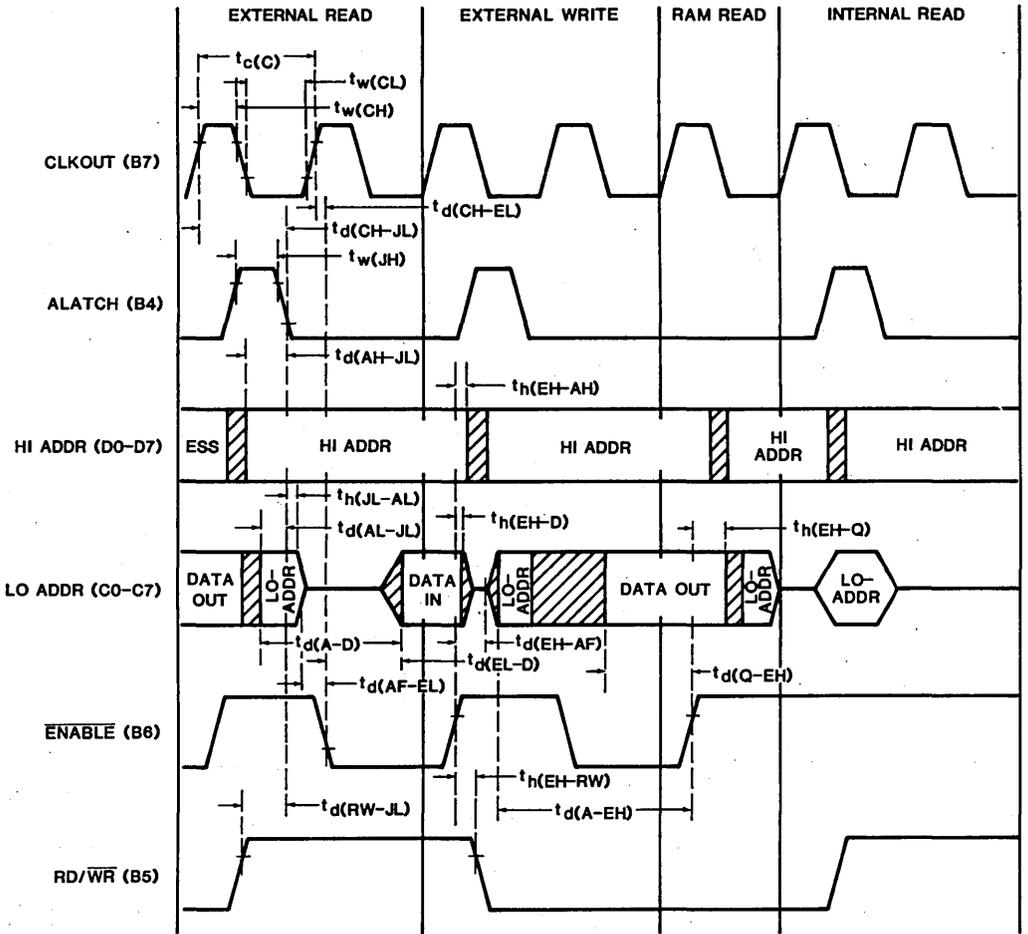
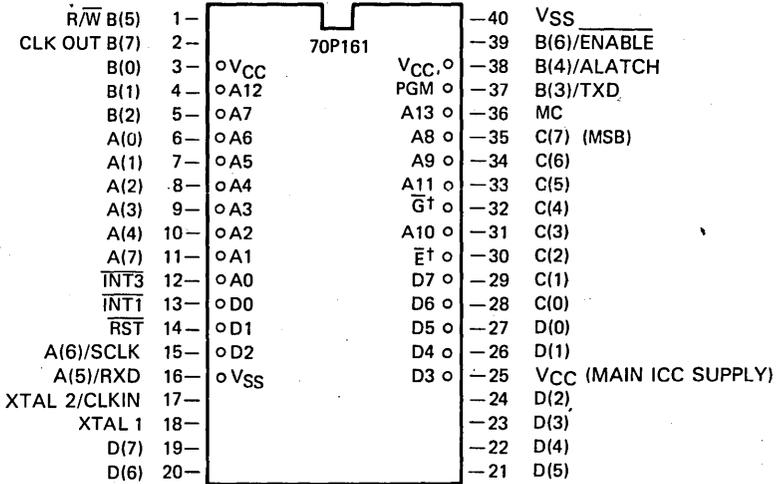


FIGURE 4-15 — READ AND WRITE CYCLE TIMING

4.3.9 Pin Description Of The SE70P161



† PIN LOW, EPROM ALWAYS ENABLED

5. MICROPROGRAMMING

5.1 TMS7000 CUSTOM MICROCODING DESCRIPTION

Standard members of the TMS7000 family implement a general purpose instruction set intended to address the needs of most potential users. A general purpose instruction set, however, does not directly address the requirements of any specific application. Microcoding is a technique which can be used to tailor the instruction set to more efficiently satisfy the particular application needs. Basic performance attributes of the TMS7000, such as speed and program size, may be greatly improved by microcoding.

Microcoding involves modifying the CPU control logic. This logic implements the instruction set of the CPU and, when modified, includes the user functions as a new assembly language instruction. The control information (called microinstructions) is contained in a ROM called the Control ROM, or CROM (see Figure 5-1 TMS7000 CPU Internal Block Diagram). This microprogram is similar to an assembly language program contained in memory. The control logic may be modified to implement a new user function by using similar methods as the masked program ROM. Modifying this microcoded control information allows a relatively inexpensive way to implement a more efficient user routine. Normally, this routine would be written in assembly language code which uses more time and ROM. In contrast, altering the instruction set of a CPU which is not microprogrammed is expensive and usually impractical due to the complexity of modifying the random logic used to implement its control section.

With custom microcoding, the new function is normally initiated by executing a single, newly defined assembly language instruction which generates a unique opcode that causes the function to execute. Microcoding can produce a 40% or greater improvement in performance depending on the function implemented.

5.1.1 Typical Applications

In a wide variety of applications, microcoding efficiently bridges the performance and cost gap between general purpose microprocessors/microcomputers and expensive high performance dedicated controllers. Applications for microcoding are from areas where extended performance and control at the bit level are required in a dedicated microprocessor/microcomputer based system. These requirements can include speed and program size improvements. Texas Instruments' microcoding capability and support for the TMS7000 makes this microcomputer family the ideal choice for these types of applications.

Some typical applications for TMS7000 custom microcoding are listed below:

- AUTOMOTIVE
 - DASHBOARD CONTROL
 - DASHBOARD DISPLAY
 - RADIOS
 - CAR COMPUTER
- TELECOMMUNICATIONS
 - MEMORY PHONES
 - AUTOMATIC DIALERS
 - PHONE LINKED COMPUTER TERMINAL
- COMPUTER PERIPHERALS
 - PRINTERS
 - DISK CONTROLLERS
 - KEYBOARDS
 - ALPHA-GRAPHIC TERMINALS
 - TAPE CONTROLLERS
 - SMART MODEMS
 - HANDHELD COMPUTERS
 - PLOTTERS
- INDUSTRIAL
 - MACHINE CONTROL
 - SPEED CONTROL
 - POSITION CONTROL
 - TEMPERATURE CONTROL
 - HIGH-LEVEL LANGUAGE
 - COMPUTER CONTROL
 - TIMER-CONTROLLER-CLOCKS
- RETAIL
 - POINT OF SALE TERMINALS
 - SCALES
 - BAR CODE READERS
 - VENDING MACHINES
 - DATA ENCRYPTION
 - REMOTE BANK TELLERS
 - METERING
- CONSUMER
 - HOME COMPUTERS
 - GAMES
 - EDUCATIONAL PRODUCTS
 - SPEECH PRODUCTS
 - SECURITY
 - SMART APPLIANCES
 - STEREO EQUIPMENT

5.1.2 Key Features

There are several advantages to using microcode to implement a given function instead of coding that function in assembly language. Among these are:

- IMPROVED EXECUTION SPEED
- REDUCED PROGRAM SIZE REQUIREMENTS AT THE ASSEMBLY LANGUAGE LEVEL
- ALGORITHM SECURITY
- LESS EXTERNAL LOGIC REQUIRED

One of the most important advantages is the improvement in execution speed of microcoding over assembly language. This improvement in execution speed results because microcode is more specific than assembly language and therefore performs less redundant operations in its execution. This results in more compact and efficient code which executes in fewer CPU cycles, thereby executing faster.

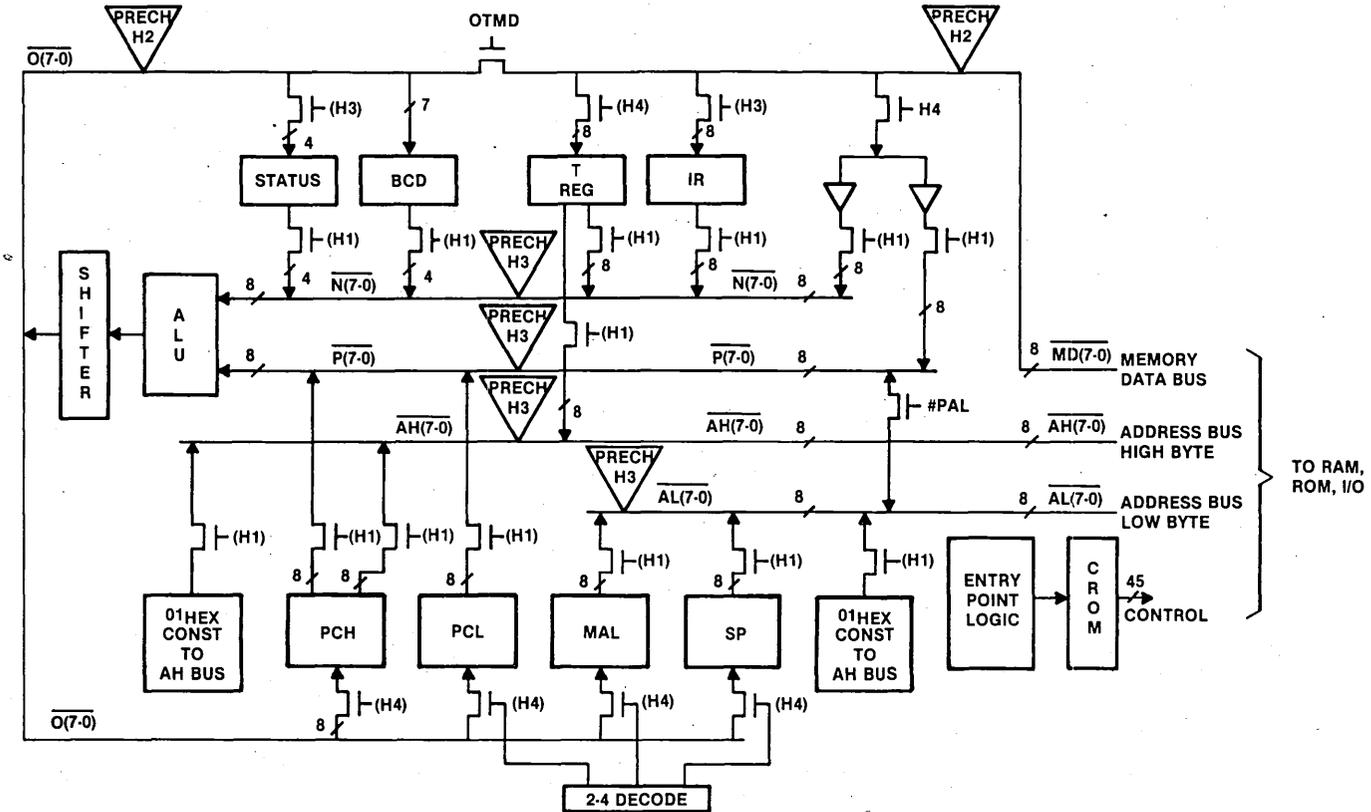
As an example of this redundancy, consider a sequence of assembly language code that operates on a byte of data. Any operation on that byte will typically involve bringing the byte into the CPU, operating on it and possibly another operand and then storing the result of this operation in memory. Subsequent operations on this result require that the data be again brought into the CPU before it can be used. In a microcoded version of the same sequence, a

significant amount of execution time can be saved since the data which is required at a later stage of processing can remain in the CPU. This eliminates the overhead of multiple memory accesses which store and then retrieve the same information.

Another advantage of microcode is that since the CPU operates more efficiently, less assembly language code is required to perform a given function. This results in reduced program size and ROM memory requirements, especially if this code is used repetitively. Specifically, a function which may take many instructions to implement in assembly language may be executed in microcode with only one assembly language instruction. Clearly, this can result in significant savings in memory requirements.

Algorithm security is another positive aspect of microcoding. Once microcoded functions are implemented on the TMS7000 chip, the algorithm used in their implementation is significantly more difficult to access than if it were an assembly language program contained in ROM. Thus, if a particular method of implementing a function is proprietary for any reason, microcoding the function will increase the security of this information. This is important, for example, in applications which implement data encryption type functions and in highly competitive markets such as toys and games.

Microcoding can also reduce the amount of additional logic circuitry required in a system. This reduction in additional circuitry in a system results because many of the functions to be performed in the external logic can be accomplished by the microcoded sequences. These functions include such operations as bit shifting, latching, counting, timing synchronization and many other functions which can be easily accomplished in microcode. Implementing these functions in microcode results in a lower system chip count which results in lower system cost and improved reliability.



TO RAM,
ROM, I/O

NOTE: Transfer gate controls in parentheses indicate only the clock phase on which the control occurs. Controls are not activated each cycle of the clock, only when the control signal is asserted.

FIGURE 5-1 — TMS7000 CPU INTERNAL BLOCK DIAGRAM

5.1.3 Microcoding Example

To illustrate the contrast between functions coded in assembly language vs. functions implemented in microcode, it is interesting to consider two sequences which perform the same operation. Figure 5-2 shows an assembly language code sequence which implements the same multiply algorithm that the TMS7000 MPY instruction uses.

	CLR	A	RESULT INITIALLY ZERO
	CLR	R3	CLEAR LOOP COUNT
LOOP1	CLRC		CLEARs STC BIT
LOOP2	RRC	A	SHIFT RESULT RIGHT 1 BIT
	RRC	B	CHECK MULTIPLIER BIT
	JNC	@LABEL	ADD OR NO ADD?
	INC	R3	INCREMENT LOOP COUNT
	ADD	R2,A	PERFORM ADDITION
	JMP	@LOOP2	PROCESS NEXT BIT
LABEL	INC	R3	INCREMENT LOOP COUNT
	CMP	%9,R3	NINE LOOPS COMPLETE?
	JNE	@LOOP1	NO, PROCESS NEXT BIT
	TSTA		IF YES, SET STATUS, EXIT

FIGURE 5-2 — ASSEMBLY LANGUAGE MULTIPLY SEQUENCE

The assembly language sequence performs an eight by eight bit multiplication. This sequence leaves the resultant 16-bit product in the A/B register pair and sets the Status Register bits according to the contents of the A Register (just as the MPY instruction does). The sequence implements the multiply function as a subroutine and assumes that the two operands are located in R1 (the B Register) and R2. It should be noted that if a general addressing scheme had been implemented, additional code would have been required.

Although this sequence implements the same multiply algorithm that the MPY instruction uses, a minimum of 358 cycles (143.2 μ s with a 2.5 MHz internal clock rate) are required for its execution, whereas the microcoded MPY instruction executes in a maximum of 48 microinstruction cycles (19.2 μ s). The magnitude of this difference illustrates how much more efficient microcoded functions can be than functions coded in assembly language.

The significant savings of microcode over assembly language often makes microcoding indispensable in meeting a design's performance goals.

5.1.4 Considerations Of Microcoding

There are several tradeoffs to consider in determining whether microcoding is appropriate for a given application. These tradeoffs include:

- DESIGN CYCLE EXTENDED
- DEVICE TESTING REQUIREMENTS INCREASED
- AVAILABLE CROM SPACE RESTRICTED TO 46 WORDS (OUT OF 160)
- ONE OR MORE ASSEMBLY LANGUAGE INSTRUCTIONS MUST BE SACRIFICED

Each of these tradeoffs require consideration in the microcoding process. A potentially longer and more complex design cycle needs to be taken into account during the early planning stages of a microcoding task. This is also true of additional testing requirements dictated by a custom microcoded CPU. Both considerations should be anticipated and provided for. Also, since the standard instruction set microcode occupies the full 160 words of CROM, some of the standard instructions must be removed to make room for custom microcode. The standard instructions to be removed should be considered carefully to avoid limiting assembly language programming.

The standard assembly language instruction set has been divided into two groups of instructions designated core and non-core. Core instructions, considered to be essential in maintaining architectural integrity, are provided with all TMS7000's and may not be removed for microcoding purposes. Non-core instructions may be removed from the standard instruction set to allow room for microcoding. Of the 160 words in the CROM, 46 are non-core. The non-core assembly language instructions are listed in Figure 5-3.

MPY	Multiply
DAC	Decimal Add with Carry
DSB	Decimal Subtract with Borrow
DECD	Decrement Double
MOVD	Move Double
SWAP	Swap
CMPA	Compare A
XCHB	Exchange B
TRAPn	Traps 8-23
Peripheral File instructions	

FIGURE 5-3 — NON-CORE ASSEMBLY LANGUAGE INSTRUCTIONS

5.1.5 Microcode Development Cycle

The microcode development support package makes development of microcode for the TMS7000 straightforward and efficient. The microcode development cycle comprises many steps. These steps are briefly summarized below:

- GENERATE SPECIFICATION FOR MICROCODE
- GENERATE AND VERIFY MICROCODE
- GENERATE AND VERIFY TEST PATTERNS
- PRODUCE AND TEST PROTOTYPE DEVICES

The first step in the microcode development cycle is to determine that microcode is appropriate for the application and to identify which functions are to be microcoded. Once this is accomplished, a specification for the microcode is generated and writing of the microcode can begin. Also at this time, the assembly language code to be contained in the TMS7000's program ROM should be generated.

The flowchart in Figure 5-4 shows the microcode development cycle in detail. Contact the TI Factory for details of project timing. Microcode can be generated by four different sources: (1) customer (2) TI's Regional Technology Center - RTC (3) 3rd party (4) TI factory. It should be noted that this flowchart depicts the flow for microcode developed by a customer; however,

Texas Instruments will generate custom microcode if required. The development cycle in this case is similar to the one shown except that the flow includes validation of the code by the customer to ensure that the desired function is implemented.

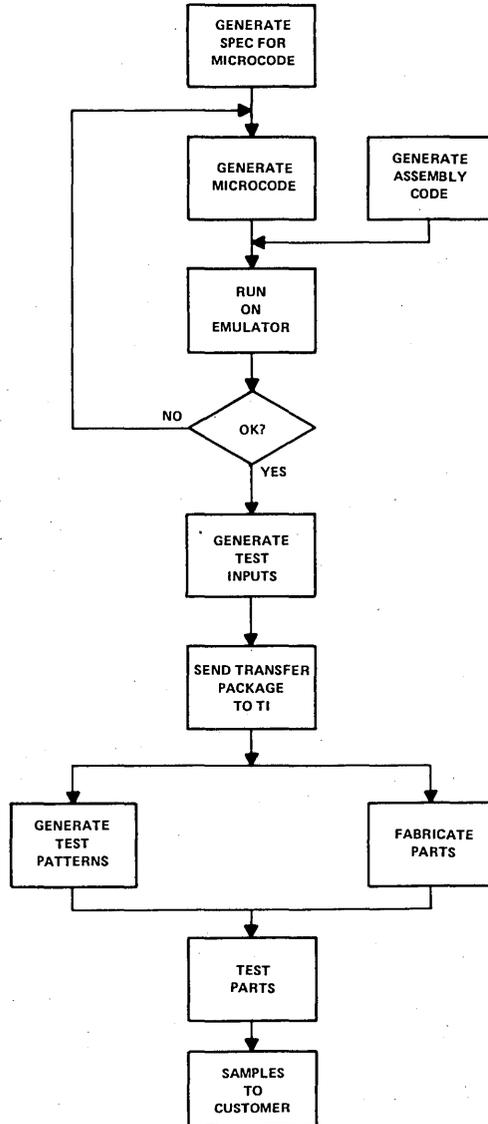


FIGURE 5-4 — MICROCODE DEVELOPMENT FLOWCHART

5.1.6 Available Support

Support for microcoding is provided through a comprehensive package of software, hardware and documentation. This support includes:

- TMS7000 MICROASSEMBLER SOFTWARE PACKAGE
- TMS7000 AMPL EMULATOR SYSTEM
- TMS7000 MICROCODE DOCUMENTATION PACKAGE, CONSISTING OF:
 - TMS7000 MICROCODE DEVELOPMENT GUIDE (MP #458)
 - TMS7000 MICROASSEMBLER USER'S GUIDE (MP #457)
 - TMS7000 MICROARCHITECTURE USER'S GUIDE (MP #061)
 - TMS7000 MICROPROGRAMMER'S REFERENCE CARD (MP #459)

5.1.6.1 *TMS7000 Microassembler Software Package*

This package is the software used for running the microassembler on the TI 990 hard disk computer. The TMS7000 microassembler (called MICASM) is the microcode assembler for the TMS7000 family which allows programmers to modify the standard TMS7000 family microcode and create a microcode object file.

5.1.6.2 *TMS7000 AMPL Emulator System*

The TMS7000 AMPL Emulator, which runs on TI 990 computers under AMPL, supports the TMS7000/TMS7020/TMS7040 devices (and their CMOS versions with the same limitations as the SE70P161, see Section 7.5) and the TMS70120. It allows in-circuit emulation of the microcoded device running at full speed or single stepping through the microcode.

5.1.6.3 *TMS7000 Microcode Documentation Package*

The Microcode Development Guide is a general microcode development aid which includes a tutorial on microcoding. This manual is normally the first document required and is helpful in determining whether microcoding is appropriate for a given application.

The TMS7000 Microassembler User's Guide describes the TMS7000 microassembler program which is used to generate microinstructions from statements containing mnemonics for microcode functions.

The Microarchitecture User's Guide contains all of the details of the internal operation of the TMS7000 that are necessary for microcoding.

The Microprogrammer's Reference Card contains a useful collection of reference information pertinent to microcoding the TMS7000.

5.2 MICROCODED BENCHMARKS

Benchmarks are a common method of comparing the performance of different computing elements executing the same function. A set of common microprocessor/microcomputer benchmarks has been microcoded to demonstrate the typical speed improvements possible through microcoding. These benchmarks and the speed improvements for each are listed in Table 5-1.

TABLE 5-1 — BENCHMARK 1-6 COMPARISON (2.5 MHz)

BENCHMARK	7000		7000	
	MICROCODED	RANK	STANDARD	RANK
BINARY ADDITION	4	1.0	6.4	.63
BCD ADDITION	5.6	1.0	10	.56
BLOCK MOVE	315	1.0	1780	.21
TABLE SEARCH	101	1.0	453	.22
BINARY TO BCD	100	1.0	295	.34
BIT I/O	10	<u>1.0</u>	20	<u>.50</u>
RELATIVE RANKING		1.0		.41

Note that the performance of the TMS7000 assembly language benchmark ranges anywhere from 21% to 63% of the performance of the microcoded version. These variations are due to which speed improvement techniques were applied and the extent to which they were able to be applied.

All benchmarks (except Benchmark 3) are responsible for fetching their own operands. Therefore the custom microcode must be entered directly from the instruction acquisition sequence of microcode. The TRAP B instructions all share the TGBO microstate which is entered directly from IAQ2, the last instruction acquisition microstate. All benchmarks will use the TGBO microstate for the first microinstruction of the benchmark. The instructions, TRAP 8 through TRAP 15, all enter the TGBO state. Non-core microinstructions were used as needed starting from the first non-core microinstructions listed in the TMS7000 standard instruction set source file.

Benchmark 3 uses shared microcode to fetch some of its operands. An unused instruction is used to enter the shared microcode. This unused instruction will execute the TESTO state after the addressing mode microcode. The TESTO microstatement is executed after the Long Addressing Function microcode fetches two of the necessary operands. The opcode used will be >89 which is currently unimplemented in the standard instruction set.

5.2.1 Benchmark Rules

The following list describes the rules used when microcoding the benchmarks:

- 1) All of the registers used in the assembly language code may be used. No other registers may be used.
- 2) The PCH, PCL, and SP registers must not be modified except where it is necessary to read operands. The program counters (PCH and PCL) may be stored on the stack to allow general use of these registers.

- 3) The microcode can assume where its operands are if operand placement is the same as in the assembly language benchmark.
- 4) The CPU's T, MAL, and IR registers are available for storage. The IR register, which uses the opcode as a basis for dispatches may be used because no function or group dispatches will be performed once the benchmark microcode is entered.

An individual description of each microcoded benchmark and what speed improvement techniques were applied follows.

5.2.2 Benchmark 1: 16 Bit Binary Addition

Two 16-bit unsigned binary integers in on-chip RAM (the register file) are added together; the result is stored back into on-chip RAM. One integer is contained in the A (MSB) and B registers, and the other operand is contained in registers R3 (MSB) and R4. The result is left in the A and B registers. The assembly language code to perform binary addition is:

```
ADD R4,B  Add LSBs together...set up carry for MSB addition
ADC R3,A  Add MSBs together...add in carry from LSB addition
```

This code occupies four bytes of memory and takes 6.4 microseconds to execute. The corresponding microcode implementation occupies one byte of memory and executes in 4 microseconds. Seven unique microstatements were required to perform binary addition. The following techniques were used to obtain the 38% speed improvement:

- 1) Elimination of instruction fetch and PC increment operations for the ADC instruction.
- 2) Benchmark 1 assumes the operands are located in registers A, B, R3 and R4. Constants are generated to address registers R3 and R4.

Only a 38% speed improvement was possible in this benchmark due to the simplicity of the function.

5.2.3 Benchmark 2: 16 Bit Binary Coded Decimal (BCD) Addition

Two unsigned 4 digit packed BCD integers in on-chip RAM are added together and the result stored back into on-chip RAM. One of the integers is contained in the A (MSB) and B registers and the other is located in registers R3 (MSB) and R4. The assembly language code is:

```
CLRC          Clear carry for LSB addition
DAC R4,B      Add with carry LSBs
DAC R3,A      Add MSBs together with carry from LSB addition
```

This code uses five bytes of memory and executes in 10 microseconds. This benchmark's microcode occupies one byte of code space and executes in 5.6 microseconds. Eleven unique microstatements were required to implement this function. This represents a speed improvement of 44%. The following techniques provided the 44% improvement:

- 1) Elimination of instruction fetch and PC increment operations.
- 2) Assumption of operand placement. Again, as in Benchmark 1, constants are generated to address registers R3 and R4.

Only a 44% speed improvement was possible due to the simplicity of the algorithm. Note that BCD arithmetic is slightly more complex than binary addition, and thus a 44% improvement was obtained, versus 38% for Benchmark 1. Also note that a CLRC instruction is required before the first addition since there is only one BCD addition instruction and it adds in the carry bit from the status registers.

5.2.4 Benchmark 3: Block Move

A block of 127 bytes in off-chip memory is moved to another location also in off-chip memory. The assembly language code to move blocks of data is:

	MOV	%127,B	Set up number of bytes to move
LOOP	LDA	@FROM-1(B)	Read a FROM block data byte
	STA	@TO-1(B)	Store byte to a TO block address
	DJNZ	B,LOOP	Decrement block move counter...jump if non-zero

This code uses 10 bytes of code, and when a block length of 127 bytes is specified, will execute in 1780 microseconds. Note that the table move is started from the end of the table. The microcode requires five operands: the number of bytes to move, the FROM addresses, and the TO addresses. The opcode and operands of Benchmark #3 will appear in memory in the following order:

LOCATION	X	BENCHMARK 3 OPCODE
	X + 1	TO MSB
	X + 2	TO LSB
	X + 3	FROM MSB
	X + 4	FROM LSB
	X + 5	NUMBER BYTES TO MOVE

Six bytes of program storage are needed and program execution will take 315 microseconds. Twenty-five unique microstatements are required. The CPU register usage is as follows:

PCH register	—	FROM MSB
PCL register	—	FROM LSB
T register	—	TO MSB
MAL register	—	TO LSB
IR register	—	Byte move counter

The microcoded block move allows a variable block move function. If the microcode is passed a block length of zero, 256 bytes will be moved. The microcode makes no check for being passed a block length of zero.

Two 16-bit addresses need to be referenced by this benchmark: the FROM block address and the TO block address. Because there are only two general purpose CPU registers available to address memory with (T and MAL), the program counter registers (PCH and PCL) are used to store one of the 16 bit block addresses. The program counter registers are saved on the stack. Therefore, two bytes of stack must be available for use by this benchmark's microcodes.

Benchmark 3 uses shared microcode to fetch the TO MSB and the TO LSB addresses. The Long Addressing function is used to fetch these two operands into the T and MAL registers. An unused opcode, >89, is available in the Long Addressing function group to be used. The opcode >89 will direct execution to the TESTO microinstruction after the long addressing

mode fetched the TO addresses. The TESTO non-core microstatement is the first non-core microinstruction used by the benchmark microcode.

The following techniques were applied to yield the 82% speed improvement:

- 1) Elimination of instruction fetch and PC increment operations.
- 2) The LDA and STA instructions move the data byte to the A register for storage. The microcode leaves the data byte inside the CPU.
- 3) The loop downcounter decrement operation is performed at the end of the block move loop. However the downcounter equal to zero check is done during the first microinstruction of the block move loop. This allows the loop to execute in fewer microstates.
- 4) The program counters are incremented when they are retrieved from the stack. Two cycles are saved because the incrementing of the program counters is done the same cycle the program counter values are passed thru the ALU to the program counter registers.

5.2.5 Benchmark 4: Table Search

Benchmark 4 searches a table looking for a key character. The assembly language code appears like this:

	MOV	%KEY,A	Set up byte to look for
	MOV	%40,B	Set up table length
LOOP	CMPA	@TABLE-1(B)	Does the table byte match key character
	JEQ	FND	If bytes match, jump
	DJNZ	B,LOOP	Decrement table length counter; jump if non zero
NFND		Key Not Found
FND		Key Found

This code occupies 11 bytes and executes in 326 microseconds. The timings are all done for the KEY not found condition. The microcode version requires four operands and occupies five bytes of code space. The microcoded instruction and its operands will appear in memory as follows:

LOCATION	X	BENCHMARK 4 OPCODE
	X+1	TABLE LENGTH
	X+2	KEY VALUE TO SEARCH FOR
	X+3	TABLE ADDRESS MSB
	X+4	TABLE ADDRESS LSB

When a table length of 40 is passed to the microcode, execution will take place in 101.6 microseconds. Again, the microcode implementation allows variable table lengths and KEY values. Twenty-one unique microstatements were required to implement the table search algorithm. The CPU register usage is as follows:

T register	—	Table address MSB
MAL register	—	Table address LSB
IR register	—	Key value to search for

The A register holds the table length original value. This value is stored for later subtraction with the current B offset value to yield the correct table offset value when the KEY is found. The table length downcounter is contained in the B register.

The assembly language version searches the table from back to front. The microcode version searches the table from front to back. If the microcode searched the table from back to front as the assembly code does, the table address would have to be reread every iteration of the loop because of the current table offsets addition into the table base address. The CMPA instruction does the table offset addition into the base address in the assembly language version. Reading the table from front to back allows the table address, contained in T and MAL, to be continually incremented to point to successive table locations. The B register, which contains the table offset downcounter, is read, decremented, written back to the B register, and then checked for having reached zero.

The original table length is stored in the A register. Once the KEY is found, the B register is subtracted from the original table length (in A) to yield the correct table offset.

To obtain the 69% speed improvement, the following techniques were used:

- 1) Elimination of instruction fetch and PC increment operations.
- 2) Elimination of unnecessary reads/writes. The table addresses and the KEY character are all stored internal to the CPU.
- 3) The table value and the KEY value are compared during the same cycle the B register is read for the decrement operation.
- 4) The decision of whether the KEY is found or not is made in the same microcycle that the B register is decremented and written.
- 5) The table address is incremented the same cycle as a microjump is done on the table empty condition.

If the KEY was not found, the B register contents are >FF. This limits the table length to 254 bytes, or >FE. Because the microcode cannot determine the address of the code to execute when the KEY is found, the PC will be incremented by two to point past the KEY not found return. This requires that a two byte jump be placed at the NFND label to jump over the KEY found code.

5.2.6 Benchmark 5: Binary To BCD Conversion

This benchmark performs binary to BCD conversion. A 16 bit binary number, contained in registers R2 (MSB) and R3, is converted to a Binary Coded Decimal value to be left in registers A (MSB) and B. The conversion is done by looping 16 times, rotating A and B through the status carry and adding the BCD number to itself (which sets up the carry for the rotations). The assembly language code is:

	CLR	A	Clear MSB result
	CLR	B	Clear LSB result
	MOV	%16,R4	Set up loop count
LOOP	RLC	B	The BCD result in A and B
	RLC	A	Is generated through the carry bit
	DAC	R3,R3	Decimal add binary MSB to itself to set up carry
	DAC	R2,R2	Decimal add binary LSB to itself to set up carry
	DJNZ	R4,LOOP	Decrement loop count and jump if non zero

This code occupies 16 bytes, and takes 295 microseconds to execute. The microcode version occupies only one byte of code space and takes 100 microseconds to execute. A speed improvement of 66% is obtained. Generally speaking, iterative functions will yield a greater speed improvement when microcoding than non-iterative functions.

The T register is used to hold the "A" register value (binary MSB). The T register is written to the A register at the end of the conversion. The binary LSB is stored in the B register as is done by the assembly code.

The loop downcounter is stored in the IR register. To decrement a value by one, it will be placed on the ALU P bus with a zero gated onto the N bus. The ALU will then perform a subtraction (PSUBN) operation with a zero carry in to decrement the operand. Because the IR only connects to the N bus, an extra cycle is required each loop iteration transferring the IR thru the ALU to the MD bus where the next microinstruction will gate the value onto the P bus to perform the decrement operation.

The 66% speed improvement was obtained through the use of the following techniques:

- 1) Elimination of instruction fetch and PC increment operations.
- 2) Assumption of operand placement. The binary number is assumed to be in R2 and R3 and the result is assumed to be placed in A and B.
- 3) Elimination of unnecessary reads/writes. The binary MSB, which the assembly code manipulates to/from the A register, is left internal to the CPU in the T register.
- 4) The constant value "3" (to address R3) and the constant "16" are generated simultaneously.
- 5) The binary MSB and LSB (assembly uses CLR A and CLR B) are cleared simultaneously.
- 6) The loop downcounter decrement, and equal to zero check, are done at the same time as the constant "2" is incremented to point to R3.

5.2.7 Benchmark 6: Bit I/O

Benchmark 6 tests the ability of microcomputers to perform bit I/O operations. An input only port (port A) and an output only port (port B) are used. If any one of three input bits are a "1" then an output bit will be set to a "1". Then, if another input bit is a "0" three output bits are toggled. All inputs may be on the same 8 bit port, but inputs and outputs must be on different ports. The assembly language code is:

	ANDP	%IMASK1,PA	AND functions checks for "1" bits on PA
	JZ	NEXT	If none are set, jump over bit set instruction
	ORP	%OMASK1,PB	Set the output bit to a "1"
NEXT	BTJOP	%IMASK2,PA,DONE	If any masked bits are a "0", jump over toggle function
	XORP	%OMASK2,PB	Toggle output bits values
	DONE	

This code occupies 15 bytes and takes 20 microseconds. The timings are calculated assuming neither of the two jumps were taken; the ORP and XORP instructions were executed. Benchmark 7 is passed the following parameters in the order they are obtained from the program counter pointers:

LOCATION	X	BENCHMARK 6 OPCODE
	X+1	IMASK1
	X+2	OMASK1
	X+3	IMASK2
	X+4	OMASK2

The microcode application of this benchmark uses five bytes of code space and executes in 10 microseconds which represents a 50% speed improvement.

Only the T register is used by this benchmark. The mask values are read into the T register. The peripheral port values are read onto the MD bus and operated on with the mask values to provide a port output value or to set up the status (for decision making).

The 50% speed improvement was gained by the application of the following techniques:

- 1) Elimination of instruction fetch and PC increment operations.
- 2) Rearrangement of algorithm functions. While the Port A decision is being made, the read of the output mask is started. Even if the output mask is not needed, the program counter still needs to be incremented to point to either the next operand or the next instruction.
- 3) Port A is an input only port but the ANDP instruction writes to this port. Writing to port A is not required by this benchmark and was removed in the microcode application of the benchmark.

5.3 MICROARCHITECTURE DESCRIPTION

This section contains a description of the internal architecture of the TMS7000. It describes primarily the operation of CPU; the memory and on-chip I/O circuitry may vary among the TMS7000 family members, and will be described in the documentation for those individual devices. This section is intended to present information regarding the internal architecture of the TMS7000 family necessary for microcoding these devices. A symbolic microinstruction assembler called MICASM is provided for assembling microcode instruction mnemonics. This assembler is described in the TMS7000 MICROASSEMBLER USER'S GUIDE (Part Number MP457).

5.3.1 TMS7000 Family Address Space

The TMS7000 family address space is divided into multiple 256-byte pages. Addresses >0000 to >007F are utilized as a 128-byte Register File or RF, and reference the on-chip RAM. On-chip ROM is located at the top of the address space, from addresses >F800 to >FFFF for the TMS7020, and >F000 to >FFFF for the TMS7040. The last 48 bytes of memory, addresses >FFD0 to >FFFF, are reserved for trap and interrupt vectors. The TMS7000 family address space is shown in Figure 5-5. Note that the TMS70120, not depicted in Figure 5-5, has 12K bytes of ROM.

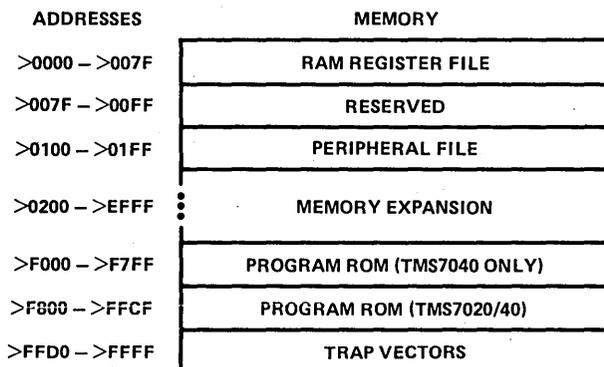


FIGURE 5-5 – TMS7000 FAMILY ADDRESS SPACE

The Peripheral File, or PF, is a special 256-byte page in the memory address space. Each location of the PF is a special control or data register. On-chip circuitry interprets PF Registers as I/O control, programmable timer, memory expansion, and other registers to control features of the chip. For example, the four I/O ports may be accessed as four registers in the PF. Accesses to the Peripheral File are recognized by the Peripheral/Memory Controller (PMC) external to the CPU. In general, all chip functions not implemented by the CPU will be implemented by the Peripheral/Memory Controller, and controlled via accesses to Peripheral File Registers.

The advantage of defining special pages for the Peripheral and Register files is that accesses to these areas may be made by specifying an offset of 8 bits, rather than a full 16-bit memory address. The Register File is located at memory addresses >0000 thru >007F and the Peripheral File is implemented in the second page of memory address space, from addresses >0100 to >01FF.

5.3.2 Basic TMS7000 Architecture

The major components of the TMS7000 architecture are the CPU, the Peripheral/Memory Controller, and the RAM and ROM. These components and their interconnections are shown in Figure 5-6.

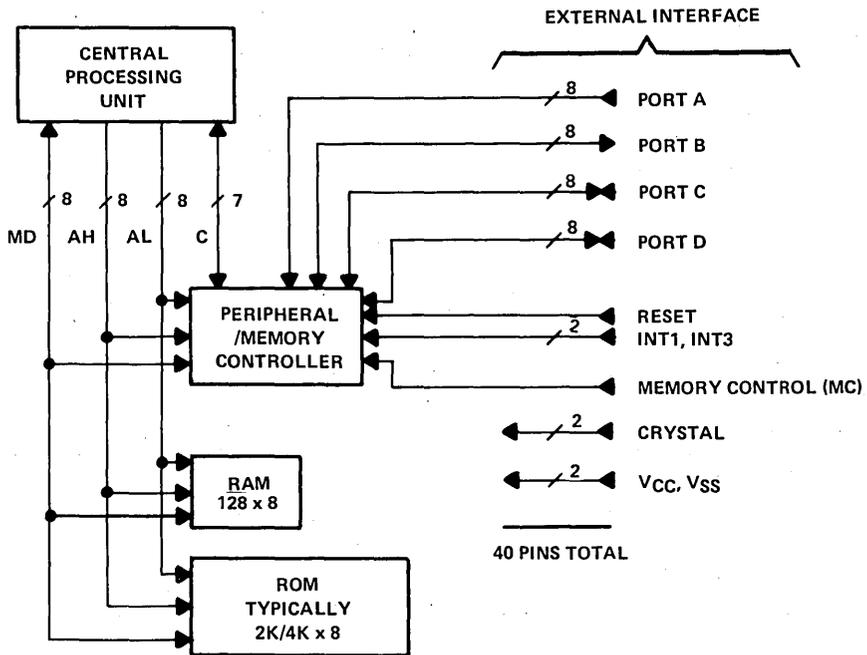


FIGURE 5-6 – TMS7000 OVERALL BLOCK DIAGRAM

The Central Processing Unit (CPU) contains the internal registers, which store the operands of an instruction, and the Arithmetic Logic Unit (ALU), which operates on the internal register values. A shifter is provided to rotate the output of the ALU before its results are either stored in an internal CPU register or written to a memory location. The CPU is described in further detail in paragraph 5.3.4.

The Peripheral/Memory Controller (PMC) is a collection of modules which interface the CPU with the I/O ports, memory, and the interrupt inputs. The CPU is connected to the PMC via the Address Low (AL), Address High (AH), Memory Data (MD), and Control (C) Buses. The MD Bus, AL Bus, and AH Bus are also connected to the on-chip RAM and ROM memories.

The Peripheral/Memory Controller (PMC) performs many functions. It interfaces the CPU to the outside world by providing control and data registers for I/O ports, interrupts, and internal timer controls. The interface control registers appear to the CPU as addresses in the Peripheral File. In the TMS7000, the PF is implemented in the second 256 byte page of memory, at addresses >0100 to >01FF. Input/output in the TMS7000 is accomplished by reading and writing bytes in the Peripheral File implemented by the PMC. In terms of the microarchitecture, the exact functions of the Peripheral File registers are family member dependent.

The Control (C) Bus connecting the PMC and the CPU carries control information required in the interface between these two subsections of the TMS7000. The C Bus is made up of seven signals, each of which is described briefly below.

- **#MEM (Memory):** set by the CPU during any memory access.
- **#MEMCNT (Memory Continue):** set by the CPU during the first cycle of two cycle memory accesses.
- **#WR (Write):** set to 1 by the CPU to indicate a memory write operation.
- **STINT (Status Interrupt Enable):** set by the CPU to allow the PMC to assert IACT.
- **IACT (Interrupt Active):** set by the PMC if a valid interrupt is active and STINT is a 1.
- **RST (Reset):** set to 1 by the PMC whenever the external RESET pin is a 0.
- **OTMD (O Bus to MD Bus Enable):** set by the PMC to enable the O Bus to drive the MD Bus.

Each of these signals is discussed in greater depth in later sections of this manual. Further details of interrupt control may be found in the TMS7000 8-Bit Microcomputer Data Manual (Part Number MP 008A).

5.3.3 Microinstruction Format

This section describes the format of the TMS7000 microinstructions, and details the internal timing of microinstruction execution.

The CROM is organized as a 64-bit wide, 160-word memory. The current microarchitecture of the TMS7000 uses 45 bits per microinstruction to control its operation. To allow for future expansion of this architecture, however, a total of 64 microinstruction bits are reserved in the architecture definition. Table 5-2 describes the format of the TMS7000 microinstruction word.

TABLE 5-2 – MICROINSTRUCTION WORD FORMAT

BITS	FIELD	FUNCTION
63-56	#JMPADDR(7-0)	BASE ADDRESS FOR NEXT INSTRUCTION
55-53	#JMPCNTL(2-0)	JUMP FUNCTION SELECTION
52	#O> PCH	GATES O BUS TO PCH REGISTER
51	#MD> T	GATES MD BUS TO T REGISTER
50	#-MD> IR	GATES MD BUS TO IR REGISTER
49-48	#LOWWRITE(1-0)	SELECTS ONE OF 3 O BUS DESTINATIONS
47	#-O> ST	GATES O BUS TO ST REGISTER
46	#MD> P	GATES MD BUS TO P BUS
45	#PCH> P	GATES PCH REGISTER TO P BUS
44	#PCL> P	GATES PCL REGISTER TO P BUS
43	#MD> N	GATES MD BUS TO N BUS
42	#T> N	GATES T REGISTER TO N BUS
41	#ST> N	GATES ST REGISTER TO N BUS
40	#BCD> N	GATES BCD CONSTANT TO N BUS
39	#IR> N	GATES IR REGISTER TO N BUS
38	#ONE> AL	GATES CONSTANT ONE TO AL BUS
37	#PAL	GATES P BUS TO AL BUS
36	#MAL> AL	GATES MAL REGISTER TO AL BUS
35	#SP> AL	GATES SP REGISTER TO AL BUS
34	#T> AH	GATES T REGISTER TO AH BUS
33	#PCH> AH	GATES PCH REGISTER TO AH BUS
32	#ONE> AH	GATES CONSTANT ONE TO AH BUS
31	#MEMCNT	FIRST ONE OF TWO CYCLE MEM. ACCESS
30	#MEM	INDICATES A MEMORY ACCESS
29	#WR	INDICATES A MEMORY WRITE
28	#-LST	UPDATES STATUS REGISTER BITS
27-24	#SHIFCNTL(3-0)	SELECTS SHIFT/ALU CARRY FUNCTIONS
23-20	#ALUCNTL(3-0)	SELECTS ALU FUNCTION
19	#ABL	LOGICAL (VS. ARITHMETIC) ALU OP'S
18-0	Reserved	

NOTE: In multiple bit fields bit 0 is the LSB.

All 160 words of the CROM are required to implement the standard instruction set of the TMS7000. Because of this, adding other microcoded functions to the TMS7000 requires that some of the standard instructions be deleted to allow space for the new instructions.

The TMS7000 Standard Instruction Set has been divided into two instruction groups designated core and non-core instructions. Non-core instructions are those instructions which Texas Instruments will allow to be removed in order to implement other microcoded functions. Core instructions may not be removed and are provided with any TMS7000 whether further microcoding has been implemented or not. Core and Non-core instructions are described in the TMS7000 Microcode Development Guide, (Part Number MP 458).

A symbolic microprogram assembler, MICASM, is available to aid microprogram generation. MICASM accepts mnemonic names for bit fields in a microinstruction word, and builds the appropriate bit patterns. The names of each bit field in the TMS7000 microinstruction word are given in Figure 5-7. They are distinguished from other signal names by preceding them with a '#'.
#'

For single bit fields, if the MICASM statement contains the name of the bit, it is asserted in the assembled instruction. For high-true signals, the bit is set to 1; for low-true signals (such as #0>ST), the bit is set to 0. For multiple-bit fields, MICASM accepts any one of a set of possible names, where each name corresponds to a bit pattern for the multi-bit field. A sample of a MICASM statement is shown in Figure 5-7.

```

.ORG    ADD0          'ADD Dual Operand Function
        Z>AH,         'AH=0 for Page 0 access
        MAL>AL,      'AL = destination register #
        MD>P,        'Source operand to P bus
        T>N,         'Destination operand to N bus PADDN,ZCI,LST
                        PADDN,ZCI,LST
        MW,          'Add them, load status register
        JUNC(NEXT);  'Write the result to destination
                        'Jump to next microinstruction

```

FIGURE 5-7 – SAMPLE OF A MICASM STATEMENT

The .ORG line establishes the address of the microinstruction in the Control ROM. The remaining lines contain symbols which set bits in the current microinstruction word. The last line indicates the next microinstruction that is to be executed.

5.3.3.1 Microinstruction Cycle Timing

Each microinstruction cycle has four overlapping clock phases; H1, H2, H3, and H4. H1 and H3 are non-overlapping, and H2 and H4 are non-overlapping. Microinstruction cycles begin on the rising edge of H1. Two versions of clock generator circuitry are available for the TMS7000. The first version uses the external crystal frequency directly to generate H1-H4. The second version divides the crystal frequency by two before generating the internal clock phases. Figure 5-8 shows the timing relationships of the four internal clock phases H1-H4 and the signal from the crystal oscillator.

* NOTE: This waveform represents the crystal oscillator output divided by two if that version of the clock generator circuitry is used.

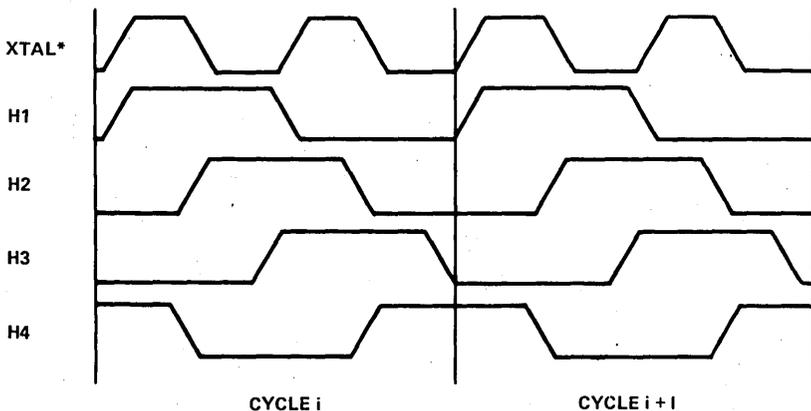


FIGURE 5-8 – MICROINSTRUCTION CYCLE PHASES

H1-H4, the four internal clock phases, are used as data transfer signals throughout the architecture. In particular, the current microinstruction is gated out of the Control ROM during H1. Microinstruction bits required during later phases (H2, H3, H4) are appropriately sampled by the hardware.

The internal implementation of the TMS7000 uses MOS dynamic ratioless logic which allows the chip to operate with lower power requirements than with other types of MOS logic. Signal lines considered to be valid during phase HX (e.g. H1) are precharged during the non-overlapping phase of HX (e.g. H3). For this reason, timing diagrams in this document will indicate signal values only during the phase in which they are valid, with a don't care indication during the phase in which they are precharged.

5.3.3.2 *Memory Cycle Timing*

Memory references to the on-chip Register File (RF) require one microinstruction cycle, and are called short memory cycles. All other references, i.e. to on-chip ROM, extended memory, or the Peripheral File, require two microinstruction cycles, and are called long memory cycles. Extended memory must be able to respond in this time period, since no wait states are provided in the TMS7000.

5.3.3.3 *Short Memory References*

The timing for a read or write to the on-chip Register File is shown in Figure 5-9.

ON-CHIP RAM MEMORY CYCLE TIMING

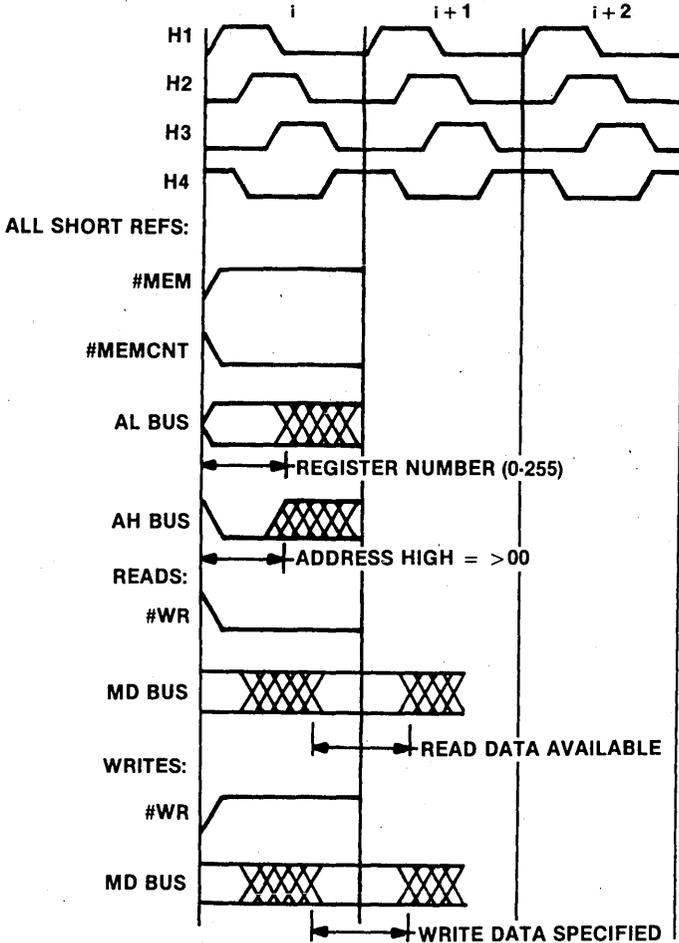


FIGURE 5-9 — ON-CHIP RAM MEMORY CYCLE TIMING

For a Register File read during cycle *i*, the microinstruction loaded at the initiation of cycle *i* asserts #MEM high and #MEMCNT low. #MEM is asserted at all times when a memory reference is active, and #MEMCNT is asserted high only during the first cycle of two-cycle (ie. long) memory cycles. #WR is set low for read operations and high for write operations. Microinstruction *i* also specifies the contents of the the address bus, placing a >00 on the AH (Address High) Bus and the register number on the AL (Address Low) Bus. During H2, the MD Bus is precharged and the RAM is accessed. For the duration of H4, the RAM output data on write operations and the RAM input data on read operations is on the MD Bus.

Because H4 of cycle *i* overlaps H1 of cycle *i*+1, the data read on cycle *i* may be loaded into registers T or IR at the end of cycle *i* or gated onto the P or N Buses at the beginning of cycle *i*+1. This characteristic of the MD Bus can be very useful in optimizing microcode performance.

Initial members of the TMS7000 family implement only 128 bytes of the 256-byte Register File; attempts to write to addresses in non-existent on-chip memory will be ignored. Attempts to read non-existent memory will produce >00.

5.3.3.4 Long Memory References

The timing for all long memory references is shown in Figure 5-10.

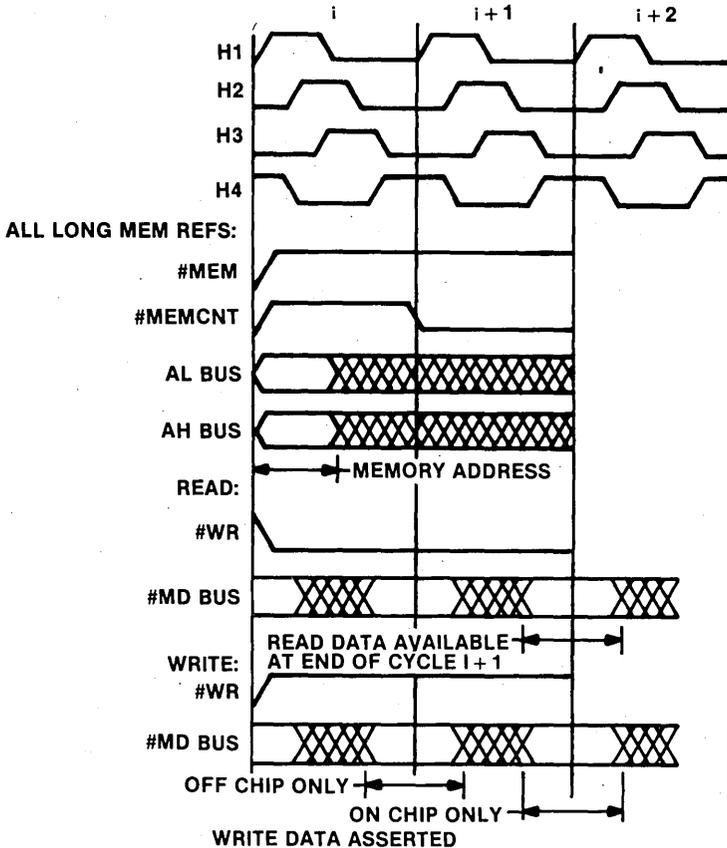


FIGURE 5-10 — LONG MEMORY CYCLE TIMING

The memory control signals #MEMCNT, #MEM, and #WR are specified in the microinstruction directly. Figure 5-10 shows these signals valid during a full microinstruction cycle because, once specified for a cycle, their state will not change during that cycle.

For all long memory references, #MEM must be asserted high for two consecutive cycles. #MEMCNT should be 1 for the first cycle, and 0 for the second cycle. #MEMCNT is asserted by specifying the MCNT symbol in the MICASM statements for the microinstruction. Various combinations of the #MEM and #WR microinstruction bits are specified by other MICASM symbols, as explained in paragraph 5.3.3.6. The 16-bit address to be accessed must be gated

onto the AH and AL Buses during the first cycle. The Peripheral/Memory Controller latches the memory address, so the address need not be asserted during the second cycle. It should be noted that this feature can be used to great advantage in microcode sequences since this allows the AH and AL Buses to be used for other functions during the second microinstruction cycle. In this manner, microcode functions may be overlapped which can result in shorter, faster executing microcode.

For read cycles, #WR is set to 0 for both cycles. The result of a read appears on the MD Bus in phase H4 of the second cycle. It may either be loaded into the T or IR Registers at the end of the second cycle or loaded into the P or N Bus at the beginning of the third cycle.

For write cycles, #WR is set to 1 for both cycles. When the write's destination is an on-chip address, the write data must be valid during H4 of the second microinstruction cycle; when the writes destination is an off-chip address, the write data is required to be valid during H4 of the first microinstruction cycle. The data used in an off-chip write is latched by the PMC during the first cycle, and therefore need not be valid during the second cycle, and conversely the data in an on-chip write need not be valid during the first cycle. This can be used advantageously in certain microcoding situations. If desired, however, data may be asserted during both cycles.

5.3.3.5 Interrupt Vector Reads

When an interrupt is received by the Peripheral/Memory Controller, the PMC asserts IACT on the Control Bus to the CPU, provided that STINT is a 1. The state of IACT may be tested by the CPU using an INT dispatch (see paragraph 5.3.5.1.5). If an interrupt is active the CPU may then read an interrupt vector supplied by the PMC on the MD Bus, indicating which interrupt has occurred. The interrupt vector read requires two cycles, as shown in the timing diagram in Figure 5-11.

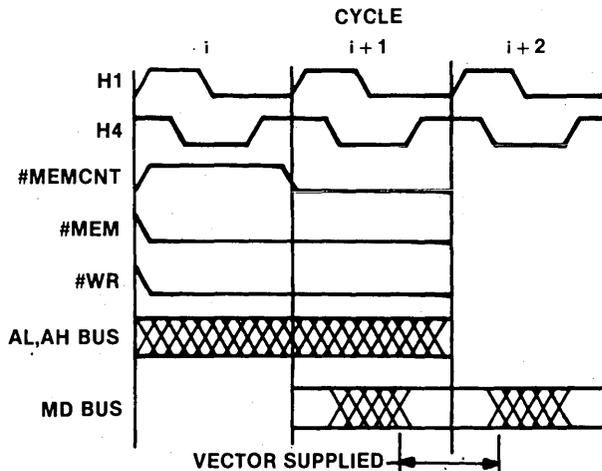


FIGURE 5-11 — INTERRUPT VECTOR READS

Notice that #MEM and #WR must be low for both cycles of the interrupt vector read. As with a long memory read, the vector is not available until the end of the second microinstruction cycle. An interrupt vector read may be coded in MICASM using the statements described in Table 5-3.

The value of the vector supplied by the PMC for each interrupt is shown in Figure 5-12. There is a distinction between the interrupt vector supplied by the PMC and the trap vector address at which the interrupt subroutine entry point address is stored.

INTERRUPT LEVEL	VECTOR SUPPLIED	TRAP VECTOR ADDRESS
0 (Reset)	---	>FFFE
1	>FE	>FFFC
2	>FD	>FFFA
3	>FC	>FFF8

FIGURE 5-12 — INTERRUPT VECTOR REFERENCES

The vector supplied by the PMC is the same as the TRAPn opcode for the TMS7000 Standard Instruction Set. In order to call the interrupt handler, the microcode generates the trap vector address from the vector supplied, and reads memory at that location to get the address of the interrupt handler subroutine. It should be noted that the interrupt trap vector addresses shown in Figure 5-12 are those implemented in the currently supplied TMS7000 Standard Instruction Set Microcode. Different trap vector addresses may be implemented if additional microcode is written to handle modified interrupt servicing.

5.3.3.6 Memory Control Signals

The three memory control signals output by the CPU and interpreted by the Peripheral/Memory Controller are:

- #MEMCNT (Memory Continue): asserted on the first cycle of a two-cycle long memory reference.
- #MEM (Memory): asserted if the microinstruction references memory of any kind (RAM, ROM, extended, peripheral).
- #WR (Write): 1 if a write is being performed; 0 if a read.

The interpretation of various combinations of these signals is described in Table 5-3.

TABLE 5-3 — MEMORY CONTROLS

#MEMCNT (previous)	#MEMCNT (current)	#MEM	#WR	Function	OTMD	MICASM Symbol
0	0	0	0	- No Mem Reference -	0	-See Note 1-
0	0	0	1	Gate O Bus to MD Bus	1	O>MD-See Note 2
0	0	1	0	Short Memory Read	0	MR
0	0	1	1	Short Memory Write	1	MW
1	0	0	0	2nd State Int. Vector	0	INTVEC
1	0	0	1	* Illegal *	1	-
1	0	1	0	2nd State Long Read	0	MR
1	0	1	1	2nd State Long Write	1	MW
0	1	0	0	1st State Int. Vector	1	MCNT, INTVEC
0	1	0	1	* Illegal *	1	-
0	1	1	0	1st State Long Read	1	MCNT, MR
0	1	1	1	1st State Long Write	1	MCNT, MW
1	1	x	x	* Illegal *	1	-

NOTES: 1. MICASM is not capable of generating this combination of memory controls directly.

2. This combination of memory control signals is also the default combination, produced by MICASM when no memory control is specified.

The MICASM symbol or symbols listed in Table 5-3 must be used to specify the appropriate combination of memory control signals. The #MEMCNT Microinstruction Bit is set independently by the MICASM symbol MCNT. The various combinations of #MEM and #WR Microinstruction Bits are set by specifying the MICASM symbols O>MD, MR, and MW. O>MD may be specified when no memory access is desired, but the ALU Output (O) Bus contents are to be gated onto the Memory Data (MD) Bus. OTMD, the signal which enables the O Bus to drive the MD Bus, is generated by the PMC and is defined as OTMD=#WR .OR. #MEMCNT. The O>MD MICASM statement has been defined only to assert OTMD during non-memory cycles by generating a unique combination of #WR and #MEMCNT which does not occur during actual memory cycles. (O>MD should not be coded during memory accesses). Note, however that the combination of memory controls produced by O>MD is also the default and will be produced by MICASM if no memory controls are coded in a particular microinstruction cycle.

MR is specified for a memory read operation, and MW for a memory write. For long memory cycles, which require two microinstructions, MCNT is specified in the first microinstruction only. MR or MW must be specified in both microinstructions.

5.3.4. Organization Of The TMS7000 CPU

This section describes the internal organization of the TMS7000 CPU. A block diagram is shown in Figure 5-13. Each of the internal registers and buses are 8 bits wide. The internal CPU buses are used to transfer information between registers and to devices external to the CPU. Normally a bus will be used to transfer data between two particular locations during a microinstruction cycle. (Buses are precharged at various times during each microinstruction and therefore cannot be used to store data). These types of transfers of information are explained in the following descriptions of the various buses and registers within the CPU. In most cases, a bus will usually have only one source or destination; however, it may be desirable to have either multiple sources or destinations for bus.

The case of multiple destinations of a bus is a simple extension of a single bus destination; a bus's contents are merely gated to several places simultaneously. This can be accomplished by

simply including the MICASM statements for each destination, i.e., MD>N and MD>P both coded in the same microinstruction cycle.

Multiple sources for a bus is more complex. Logically this may be coded in MICASM in a straightforward manner, just as multiple bus destinations are, however the result is quite different. The contents of a bus when multiple sources are specified is the logical OR of the two sources. This may be used advantageously in saving microcode in some situations with one restriction: the TMS7000 Emulator cannot be used to debug the microcode. It should be emphasized that this technique should be used only when absolutely necessary and the Emulator may not be used to check the microcode, which can make a design very difficult to debug.

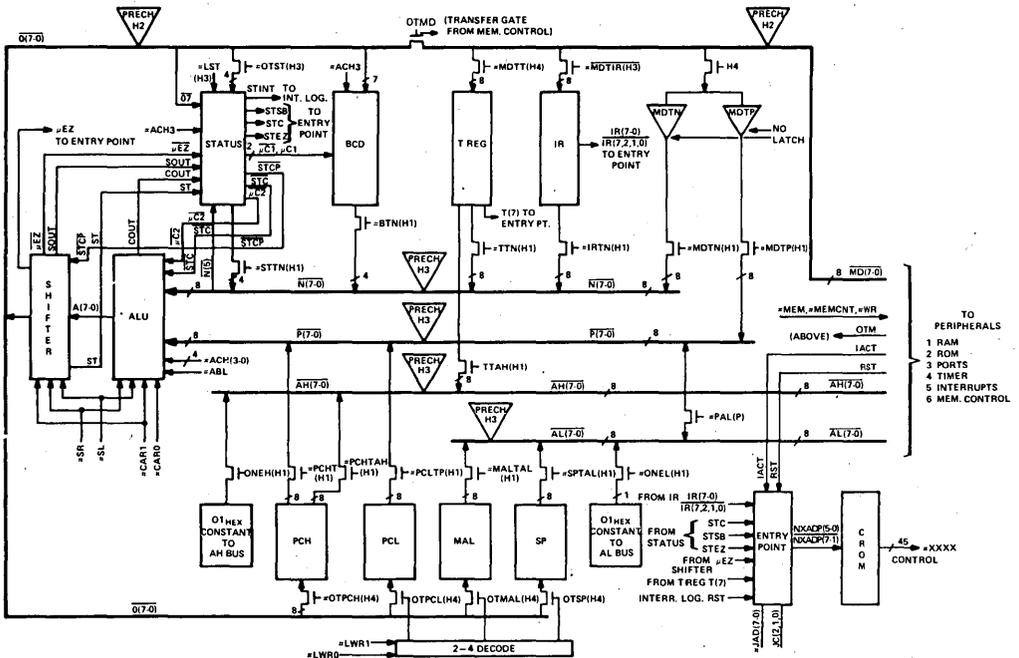


FIGURE 5-13 - INTERNAL ORGANIZATION OF THE TMS7000 CPU

5.3.4.1 THE P BUS

The P Bus is one of the inputs to the Arithmetic Logic Unit, or ALU. It is called P for positive because it always contains the positive or left-hand operand; in subtract operations, the ALU always computes P-N and in add operations, P+N is computed. The P Bus is loaded from the MD Bus, the AL Bus, the PCH Register, the PCL Register or with the constant >00 or >01. Any of the AL Bus sources may be placed on the P Bus by gating them onto the AL Bus and asserting the #PAL microinstruction bit, connecting the P Bus to the AL Bus. A P Bus source must be coded in each microinstruction cycle. All of the possible P Bus sources are shown in Figure 5-14.

P BUS SOURCE	MICASM SYMBOL(s)	HEX REPRESENTATION
MD Bus	MD>P	0000 4000 0000 000
PCH Register	PCH>P	0000 2000 0000 000
PCL Register	PCL>P	0000 1000 0000 000
MAL Register	MAL>AL, AL>P	0000 0030 0000 000
SP Register	SP>AL, AL>P	0000 0028 0000 000
>01 constant	ONE>AL, AL>P	0000 0060 0000 000
>00 constant	Z>P or DC>P	0000 0000 0000 000

FIGURE 5-14 — P BUS SOURCES

The hex representation in Figure 5-14 indicates the bits in a microinstruction that are affected when the MICASM symbol shown is specified for the P Bus source. Note that if a microinstruction requires no source on the P Bus, the MICASM symbol DC>P must be specified to indicate a don't care condition on the bus.

The P Bus is loaded at the beginning of a microinstruction cycle, on phase H1.

5.3.4.2 The N Bus

The N Bus is the second input to the ALU. It is called N for negative because in an ALU subtract operation, the N Bus contains the negative or right-hand operand. The N Bus is loaded from the MD Bus, the T Register, the IR Register, the Status Register, the BCD Constant Register or the constant >00. The source of the N Bus is indicated directly by a bit in the microinstruction word. If the bit is 1, the source is gated onto the N Bus. An N Bus source must be coded in each microinstruction cycle. All the possible N Bus sources are shown in Figure 5-15.

N BUS SOURCE	MICASM SYMBOL(s)	HEX REPRESENTATION
MD Bus	MD>N	0000 0800 0000 0000
T Register	T>N	0000 0400 0000 0000
Status Register	ST>N	0000 0200 0000 0000
BCD Constant	BCD>N	0000 0100 0000 0000
IR Register	IR>N	0000 0080 0000 0000
>00 constant	Z>N or DC>N	0000 0000 0000 0000

FIGURE 5-15 — N BUS SOURCES

If a microinstruction does not require an operand on the N Bus, the MICASM symbol DC>N must be specified to indicate a don't care condition on the bus.

The N Bus is loaded at the beginning of a microinstruction cycle, on phase H1.

5.3.4.3 The AL Bus

The AL (Address Low) Bus holds the the lower 8 bits of all memory addresses. This includes references to the Register File, Peripheral File, on-chip, and extended memory. The AL Bus is loaded during phase H1, at the beginning of a microinstruction cycle. The sources of the AL Bus are the MAL Register, the SP Register, or the constant >00 or >01. The constant >01 is provided to efficiently address RAM location >01, (the B register of the standard TMS7000). This also facilitates addressing registers 16 and 17 (>10 and >11). An AL Bus source must be specified in each microinstruction cycle.

The AL Bus may also be connected to the P Bus by asserting the #PAL microinstruction bit, which can be accomplished by coding the P>AL MICASM instruction. In this manner, the AL Bus sources (MAL, SP, or the constant >00 or >01) may be gated onto the AL Bus and then onto the P Bus to be operated on by the ALU. Likewise, the P Bus sources (PCH, PCL, and MD Bus contents) may be gated onto the P Bus and then onto the AL Bus to serve as low order address lines. The MD Bus contents transferred are those present at the start of the microinstruction, i.e., those output by the previously executed microinstruction. All of the possible sources of the AL Bus are listed in Figure 5-16.

AL BUS SOURCE	MICASM SYMBOL(s)	HEX REPRESENTATION
MAL Register	MAL>AL	0000 0010 0000 0000
SP Register	SP>AL	0000 0008 0000 0000
PCL Register	PCL>P, P>AL	0000 1020 0000 0000
PCH Register	PCH>P, P>AL	0000 2020 0000 0000
MD Bus	MD>P, P>AL	0000 4020 0000 0000
>01 Constant	ONE>AL	0000 0040 0000 0000
>00 Constant	Z>AL or DC>AL	0000 0000 0000 0000

FIGURE 5-16 — AL BUS SOURCES

If no AL Bus source is required, the MICASM symbol DC>AL must be specified to indicate a don't care condition on the bus:

5.3.4.4 The AH Bus

The 8-bit AH (Address^hHigh) Bus contains the high-order byte of the address referenced by the CPU. It is loaded during H1, at the beginning of a microinstruction cycle. It may be loaded with the contents of the PCH Register, the T Register, or the constant >00 or >01. The high byte of the program counter is intended to be stored in PCH; the T Register is intended to hold the high byte of other addresses in memory. The constant >01 is provided to efficiently access addresses in the Peripheral File (i.e., addresses of the form >01XX). An AH Bus source must be coded in each microinstruction cycle. The sources of the AH Bus are summarized in Figure 5-17.

AH BUS SOURCE	MICASM SYMBOL(s)	HEX REPRESENTATION
PCH Register	PCH>AH	0000 0002 0000 0000
T Register	T>AH	0000 0004 0000 0000
>01 Constant	ONE>AH	0000 0001 0000 0000
>00 Constant	Z>AH or DC>AH	0000 0000 0000 0000

FIGURE 5-17 — AH BUS SOURCES

If no AH Bus source is required, the MICASM symbol DC>AH must be specified to indicate a don't care condition on the bus.

5.3.4.5 The O Bus

The O (Output) Bus always contains the output of the ALU-Shifter combination. Its contents may be loaded onto the MD Bus, or into the Status, PCH, PCL, MAL, or SP Registers. The Status Register is loaded by the low-true microinstruction bit #—O>ST. The PCH Register is loaded by the high-true microinstruction bit #O>PCH. The load signals for the other destination registers (MAL, PCL, and SP) are encoded in the microinstruction bits #LOWWRITE(1-0), as shown in Figure 5-18. Note that since these bits are encoded, these three O Bus destinations are mutually exclusive; that is, only one of these destinations may be specified in a given microinstruction cycle.

#LOWWRITE		O BUS DESTINATION	MICASM SYMBOL
1	0		
0	0	—none—	---
0	1	MAL Register	O>MAL
1	0	PCL	O>PCL
1	1	SP	O>SP

FIGURE 5-18 — LOWWRITE (1-0) DESCRIPTION

There is no microinstruction bit that directly loads the MD Bus from the O Bus, because the MD Bus contents are under control of the Peripheral/Memory Controller (PMC). This transfer is controlled by the OTMD signal sent from the PMC to the CPU on the C Bus. OTMD is asserted on every memory write cycle, (on-chip or extended memory), and on the first state of every long memory cycle. This is diagrammed in Table 5-3.

The O Bus is normally gated onto the MD Bus unless otherwise required in a memory cycle. Optionally, the O>MD symbol may be coded in a MICASM statement. MICASM sets up the appropriate values of the #MEM and #WR microinstruction bits so that OTMD will be asserted by the Peripheral/Memory Controller. The O Bus contents may then be loaded into the T or IR Registers from the MD Bus. Refer to paragraph 5.3.3.6 for a description of OTMD.

To write the O Bus contents to memory, the memory control signals must be specified. The destinations of the O Bus are identified in Figure 5-19.

O BUS DESTINATION	MICASM SYMBOL	MICROINSTRUCTION FIELD HEX REPRESENTATION
ST Register	O>ST	0000 0000 0000 0000 (Low True)
PCH Register	O>PCH	0010 8000 0000 0000
PCL Register	O>PCL	0002 8000 0000 0000
MAL Register	O>MAL	0001 8000 0000 0000
SP Register	O>SP	0003 8000 0000 0000
T Register	*[O>MD],MD>T	0008 8000 2000 0000
IR Register	*[O>MD],MD>IR	0004 8000 2000 0000
Short Mem Cycle	MW	0000 8000 6000 0000
Long Mem, Cycle 1	MCNT,MW	0000 8000 E000 0000
Long Mem, Cycle 2	MW	0000 8000 6000 0000

} Only One
Of Three

* Specifying O>MD here is optional

FIGURE 5-19 — O BUS DESTINATIONS

The O Bus is loaded during phase H4 of the microinstruction cycle. It contains the result of the ALU and Shifter operations specified in the current microinstruction.

5.3.4.6 The MD Bus

The MD (Memory Data) Bus is a bidirectional bus that transfers data to and from the CPU. Data is valid on MD during phase H4 of a microinstruction cycle, which spans two microinstructions. Thus, data may be read from the MD Bus onto the P or N Bus at the beginning of a cycle (H1), and the ALU results then loaded back onto the MD Bus at the end of the cycle (H4). It is important to note that when using data from the MD Bus during H1 of a particular microinstruction cycle, the actual data available will be the contents loaded onto the MD Bus during the end of the previous cycle.

At the end of a cycle, the MD Bus may be loaded in one of three ways:

- 1) The O Bus contents may be gated onto the bus.
- 2) The on-chip RAM or ROM may place data onto the bus.
- 3) The Peripheral/Memory Controller may place data onto the bus.

The MD Bus contents are controlled by the Peripheral/Memory Controller (PMC). The PMC sends the OTMD signal to the CPU to signal loading the MD Bus from the O Bus. The CPU requests use of the MD Bus by asserting combinations of the #MEM, #MEMCNT, and #WR signals, as shown in Table 5-3. The PMC sends signals to the on-chip ROM and RAM to control their accesses to the bus.

The timing of read and write accesses to memory is explained in paragraph 5.3.2.3. For short memory reads, data is available at the end of the microinstruction that initiated the read. This data may be loaded into the T or IR Registers during that microinstruction by specifying the MD>T or MD>IR MICASM symbols, respectively. The data may be loaded into the P or N Bus on the next microinstruction by specifying the MD>P or MD>N symbols in the MICASM statement for the next microinstruction. For short memory cycle writes, the O Bus data is placed on the MD Bus, and the MW MICASM symbol specified. For long memory reads, the desired address is placed on the AH and AL lines, and the MR and MCNT symbols specified in the first of the two cycles required. At the end of the second cycle, data is available on MD. (The memory address is latched by the PMC on the first cycle, and need not be asserted on the second cycle). For long memory writes, the address is specified in the first cycle, and the data is placed on the MD Bus for the first and/or second cycles. The destinations of the MD Bus in the CPU are described in Figure 5-20.

MD BUS DESTINATION	MICASM WHEN LOADED	SYMBOL
T Register	End of Cycle	MD>T
IR Register	End of Cycle	MD>IR
P Bus	Start of Cycle	MD>P
N Bus	Start of Cycle	MD>N

FIGURE 5-20 — MD BUS DESTINATIONS

5.3.4.7 ALU Operation

The Arithmetic Logic Unit (ALU) accepts as inputs the values on the P and N Buses, and outputs its result to the Shifter. The ALU operation is controlled by the #ALUCNTL(3-0) and #ABL lines from the current microinstruction. The ALU operates on the values loaded on the P and N Buses during H1 of the current microinstruction and produces an 8-bit output which is input to the Shifter, and a carry bit (COUT), which is an arithmetic carry bit based on the 8-bit ALU operation. To specify the carry-in, the ALU accepts the #SHIFTCNTL(3-0) bits from the current microinstruction. An overall block diagram of the ALU appears in Figure 5-21.

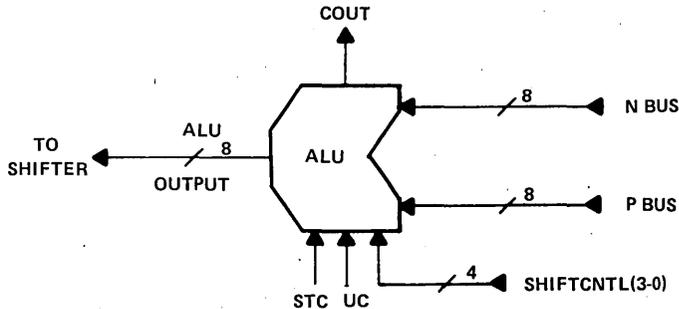


FIGURE 5-21 -- ALU BLOCK DIAGRAM

The available operations of the ALU are defined in Figure 5-22.

#ALUCNTL (3-0) #ABL	HEX REPRESENTATION	MICASM SYMBOL	ALU OUTPUT
0000 0	0000 0000 0000 0000	PADDN	P + N + CI
0000 1	0000 0000 0008 0000	XNOR	P XOR N
0001 1	0000 0000 0018 0000	AND	P AND N
0010 1	0000 0000 0028 0000	IPORN	(NOT P) OR N
0011 1	0000 0000 0038 0000	PASSN	N
0100 1	0000 0000 0048 0000	PORIN	P OR (NOT N)
0101 1	0000 0000 0058 0000	PASSP	P
0110 1	0000 0000 0068 0000	FF	>FF
0111 1	0000 0000 0078 0000	OR	P OR N
1000 1	0000 0000 0088 0000	NOR	P NOR N
1001 1	0000 0000 0098 0000	ZERO	>00
1010 1	0000 0000 00A8 0000	INVP	NOT P
1011 1	0000 0000 00B8 0000	IPANDN	(NOT P) AND N
1100 1	0000 0000 00C8 0000	INVN	NOT N
1101 1	0000 0000 00D8 0000	PANDIN	P AND (NOT N)
1110 1	0000 0000 00E8 0000	NAND	P NAND N
1111 0	0000 0000 00F0 0000	PSUBN	P - N - 1 + CI
1111 1	0000 0000 00F8 0000	XOR	P XOR N

FIGURE 5-22 -- ALU FUNCTIONS

The Carry-in Bit of the ALU (CI) is specified by the #SHIFTCNTL(3-0) bits of the microinstruction, which are described in full in the next section. For operations requiring no shifting of the ALU contents, the possible carry-in bits are defined in Figure 5-23.

#SHIFTCNTL 3 2 1 0	MICASM SYMBOL	ALU CARRY IN (CI)
0 0 0 0	ZCI	0
0 0 0 1	ONECI	1
0 0 1 0	UCI	UC — Micro Carry Bit
0 0 1 1	STCI	STC — Status Carry Bit

FIGURE 5-23 — ALU CARRY IN VALUES

The Micro Carry Bit (UC) is the carry-out from the ALU operation of the immediately preceding microinstruction. This is not the same as the Shift-out Bit (SOUT) from the Shifter operation of the previous microinstruction. The Status Carry Bit (STC) is the Carry bit of the Status Register.

The arithmetic Carry-out Bit from the ALU (COUT) is 1 if there is a carry-out during an add (PADDN) or subtract (PSUBN) operation in the ALU. For an add operation, COUT = 1 indicates there was a carry, i.e., the sum of the (unsigned) operands exceeds 255. For a subtract operation, COUT = 0 indicates there was a borrow, i.e., the P operand was lower than the N operand (unsigned). For all other operations, i.e., logical operations, COUT is set to 0. COUT is sent to the Status Register circuitry for possible loading into STC, the Status Carry Bit.

As an example of ALU operation, the following symbols appearing in a MICASM statement,

PADDN,ZCI

will cause the ALU to calculate the sum of the P and N Bus contents. To calculate the difference between the P and N Bus contents,

PSUBN,ONECI

must be specified. A 1 must be carried in since no borrow was desired. Figure 5-24 details two microcode examples. The microinstructions read the current byte addressed by the PC, place it in the T Register, and increment the PC.

.ORG IMMED1	' Read immediate byte, 1st cycle
PCL>P,P>AL,	' Define location of microinstruction
PCH>AH,	' Place PCL on AL Bus via P Bus
Z>N,	' Place PCH on AH Bus
PADDN,ONECI,	' Place Zero on N Bus
O>PCL,	' Increment PCL by 1 (sets Micro Carry UC
MCNT,MR,	' Place result back in PCL
JUNC(IMMED2);	' 1st cycle of long read
	' Goto next cycle
.ORG IMMED2	' Read immediate byte, 2nd cycle
DC>AH,DC>AL,	' Don't care what's on AH and AL since address was
	latched on 1st cycle
PCH>P,	' Place PCH on P Bus
Z>N,	' Place Zero on N Bus
PADDN,UCI,	' Add micro carry from PCL increment
O>PCH,	' Place result back in PCH
MR,	' Meanwhile, continue memory read
MD>T,	' And place the byte read into T
JUNC(NEXT);	' Then goto next instruction

FIGURE 5-24 — MICROCODE EXAMPLE

Notice that an increment was done in IMMED1 by using an ALU carry-in of 1. The second instruction (IMMED2) incremented the high byte of the PC only if the Micro Carry Bit (UC) generated by IMMED1 was 1.

5.3.4.8 Shifter Operation

The Shifter performs a variety of 1-bit shift operations on the output of the ALU. The #SHIFTCNTL(3-0) lines control the following ALU and Shifter characteristics:

- The ALU Carry-in Bit (CI)
- The shift direction (L or R)
- The bit shifted into the Shifter

Figure 5-25 shows the various combinations of shift control lines.

#ShiftCntl 3 2 1 0	ALU Ci	Shift Direction	Shift-In Bit	MICASM Symbol
0 0 0 0	0	No Shift	—	ZCI
0 0 0 1	1		—	ONECI
0 0 1 0	UC		—	UCI
0 0 1 1	STC		—	STCI
0 1 0 0	0	Shift Left	ALU(7)	RLO
0 1 0 1	1		ALU(7)	RLZ
0 1 1 0	0		STC	RLCO
0 1 1 1	1		STC	RLCZ
1 0 0 0	0	Shift Right	ALU(0)	RRO
1 0 0 1	1		ALU(0)	RRZ
1 0 1 0	0		STC	RRCO
1 0 1 1	1		STC	RRCZ
1 1 X X	*	Invalid	*	*

FIGURE 5-25 – SHIFT/ALU CARRY-IN CONTROL

For #SHIFTCNTL=00XX, no shifting is performed, and the ALU Carry-in Bit Ci is as described in the ALU description, above. For #SHIFTCNTL=010X, the ALU output is rotated left, with the most significant bit, ALU(7), shifted in from the right. For #SHIFTCNTL=011X, the ALU output is rotated left through the Status Carry Bit, STC. For #SHIFTCNTL=100X, the ALU output is rotated right, and for #SHIFTCNTL=101X, the output is rotated right through the carry bit. The MICASM symbols represent this, with the last character indicating the value of the ALU Ci bit. #SHIFTCNTL=11XX is an invalid command and must never be specified.

The Shift-out Bit (SOUT) shifted out in a rotate instruction is sent to the Status Register. It will be loaded as the new Status Carry Bit (STC) if the #-LST microinstruction bit is set. Operation of each of the shift instructions is diagrammed in Figure 5-26.

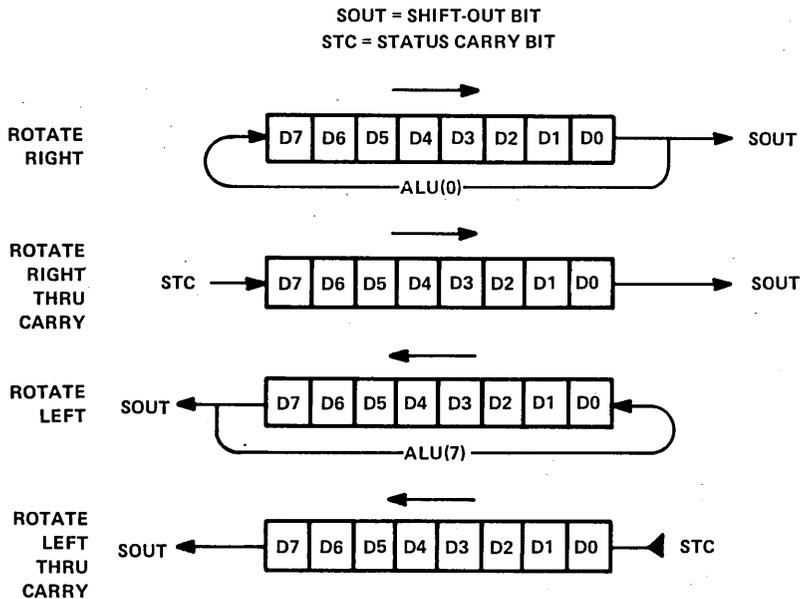


FIGURE 5-26 – SHIFTER OPERATION

5.3.4.9 IR Register

The Instruction Register (IR) is a register intended to hold the assembly language opcode. It is loaded from the MD Bus by specifying the MD > IR symbol in a MICASM statement. It may be loaded onto the N Bus with the IR > N MICASM symbol.

The TMS7000 Microarchitecture is designed to dispatch (branch) on various subfields of the IR contents, providing for the execution of appropriate microcode for each assembly language instruction. The IR may be considered to have two possible formats:

- 1) Format 0 is indicated by a 0 in IR(7), the most significant bit of the IR Register. In this format, bits IR(6-4) form a 3-bit Group field and bits IR(3-0) form a 4-bit Function field.
- 2) Format 1 is indicated by a 1 in IR(7). In this format, bits IR(6-3) form a 4-bit Group field and bits IR(2-0) form a 3-bit Function field.

The formats of the IR Register are diagrammed in Figure 5-27.

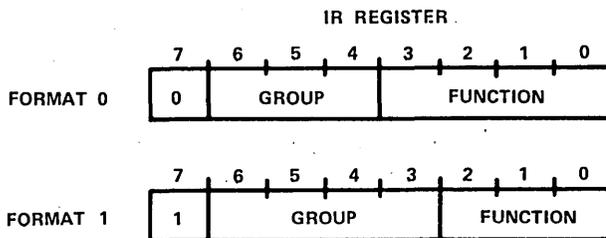


FIGURE 5-27 — IR REGISTER FORMATS

The terms group and function refer to logical subsets of assembly language opcodes. In the TMS7000 standard instruction set the Group field in an opcode indicates the addressing mode of the instruction, and the Function field indicates the arithmetic or logical operation performed on the operands. The microarchitecture is designed to allow significant sharing of microinstructions among opcodes within the same group or function. In the microcode for the standard TMS7000, for instance, all opcodes of the form >1X share microcode which fetches the A Register and a general RF register.

The mechanisms for dispatching on the Group and Function field values in the IR are described in Section 4. Dispatching on an IR subfield may be performed on the first microinstruction after the IR is loaded. Thereafter, dispatching may be performed by microinstructions up to and including the next one that reloads the IR. If no dispatching is required, then the IR may be used as a general purpose 8-bit register.

5.3.4.10 The Status Register

The Status Register (ST) is an 8-bit register with contents indicating various conditions of the CPU. Each bit of the Status Register has a special meaning and separate circuitry devoted to it. The format of the ST Register is shown in Figure 5-28.

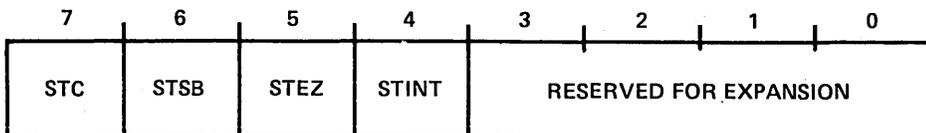


FIGURE 5-28 — STATUS REGISTER

STC is the Status Carry Bit. It holds either the carry-out of the ALU, the shift-out of the Shifter, or the decimal arithmetic carry-out. STSB is the Status Sign Bit. It contains the most significant bit of the O Bus contents. STEZ is the Status Equal to Zero Bit. It contains a 1 when all bits of the O Bus are zero. STINT is the Status Interrupt Enable Bit. Bits 3-0 of the Status Register are reserved for future expansion. These bits will be zeros when the ST Register is loaded onto the N Bus.

The existing Status Register Bits may be modified in one of two ways:

- 1) All bits may be replaced by the contents of the O Bus.
- 2) The STC, STSB, and STEZ bits may be set according to their particular input circuitry. The STINT Bit is unaffected in this case.

The Status Register Sources are summarized in Figure 5-29.

ST REGISTER SOURCE	MICASM SYMBOL	HEX REPRESENTATION
O Bus Input	O>ST	0000 0000 1000 0000 (Asserted low)
Circuitry	LST	0000 8000 0000 0000 (Asserted low)

FIGURE 5-29 – ST REGISTER SOURCE

The O Bus is gated into the Status Register if the #O>ST Microinstruction Bit is asserted low. This may be specified by the O>ST symbol appearing in a MICASM statement. The STC, STSB, and STEZ Bits are loaded when the #LST Microinstruction Bit is asserted low. This may be specified by the LST symbol appearing in the MICASM statement. There is no way to individually load the STC, STSB, and STEZ Bits; they must be loaded together. This feature permits an efficient implementation of the TMS7000 status logic, typically a very costly item in single-chip microarchitectures. The special circuitry defining the value of the STC, STSB, and STEZ Registers is described in the following paragraphs.

5.3.4.10.1 The Status Carry Bit (STC)

When the #LST signal is asserted by coding the LST MICASM instruction, the STC Bit will be loaded from one of three sources:

- 1) The ALU Arithmetic Carry-out Bit (COUT); This is the carry/borrow bit generated by the ALU on arithmetic operations. COUT is loaded if no Shifter operation is specified, i.e., #SHIFTCNTL = 00XX.
- 2) The Shifter Shift-out Bit (SOUT). This is the bit shifted out in Shifter operations. If a Shifter operation is specified—i.e., #SHIFTCNTL > 0011—then SOUT is loaded into the STC Bit (whether a rotate thru carry was specified or not).
- 3) The BCD Decimal Carry/Borrow Out Bit (DCOUT). This is the carry bit computed by the decimal adjust hardware within the BCD Constant Register. It is loaded into the STC Status Carry Bit if the #BCD > N Bit is set, indicating a decimal adjust constant is loaded onto the N Bus.

5.3.4.10.2 The Status Sign Bit (STSB)

When #LST is asserted, the input to the STSB Bit is O(7), the most significant bit of the O Bus.

5.3.4.10.3 The Status Equal To Zero Bit (STEZ)

When #LST is asserted, the input to the STEZ Bit is the Micro Equal-to-Zero Bit, UEZ. The UEZ Bit is simply the logical NOR of all O Bus lines. That is, if all O Bus lines are 0, UEZ is set to 1. Otherwise, it is set to 0.

5.3.4.10.4 The Status Interrupt Enable Bit (STINT)

The STINT Bit may only be modified by loading the O Bus contents into the Status Register. The STINT Bit corresponds to bit O(4) in this case. STINT is output from the CPU to the Peripheral/Memory Controller on the C (Control) Bus between the CPU and PMC. If STINT = 0, the PMC will not pass an interrupt to the CPU via the IACT line (also in the C Bus). If STINT = 1, the PMC will assert IACT on an interrupt. By dispatching on the IACT bit, the microcode is able to test for interrupts.

Due to propagation delays, the effect of loading STINT on IACT takes two microinstruction cycles to be asserted. Accordingly, if STINT is updated in cycle i , IACT will not be valid until cycle $i + 2$. Thus a JINT dispatch on IACT will not jump correctly if coded in state $i + 1$.

5.3.4.11 BCD Constant Register

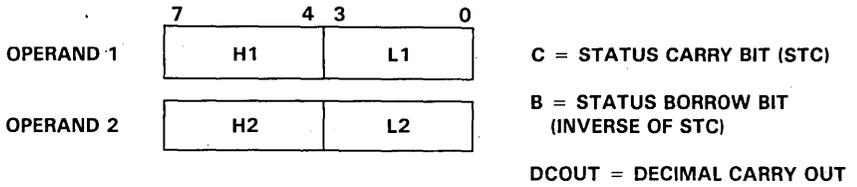
The BCD Constant Register is a module which generates a correction constant for binary coded decimal arithmetic operations. Decimal numbers on the TMS7000 are represented with 2 binary coded decimal digits per byte, with the least significant digit in the least significant nibble, bits 3-0, of a byte. For example, the decimal number 78 would be represented in binary as '01111000', or >78 . To perform decimal addition on two BCD Bytes X and Y, the following operations must be performed:

- 1) The binary sum of X and Y is computed, with the STC Bit carried in, and the result saved temporarily.
- 2) A decimal correction constant is computed by the BCD hardware.
- 3) The correction constant is added to the saved result to produce the final BCD sum.

Each of these operations requires a microinstruction cycle.

The STC Bit is added in order to permit adding multiprecision strings of BCD digits. Decimal subtraction (with borrow) is similar to the above procedure. The binary difference $X - Y$ is first computed, and the correction constant then subtracted from the result.

Figure 5-30 indicates the decimal correction constant and decimal carry out bit generated for decimal addition and subtraction.



	$L1 + L2 + C < 10$	$10 \leq L1 + L2 + C$		$L1 - B \geq L2$	$L1 - B < L2$
$H1 + H2 < 9$	> 00	> 06	$H1 > H2$	> 00 DCOU	> 06 DCOU
$H1 + H2 = 9$	> 00	> 66 DCOU	$H1 = H2$	> 00 DCOU	> 66
$H1 + H2 > 9$	> 60 DCOU	> 66 DCOU	$H1 < H2$	> 60	> 66

DECIMAL ADD WITH CARRY

DECIMAL SUBTRACT WITH BORROW

FIGURE 5-30 – BCD CORRECTION CONSTANT GENERATION

The BCD constant logic uses signals from the ALU such as the 8-bit carry (COU), the ALU operation code #ALUCNTL(3-0), and ALU outputs on the O Bus to determine the correction constant and Decimal Carry-out Bit (DCOU). Like the binary arithmetic carry, DCOU is 1 if a carry is required after an addition, and 0 if a borrow is required after a subtraction. Figure 5-30 indicates the conditions in which DCOU is 1. DCOU is sent to the Status Register for possible loading into the STC Status Carry Bit.

Three microinstruction cycles are required to perform a decimal arithmetic operation. The timing for a decimal arithmetic operation is shown in Figure 5-31.

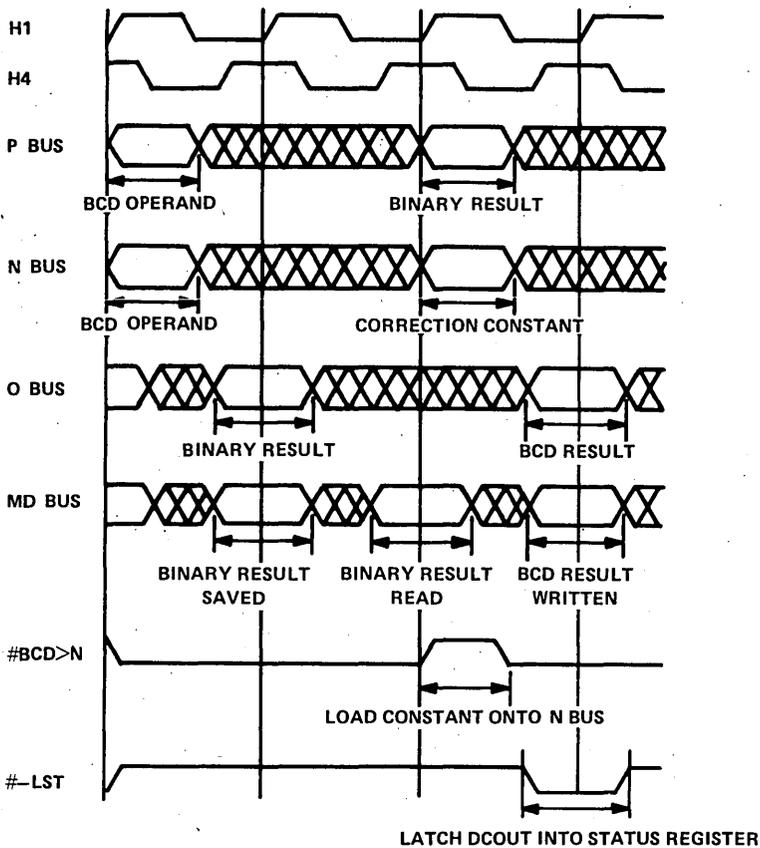


FIGURE 5-31 — BCD ARITHMETIC OPERATION TIMING

The first state loads the BCD operands onto the P and N Buses, and performs the appropriate ALU operation (PADDN or PSUBN) to produce the binary result. The binary result must be stored in a temporary location for use in the third state. The BCD operation diagrammed in Figure 5-31 assumes the result is stored in the RF. The second state reads this binary result from the Register File and leaves it on the MD Bus. This state allows the BCD constant hardware to determine the correction constant and Decimal Carry-out Bit, DCOUT. The third state loads the binary result onto the P Bus and the correction constant onto the N Bus and performs the appropriate ALU operation to produce the correct BCD result. The Status Register should be loaded in this state by coding an LST instruction in MICASM.

The MICASM statement shown in Figure 5-32 implement a decimal add with carry. A source operand is added to a destination operand, and the result stored in the destination operand (a register in the RF). The T Register is assumed to contain the source operand, the MD Bus contains the destination operand, and the MAL Register contains the register number of the destination operand.

.ORG DAC0	' Decimal Add w/ Carry, first state
Z>AH,	' Place destination register address
MAL>AL,	' on address bus: AH=0, AL=MAL
MD>P,	' Dest. operand to P Bus
T>N,	' Source operand to N Bus
PADDN,STCI,	' Add them, including carry from last DAC
MW,	' Store binary result in dest. register
JUNC(DAC1);	' Goto DAC1
.ORG DAC1	' DAC, second state
Z>AH,	' Read binary result back. Put dest. addr
MAL>AL,	' on addr. bus: AH=0, AL=MAL
DC>P, DC>N,	' Don't cares to P and N Bus
PADDN,ZCI,	' Maintain ALU operation code (PADDN)
MR,	' Read binary result, placed on MD Bus
JUNC(DAC2);	' Goto DAC2
.ORG DAC2	' DAC, third state
Z>AH,	' Put destination address on Address Bus
MAL>AL,	' (AH=0, AL=MAL)
MD>P,	' Put binary result on P Bus
BCD>N,	' Put BCD correction constant on N Bus
PADDN,ZCI,	' Add them (with no carry)
LST,	' Load Status register with decimal carry
MW,	' Store BCD result to destination register
JUNC(NEXT);	' Goto next microinstruction.

FIGURE 5-32 — MICASM STATEMENT

For a decimal subtract operation, the PADDN symbols should be replaced with PSUBN. State DAC2 should subtract the BCD constant via the MICASM symbols PSUBN,ONECI. A carry-in of 1 is needed since no borrow is required.

5.3.4.12 Other Registers

The remaining registers implemented in the TMS7000 CPU include five storage registers and two constant registers. Two of the storage registers, the PCH and PCL, are used to hold the high and low bytes of the Program Counter. The Program Counter contents are normally essential to CPU operation, hence the PCH and PCL registers are almost never used as general purpose storage.

Two other storage registers, the Temporary or T Register and the MAL or Memory Address Low Byte Register may be paired to store the high and low bytes of a memory address, or used separately with the T Register serving as temporary storage and a memory address being generated from the MAL and a constant.

There are two constant registers used for generating the constant >01; one for each of the AH and AL Buses. Thus either of these buses may be loaded with either >00 or >01 if necessary. This capability is used for, among other things, generating RF and PF Addresses.

The SP or Stack Pointer is normally used to hold a pointer to the stack in RAM, but may be used as temporary data storage if a stack is not implemented or if the SP contents are not needed.

5.3.5 Microinstruction Sequence Control Overview

This section describes the mechanisms used in controlling the sequence of microinstruction execution, which include generation of the next microinstruction address in both conditional and unconditional branching. Included is a description of dispatching capabilities which can be used to share microstates among several assembly language instructions.

Microinstructions are stored in the Control ROM, or CROM, on the TMS7000 chip. A characteristic of horizontally microprogrammed architectures like the TMS7000 is that each microinstruction indicates the address at which the next microinstruction to be executed is located. In the TMS7000, the next microaddress is specified by two fields:

- 1) #JMPADDR(7-0), an 8-bit field indicating a base address in CROM.
- 2) #JMPCNTL(2-0), a 3-bit code indicating one of 8 dispatch offsets from the base address in #JMPADDR.

If #JMPCNTL(2-0) = 000, then the #JMPADDR field is simply the address of the next microinstruction. If #JMPCNTL(2-0) is nonzero, it indicates what data will replace the low order bits of #JMPADDR, and thus form the next microaddress. This technique is called dispatching, and is extremely easy to implement in MOS technology.

All conditional branching in microcode is accomplished by means of dispatching. A base address is specified in the #JMPADDR(7-0) bits of the microinstruction. The #JMPCNTL(2-0) lines indicate what data is used to form the low order bits of the base address to generate the new microinstruction address. As an example, Figure 5-33 depicts dispatching on the IR(3-0) Bits.

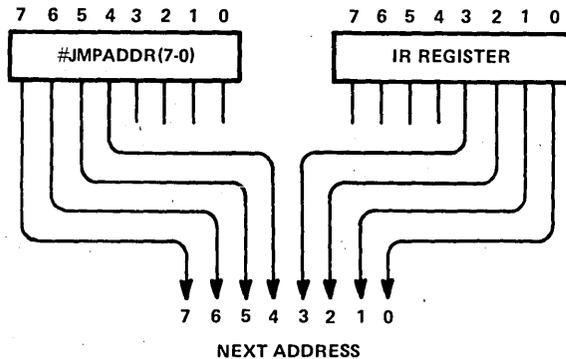


FIGURE 5-33 — MICROINSTRUCTION DISPATCH EXAMPLE

The dispatch field bits, like IR(3-0), actually replace the low order address bits in the #JMPADDR(7-0) field; they are not OR'ed with them. For example, suppose #JMPADDR was specified to be > 11, and the #JMPCNTL(2-0) lines are set to 110, indicating a dispatch on STC, the Status Carry Bit. If STC were 0, the next microaddress would be > 10.

Figure 5-34 summarizes the possible dispatch fields and the MICASM code to indicate the next address.

#JMPCNTL 2 1 0	NEXT ADDRESS								MICASM Format
	7	6	5	4	3	2	1	0	
0 0 0	J7	J6	J5	J4	J3	J2	J1	J0	JUNC(addr)
0 0 1	J7	J6	J5	J4	IR3	IR2	IR1	IR0	IRL(baseaddr)
0 1 0	J7	J6	J5	J4	J3	J2	J1	T7	JT7 (oneaddr,zeroaddr)
0 1 1	J7	J6	J5	J4	J3	J2	J1	UEZ	JUZ(oneaddr,zeroaddr)
1 0 0	J7	J6	J5	J4	J3	J2	0	IACT	INT(oneaddr,zeroaddr)
1 0 1	J7	J6	J5	IR7	IR6	IR5	IR4	(1)	IRH(baseaddr)
1 1 0	J7	J6	J5	J4	J3	J2	J1	STC	JC(oneaddr,zeroaddr)
1 1 1	J7	J6	J5	J4	J3	J2	J1	MJMP	MJMP(oneaddr,zeroaddr)

(1) IR3 .or. (.not. IR7)

Jn — #JMPADDR(n)
 IRn — IR Register bit n
 T7 — T register sign bit (bit 7)
 UEZ — 1 if 0 bus = > 00, 0 otherwise
 IACT — Interrupt Active line from PMC
 STC — Status Carry Bit
 MJMP — Macro jump: test Status Register bits
 baseaddr — Base micro-address for dispatch
 oneaddr — Next micro-address if bit 0 is 1
 zeroaddr — Next micro-address if bit 0 is 0

FIGURE 5-34 — NEXT MICRO ADDRESS GENERATION

5.3.5.1 Dispatch Conditions

Each of the dispatch possibilities is further explained in the following sections.

5.3.5.1.1 Unconditional Branching (JUNC)

If conditional branching of the microcode is not desired, #JMPCNTL should be set to 000. The symbol

JUNC(addr)

appearing in a MICASM statement will cause the TMS7000 to branch unconditionally to the microinstruction at address addr after the current microinstruction is executed. The addr field may be a constant or, more practically, a symbol equated to the desired address of the microinstruction. The address addr is loaded into the #JMPADDR(7-0) field of the current microinstruction.

5.3.5.1.2 Function Dispatch (IRL)

When #JMPCNTL = 001, the next microinstruction is determined by the low four bits of the the IR Register. This is specified in MICASM as:

IRL(baseaddr)

The baseaddr is loaded into the #JMPADDR(7-0) field of the microinstruction. The next micro address is determined by replacing the bits 3-0 of the base address with bits 3-0 of the IR Register. To avoid confusion, it is convenient to make the base address a multiple of 16 i.e., bits baseaddr(3-0) = 0, since they will be ignored. The IRL dispatch is indicated pictorially in Figure 5-35.

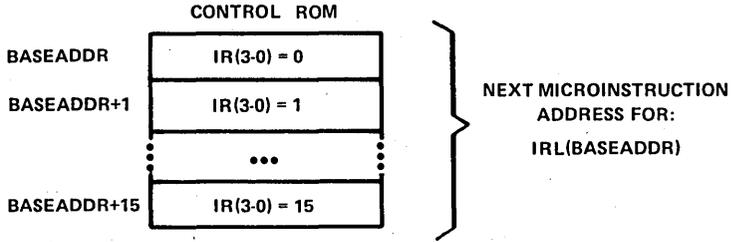


FIGURE 5-35 – IRL DISPATCH

An IRL dispatch is a dispatch on the Function field of the IR. In the TMS7000 Standard Instruction Set the Function field indicates the arithmetic operation to be performed. This is contrasted with the Group field, bits 7-4, which indicates the addressing mode of the instruction. Even though Format 1 instructions have a 3-bit Function field; IR(2-0), the IRL dispatch still performs a 16-way branch on the lower 4 bits of the IR Register. The Function dispatch for Format 1 opcodes thus depends on the value of the IR(3) Bit.

5.3.5.1.3 Test Sign Bit (JT7)

The sign bit of the contents of the T Register may be dispatched on by specifying #JMPCNTL=010. This is indicated by

JT7(oneaddr,zeroaddr)

in a MICASM statement. The oneaddr field should be the 8-bit address of the microinstruction to be executed if T(7) is 1, and the zeroaddr field is the address of the microinstruction to be executed if T(7) is 0. This is shown in Figure 5-36.

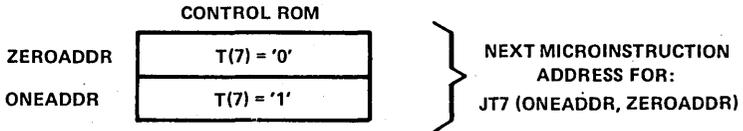


FIGURE 5-36 – JT7 DISPATCH

Typically, zeroaddr and oneaddr are MICASM labels initialized by an .EQU statement. It is required that zeroaddr be even and that oneaddr = zeroaddr + 1.

5.3.5.1.4 Test If Zero (JUZ)

The microcode may test the value on the O Bus of the immediately preceding microinstruction by specifying #JMPCNTL = 011. This is indicated by

JUZ(oneaddr,zeroaddr)

appearing in a MICASM statement. When JUZ appears in microinstruction *i*, it tests the O Bus contents of the previously executed microinstruction, *i*-1. The entry-point logic replaces JMPADDR(0) with the UEZ Bit from the Status Register, which is 1 when the O Bus is all zeroes (>00) and 0 otherwise. The symbol oneaddr denotes the address to which control is transferred if the O Bus was zero, i.e., if UEZ = 1. The symbol zeroaddr denotes the address jumped to if the O Bus was nonzero, i.e., if UEZ = 0. Like the JT7 MICASM symbol, zeroaddr must be even and oneaddr must equal zeroaddr + 1. The dispatch on the UEZ Bit is depicted in Figure 5-37.

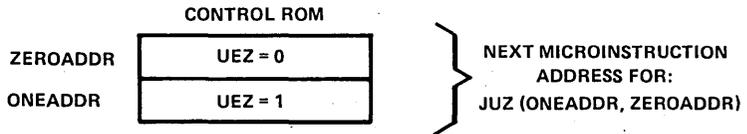


FIGURE 5-37 — JUZ DISPATCH

5.3.5.1.5 Test If Interrupt (INT)

The microcode may test for a pending interrupt by dispatching on the IACT (Interrupt Active) signal input from the Peripheral/Memory Controller. This is accomplished by specifying #JMPCNTL = 100, or in a MICASM statement by:

INT(oneaddr,zeroaddr)

As with the JT7 and JUZ instructions, oneaddr denotes the microinstruction address branch to if IACT = 1, and Zeroaddr is the address branched to if IACT = 0. Zeroaddr and oneaddr must be adjacent, as depicted in Figure 5-38.

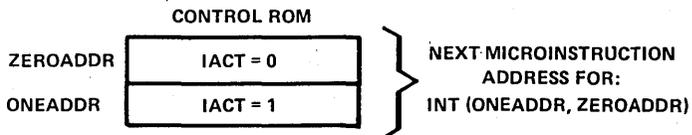


FIGURE 5-38 — INT DISPATCH

The IACT line is asserted by the Peripheral/Memory Controller (PMC) when an interrupt condition is detected. IACT can be asserted only when STINT (Status Interrupt Enable) is 1. Operation of the PMC in asserting interrupts is further explained in the TMS7000 8-Bit Microcomputer Data Manual (Part Number MP 008A).

5.3.5.1.6 Group Dispatch (IRH)

Dispatching on the Group field of the IR Register is accomplished by specifying 101 in the #JMP_CNTL field. This is indicated by coding

IRH(baseaddr)

in a MICASM statement. The baseaddr field is loaded into the #JMPADDR field of the microinstruction being defined.

There are 24 groups defined, 8 in Format 0 (IR(7)=0) and 16 in Format 1 (IR(7)=1). The groups are numbered in Figure 5-39.

FORMAT 0		FORMAT 1	
IR	GROUP NUMBER	IR	GROUP NUMBER
0000XXXX	0	10000XXX	8L
0001XXXX	1	10001XXX	8H
0010XXXX	2	10010XXX	9L
0011XXXX	3	10011XXX	9H
0100XXXX	4	10100XXX	AL
0101XXXX	5	10101XXX	AH
0110XXXX	6	10110XXX	BL
0111XXXX	7	10111XXX	BH
		11000XXX	CL
		11001XXX	CH
		11010XXX	DL
		11011XXX	DH
		11100XXX	EL
		11101XXX	EH
		11110XXX	FL
		11111XXX	FH

FIGURE 5-39 — TMS7000 GROUP NUMBERS

The IRH(baseaddr) symbol performs a 24-way dispatch on the Group field. This is done by replacing the low order bits of #JMPADDR with a function of the Group number. The high nibble of the IR Register, IR(7-4), is placed in the low nibble of the next address, shifted by 1 bit. The low order bit of the next address, is defined as NEXTADDRESS(0) = IR(3).OR.(.NOT.IR(7)). For Format 0 instructions, NOT IR(7) = 1, and NEXTADDRESS(0) always equals 1. Thus, the machine will jump to microaddress baseaddr + (group * 2) + 1 for Format 0 group numbers. For Format 1 instructions, NOT IR(7) = 0, and NEXTADDRESS(0) equals IR(3). Thus, the machine will jump to microaddress baseaddr + (group * 2) + IR(3) for Format 1 group numbers. The group names given in Figure 5-39 are the first hex digit in the two-digit hex representation of the IR Register contents. Format 1 names have an L if IR(3) = 0 and H if IR(3) = 1. The operation of the Group decode is shown in Figure 5-40.

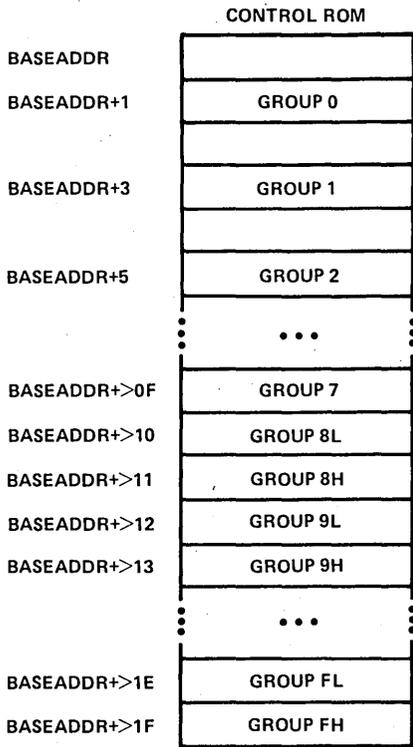


FIGURE 5-40 – IRH DISPATCH

The CROM addresses `baseaddr`, `baseaddr + 2`, `baseaddr + 4`, etc., may be used for other microinstructions. The microcode for the TMS7000 Standard Instruction Set uses the IRH dispatch immediately after the assembly language instruction is loaded into the IR. Each group corresponds to an addressing mode for the instruction, and the microcode executed after the dispatch fetches the appropriate operands. Typically, a Function, or IRL, dispatch is then performed, and the microcode branches to perform the appropriate ALU function on the operands. In this manner, the operand fetch microinstructions are shared among the assembly language instructions and each instruction has its own microcode to perform the function of that instruction.

5.3.5.1.7 Test If Carry (JC)

The microcode may test the value of the carry bit in the Status Register by performing a dispatch on the STC Bit. This is indicated by specifying `#JMPCNTL(2-0) = 110`, or

`JC(oneaddr,zeroaddr)`

appearing in a MICASM statement. The bit tested is the value of the STC (Status Carry) Bit after the execution of the immediately preceding microinstruction, i.e., the microinstruction executed prior to the one containing the `JC(oneaddr,zeroaddr)` statement. The STC Bit is placed in bit 0 of `#JMPADDR`, and the result used as the next microinstruction address.

If the STC Bit is 1, control transfers to oneaddr, and if STC=0, control transfers to zeroaddr. The locations zeroaddr and oneaddr must be adjacent, with zeroaddr on an even address and oneaddr on the subsequent odd address. This is diagrammed in Figure 5-41.

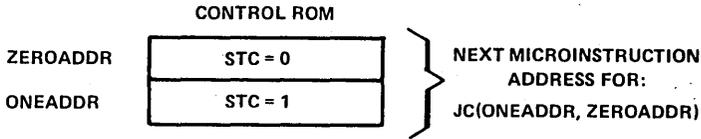


FIGURE 5-41 – JC DISPATCH

5.3.5.1.8 Test Status Register (MJMP)

The contents of the status register may be tested with the Macro Jump dispatch by specifying #JMPCNTL(2-0) = 111. This is indicated by

MJMP(oneaddr,zeroaddr)

appearing in the MICASM statement for a microinstruction. The MJMP dispatch tests eight possible conditions of the Status Register, indicated by the 3 bits in IR(2-0). If the condition is true, control transfers to oneaddr. If the condition is not true, control transfers to zeroaddr. The conditions tested are indicated in Figure 5-42.

IR(2-0)	CONDITION TESTED			COMMENT
	STC	STSB	STEZ	
0 0 0	X	X	X	Unconditionally Jump
0 0 1	X	1	X	Jump if Negative
0 1 0	X	X	1	Jump if Zero
0 1 1	1	X	X	Jump if Carry
1 0 0	X	0	0	Jump if Positive
1 0 1	X	0	X	Jump if Positive or Zero
1 1 0	X	X	0	Jump if Not Zero
1 1 1	0	X	X	Jump if No Carry

FIGURE 5-42 – MACRO JUMP CONDITIONS

The Xs in the Condition Tested column indicate don't care conditions.

The result of the condition test is placed in Bit 0 of #JMPADDR to form the new microinstruction address. The address oneaddr must be the odd address immediately following zeroaddr, as shown in Figure 5-43.

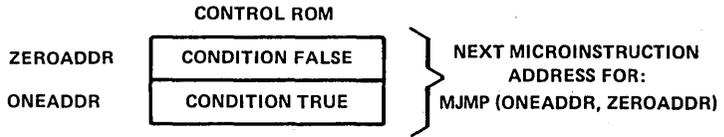


FIGURE 5-43 – MJMP DISPATCH

The MJMP dispatch is used in the microcode of the TMS7000 Standard Instruction Set to implement the conditional branch instruction.

5.3.6 Reset Operation

When the **RESET** pin is asserted externally, the PMC asserts the RST signal on the C Bus between the PMC and CPU. The entry-point logic immediately forces the next microinstruction address to be >FF. Unlike the normal interrupt facility, the microcode does not poll the RST line; rather, the microinstruction at CROM address >FF is unconditionally forced to be the next microinstruction executed.

In the TMS7000 Standard Instruction Set, the sequence of microinstructions executed upon reset fetch a subroutine entry point address at address >FFFE in memory (in the on-chip ROM) and branch to the subroutine.

6. DESIGN AIDS

6.1 INTERFACING THE TMS7000 TO PERIPHERAL AND MEMORY DEVICES

6.1.1 Introduction

All TMS7000 family devices feature 32 pins which can be used for general purpose I/O. However, several of these pins may be reconfigured to form an off-chip memory expansion bus. This reconfiguring allows the microcomputer to reference up to 64K bytes of ROM, RAM, or other peripheral devices. Two sample designs are presented which interface external peripheral and memory devices to the TMS7000.

All TMS70XX* devices are software compatible and differ only in special hardware features such as on-chip ROM size, extra timers, serial ports, etc. The timing data of the devices used in the two sample circuits are listed in Table 6-1. The timing information is taken from the data manual of that particular device. The timing data specified for the TMS70XX assumes a /4 clock option and a 10 MHz input clock frequency. Timing data for a 9 MHz clock was interpolated by multiplying the values specified in the data manual by 10/9. Refer to the timing diagram in Figure 6-1.

TABLE 6-1 — TIMING DATA FOR SAMPLE CIRCUITS

TMS70XX(U3)
TIMING DATA (+4 OPTION)

PARAMETER	TEST CONDITIONS	MIN	MAX	UNIT
t _d (A-D) Access time, data in from valid address	f = 9 MHz	444	522	ns
	f = 10 MHz	400	470	
t _d (EL-D) Data-in after $\overline{\text{ENABLE}}$ falling	f = 9 MHz	172	211	ns
	f = 10 MHz	155	190	
t _d (EH-AF) $\overline{\text{ENABLE}}$ rising to next address drive	f = 9 MHz	67	94	ns
	f = 10 MHz	60	85	
t _h (EH-RW) R/W hold after $\overline{\text{ENABLE}}$ rise	f = 9 MHz	44	111	ns
	f = 10 MHz	40	100	
t _h (EH-D) Data-in hold after $\overline{\text{ENABLE}}$ rise	f = 9 MHz	0		ns
	f = 10 MHz	0		
t _h (EH-Q) Data-out hold after $\overline{\text{ENABLE}}$ rise	f = 9 MHz	72	89	ns
	f = 10 MHz	65	80	
t _d (Q-EH) Data-out valid before $\overline{\text{ENABLE}}$ rise	f = 9 MHz	255	322	ns
	f = 10 MHz	230	290	

* TMS70XX refers to all family devices except as noted.

TABLE 6-1 — TIMING DATA FOR SAMPLE CIRCUITS (CONTINUED)

TMS9918A(U5)					
TIMING REQUIREMENTS					
PARAMETER			NOM	UNIT	
$t_{su}(D-WH)$	Data setup time before CSW high		100	ns	
$t_h(WH-D)$	Data hold time after CSW high		30		
SWITCHING CHARACTERISTICS					
PARAMETER			TYP	MAX	UNIT
$t_a(CSR)$	Data access time from CSR low		100	150	ns
t_{PVX}	Data disable time after CSR high		65	100	
TMS2516-35(U11)					
SWITCHING CHARACTERISTICS					
PARAMETER			TYP	MAX	UNIT
$t_a(A)$	Access time from address		250	350	ns
$t_a(S)$	Access time from chip select			120	
$t_{dis}(S)$	Output disable time from chip select during read mode only			100	
TMS4016-25(U10)					
TIMING REQUIREMENTS					
PARAMETER			MIN	MAX	UNIT
$t_{su}(D)$	Data setup time		100		ns
$t_h(D)$	Data hold time		10		
SWITCHING CHARACTERISTICS					
PARAMETER			MIN	MAX	UNIT
$t_a(A)$	Access time from address			250	ns
$t_a(S)$	Access time from chip select low			120	
$t_{dis}(S)$	Output disable time after chip select high			80	
74LS00(U1), 74S32(U2), 74LS373(U4), 74LS245(U6), 7408(U7), 74LS04(U8), and 74S138(U9)					
SWITCHING CHARACTERISTICS					
PARAMETER			TYP	MAX	UNIT
t_{pd}	74LS00(U1) propagation delay time		10	15	ns
t_{pd}	74S32(U2) propagation delay time		4	7	ns
t_{pd}	74LS373(U4) propagation delay time		12	18	ns
t_{pd}	74LS245(U6) propagation delay time		8	12	ns
t_{PLZ}	74LS245(U6) output disable time from low level		15	25	ns
t_{pd}	7408(U7) propagation delay time		17.5	27	ns
t_{pd}	74LS04(U8) propagation delay time		10	15	ns
t_{PHL}	74S138(U9) propagation delay time, high-to-low level from enable to any output (2-levels of logic)		7	11	ns
t_{PLH}	74S138(U9) propagation delay time, low-to-high level from enable to any output (2-levels of logic)		5	8	ns

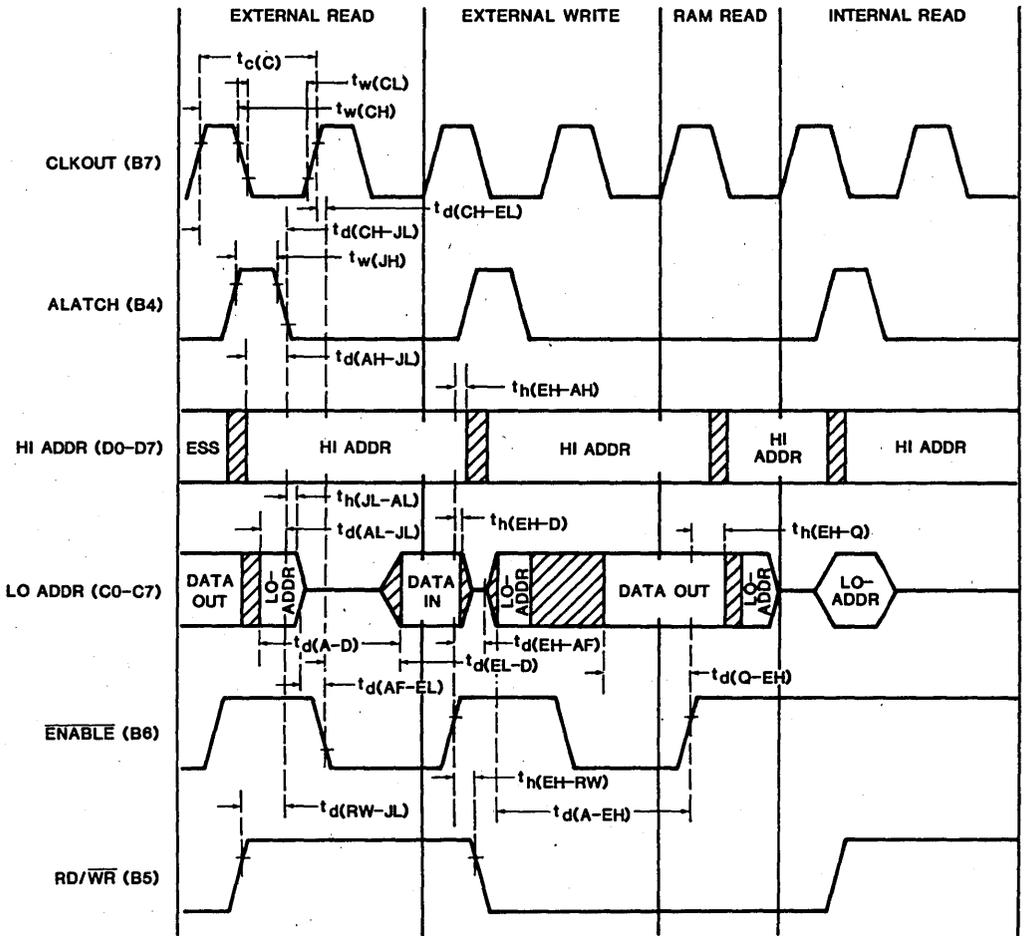


FIGURE 6-1 — TMS70XX READ AND WRITE CYCLE TIMING

6.1.2 Peripheral Expansion Mode Example

The schematic in Figure 6-2 is a TMS70XX — TMS9918A VDP (Video Display Processor) logic design using a minimum number of parts. The TMS70XX is configured for the Peripheral Expansion Mode, so only the C port and half of the B port are dedicated to the TMS9918A memory map interface. The C port becomes the multiplexed address/data bus and the upper nibble of the B port becomes the interface control bus. A 9 MHz crystal is used for the 70XX because the read access time of the TMS9918A is too long for a 70XX running at the full speed of 10 MHz (with divide by 4 clock option). The A port, D port, and the other half of the B port (lower nibble) of the 70XX remain available as I/O ports for other system functions. The A port is input only (I/O on the 7041), the D port is I/O, and the lower B port nibble is output only. U4 latches the 8-bit address from the address/data bus during read and write memory cycles. U6 is a bidirectional data buffer which is necessary for a fast disable time of read data on the address/data bus before the next processor read/write cycle. A very simple address decode is accomplished with U1 and U2.

There are 246 bytes of external memory mapped addressing possible with the TMS70XX in Peripheral Expansion Mode (238 bytes for the 7041). A complete address decoding scheme is not necessary because the TMS9918A is the only peripheral device depicted in this design. Eight address lines (A7 - A0) are available in the Peripheral Expansion Mode and three of these are needed for address decoding in this application. The MODE input pin of the TMS9918A is used to decode the two separate memory addresses it requires. A5 is used to enable write cycles to the TMS9918A and A6 is used to enable read cycles from the TMS9918A. Separate addresses are used for VDP read and write because of the read-before-write nature of many of the 70XX instructions (see paragraph 6.1.4, Software Considerations). The TMS9918A select starts at >0120 and >0140 and will not interfere with any of the dedicated or reserved peripheral file addresses of the 70XX. A0 is connected to the MODE input of the TMS9918A. The four 16-bit addresses are decoded as follows.

<u>A15</u>	<u>A8</u>	<u>A7</u>	<u>A0</u>		
0000	0001	X01X	XXX0		
0000	0001	X01X	XXX1	Write only addresses	(X = don't care)
0000	0001	X10X	XXX0		
0000	0001	X10X	XXX1	Read only addresses	

6.1.2.1 Read Cycle Timing For The Peripheral Expansion Mode

In a TMS70XX read cycle, the read data from the TMS9918A should be available as soon as 172 ns ($t_{d(EL-D)}$) after \overline{ENABLE} signal falls low. The TMS9918 will deliver data 150 ns maximum from \overline{CSR} low. The minimum access time calculated for this circuit is:

$$t_{d(EL-D)} = \text{Maximum delay time from } \overline{ENABLE} \text{ low to read data valid}$$

$$t_{d(EL-D)} = t_a(\overline{CSR}) + t_{dpU2} + t_{pdU6} = 150 + 7 + 12 = 169 \text{ ns.}$$

As mentioned earlier, U6 is a bidirectional data buffer which is necessary for a fast disable time of read data on the address/data bus before the next processor read/write cycle. The minimum $\overline{\text{ENABLE}}$ rise to the next address drive time of the TMS70XX running at 9 MHz ($t_{d(\text{EH-AF})}$) is 67 ns, so the design goal is to have a data disable time of less than or equal to 67 ns in the read cycle. The TMS9918's data disable time from $\overline{\text{CSW}}$ high (t_{pvx}) is at maximum 100 ns. U6 is used to solve this possible data bus conflict problem. The maximum data bus disable time is calculated next.

$$t_{d(\text{EH-AF})} = \text{Maximum time data bus is tristate after } \overline{\text{ENABLE}} \text{ high}$$

$$t_{d(\text{EH-AF})} = t_{\text{pdU2}} + t_{\text{pdU7}} + t_{\text{PLZU6}} = 7 + 27 + 25 = 59 \text{ ns}$$

It is necessary to ensure that the $\overline{\text{RW}}$ signal does not change state before any buffers driving the data bus are disabled. For example, if the U6 bidirectional buffer were enabled ($\overline{\text{G}}$ low) and the $\overline{\text{RW}}$ signal changed state (DIR low-high or high-low) then the previous buffer inputs would become buffer outputs and cause possible bus conflict in the system. The $\overline{\text{RW}}$ signal from the TMS70XX is held in a steady-state for at least 44 ns after $\overline{\text{ENABLE}}$ goes high ($t_{h(\text{EH-RW})}$). Consequently, the $\overline{\text{G}}$ signal to U6 must be high within 44 ns of $\overline{\text{ENABLE}}$ going high.

$$t_{h(\text{EH-G})} = \text{Maximum time } \overline{\text{G}} \text{ goes high after } \overline{\text{ENABLE}} \text{ rise}$$

$$t_{h(\text{EH-G})} = t_{\text{pdU2}} + t_{\text{pdU7}} = 7 + 27 = 34 \text{ ns}$$

6.1.2.2 Write Cycle Timing For The Peripheral Expansion Mode

In a Write Cycle the TMS9918A expects the write data from the TMS70XX to be valid for approximately 100 ns ($t_{\text{su(D-WH)}}$) before the $\overline{\text{CSW}}$ signal goes inactive (high). The circuit will easily meet this requirement as shown next.

$$t_{\text{su(D-WH)}} = \text{Minimum time data is valid before } \overline{\text{CSW}} \text{ high}$$

$$t_{\text{su(D-WH)}} = (t_{d(\text{Q-EH})} + t_{\text{pdU2}}) - t_{\text{pdU6}} = (255 + 4) - 12 = 247 \text{ ns}$$

The TMS9918A expects a data hold time of about 30 ns ($t_{h(\text{WH-D})}$) after $\overline{\text{CSW}}$ rises. The data hold time in this circuit is calculated as follows.

$$t_{h(\text{WH-D})} = \text{Minimum time data is valid after } \overline{\text{CSW}} \text{ rise}$$

$$t_{h(\text{WH-D})} = t_{\text{pdU7}} + t_{\text{PLZU6}} = 17.5 + 15 = 32.5 \text{ ns}$$

PERIPHERAL EXPANSION MODE EXAMPLE
TMS70XX TO TMS9918A/9928A/9929A)

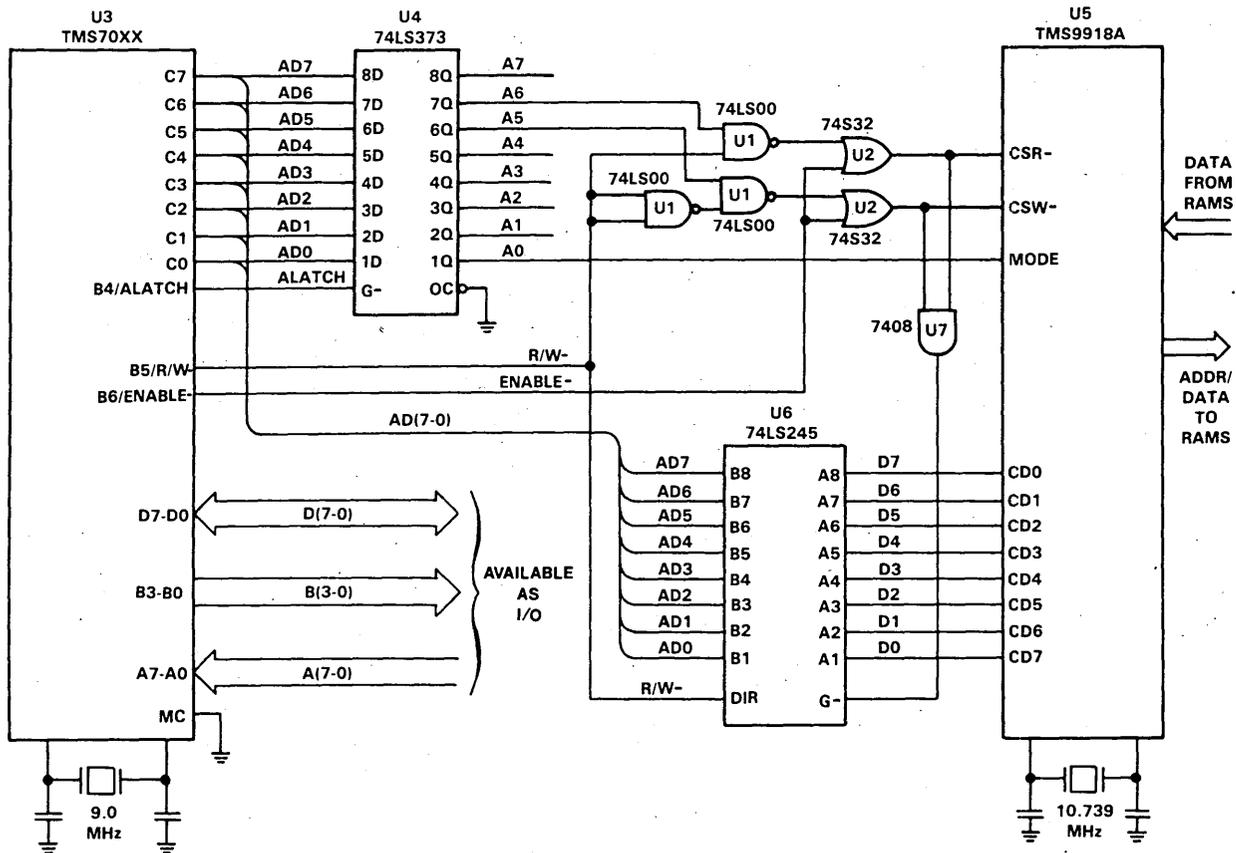


FIGURE 6-2 — PERIPHERAL EXPANSION MODE EXAMPLE

6.1.3 Microprocessor Mode Example

In the Full Expansion Mode and the Microprocessor Mode, all 16-bits of addressing is available on the C and D ports of the TMS70XX. The on-chip ROM (if any), RAM, and limited I/O of the 70XX can still be used in the Full Expansion Mode, but the ROM is disabled in the Microprocessor Mode and its address space is available externally.

The schematic in Figure 6-4 is an example of a memory interface to a 10 MHz TMS70XX operating in the Microprocessor Mode. The Mode Control (MC) pin is tied to VCC to place the 70XX in this mode. The D port becomes the most significant 8-bit address bus (A15 — A8). The C port becomes the multiplexed least significant 8-bit address bus (A7 — A0) and full 8-bit data bus, just as in the Peripheral Expansion Mode. The memory control signals are brought out on the upper nibble of the B port, just as in the Peripheral Expansion Mode. The A port remains an input only port (I/O port on the TMS7001/TMS7041) and the lower nibble of the B port remains an output only port.

The least significant 8-bits of the 16-bit address (A7 — A0) are latched into U4 by the ALATCH from the address/data bus during read/write memory cycles. U6 is a bidirectional data buffer which is necessary for a fast disable time of read data to the 70XX before the next read/write cycle. A full address decode is accomplished with U8 and U9. Eight memory select lines (SEL0 to SEL7) are generated by U9 and are each individually activated on an address block of 2048 bytes. Figure 6-3 lists the address range decoded by each select pin.

Pin	Address Range
<u>SEL0</u>	>C000 — >C7FF
<u>SEL1</u>	>C800 — >CFFF
<u>SEL2</u>	>D000 — >D7FF
<u>SEL3</u>	>D800 — >DFFF
<u>SEL4</u>	>E000 — >E7FF
<u>SEL5</u>	>E800 — >EFFF
<u>SEL6</u>	>F000 — >F7FF
<u>SEL7</u>	>F800 — >FFFF

FIGURE 6-3 — MEMORY ADDRESS DECODE

The example schematic in Figure 6-4 shows a TMS4016-25 static RAM selected by SEL0 and a TMS2516-35 EPROM selected by SEL7. Any combination of ROM, RAM or other peripheral device could be added into the circuit and enabled by the other SEL pins, provided that their timing requirements allow them to be interfaced to the TMS70XX.

6.1.3.1 Read Cycle Timing For The Microprocessor Mode

The minimum address to data access time required by the TMS70XX is 400 ns ($t_{d(A-D)}$). The following equation is used to check if U10 and U11 can deliver read data in less than or equal to 400 ns.

$$\begin{aligned}t_{d(A-D)} &= \text{Max read data valid time from address (A10 — A0)} \\t_{d(A-D)} &= t_a(A)U10 + t_{pd}U4 + t_{pd}U6 = 250 + 18 + 12 = 280 \text{ ns} \\t_{d(A-D)} &= t_a(A)U11 + t_{pd}U4 + t_{pd}U6 = 350 + 18 + 12 = 380 \text{ ns}\end{aligned}$$

The minimum $\overline{\text{ENABLE}}$ to data access time required by the TMS70XX is 155 ns ($t_{d(\text{EL-D})}$). Consequently, the chip select to data access of U10 and U11 must be less than or equal to 155 ns.

$$\begin{aligned} t_{d(\text{EL-D})} &= \text{Maximum delay time read data is valid from } \overline{\text{ENABLE}} \text{ low} \\ t_{d(\text{EL-D})} &= t_{a(\text{A})\text{U10}} + t_{\text{PHLU9}} + t_{\text{pdU6}} = 120 + 11 + 12 = 143 \text{ ns} \\ t_{d(\text{EL-D})} &= t_{a(\text{S})\text{U11}} + t_{\text{PHLU9}} + t_{\text{pdU6}} = 120 + 11 + 12 = 143 \text{ ns} \end{aligned}$$

The minimum $\overline{\text{ENABLE}}$ rise time to the next address drive time of the TMS70XX is 60 ns ($t_{d(\text{EH-AF})}$). The data bus is not to be driven by any external devices within this time: this is the main purpose of U6.

$$t_{d(\text{EH-AF})} = \text{Maximum time data bus is tristate after } \overline{\text{ENABLE}} \text{ high}$$

$$t_{d(\text{EH-AF})} = (2 \times t_{\text{pdU8}}) + t_{\text{PLZU6}} = (2 \times 15) + 25 = 55 \text{ ns}$$

As mentioned earlier, to avoid any possible bus conflict, the data direction of U6 must not be reversed by the $\overline{\text{RW}}$ signal while this device is enabled ($\overline{\text{G}}$ low). Therefore, $\overline{\text{G}}$ of U6 must be high within the time $\overline{\text{ENABLE}}$ goes high and $\overline{\text{RW}}$ changes state.

$$t_{h(\text{EH-G})} = \text{Maximum time } \overline{\text{G}} \text{ goes high after } \overline{\text{ENABLE}} \text{ rise}$$

$$t_{h(\text{EH-G})} = 2 \times t_{\text{pdU8}} = 2 \times 15 = 30 \text{ ns}$$

6.1.3.2 Write Cycle Timing For The Microprocessor Mode

The output data from the TMS70XX must be valid long enough before $\overline{\text{ENABLE}}$ rises to satisfy the TMS4016-25 RAM. The following equation derives the minimum time that write data will be valid to the memory devices while they are selected.

$$t_{\text{su}(\text{D})\text{U10}} = \text{Minimum time data is valid before } \overline{\text{S}} \text{ rise}$$

$$t_{\text{su}(\text{D})\text{U10}} = (t_{d(\text{Q-EH})} + t_{\text{PLHU9}}) - t_{\text{pdU6}} = (230 + 5) - 12 = 223 \text{ ns}$$

A $t_{\text{su}(\text{D})\text{U10}}$ of 223 ns easily exceeds the minimum data setup requirement of 100 ns for the TMS4016-25 RAM. The 4016 requires a minimum data hold time of 10 ns after $\overline{\text{S}}$ rises ($t_{h(\text{D})\text{U10}}$), so the value for $t_{h(\text{D-S})}$ must be greater than or equal to 10 ns. The purpose of the two inverters (U8) going to the $\overline{\text{G}}$ input of U6 is to ensure sufficient data hold time for the RAM.

$$t_{h(\text{D})\text{U10}} = \text{Minimum data hold time to U10 after } \overline{\text{S}} \text{ rise}$$

$$t_{h(\text{D})\text{U10}} = [(2 \times t_{\text{pdU8}}) + t_{\text{PLZU6}}] - t_{\text{PLHU9}} = [(2 \times 10) + 15] - 8$$

$$t_{h(\text{D})\text{U10}} = 27 \text{ ns}$$

MICROPROCESSOR MODE EXAMPLE
(TMS70XX TO TMS2516 EPROM AND TMS4016 RAM)

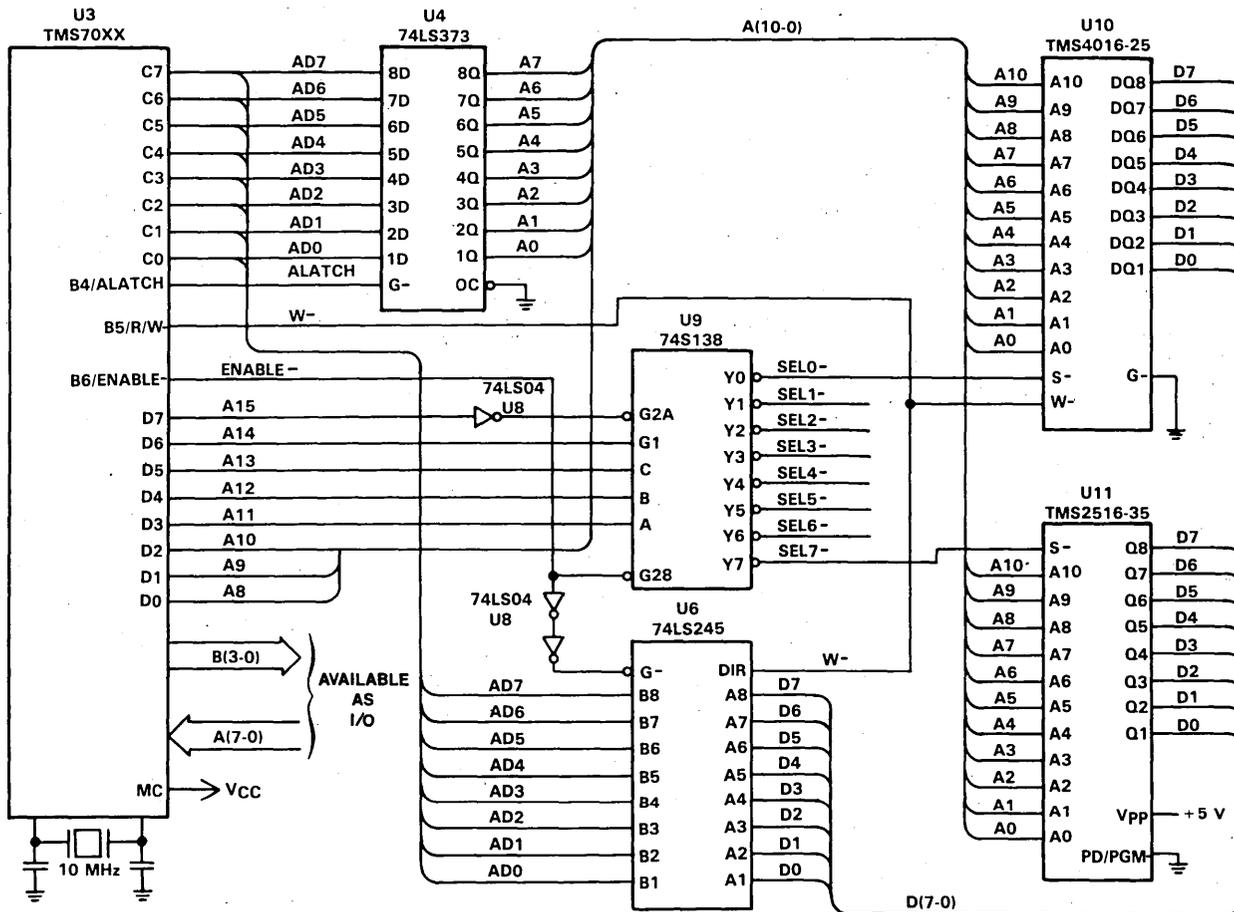


FIGURE 6-4 — MICROPROCESSOR MODE EXAMPLE

6.1.4 Software Considerations

The TMS70XX microcomputer features a variety of instructions which allow easy access to external memory mapped devices. The address space from >0100 to >01FF serves as the peripheral file. A special set of instructions are dedicated to the peripheral file for more efficient I/O communication to peripheral devices memory mapped in this space. The Peripheral Expansion Mode of the 70XX allows this space to be available externally. All instructions dedicated to the peripheral file use the letter 'P' at the end of the opcode mnemonic. These instructions are MOVP, BTJOP, BTJZP, ANDP, ORP, and XORP (see Section 3.3.3.2).

As indicated previously in the Peripheral Expansion Mode example, separate addresses are used for reads and writes. Due to processor design, many of the TMS70XX instructions perform a read-before-write cycle on the destination operand. This is true with the peripheral file instructions that would most likely be used to write to the TMS9918A:

```
MOVP A,Pn
MOVP B,Pn
MOVP %IOP,Pn
```

where:

```
A, B = accumulators
n    = peripheral file number
IOP  = immediate data value
```

These will read the peripheral file address before writing to it. If the \overline{CSW} and \overline{CSR} pins of the TMS9918A are decoded at the same address, a false read would occur when using these instructions. Therefore read and write addresses must be decoded separately. There is a method to allow the use of the same address for reading and writing in the TMS9918A example. This method is to use an instruction that does not read-before-write on the destination address.

```
STA @LABEL
STA @LABEL(B)
STA *Rn
```

where:

```
LABEL = 16-bit destination address
B     = index register
n     = register pair number
```

The instructions listed above will not perform an unnecessary read cycle on the destination address before writing to it. The TMS9918A address decode could be simplified by using just two address lines (A5,A0) instead of three (A6,A5,A0) when using these instructions.

A program can be executed from anywhere in the TMS70XX 64K byte address space where memory is available. This includes the 128 byte register file which is located at >0000 to >007F. Caution should be taken if a program is allowed to execute in the peripheral file address space because some of these locations are reserved for special on-chip functions. The Full Expansion Mode and Microprocessor Mode allow the use of additional external memory. The Microprocessor Mode example shows that RAM can be added externally as well as EPROM. A program can write to and read from this RAM by using the extended instructions LDA and STA. Direct, indirect, and indexed addressing modes are possible with the following instructions.

```
LDA    @LABEL
LDA    *Rn
LDA    @LABEL(B)
STA    @LABEL
STA    *Rn
STA    @LABEL(B)
```

where:

```
LABEL  = 16-bit source/destination address
n      = register pair number
B      = index register
```

The TMS70XX is a versatile single-chip microcomputer that can be reconfigured to address external peripheral and memory devices. This allows the TMS70XX to meet system requirements that could not be satisfied with single-chip mode.

6.2 SERIAL COMMUNICATION WITH THE TMS7000 FAMILY

This section is intended to assist the TMS7000 family user in performing serial communication via a UART (Universal Asynchronous Receiver Transmitter) function. It describes the implementation of the UART function in software using the TMS7040 and with the on-chip serial port using the TMS7041.

6.2.1 Communication Formats

Serial communications occur in one of two basic formats; synchronous or asynchronous. These formats are similar in that they both require framing bits to be added to the data to enable proper detection of the data at the receiving end.

In synchronous format, blocks of data are sent as a continuous string of characters where the string is preceded and terminated by framing bits; the preceding framing bits are used by the receiving device to synchronize its clock with the transmitter's clock.

In asynchronous format, as shown in Figure 6-5, each character to be transmitted is preceded by a START framing bit and followed by a parity bit (if parity is enabled), then one or more STOP framing bits.

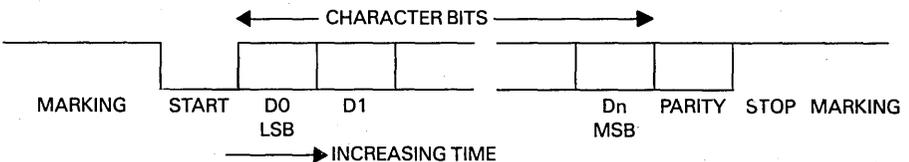


FIGURE 6-5 — ASYNCHRONOUS COMMUNICATION FORMAT

The START bit is a logical zero, or SPACE. It notifies the receiver to start assembling a character and allows the receiver to synchronize itself with the transmitter.

A PARITY bit is an additional bit added to a character for error checking. The PARITY bit is set to '0' or '1' in order to make the number of '1's in the character (including the PARITY bit) even or odd depending on whether even or odd parity is selected.

The STOP bit is a logical one or MARK. One or more STOP bit(s) are added to the end of the character to ensure that the START bit of the next character will cause a transition on the communication line and give the receiver time to catch up with the transmitter if its basic clock happens to be running slightly slower than that of the transmitter.

6.2.2 Design Constraints For The Software And Hardware UARTS

The purpose of this design is to implement the UART function using the TMS7000 family. There are two main routines to be written: the 'transmit' routine that transmits the character in the A Register and 'receive' routine that receives the character and stores it in the A register. The routines for the software UART will be called SWXMIT and SWRCVD; likewise, the routines for the hardware UART will be called HWXMIT and HWRCVD. Both the software and hardware UART implementations use the same I/O pins as shown in Figure 6-6.

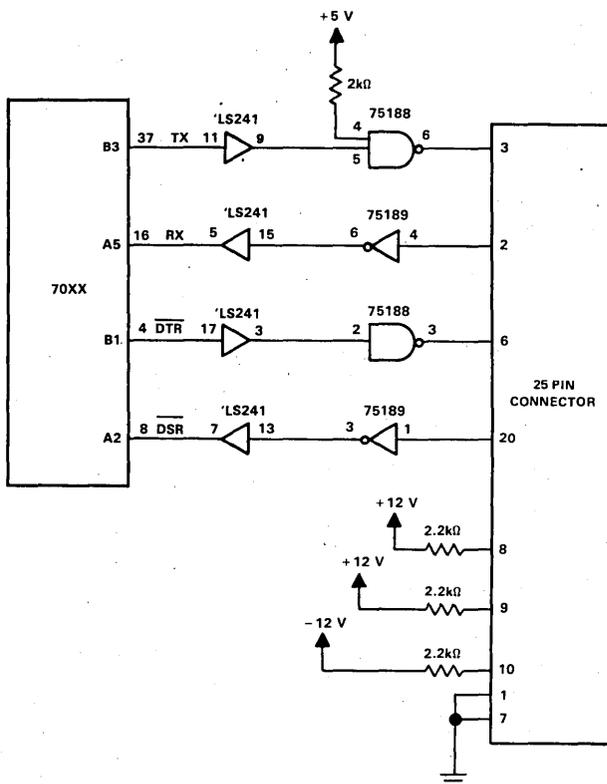


FIGURE 6-6 — I/O INTERFACE

Port A5 (pin 16) and Port B3 (pin 37) are used for receiving data and transmitting data. Port A2 (pin 8) and Port B1 (pin 4) are used for the inputting and outputting of the handshaking signals. Table 6-2 defines the pin assignments and the function of each pin.

TABLE 6-2 – I/O PIN ASSIGNMENT

SIGNATURE	PIN	I/O	FUNCTION
A2	8	I	Data Set Ready ($\overline{\text{DSR}}$)
A5	16	I	Receive Data (RXD)
B1	4	O	Data Terminal Ready ($\overline{\text{DTR}}$)
B3	37	O	Transmit Data (TXD)

The flowcharts together with the complete program listings for the XMIT and RCVD routines are included later in this section.

6.2.2.1 Design Of The Software UART For The TMS7040

Listed below is the register assignment for the software UART:

REGISTER	NAME	FUNCTION
R34 = BDCNT1	BIT COUNTER	STORE DELAY CONSTANT
R35 = BDCNT2	BIT COUNTER	STORE DELAY CONSTANT
R36 = HFBAUD	HALF BAUD RATE	STORE HALF BIT DELAY CONSTANT
R37 = MODE	MODE REGISTER	SET MODE OF OPERATION
R38 = BITCNT	COUNTER INITIALIZER	FOR # OF BITS TO BE XMITTED
R39 = BITIME	TIMER INITIALIZER	FOR DELAY
R40 = DLAYR1	DELAY LOOP1	USED IN DELAY LOOPING
R41 = DLAYR2	DELAY LOOP2	USED IN DELAY LOOPING
R42 = UATREG	UART REGISTER	TEMPORARY REGISTER
R43 = TMP	TEMPORARY REGISTER	TEMPORARY REGISTER
R44 = STAT	STATUS REGISTER	FOR CHECKING PARITY ERROR
R45 = RCHAR	RECEIVED CHARACTER	STORE THE RECEIVED CHARACTER
R46 = SHFCNT	SHIFT COUNTER	FOR BIT POSITION ADJUSTMENT

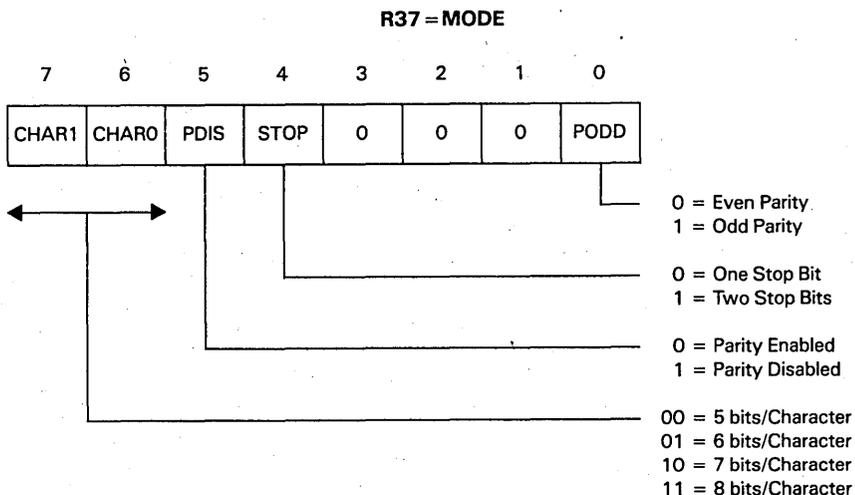
Each register has been assigned a name and its function is listed beside it.

- R34 and R35 provide the PIN constants for looping in the delay subroutine.
- R36 provides the delay constant for sampling the start bit at the half bit position.
- R37 controls the number of STOP bit(s), odd/even/no parity and the number of bits in the character.
- R38 controls the number of bits to be transmitted.
- R39 provides the delay constant for time compensation.
- R40 and R41 are used in the actual delay looping in the delay subroutine. They are loaded from R34 (BDCNT1) and R35 (BDCNT2) at the beginning of the delay subroutine.

- R42 contains a parity error flag at bit 0.
- R45 is used to store the received character.
- R46 is used to make the bit position adjustment so that the received data is right-justified.

Mode Register R37 = MODE

MODE is accessed through R37 in the register file. It describes the character format of the software UART.



Parity Odd (PODD) Bit 0 — If this bit is set to a 1, then odd parity is selected. The parity bit will be set to 0 or 1 in order to make the total number of 1's in the character (including the PARITY bit) odd.

Bit 1 to Bit 3 are reserved and must be set to 0's.

Number of Stop bits (STOP) Bit 4 — This bit determines the number of STOP bit(s) to be sent. Setting this bit to a 0 selects one STOP bit and setting it to a 1 selects two STOP bits.

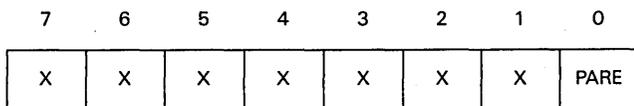
Parity Disable (PDIS) Bit 5 — If this bit is set to a 1, then no PARITY bit is generated during transmission or expected during reception.

Number of Bits per Character (CHAR0, 1) Bit 6, 7 — A character is programmable to 5, 6, 7 or 8 bits. Characters less than 8 bits are right-justified.

Status Register R44 = STAT

STAT is accessed through R44 of the register file. It is used for determining the parity error in the received character.

R44 = STAT



0 = No Parity Error
1 = Parity Error

Parity Error (PARE) Bit 0 - This bit indicates that a parity error is encountered on the received character if this bit is set to a 1 after a character is received.

SOFTWARE UART ROUTINE DESCRIPTION

The details of the routines for the software UART can be best understood by going through Figure 6-7, 6-8 and the program listings in this section.

In the SWXMIT routine, the character is contained in the A Register. This character is to be transmitted through the transmit line (TXD) according to the format specified in the MODE (R37) register.

The following is a portion of the SWXMIT routine listing:

		0008	XMIT1	EQU	>08	TRANSMIT '1'
						MASK (OR)
		00FD	RTS	EQU	>FD	READY TO SEND (AND)
		0004	DSR	EQU	>04	DATA SET
						READY (TEST)
		0004	UARTIN	EQU	P4	PORT A-UART
						INPUT (1)
		0006	UARTOT	EQU	P6	PORT B-UART
						OUTPUT (1)
0032	F006	C8	SWXMIT	PUSH	B	SAVE CONTENTS
						OF THE B REG.
0033	F007	A4		ORP	%XMIT1,UARTOT	PLACE A 'MARK'
						ON XMIT LINE
		F008	08			
		F009	06			
0034	F00A	A3	ANDP		%RTS,UARTOT	ASSERT 'RTS'
		F00B	FD			
		F00C	06			
0035	F00D	A6	WAIT	BTJOP	%DSR,UARTIN,WAIT	WAIT FOR
						HANDSHAKING
		F00E	04			
		F00F	04			
		F010	FC			

The SWXMIT routine listing starts by saving the B Register value on the stack so that the value can be restored after the execution of the routine.

Symbols refer to SWXMIT flowchart in Figure 6-7.

- A It places a 'MARK' or 1 on the transmit line (TXD) and then places a 0 on the output handshaking line ($\overline{\text{DTR}}$) informing the receiving end that it is Ready To Send the character. It waits for the input handshaking line ($\overline{\text{DSR}}$) to be pulled to a 0 by the receiving end. Refer to SWXMIT listing immediately above.
- B Once it receives a 0, it starts initializing the Bit Counter (R38) and the Timer Initializer (R39).
- C It jumps to 'LOOP2' to send out the START bit. After calling the delay subroutine, it jumps back to 'LOOP1' and starts sending the character bits. The total number of bits to be sent is determined by Bit Counter (R38).
- D After all the character bits have been transmitted, it tests the MODE (R37) register for parity. If parity is enabled, it will output the parity bit, otherwise; it jumps to the STOP bit and outputs the number of STOP bit(s) specified in bit 4 of MODE (R37).
- E After sending the STOP bit(s), it places a 1 on the output handshaking line ($\overline{\text{DTR}}$) and restores B-register.

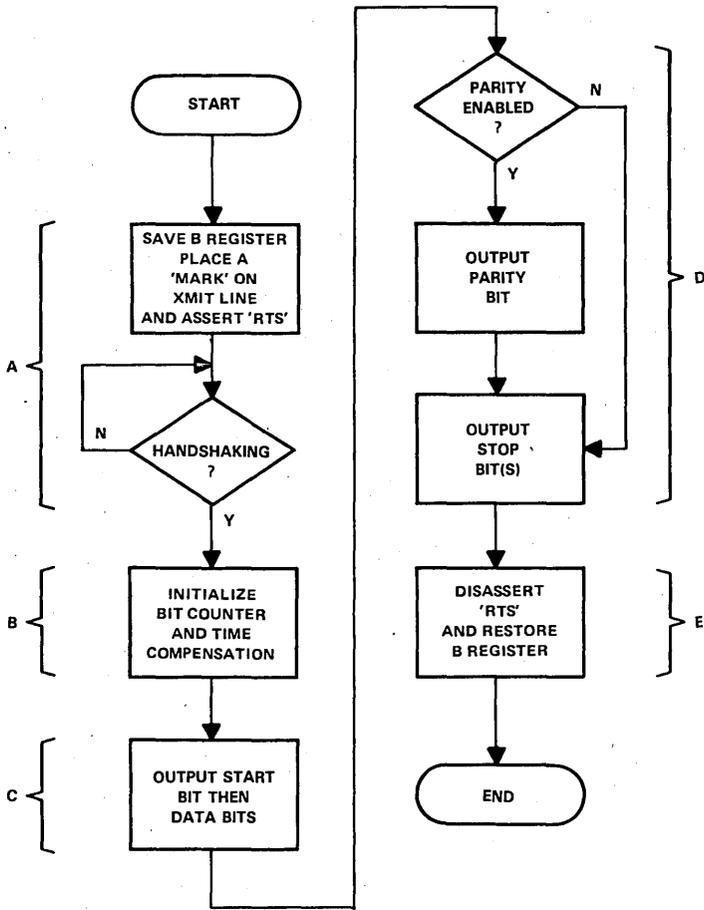


FIGURE 6-7 — SWXMIT ROUTINE FLOWCHART

In the SWRCVD routine, the character is received through the receive line A5(RXD) and stored in the A Register.

The following is a portion of the SWRCVD routine listing:

	00FD	RTS	EQU	>FD		READY TO SEND (AND)	
	0020	DIN	EQU	>20		DATA IN (TEST)	
	0004	UARTIN	EQU	P4		PORT A-UART	
						INPUT (1)	
	0006	UARTOT	EQU	P6		PORT B-UART	
						OUTPUT (1)	
0113	FOAE	A3		ANDP	%RTS,UARTOT	ASSERT 'DTR'	
	FOAF	FD					
	FOB0	06					
0114	FOB1	A7	MARKCK	BTJZP	%DIN,UARTIN,MARKCK	LOOP UNTIL	
						MARK OCCURS	
	FOB2	20					
	FOB3	04					
	FOB4	FC					
0115	FOB5	A6	STRBIT	BTJOP	%DIN,UARTIN,STRBIT	LOOP UNTIL	
						SPACE OCCURS	
	FOB6	20					
	FOB7	04					
	FOB8	FC					
0116	FOB9	32		MOV	HFBAUD,B	INITIALIZE	
						COUNTER	
	FOBA	24					
0117	FOBB	00	HERE2	NOP		TIME	
						COMPENSATION (4)	
0118	FOBC	00		NOP		(4)	
0119	FOBD	CA		DJNZ	B,HERE2	WAIT HALF A BIT (7 + 2)	
	FOBE	FC					
0120	*	SAMPLE START BIT AT HALF BIT					
0121	FOBF	A6		BTJOP	%DIN,UARTIN,STRBIT	BRANCH IF	
						FALSE START	
	FOC0	20					
	FOC1	04					
	FOC2	F2					

Symbols refer to SWRCVD flowchart in Figure 6-8:

- A It starts by saving B-register, initializing the Bit Counter (R38) and the Shift Counter (R46).
- B Then, it places a 0 on the output handshaking line (DTR-) informing the transmitting end that it is Ready To Receive the character. It checks the receive line (RXD) for 'MARK' or 1. After this condition is satisfied, it waits for the START bit to occur. Once the START bit is detected, it waits half a bit and samples again. Refer to the listing immediately above.
- C If the START bit is valid after half bit, it starts assembling the character bits after calling the delay subroutine for one bit delay. The received character is stored in RCHAR (R45).
- D It checks for a parity error and sets the STAT (R44) accordingly. The character received is also made right-justified.
- E Then, it places a 1 on the output handshaking line (DTR-) and moves the character from R45 to A-register. Finally it restores B-register.

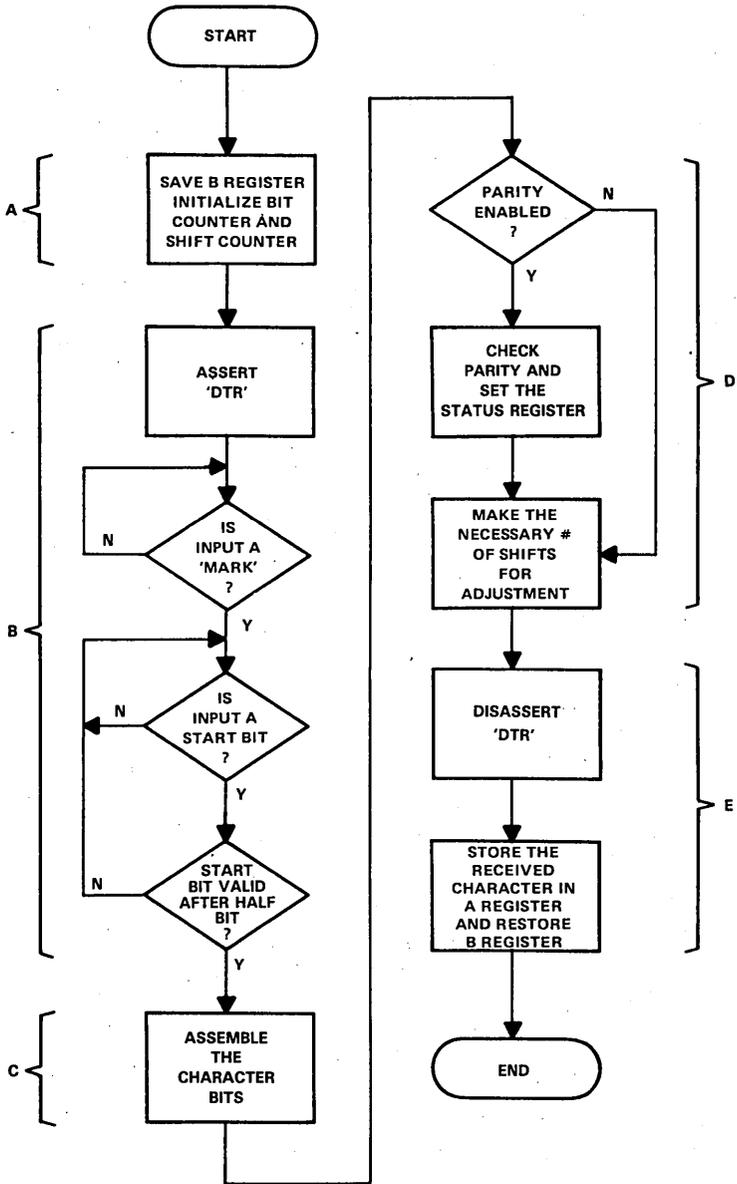


FIGURE 6-8 — SWRCVD ROUTINE FLOWCHART

CALCULATION OF THE DELAY CONSTANTS AND FORMULAS

Figure 6-9 describes how the delay works and how the bit time is calculated.

Let T = time per bit in micro seconds.

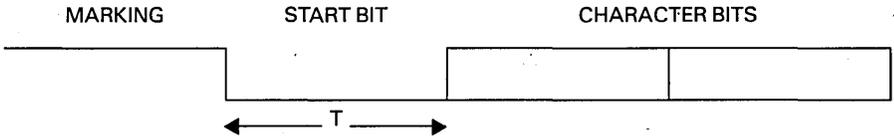


FIGURE 6-9 – DELAY CONSTANTS CALCULATION

For instance, if the microcomputer is operating at the crystal/clockin frequency $f_{osc} = 10$ MHz with the divide by 4 option ($f_{osc} = 5$ MHz with the divide by 2 option) as shown in Figure 6-9. The clockout cycle time $t_{c(c)} = 400$ n seconds. Table 6-3 shows the total number of cycles needed in the delay loop for the corresponding baud rates.

TABLE 6-3 – CYCLE CALCULATION

BAUD RATE	T IN MICRO SEC	# OF CYCLES NEEDED	TOTAL # OF CYCLES IN DELAY LOOP*
300	3333	8333	8221
600	1667	4167	4055
1200	8333	2083	1971
2400	417	1042	930
4800	208	521	409
9600	104	260	148

*NOTE: There are 112 cycles needed to manipulate the next bit to be sent out.

Refer to the delay subroutine in the SWUART program listing at the end of this section. The following is a sample of that program.

				# OF CYCLES PER INSTRUCTION
DELAY	MOV	BDCNT2,DLAYR2	INITIALIZE OUTER COUNT	10
ENTRY	MOV	BDCNT1,DLAYR1	INITIALIZE INNER COUNT	10
HERE1	DJNZ	DLAYR1,HERE1	INNER COUNT	9+2
	DJNZ	DLAYR2,ENTRY	OUTER COUNT	9+2
	RETS			7

Let A = Value in BDCNT1 and B = Value in BDCNT2 where A and B range from 1 to 255.

Therefore, total number of CYCLES in the delay subroutine

$$\begin{aligned}
 &= (11A + 21)B + 17 - 2(B + 1) \\
 &= 11AB + 19B + 15
 \end{aligned}$$

For example, if the total number of cycles in the delay subroutine = 4055, A = 35 and B = 10 are needed.

A simple program can optimize the value of A and B to provide the correct number of delay cycles.

Values of A and B with different crystal frequencies are provided in Table 6-5 at the end of this section.

Figure 6-10 describes how the start 'half bit' works and is calculated. Listed below is a sample of the start bit detection program found in the SWRCVD routine.

			# OF CYCLES PER INSTRUCTION	
STRBIT	BTJOP	%DIN,UARTIN,STRBIT	LOOP UNTIL START BIT OCCURS	12
	MOV	HFBAUD,B	INITIALIZE COUNTER	8
HERE2	NOP		TIME COMPENSATION	4
	NOP		TIME COMPENSATION	4
	DJNZ	B,HERE2	WAIT HALF A BIT	7 + 2
	BTJOP	%DIN,UARTIN,STRBIT	SAMPLE AGAIN, BRANCH IF FALSE START BIT	

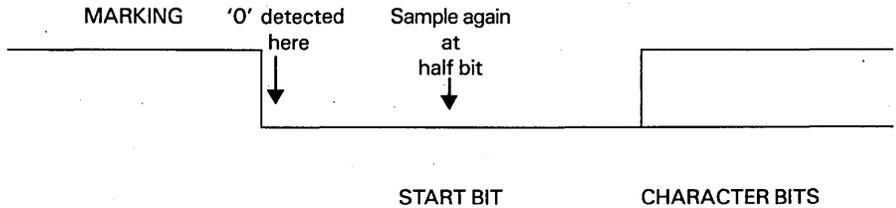


FIGURE 6-10 — START BIT DETECTION

Once the START bit is detected, the program will wait half a bit time and sample again as shown in Figure 6-10. This sequence provides false start bit rejection and also locates the center of bits in frame for assembling the character. Refer to Figure 6-8 SWRCVD flowchart - symbol B. Table 6-4 shows the number of cycles needed for a half bit delay.

TABLE 6-4 — HALF BIT CYCLES CALCULATION

BAUD RATE	# OF CYCLES FOR HALF BIT	# OF CYCLES NEEDED FOR HBAUD DELAY*
300	4167	4147
600	2083	2063
1200	1042	1022
2400	521	501
4800	260	240
9600	130	110

* NOTE: 20 Cycles are used to set up the half bit delay after the start bit is detected.

Let X = value in HFBAUD where X ranges from 1 to 255.

Therefore, $[4 + 4 + (7 + 2)]X - 2 = \# \text{ OF CYCLES NEEDED FOR HFBAUD DELAY}$

For example, if the number of cycles needed for a HBAUD delay is 4147, a value of X equal to 244 is needed.

X values with different crystal frequencies are provided in Table 6-5.

The crystal-dependent constants (BDCNT1, BDCNT2 and HFBAUD) used in the software UART are given in Table 6-5. These constants must be loaded into the corresponding registers and the MODE register must be set before SWXMIT or SWRCVD is called.

TABLE 6-5 – CRYSTAL-DEPENDENT CONSTANTS FOR THE SOFTWARE UART

BAUD RATE	BDCNT1(A)	BDCNT2(B)	HFBAUD(X)
300	18	1D	F4
600	23	0A	79
1200	06	17	3C
2400	1A	03	1E
4800	22	01	0E
9600	0A	01	07

CRYSTAL-DEPENDENT CONSTANTS GIVEN IN HEX:
10MHz CRYSTAL WITH DIVIDE BY 4 OSCILLATOR
or 5MHz CRYSTAL WITH DIVIDE BY 2 OSCILLATOR

BAUD RATE	BDCNT1(A)	BDCNT2(B)	HFBAUD(X)
300	1C	14	C3
600	90	02	61
1200	0B	0B	30
2400	09	06	17
4800	07	03	0B
9600	02	02	05

CRYSTAL-DEPENDENT CONSTANTS GIVEN IN HEX:
8MHz CRYSTAL WITH DIVIDE BY 4 OSCILLATOR
or 4MHz CRYSTAL WITH DIVIDE BY 2 OSCILLATOR

TABLE 6-5 — CRYSTAL-DEPENDENT CONSTANTS FOR THE SOFTWARE UART (CONTINUED)

BAUD RATE	BDCNT1(A)	BDCNT2(B)	HFBAUD(X)
300	23	0A	79
600	06	17	3C
1200	1A	03	1E
2400	22	01	0E
4800	0A	01	07
CRYSTAL-DEPENDENT CONSTANTS GIVEN IN HEX: 5MHz CRYSTAL WITH DIVIDE BY 4 OSCILLATOR or 2.5MHz CRYSTAL WITH DIVIDE BY 2 OSCILLATOR			

BAUD RATE	BDCNT1(A)	BDCNT2(B)	HFBAUD(X)
300	90	02	61
600	0B	0B	30
1200	09	06	17
2400	07	03	0B
4800	02	02	05
CRYSTAL-DEPENDENT CONSTANTS GIVEN IN HEX: 4MHz CRYSTAL WITH DIVIDE BY 4 OSCILLATOR or 2MHz CRYSTAL WITH DIVIDE BY 2 OSCILLATOR			

BAUD RATE	BDCNT1(A)	BDCNT2(B)	HFBAUD(X)
300	83	04	AE
600	80	02	57
1200	10	07	2B
2400	11	03	15
4800	02	06	0A
9600	01	02	04
CRYSTAL-DEPENDENT CONSTANTS GIVEN IN HEX: 3.579MHz CRYSTAL WITH DIVIDE BY 2 OSCILLATOR			

TABLE 6-5 – CRYSTAL-DEPENDENT CONSTANTS FOR THE SOFTWARE UART (CONTINUED)

BAUD RATE	BDCNT1(A)	BDCNT2(B)	HFBAUD(X)
300	80	02	57
600	10	07	2B
1200	11	03	15
2400	02	06	0A
4800	01	02	04
CRYSTAL-DEPENDENT CONSTANTS GIVEN IN HEX: 3.579MHz CRYSTAL WITH DIVIDE BY 4 OSCILLATOR			

BAUD RATE	BDCNT1(A)	BDCNT2(B)	HFBAUD(X)
300	1B	11	A1
600	02	40	50
1200	37	02	27
2400	0B	04	13
4800	12	01	09
9600	02	01	04
CRYSTAL-DEPENDENT CONSTANTS GIVEN IN HEX: 3.3MHz CRYSTAL WITH DIVIDE BY 2 OSCILLATOR			

BAUD RATE	BDCNT1(A)	BDCNT2(B)	HFBAUD(X)
300	0B	0B	30
600	09	06	17
1200	07	03	0B
2400	02	02	05
CRYSTAL-DEPENDENT CONSTANTS GIVEN IN HEX: 2MHz CRYSTAL WITH DIVIDE BY 4 OSCILLATOR or 1MHz CRYSTAL WITH DIVIDE BY 2 OSCILLATOR			

6.2.2.2 Hardware UART (TMS70X1)

The serial port consists of a receiver (RX), transmitter (TX), and TIMER3 (T3). The complete functional definition of the serial port is configured by the TMS7041 software. A set of control words must first be sent out to the serial port to initialize it, so that it will support the UART function.

The serial port is controlled and accessed through registers in the peripheral file. The registers associated with the serial port are shown in Table 6-6.

TABLE 6-6 – SERIAL PORT REGISTERS

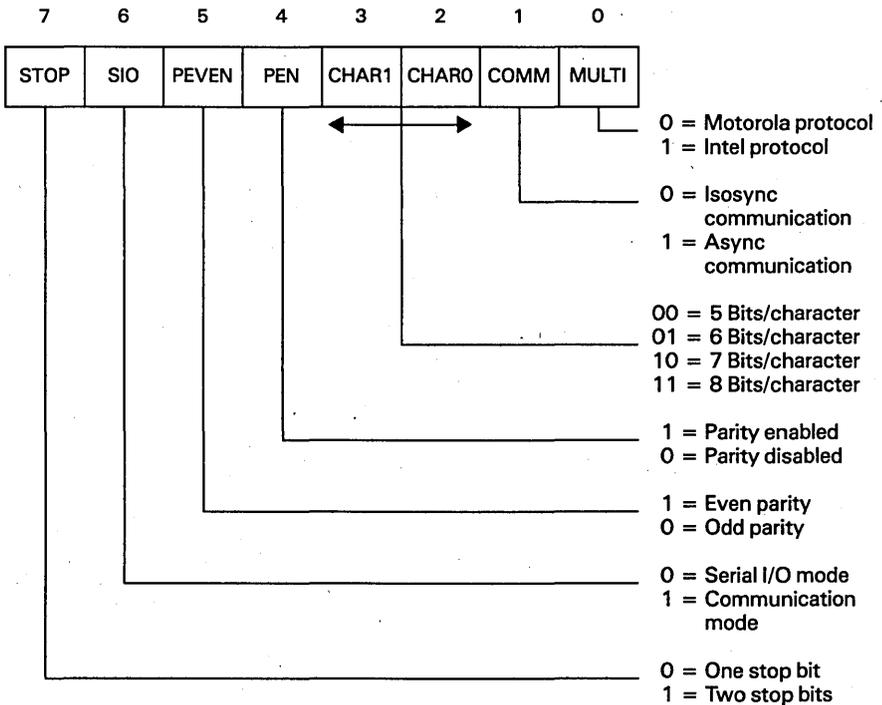
REGISTER	NAME	TYPE	FUNCTION
P17	SMODE	WRITE	Serial Port Mode
P17	SCTLO	WRITE	Serial Port Control-0
P17	SSTAT	READ	Serial Port Status
P20	T3DATA	R/W	Timer 3 Data
P21	SCTL1	R/W	Serial Port Control-1
P22	RXBUF	READ	Receiver Buffer
P23	TXBUF	WRITE	Transmission Buffer

The following diagrams are bit assignments of the peripheral file registers. They are included here for reference. It is suggested that the reader consult Section 2.7 for a complete description and explanation regarding their usage.

Mode Register (SMODE)

SMODE is accessed through P17 in the peripheral file. It is used to control the character format and type of communications mode (asynchronous or isosynchronous).

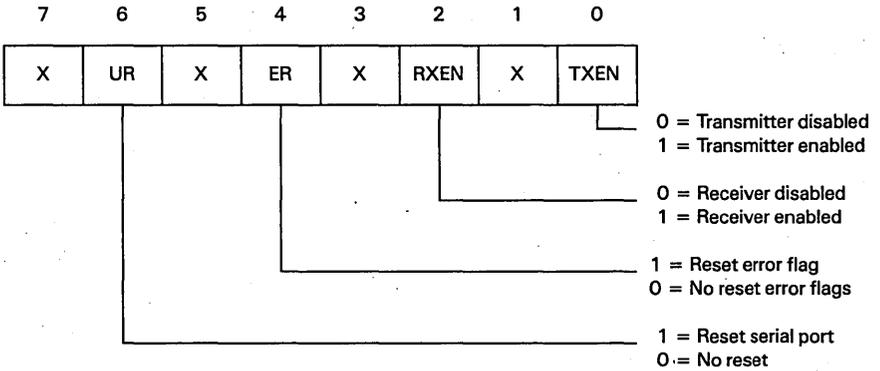
P17 = SMODE



Serial Control 0 Register (SCTLO)

SCTLO is accessed through P17 of the peripheral file. The SCTLO register is used to control the serial port functions, such as transmit and receive enable, clearing of error flags and software reset.

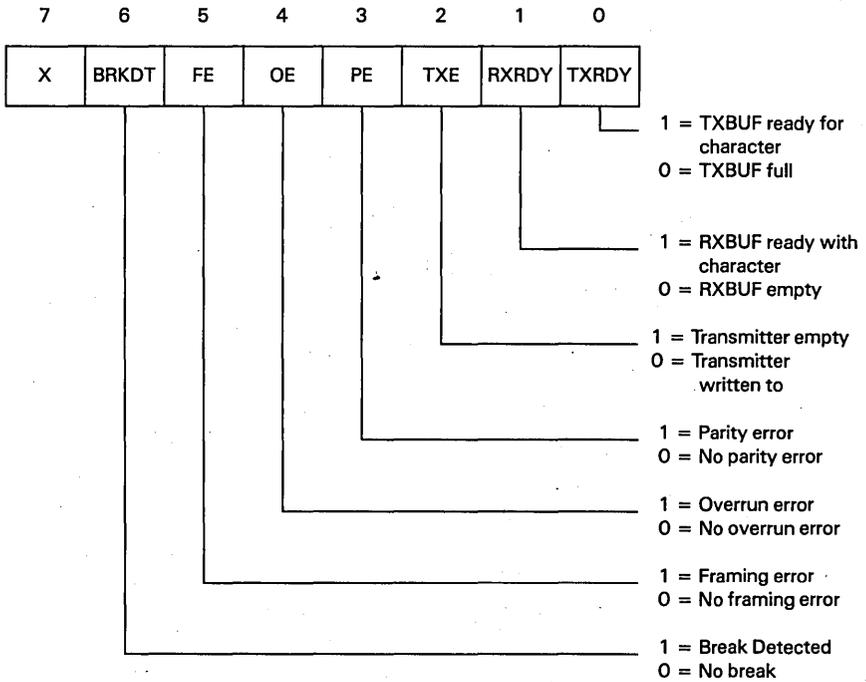
P17 = SCTLO



Status Register (SSTAT)

The Status is accessed through P17 of the Peripheral File. It is used for determining the status of the serial port.

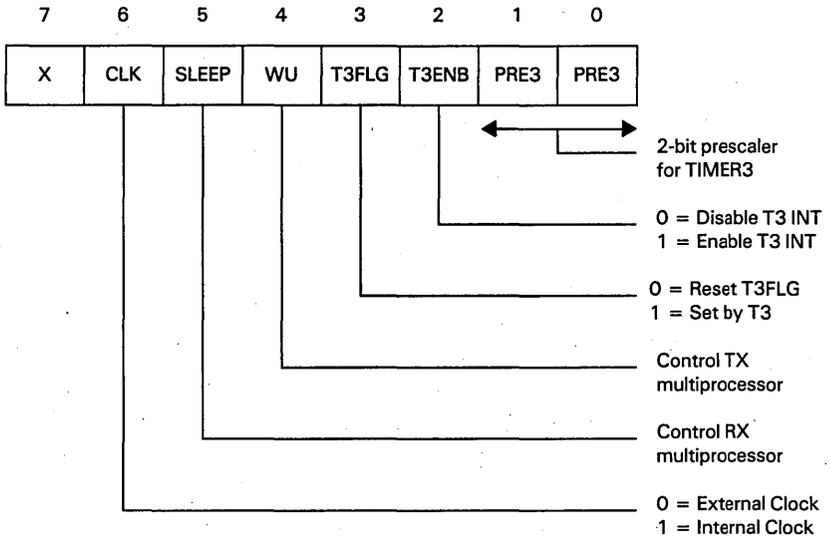
P17 = SSTAT



Serial Control 1 Register (SCTL1)

The SCTL1 is accessed through P21 in the peripheral file. This register is used to control the source of SCLK, multiprocessor communications, TIMER3 interrupt, and the TIMER3 prescaler value.

P17 = SCTL1



DESCRIPTION

SMODE is only accessible after a RESET operation (hardware or software). The first write operation to location P17 in the peripheral file, immediately following a RESET, will access the SMODE register. All subsequent writes to P17 will access the control register (SCTL0).

INT4 is dedicated to the serial port. Three sources can generate an interrupt through INT4: the transmitter (TX), the receiver (RX), and TIMER3 (T3). The serial port can be driven by an internal TIMER3 or external baud rate generator.

In this HWUART program, the T3 interrupt is disabled and the internal TIMER3 is chosen for the serial clock. The INT4 service routine as shown in Figure 6-11 must determine which flag caused the INT4 and take the necessary action. The INT4 vector is stored in memory addresses >FFF6 and >FFF7.

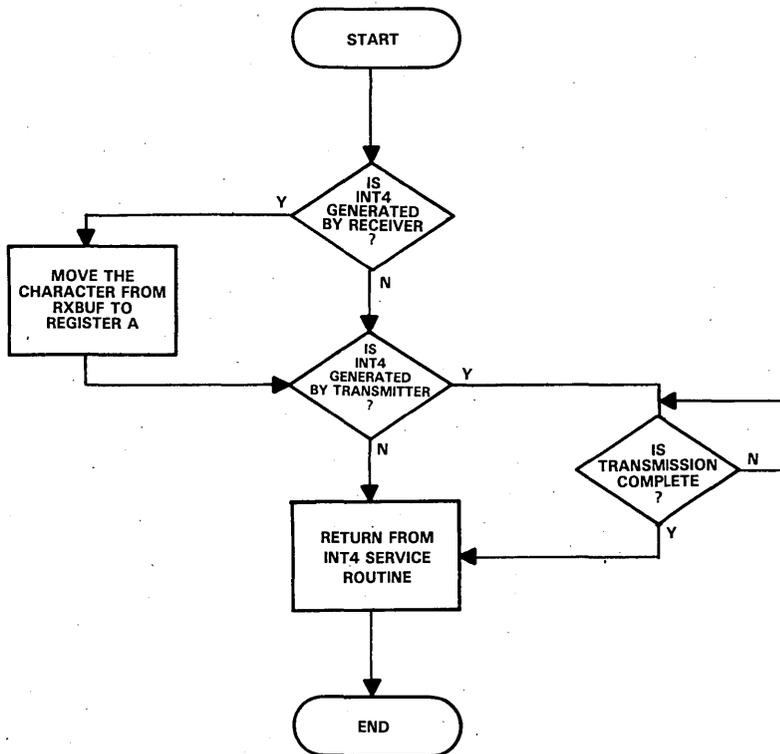


FIGURE 6-11 — INTERRUPT 4 SERVICE ROUTINE

In the HWXMIT routine (refer to Figure 6-12 HWXMIT flowchart and the listing at the end of the section) the peripheral file registers are set in the following orders:

- | | |
|-----------------|--------------------------------|
| 1) P5 = ADDR | Port A Direction Register |
| 2) P16 = IOCNT1 | I/O Control Register 1 |
| 3) P17 = SCTL0 | Serial Port Control Register 0 |
| 4) P21 = SCTL1 | Serial Port Control Register 1 |

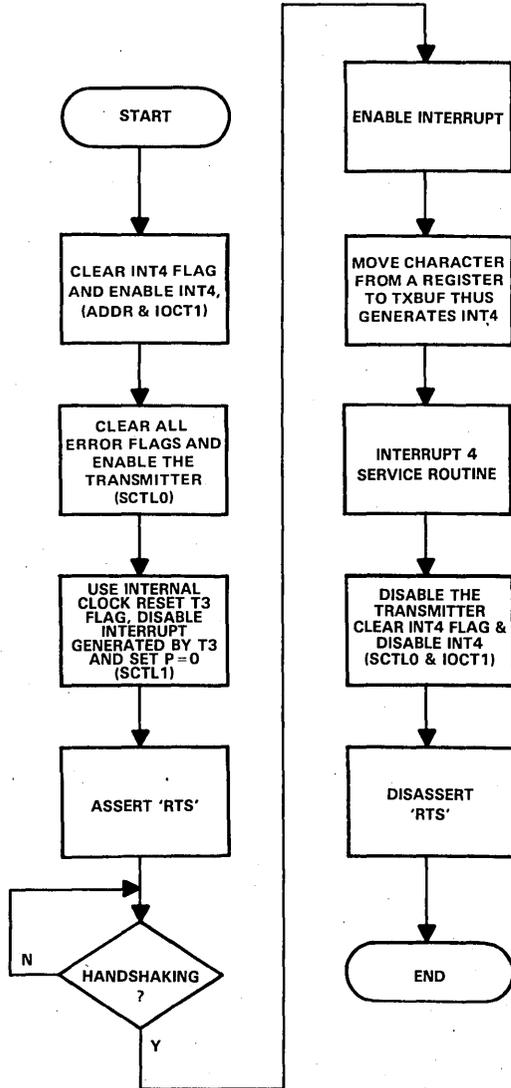


FIGURE 6-12 – HWXMIT ROUTINE FLOWCHART

It then places a '0' on the output handshaking line B1(DTR-) informing the receiving end that it is ready to send. After receiving the ready signal A2(DSR- = 0) from the receiving end, it enables the maskable interrupt, moves the character from the A Register to TXBUF thus generating an INT4. Upon returning from the INT4, it disables the transmitter and places a '1' on the output handshaking line B1(DTR-).

In the HWRCVD routine (refer to Figure 6-13 HWRCVD flowchart and the listing at the end of the section) the peripheral file registers are set in the following order:

- | | |
|-----------------|--------------------------------|
| 1) P5 = ADDR | Port A Direction Register |
| 2) P16 = IOCNT1 | I/O Control Register 1 |
| 3) P17 = SCTL0 | Serial Port Control Register 0 |
| 4) P21 = SCTL1 | Serial Port Control Register 1 |

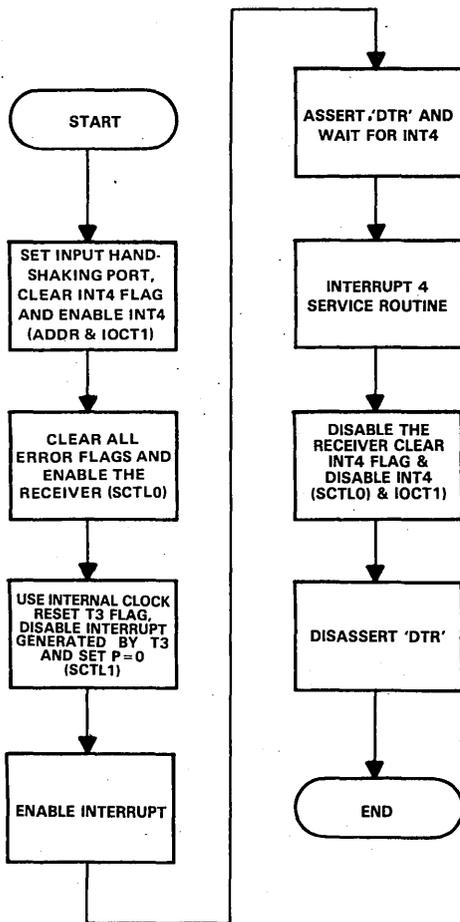


FIGURE 6-13 – HWRCVD ROUTINE FLOWCHART

In the transmit operation, the maskable interrupts are enabled and a '0' is placed on the output handshaking line B1(DTR-) informing the transmitting end that it is ready to receive the character. It waits for the INT4 generated by the Receiver to occur. Upon returning from the INT4, the routine clears the INT4 flag and disables INT4. Then, it sets the output handshaking line B1(DTR-) to a 1.

The baud rate generated by TIMER3 is user programmable and is determined by the value of the 2-bit prescaler and the 8-bit timer latch. The equation for determining the baud rates for asynchronous mode is as follows:

$$\text{ASYNCHRONOUS BAUD RATE} = \frac{\phi}{64 * (P + 1) * (L + 1)}$$

where:

- ϕ = Internal clock frequency
- P = TIMER3 prescaler value
- L = TIMER3 latch value (to be stored in T3DATA Register)

For instance, if the microcomputer is operating at the crystal/clockin frequency $f_{osc} = 10$ MHz with the divide by 4 option ($f_{osc} = 5$ MHz with the divide by 2 option), the internal clock frequency, ϕ , equals 2.5 MHz. The corresponding P and L values in hex are listed in Table 6-7.

TABLE 6-7 - P AND L VALUES IN HEX

BAUD RATE	P	L
300	0	81
600	0	40
1200	0	20
2400	0	0F
4800	0	07
9600	0	03
19200	0	01
38400	0	00

The SMODE register, the T3DATA register, and the INT4 vectors (in this case, memory addresses $>FFF6 = F0$, $>FFF7 = 42$) must be set before the HWXMIT or HWRCVD routine is called.

6.2.2.3 RS-232-C Interface

The RS-232-C interface consists of SN75188 line drivers and SN75189A line receivers as shown in Figure 6-6. The A port (input) of the TMS70XX (software and hardware UART) is used on all data and handshaking receptions. The B port (output) is used on all data and handshaking transmissions. As shown in Figure 6-6, the receive-data line goes to connector pin 2 and the transmit-data line to pin 3. The handshaking signal DSR (Data Set Ready) is received through pin 20 and DTR (Data Terminal Ready) is transmitted through pin 6. This configuration forms a port suitable for connection to an RS-232-C compatible terminal.

Before the data is transmitted, the TMS70XX will place +12V through the line driver SN75188 on connector pin 6 and wait until pin 20 rises above +4 V. After the handshaking signal is received, the data is then transmitted. If at any time the DSR is not asserted, it will wait in a loop until it is asserted.

CABLING EXAMPLES

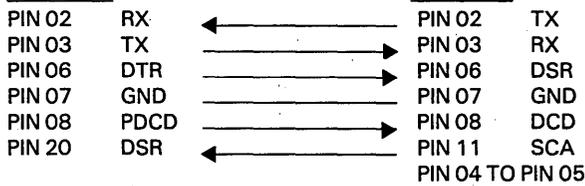
25 PIN CONNECTOR PORT

I/O

PIN 01	PROTECTIVE	GND
PIN 02	DATA RX	1
PIN 03	DATA TX	0
PIN 06	DTR(HANDSHAKE OUTPUT)	0
PIN 07	SIGNAL GND	
PIN 08	+12V	
PIN 09	+12V	
PIN 10	-12V	
PIN 20	DSR(HANDSHAKE INPUT)	1

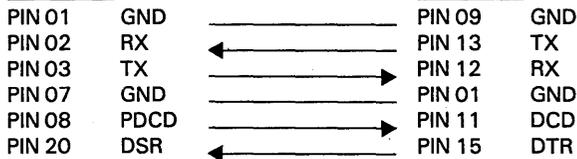
EIA PORT

820 KSR



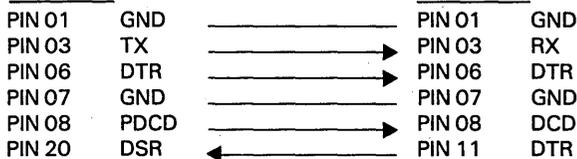
EIA PORT

743 KSR



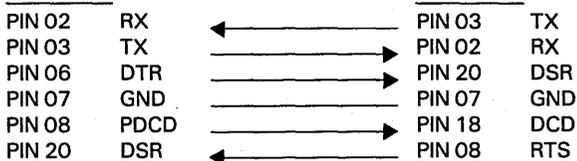
EIA PORT

810 LP



EIA PORT

990 CARD



6.2.2.4 *Other Design Approaches*

In the example given above, the microcomputer operates at the maximum internal clock rate of 2.5 MHz. For the TMS7000 family members with different timing requirements, the new crystal-dependent constants and the value of T3DATA can be determined by the given formulas. This allows both the software and hardware UARTs to operate at other baud rates.

In the software UART, TIMER1 may be used instead of a software delay loop. This can greatly increase the microcomputer's throughput.

```

0001          OPTION XREF
0002          IDT   'SWUART'
0003          *****
0004          0008  XMIT1  EQU   >08          TRANSMIT '1' MASK (OR)
0005          00F7  XMIT0  EQU   >F7          TRANSMIT '0' MASK (AND)
0006          0002  NRTS   EQU   >02          NOT READY TO SEND (OR)
0007          00FD  RTS    EQU   >FD          READY TO SEND (AND)
0008          0004  DSR    EQU   >04          DATA SET READY (TEST)
0009          0020  DIN    EQU   >20          DATA IN (TEST)
0010          *
0011          0022  BDCNT1 EQU   R34          BIT COUNTER (1)
0012          0023  BDCNT2 EQU   R35          BIT COUNTER (1)
0013          0024  HFBAUD EQU   R36          HALF BAUD RATE (1)
0014          0025  MODE   EQU   R37          MODE REGISTER (1)
0015          0026  BITCNT EQU   R38          COUNTER INITIALIZER(1)
0016          0027  BITIME EQU   R39          TIMER INITIALIZER (1)
0017          0028  DLAYR1 EQU   R40          DELAY LOOP1 (1)
0018          0029  DLAYR2 EQU   R41          DELAY LOOP2 (1)
0019          002A  UATREG EQU   R42          UART REGISTER (1)
0020          002B  TMP    EQU   R43          TEMPORARY (1)
0021          002C  STAT   EQU   R44          STATUS REGISTER (1)
0022          002D  RCHAR  EQU   R45          RECEIVED CHARACTER(1)
0023          002E  SHFCNT EQU   R46          SHIFT COUNTER (1)
0024          *
0025          0004  UARTIN  EQU   P4          PORT A-UART INPUT (1)
0026          0006  UARTOT  EQU   P6          PORT B-UART OUTPUT(1)
0027  F006          AORG   >F006
0028          *
0029          *=====
0030          * CHARACTER TO BE TRANSMITTED IS IN THE A REGISTER
0031          *=====
0032  F006  C8  SWXMIT  PUSH  B          SAVE CONTENTS OF THE B REG.
0033  F007  A4          ORP   %XMIT1,UARTOT  PLACE A 'MARK' ON XMIT LINE
          F008  08
          F009  06
0034  F00A  A3          ANDP  %RTS,UARTOT    ASSERT 'RTS'
          F00B  FD
          F00C  06
0035  FOOD  A6  WAIT   BTJOP %DSR,UARTIN,WAIT  WAIT FOR HANDSHAKING
          F00E  04
          FOOF  04
          F010  FC
0036          *
0037  F011  42          MOV   MODE,BITCNT    INITIALIZE
          F012  25
          F013  26
0038  F014  DE          RL   BITCNT         BIT
          F015  26
0039  F016  DE          RL   BITCNT         COUNTER
          F017  26
0040  F018  73          AND   %>03,BITCNT
          F019  03
          F01A  26
0041  F01B  78          ADD   %>07,BITCNT
          F01C  07
          F01D  26
0042          *

```

```

0043 F01E 72      MOV    %>05,BITIME      INITIAL TIME CONSTANT
      F01F 05
      F020 27
0044 F021 D5      CLR    TMP                SET UP START BIT
      F022 2B
0045 F023 42      MOV    MODE,UATREG
      F024 25
      F025 2A
0046 F026 E0      JMP    LOOP2
      F027 10
0047 F028 72     LOOP1  MOV    %>01,TMP      SET UP MASK FOR A REG    9
      F029 01
      F02A 2B
0048 F02B 43      AND    A,TMP             COPY LSB OF A REG TO TMP 10
      F02C 00
      F02D 2B
0049 F02E 48      ADD    TMP,UATREG       CALCULATE PARITY        10
      F02F 2B
      F030 2A
0050 F031 BC      RR     A                 SET UP NEXT BIT FOR XMIT 5
0051 F032 DE      RL    TMP               ADJUST FOR BIT LOCATION  7
      F033 2B
0052 F034 DE      RL    TMP               ADJUST FOR BIT LOCATION  7
      F035 2B
0053 F036 DE      RL    TMP               ADJUST FOR BIT LOCATION  7
      F037 2B
0054 F038 91     LOOP2  MOV    UARTOT,B      COPY P6 INTO B REG      8
      F039 06
0055 F03A 53      AND    %XMIT0,B         MASK OUT BIT OF B REG   7
      F03B F7
0056 F03C 34      OR     TMP,B            SET UP B REG FOR XMIT P6 8
      F03D 2B
0057 F03E 92      MOV    B,UARTOT        DATA BIT XMITTED TO EIA 9
      F03F 06
0058 F040 8E      CALL  @DELAY           DELAY TO PROPER BAUD RATE14
      F041 F084
0059 F043 DA      DJNZ  BITCNI,LOOP1     JUMP TO XMIT LOOP      9+2
      F044 26
      F045 E2
0060 F046 76      BTJO  %>20,MODE,STOPB  JUMP TO STOPB IF PARITY IS
      F047 20
      F048 25
      F049 18
0061          *
0062 F04A DA      SELF1 DJNZ  BITIME,SELF1  DISABLED
      F04B 27      TIME COMPENSATION
      F04C FD
0063 F04D 00      NOP
0064 F04E 77      BTJZ  %>01,UATREG,PARZ
      F04F 01
      F050 2A
      F051 05
0065 F052 A4      ORP   %XMIT1,UARTOT    OUTPUT PARITY BIT = ONE
      F053 08
      F054 06
0066 F055 E0      JMP   PDONE
      F056 05

```

0067	F057	A3	PARZ	ANDP	%XMIT0,UARTOT	OUTPUT PARITY BIT = ZERO
	F058	F7				
	F059	06				
0068	F05A	00		NOP		
0069	F05B	00		NOP		TIME COMPENSATION
0070	F05C	8E	PDONE	CALL	@DELAY	
	F05D	F084				
0071	F05F	00		NOP		
0072	F060	00		NOP		
0073	F061	00		NOP		TIME COMPENSATION
0074			*			
0075	F062	D7	STOPB	SWAP	UATREG	GET NUMBER
	F063	2A				
0076	F064	73		AND	%>01,UATREG	OF STOP BIT
	F065	01				
	F066	2A				
0077	F067	D3		INC	UATREG	INC BY 1 FOR COMPENSATION
	F068	2A				
0078	F069	72		MOV	%>03,BITIME	
	F06A	03				
	F06B	27				
0079	F06C	DA	SELF2	DJNZ	BITIME,SELF2	TIME COMPENSATION
	F06D	27				
	F06E	FD				
0080	F06F	A4		ORP	%XMIT1,UARTOT	OUTPUT STOP BIT
	F070	08				
	F071	06				
0081	F072	72	SECOND	MOV	%>07,BITIME	
	F073	07				
	F074	27				
0082	F075	DA	SELF3	DJNZ	BITIME,SELF3	TIME COMPENSATION
	F076	27				
	F077	FD				
0083	F078	00		NOP		
0084	F079	8E		CALL	@DELAY	
	F07A	F084				
0085	F07C	DA		DJNZ	UATREG,SECOND	JUMP FOR SECOND STOP BIT
	F07D	2A				
	F07E	F3				
0086	F07F	A4		ORP	%NRTS,UARTOT	DISASSERT 'RTS'
	F080	02				
	F081	06				
0087	F082	C9		POP	B	RESTORE B REGISTER
0088	F083	0A		RETS		
0089			*			
0090						
0091						
0092	F084	42	DELAY	MOV	BDCNT2,DLAYR2	INITIALIZE OUTER COUNT 10
	F085	23				
	F086	29				
0093	F087	42	ENTRY	MOV	BDCNT1,DLAYR1	INITIALIZE INNER COUNT 10
	F088	22				
	F089	28				
0094	F08A	DA	HERE1	DJNZ	DLAYR1,HERE1	INNER COUNT 9+2
	F08B	28				
	F08C	FD				
0095	F08D	DA		DJNZ	DLAYR2,ENTRY	OUTER COUNT 9+2

```

      F08E 29
      F08F F7
0096 F090 0A          RETS                                     7
0097
0098                *=====
0099                * SOFTWARE UART RECEIVE ROUTINE
0100 F091 C8 SWRCVD PUSH B          SAVE B REG
0101 F092 42          MOV  MODE,BITCNT      INITIALIZE
      F093 25
      F094 26
0102 F095 DE          RL   BITCNT          BIT
      F096 26
0103 F097 DE          RL   BITCNT          COUNTER
      F098 26
0104 F099 73          AND   %>03,BITCNT
      F09A 03
      F09B 26
0105 F09C 42          MOV  BITCNT,SHFCNT
      F09D 26
      F09E 2E
0106 F09F 74          OR   %>FC,SHFCNT
      FOA0 FC
      FOA1 2E
0107 FOA2 D4          INV  SHFCNT          GET NUMBER OF SHIFT
      FOA3 2E
0108 FOA4 78          ADD  %>05,BITCNT
      FOA5 05
      FOA6 26
0109                *
0110 FOA7 D5          CLR  RCHAR          CLEAR INCOMING CHAR REGISTER
      FOA8 2D
0111 FOA9 73          AND  %>FE,STAT      SET STATUS BIT-0 TO ZERO
      FOAA FE
      FOAB 2C
0112 FOAC D5          CLR  TMP          CLEAR TMP REG
      FOAD 2B
0113 FOAE A3          ANDP %RTS,UARTOT    ASSERT 'DTR'
      FOAF FD
      FOB0 06
0114 FOB1 A7          MARKCK BTJZP %DIN,UARTIN,MARKCK LOOP UNTIL MARK OCCURS
      FOB2 20
      FOB3 04
      FOB4 FC
0115 FOB5 A6          STRBIT BTJOP %DIN,UARTIN,STRBIT LOOP UNTIL SPACE OCCURS
      FOB6 20
      FOB7 04
      FOB8 FC
0116 FOB9 32          MOV  HFBAUD,B      INITIALIZE COUNTER
      FOBA 24
0117 FOBB 00          HERE2 NOP          TIME COMPENSATION(4)
0118 FOBC 00          NOP          (4)
0119 FOBD CA          DJNZ B,HERE2      WAIT HALF A BIT (7+2)
      FOBE FC
0120                * SAMPLE START BIT AT HALF BIT
0121 FOBF A6          BTJOP %DIN,UARTIN,STRBIT BRANCH IF FALSE START
      FOC0 20
      FOC1 04

```

```

FOC2 F2
0122 FOC3 72      MOV  %>01,UATREG
      FOC4 01
      FOC5 2A
0123 FOC6 43      AND  MODE,UATREG      MASK PARITY BIT
      FOC7 25
      FOC8 2A
0124 FOC9 00      NOP
0125 FOCA 00      NOP      TIME COMPENSATION
0126 FOCB 00      NOP
0127 FOCC 00      NOP
0128 FOCD 00      NOP
0129 FOCE 00      NOP
0130 FOCF 8E      SAMPLE CALL @DELAY      DELAY FOR PROPER BAUD RATE
      FOD0 F084
0131          *
0132 FOD2 52      MOV  %>04,B      TIME COMPENSATION
      FOD3 04
0133 FOD4 CA      RECHER DJNZ B,RECHER      DELAY CNT IS THE SAME
      FOD5 FE
0134          *
0135          * SAMPLE DATA BIT HERE      XMIT AND RCVD
0136 FOD6 A7      BTJZP %DIN,UARTIN,ZERO      READ DATA BIT JUMP IF 0
      FOD7 20
      FOD8 04
      FOD9 05
0137 FODA D3      INC  UATREG
      FODB 2A
0138 FODC 07      SETC
0139 FODD E0      JMP  BYPASS
      FODE 05
0140 FODF 00      ZERO  NOP      TIME COMPENSATION TO ENSURE
0141 FOE0 00      NOP
0142 FOE1 00      NOP      ZERO BIT IS THE SAME LENGTH
0143 FOE2 00      NOP
0144 FOE3 00      NOP      AS ONE BIT
0145 FOE4 DD      BYPASS RRC  RCHAR      PLACE BIT INTO MSB RCHAR
      FOE5 2D
0146 FOE6 00      NOP
0147 FOE7 D2      DEC  BITCNT
      FOE8 26
0148 FOE9 E6      JNZ  SAMPLE
      FOEA E4
0149 FOEB 8E      CALL @DELAY
      FOEC F084
0150          *
0151 FOEE 76      BTJO %>20,MODE,THERE      IS PARITY BIT ENABLE?
      FOEF 20
      FOF0 25
      FOF1 13
0152          *
0153 FOF2 52      MOV  %>03,B      IF NO, JUMP TO DONE
      FOF3 03      TIME COMPENSATION
0154 FOF4 CA      CYCLE DJNZ B,CYCLE      34 CLOCK CYCLES LOOP
      FOF5 FE
0155          *
0156          * SAMPLE PARITY BIT HERE

```

0157	FOF6	A7		BTJZP	%DIN,UARTIN,PZ	JUMP TO PZ IF PARITY BIT=0
	FOF7	20				
	FOF8	04				
	FOF9	02				
0158	FOFA	D3		INC	TMP	INCREMENT TMP IF PARITY
	FOFB	2B				
0159			*			BIT=1
0160	FOFC	45	PZ	XOR	TMP,UATREG	CHECK FOR PARITY ERROR
	FOFD	2B				
	FOFE	2A				
0161	FOFF	73		AND	%>01,UATREG	MASK, SAVE
	F100	01				
	F101	2A				
0162	F102	44		OR	UATREG,STAT	AND PUT THE RESULT IN
	F103	2A				
	F104	2C				
0163			*			STAT BIT 0
0164	F105	D5	THERE	CLR	TMP	
	F106	2B				
0165	F107	4D	SHIFT	CMP	SHFCNT,TMP	MAKE THE
	F108	2E				
	F109	2B				
0166	F10A	E2		JZ	DONE	NECESSARY
	F10B	06				
0167	F10C	B0		CLRC		NUMBER OF
0168	F10D	DD		RRC	RCHAR	SHIFT
	F10E	2D				
0169	F10F	DA		DJNZ	SHFCNT,SHIFT	
	F110	2E				
	F111	F5				
0170			*			
0171	F112	A4	DONE	ORP	%NRTS,UARTOT	DISASSERT 'DTR'
	F113	02				
	F114	06				
0172	F115	12		MOV	RCHAR,A	MOVE THE DATA BIT TO A
	F116	2D				
0173	F117	C9		POP	B	RESTORE B REG
0174	F118	0A		RETS		
0175				END		

NO ERRORS, NO WARNINGS

LABEL	VALUE	DEFN	REFERENCES	PAGE 0007																
BDCNT1	0022	0011	0093																	
BDCNT2	0023	0012	0092																	
BITCNT	0026	0015	0037	0038	0039	0040	0041	0059	0101	0102	0103									
			0104	0105	0108	0147														
BITIME	0027	0016	0043	0062	0078	0079	0081	0082												
BYPASS	F0E4	0145	0139																	
CYCLE	F0F4	0154	0154																	
DELAY	F084	0092	0058	0070	0084	0130	0149													
DIN	0020	0009	0114	0115	0121	0136	0157													
DLAYR1	0028	0017	0093	0094																
DLAYR2	0029	0018	0092	0095																
DONE	F112	0171	0166																	
DSR	0004	0008	0035																	
ENTRY	F087	0093	0095																	
HERE1	F08A	0094	0094																	
HERE2	F0BB	0117	0119																	
HFBAUD	0024	0013	0116																	
LOOP1	F028	0047	0059																	
LOOP2	F038	0054	0046																	
MARKCK	F0B1	0114	0114																	
MODE	0025	0014	0037	0045	0060	0101	0123	0151												
NRTS	0002	0006	0086	0171																
PARZ	F057	0067	0064																	
PDONE	F05C	0070	0066																	
PZ	F0FC	0160	0157																	
RCHAR	002D	0022	0110	0145	0168	0172														
RECHER	F0D4	0133	0133																	
RTS	00FD	0007	0034	0113																
SAMPLE	F0CF	0130	0148																	
SECOND	F072	0081	0085																	
SELF1	F04A	0062	0062																	
SELF2	F06C	0079	0079																	
SELF3	F075	0082	0082																	
SHFCNT	002E	0023	0105	0106	0107	0165	0169													
SHIFT	F107	0165	0169																	
STAT	002C	0021	0111	0162																
STOPB	F062	0075	0060																	
STRBIT	F0B5	0115	0115	0121																
SWRCVD	F091	0100																		
SWXMIT	F006	0032																		
THERE	F105	0164	0151																	
TMP	002B	0020	0044	0047	0048	0049	0051	0052	0053	0056	0112									
			0158	0160	0164	0165														
UARTIN	0004	0025	0035	0114	0115	0121	0136	0157												
UARTOT	0006	0026	0033	0034	0054	0057	0065	0067	0080	0086	0113									
			0171																	
UATREG	002A	0019	0045	0049	0064	0075	0076	0077	0085	0122	0123									
			0137	0160	0161	0162														
WAIT	F00D	0035	0035																	
XMITO	00F7	0005	0055	0067																
XMIT1	0008	0004	0033	0065	0080															
ZERO	F0DF	0140	0136																	

```

0001          OPTION XREF
0002          IDT   'HWUART'
0003          *****
0004      0008  XMIT1  EQU   >08          TRANSMIT '1' MASK (OR)
0005      00F7  XMIT0  EQU   >F7          TRANSMIT '0' MASK (AND)
0006      0002  NRTS   EQU   >02          NOT READY TO SEND (OR)
0007      00FD  RTS    EQU   >FD          READY TO SEND (AND)
0008      0004  DSR    EQU   >04          DATA SET READY (TEST)
0009          *
0010          *-----
0011          * P. REGISTER DEFINITION
0012          *-----
0013      0000  IOCNT0 EQU   P0          I/O CONTROL REGISTER 0
0014      0004  UARIN  EQU   P4          PORT A-UART INPUT
0015      0005  ADDR   EQU   P5          PORT A DIRECTION
0016      0006  UAROT  EQU   P6          PORT B-UART OUTPUT
0017      0010  IOCNT1 EQU   P16         I/O CONTROL REGISTER 1
0018      0011  SMODE  EQU   P17         SERIAL PORT MODE
0019      0011  SCTL0  EQU   P17         SERIAL PORT CONTROL-0
0020      0011  SSTAT  EQU   P17         SERIAL PORT CONTROL STATUS
0021      0014  T3DATA EQU   P20         TIMER 3 DATA
0022      0015  SCTL1  EQU   P21         SERIAL PORT CONTROL-1
0023      0016  RXBUF  EQU   P22         RECEIVER BUFFER
0024      0017  TXBUF  EQU   P23         TRANSMITTER BUFFER
0025          *
0026          * REGISTERS DEFINITION
0027          *-----
0028          *
0029          *-----
0030      F006          AORG   >F006
0031      F006      A3   HWXMIT ANDP  %>FB,ADDR          SET A2=INPUT
0032          F007      FB
0033          F008      05
0034      F009      A2          MOVP  %>03,IOCNT1          CLEAR INT4 FLAG & ENABLE INT4
0035          F00A      03
0036          F00B      10
0037      F00C      A2          MOVP  %>11,SCTL0          NO RESET OF SERIAL PORT
0038          F00D      11
0039          F00E      11
0040          *
0041          *
0042      F00F      A2          MOVP  %>40,SCTL1          CLEAR ALL ERROR FLAGS AND
0043          F010      40          ENABLE TXEN=1, RXEN=0 DISABLED
0044          F011      15          USE INTERNAL CLK,RESET T3FLAG
0045          *
0046          *
0047      F012      A3          ANDP  %RTS,UAROT          DISABLE T3 INTERRUPT& SET P=0
0048          F013      FD          ASSERT 'RTS'
0049          F014      06
0050      F015      A6   WAIT  BTJOP %DSR,UARIN,WAIT WAIT FOR HANDSHAKING
0051          F016      04
0052          F017      04
0053          F018      FC
0054      F019      05          EINT          ENABLE MASKABLE INTERRUPT
0055      F01A      82          MOVP  A,TXBUF
0056          F01B      17
0057      F01C      01          IDLE          WAITING FOR INT4
0058      F01D      A2          MOVP  %>02,IOCNT1          CLEAR INT4 FLAG & DISABLE INT4
    
```

```

    F01E 02
    F01F 10
0044 F020 A2      MOV  P  %>00,SCTL0      NO.RESET, DISABLE XMIT TXEN=0
    F021 00
    F022 11
0045          *
0046 F023 A4      ORP   P  %NRTS,UARTOT    RXEN=0
    F024 02      DISASSERT 'RTS'
    F025 06
0047 F026 0A      RETS
0048          *
0049          *
0050          *
0051          *
0052 F027 A3      HWRCVD ANDP  %>FB,ADDR      SET A2=INPUT
    F028 FB
    F029 05
0053 F02A A2      MOV  P  %>03,IOCNT1      CLEAR INT4 FLAG & ENABLE INT4
    F02B 03
    F02C 10
0054 F02D A2      MOV  P  %>14,SCTL0      NO RESET OF SERIAL PORT
    F02E 14
    F02F 11
0055          *
0056          *
0057 F030 A2      MOV  P  %>40,SCTL1      CLEAR ALL ERROR FLAGS & ENABLE
    F031 40      RECEIVER RXEN=1, TXEN=0 DISABLE
    F032 15      INTERNAL CLK, P=0
0058          *
0059 F033 05      EINT      RESET T3FLAG & DIASBLE T3 INT
0060 F034 A3      ANDP  P  %RTS,UARTOT    ENABLE MASKABLE INTERRUPT
    F035 FD      ASSERT 'DTR'
    F036 06
0061 F037 01      IDLE     WAITING FOR INT4
0062 F038 A2      MOV  P  %>00,SCTL0      DISABLE RCVER RXEN=0; TXEN=0
    F039 00
    F03A 11
0063 F03B A2      MOV  P  %>02,IOCNT1      CLEAR INT4 FLAG& DISABLE INT4
    F03C 02
    F03D 10
0064 F03E A4      ORP   P  %NRTS,UARTOT    DISASSERT 'DTR'
    F03F 02
    F040 06
0065 F041 0A      RETS
0066          *
0067          *
0068          * INTERRUPT 4 SERVICE ROUTINE
0069          *
0070 F042 A7      BTJZP %>02,SSTAT,TX
    F043 02
    F044 11
    F045 02
0071 F046 80      MOV  P  RXBUF,A          INT4 GENERATED BY HWRCVD
    F047 16
0072 F048 A7      TX      BTJZP %>01,SSTAT,FIN    JUMP TO FINISH
    F049 01
    F04A 11

```

```
      F04B  04
0073 F04C  A7  LOOP  BTJZP %>04,SSTAT,LOOP  INT4 GENERATED BY HWXMIT
      F04D  04
      F04E  11
      F04F  FC
0074 F050  0B  FIN   RETI      INTERRUPT VECTOR STORE
0075          *
0076          END
NO ERRORS, NO WARNINGS
```

LABEL	VALUE	DEFN	REFERENCES
ADDR	0005	0015	0031 0052
DSR	0004	0008	0039
FIN	F050	0074	0072
HWRCVD	F027	0052	
HWXMIT	F006	0031	
IOCNT0	0000	0013	
IOCNT1	0010	0017	0032 0043 0053 0063
LOOP	F04C	0073	0073
NRTS	0002	0006	0046 0064
RTS	00FD	0007	0038 0060
RXBUF	0016	0023	0071
SCTLO	0011	0019	0033 0044 0054 0062
SCTL1	0015	0022	0036 0057
SMODE	0011	0018	
SSTAT	0011	0020	0070 0072 0073
T3DATA	0014	0021	
TX	F048	0072	0070
TXBUF	0017	0024	0041
UARTIN	0004	0014	0039
UARTOT	0006	0016	0038 0046 0060 0064
WAIT	F015	0039	0039
XMIT0	00F7	0005	
XMIT1	0008	0004	

6.3 INSTRUCTION SET APPLICATION NOTES

This section provides supplemental information about the instruction set as an aid to program development. Refer to the TMS7000 ASSEMBLY LANGUAGE PROGRAMMER'S GUIDE (MP 916) for further application notes.

6.3.1 The Status Register

The Status Register has four status bits that provide conditional execution of a variety of arithmetic and logical tasks (see Figure 6-14). The Carry (C), Sign (N), Zero (Z), and Interrupt enable (I) occupy bits 7-4 of the Status Register. The global INTERRUPT ENABLE (I) bit is only affected by the EINT, DINT, and POP ST instructions. The C, N, and Z bits are affected by a number of instructions. Table 6-8 classifies the instruction set according to the status bits affected by each instruction.

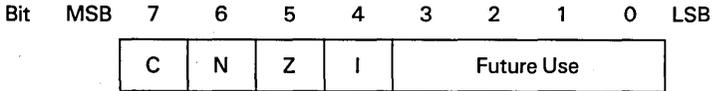


FIGURE 6-14 — STATUS REGISTER

Among the initialization-type instructions, two of the most useful are the compare instructions CMP and CMPA. Section 6.3.1.1 describes the way in which CMP and CMPA can be used to create the necessary status conditions for either a logical-type (unsigned) or arithmetic-type (signed) jump instruction. In Section 6.3.1.2, the effects of addition and subtraction on the Status Register are diagrammed for both signed and unsigned systems. Finally, Section 6.3.1.3 describes how SWAP and the rotation instructions RR, RRC, RL, and RLC can be used to clear, set, shift, or test the various status bits as required.

TABLE 6-8 — CLASSIFICATION OF INSTRUCTIONS
ACCORDING TO STATUS BITS AFFECTED

INSTRUCTION TYPES					
INIT STATUS REG.	CLEAR CARRY, SET N,Z ON A	CLEAR CARRY, SET N,Z ON B	CLEAR CARRY, SET N,Z ON RESULT	CONDITIONAL STATUS	NO STATUS AFFECTED
DINT EINT POP ST RETI SETC CLRC	LDA STA TSTA	TSTB XCHB	AND ANDP BTJO BTJOP BTJZ BTJZP CLR INV MOV MOVD MOVP OR ORP POP PUSH XOR XORP	ADC ADD CMP CMPA DAC DEC DECD DSB SBB SUB	BR CALL DJNZ IDLE J(CND) JMP NOP PUSH ST RETS STSP TRAP LDSP

6.3.1.1 Compare And Jump Instructions

The compare instructions CMP and CMPA, affect the C, N, and Z bits in the Status Register by subtracting a source operand (s) from a destination operand (d). The result of $(d) - (s)$ is not stored and is computed as follows:

$$(d)-(s) = (d) + (\bar{s}) + 1 = 8\text{-bit Result}$$

where (\bar{s}) is a direct one-for-one bitwise inversion, one's complement, of (s). The C bit serves as a "no borrow" bit and is set to '1' if (d) is greater than or equal to (s). The N bit is set to the same value as the MSB of the result. For two's complement (signed) systems, $N = 1$ indicates a negative number, and $N = 0$ a positive number. The Z bit is set to '1' if the source is equal to the destination ($(d) = (s)$). The CMP instruction uses the contents of a register (Rn) in the Register File as the destination operand, and either an immediate operand (IOP) or the contents of another Rn as the source operand. The CMPA instruction uses the contents of the A register as the destination operand and one of the Extended Addressing modes (Direct, Register File Indirect, or Indexed) is used to generate the source operand. Table 6-9 illustrates the limits of both signed and unsigned systems by listing the status bits affected for various source and destination operands substituted into the $(d) - (s)$ expression.

TABLE 6-9 — COMPARE INSTRUCTION EXAMPLES: STATUS BIT VALUES

SRC	DEST	D-S	C	N	Z	INSTRUCTIONS THAT WILL JUMP
FF	00	01	0	0	0	JL JNC JNE JNZ JP JPZ
00	FF	FF	1	1	0	JHS JC JNE JNZ JN
00	7F	7F	1	0	0	JHS JC JNE JNZ JP JPZ
81	00	7F	0	0	0	JL JNC JNE JNZ JP JPZ
00	81	81	1	1	0	JHS JC JNE JNZ JN
80	00	80	0	1	0	JL JNC JNE JNZ JN
00	80	80	1	1	0	JHS JC JNE JNZ JN
7F	80	01	1	0	0	JHS JC JNE JNZ JP JPZ
80	7F	FF	0	1	0	JL JNC JNE JNZ JN
7F	7F	00	1	0	1	JHS JC JEQ JZ JPZ
7F	00	81	0	1	0	JL JNC JNE JNZ JN

Since the compare instructions do not alter the source and destination operands, these instructions can be executed prior to a conditional jump instruction to test for a particular relationship between the source and destination operands. Table 6-10 lists the necessary status bit conditions for each of the conditional jump instructions and the type of system in which it is applicable, i.e., signed or unsigned.

TABLE 6-10 – STATUS BIT VALUES FOR CONDITIONAL JUMP INSTRUCTIONS

MNEMONIC	INSTRUCTION	CONDITION ON WHICH JUMP IS TAKEN	STATUS BIT VALUES FOR JUMP:			SIGNED	UN-SIGNED
			C	N	Z		
JC/JHS	Jump If Carry/Jump If Higher Or Same	(d)unsigned >= (s)	1	X	X	Y	Y
JNC/JL	Jump If No Carry/Jump If Lower	(d)unsigned <= (s)	0	X	X	Y	Y
JZ/JEQ	Jump If Zero/Jump If Equal	(d) = (s)	X	X	1	Y	Y
JNZ/JNE	Jump If Non-zero/Jump If Not Equal	(d) <> (s)	X	X	0	Y	Y
JP	Jump If Positive	(d)-(s) = pos #	X	0	0	Y	N
JN	Jump If Negative	(d)-(s) = neg #	X	1	X	Y	N
JPZ	Jump If Positive Or Zero	(d)-(s) = pos # or 0	X	0	X	Y	N

X = Don't care

Table searches are efficiently performed through the use of the compare A register extended (CMPA) instruction. In the following example, A 150 byte table is searched for a match with a 6 byte string:

```

*
SEARCH  MOV  %150+1,R2    Table length = 150 bytes
LOOP1   MOV  %6,B        String length = 6 bytes
LOOP2   XCHB R2         Swap pointers, long string in B
        DEC  B          Table end ? if so, no match found
        JZ   NOFIND
        LDA  @TABLE-1(B) Load test character
        XCHB R2         Swap pointers, string pointer in B
        CMPA @STRING-1(B) Match ?
        JNE  LOOP1      If not, reset string ptr. else test
        DJNZ B,LOOP2    next character.
MATCH   EQU  $          Match found
*
NOFIND  EQU  $          No match found
    
```

The indexed addressing mode is used in this example and has the capability to search a 256 byte string if needed. The B register alternates between a pointer into the 6 byte test string and a pointer into the longer table string.

6.3.1.2 Addition And Subtraction Instructions

The TMS7000 instruction set supports both single and multi-precision addition and subtraction for either binary or BCD, signed (two's complement) or unsigned data.

The following example illustrates how to perform a 32-bit addition with the ADD and ADC instructions:

```

ADD      R30,R120
ADC      R29,R119
ADC      R28,R118
ADC      R27,R117
    
```

Since no initial carry-in is desired, the first instruction is ADD. The ADC instruction is then executed three times in succession to transfer the carry through all 32 bits.

The following example illustrates how to perform a 24-bit subtraction with the SUB and SBB instructions:

```

SUB      R4,R127
SBB      R3,R126
SBB      R2,R125
    
```

Since no initial borrow-in is desired, the first instruction is SUB. The SBB instruction is then executed twice in succession to achieve the 24-bit result. The addition and subtraction instructions, their execution results, and the status bits affected are listed in Table 6-11.

TABLE 6-11 – ADD AND SUBTRACT INSTRUCTIONS

INSTR	DESCRIPTION	EXECUTION RESULTS	STATUS BITS AFFECTED
ADD	Add	$[(s) + (d)] \rightarrow (d)$	C: 1 on carry out of [...] N: set on result Z: set on result
ADC	Add w/Carry	$[(s) + (d) + C] \rightarrow (d)$	C: 1 on carry out of [...] N: set on result Z: set on result
DAC	Dec Add w/C	$[(s) + (d) + C] \rightarrow (d)$ – Decimal BCD –	C: 1 if $[..] \geq 100$ decimal N: set on result Z: set on result
SUB	Subtract	$[(d)-(s)] \rightarrow (d)$	C: 1 if $[..] \geq 0$ N: set on result Z: set on result
SBB	Sub w/Borrow	$[(d)-(s)-1 + C] \rightarrow (d)$	C: 1 if no borrow N: set on result Z: set on result
DSB	Dec Sub w/B	$[(d)-(s)-1 + C] \rightarrow (d)$ – Decimal BCD –	C: 1 if no borrow N: set on result Z: set on result

The overflow/underflow conditions for both signed and unsigned systems are summarized in Figures 6-15 and 6-16, respectively. Note that an Exclusive OR of the C and N bits ANDed with the Exclusive OR of the MSBs of the operands can always be used as a check for an overflow or underflow for subtraction in a signed system (if $(C \text{ XOR } N) \text{ AND } (\text{MSB1 XOR MSB2}) = 1$ then out of range). When adding two signed numbers, the test for an out of range condition is similar to the subtraction method. When an Exclusive OR of the C and N bits ANDed with the inverse of the Exclusive OR of the MSBs of the two operands equals one then an overflow or underflow has occurred (if $(C \text{ XOR } N) \text{ AND } (\text{NOT}(\text{MSB1 XOR MSB2})) = 1$ then out of range).

* ROUTINE TO CHECK FOR SIGNED UNDERFLOW OR OVERFLOW

* If $(N \text{ XOR } N) \text{ AND } (\text{MSB1 XOR MSB2}) = 1$ then out of range

	MOV	OPRND1,A	
	XOR	OPRND2,A	get XOR of the MSBs
	SUB	OPRND1,OPRND2	Subtract 2 signed numbers
	JN	ISNEG	
NOTNEG	JNC	NOERR	N = 0
	JMP	CXORN1	C XOR N = 1, First part of equation is true
*			
ISNEG	JC	NOERR	N = 1
CXORN1	TSTA		C XOR N = 1; set flags for MSB1 XOR MSB2
*	JPZ	NOERR	If $(N \text{ XOR } C) \text{ AND } (\text{MSB1 XOR MSB2}) = 1$ then out of range. For addition change this instruction to JN NOERR
*			
OUTRNG	...		Out of Range. Underflow or overflow
*			
NOERR	...		No underflow or overflow

In an unsigned system, the C bit always reveals the overflow/underflow status as follows: addition overflow if $C = 1$ after addition, and subtraction underflow if $C = 0$ after subtraction. Figures 6-15 and 6-16 show the >00 to $>FF$ boundary as being detectable by the C bit. The decrement instructions DEC and DECD set the C bit to 0 if the >00 to $>FF$ boundary is crossed, i.e., the 0 to 255 boundary in the unsigned system, and the 0 to -1 boundary in the signed system.

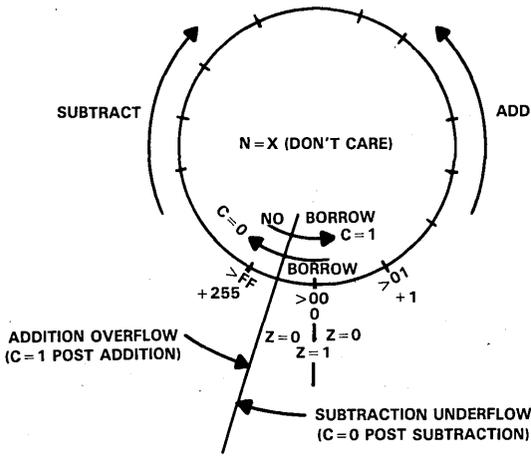


FIGURE 6-15 — UNSIGNED SYSTEM WITH 8 BITS OF MAGNITUDE: 0-255 (>00->FF)

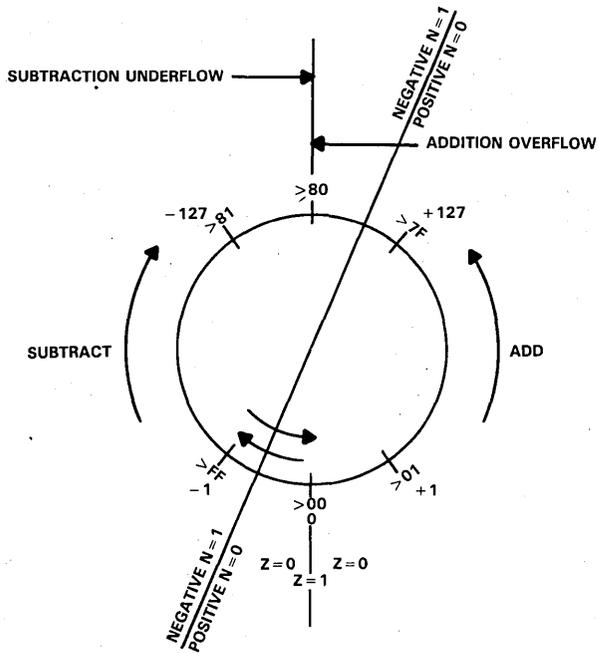


FIGURE 6-16 — SIGNED SYSTEM WITH 7 BITS OF MAGNITUDE: -127 to +127 (>81->7F)

The following subroutine shows the use of the addition instructions in adding two multi-digit numbers together. Each of the numbers is a packed BCD strings of less than 256 bytes (512 digits) stored at memory locations STR1 and STR2 . This routine adds the two strings together and places the result in STR2 placing the result in STR2. The strings must be stored with the most significant byte in the lowest number register. With most all of the TMS7000 family instruction set, it is convenient to store all numbers and addresses with the most significant byte in the lower numbered location.

```

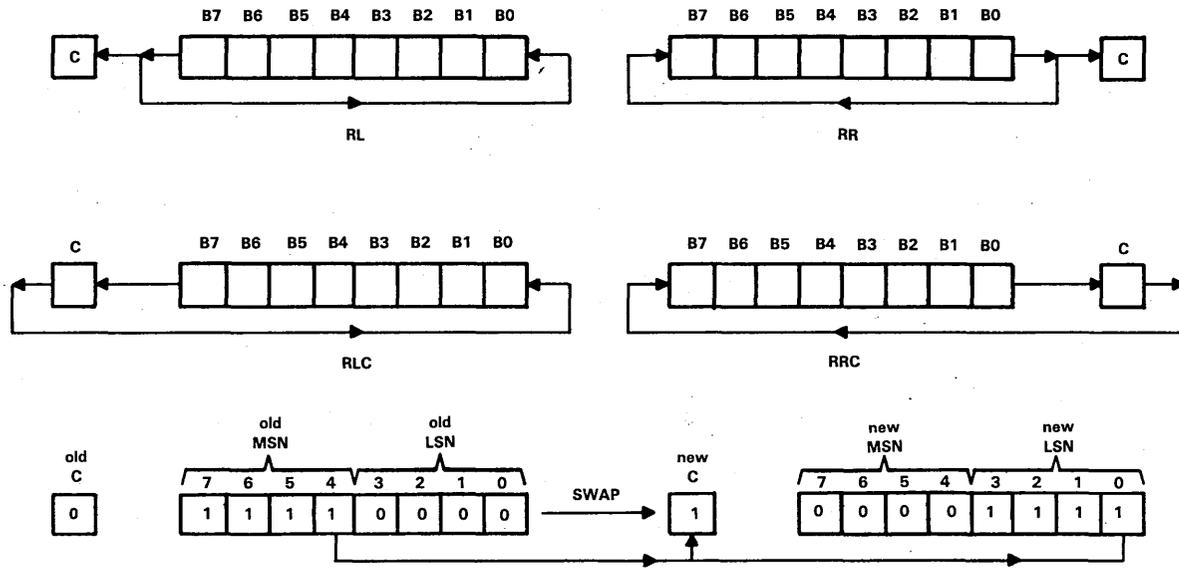
*          DECIMAL ADDITION SUBROUTINE
*          ON INPUT : B = LENGTH OF STRING ( NUMBER OF BYTES )
*                   STACK MUST HAVE 3 AVAILABLE BYTES.
*
*          ON OUTPUT STR2 = STR1 + STR2
*
ADDBCD  CLRC          Clear carry bit
        PUSH  ST      Save status of stack
LOOP    LDA   @STR1-1(B)  Load current byte
        MOV   A,R2     Save it in R2
        LDA   @STR2-1(B)  Load next byte of STR2
        POP   ST      Restore carry from last add
        DAC   R2,A     Add decimal bytes
        PUSH  ST      Save the carry from this add
        STA   @STR2-1(B)  Store result
        DJNZ  B,LOOP   Loop until done
        POP   ST      Restore stack to starting position
        RETS         Back to calling routine
*

```

Notice the use of the indexed addressing mode to reference the bytes of the decimal strings. Notice also the need to push the status register between decimal additions, to save the decimal carry bit. The B register is used to keep count of the number of bytes that have been added.

6.3.1.3 SWAP and Rotation Instructions

The rotation operations performed by the four rotation instructions Rotate Right (RR), Rotate Right Through Carry (RRC), Rotate Left (RL), and Rotate Left Through Carry (RLC) are illustrated in Figure 6-17. The SWAP instruction executes the equivalent of four consecutive RL instructions, with the C bit in the Status Register set equal to Bit 4 of the original operand or Bit 0 of the result, i.e., LSB of the result. A SWAP instruction example is also given in Figure 6-17.



Note: N and Z set on result for RL, RLC, RR, RRC, and SWAP.

FIGURE 6-17 — SWAP AND ROTATION OPERATIONS

6.3.2 Stack Operations

The stack is located in RAM and can be tailored to the specific needs of the user. One powerful application of the stack is the establishment of tables. For example, Figure 6-18 illustrates a dispatch table with an interpretive program counter (IPC). An IPC is used in some high level languages, such as PASCAL, to give the proper execution sequence. The IPC can be contained in any register and it points to an interpretive pseudo code (PCODE) byte that in turn specifies one of 256 dispatch routines. The overall effect of this function is that a program can execute one of a large number of different routines depending on a single value stored in a register. Two separate 256-byte sections are required for the high and low address bytes of each dispatch routine. The first entry of each section (ROV0) corresponds to PCODE = 0, and the second entry (ROV1) to PCODE = 1, etc.

*	IPC	EQU	R3	Interpretive Program Counter
		LDA	*IPC	Get the input Code, range = 0-255
		DECD	IPC	Point to the next input code
		MOV	A,B	PCODE Index Register
		LDA	@DTABLE(B)	Lookup Address MSB
		PUSH	A	Put MSB on stack
		LDA	@DTABLE + 256(B)	Lookup Address LSB
		PUSH	A	Put LSB on stack
		RETS		Jump to the Address on the stack
*				
*	DTABLE	BYTE	ROV0/256	Beginning of MSB table
		BYTE	ROV1/256	
		BYTE	ROV255/256	LSB table starts here
*		BYTE	ROV0	— Warning Messages May
		BYTE	ROV1	Appear Here But They Do
				Not Affect Results
		BYTE	ROV255	

FIGURE 6-18 — EXAMPLE OF A DISPATCH TABLE WITH AN INTERPRETIVE PROGRAM COUNTER (IPC)

It should be noted that the assembler expressions have 16-bit values. For those instructions requiring an 8-bit operand, the expression is truncated to the least significant 8 bits. A warning message may result from this truncation, but the value will be correct. Thus, the following instructions place byte values >AA, >55, and >55 at memory locations >8000, >8001, and >8002, respectively:

	AA55	LABEL	EQU	>AA55	
8000			AORG	>8000	
8000	AA55		DATA	LABEL	
8002	55		BYTE	LABEL	LSB only

The most significant byte (MSB) of an expression can be obtained by dividing the value by 256 (2⁸) as shown below:

	AA55	LABEL	EQU	>AA55	
8000			AORG	>8000	
8000	AA55		DATA	LABEL	
8002	AA		BYTE	LABEL/256	MSB only

6.3.3 Subroutine Instructions

There are two types of instructions for invoking subroutines: CALL and TRAP. Both instructions save the current value of the Program Counter (PC) on the stack before transferring control to the subroutine. Since the return address is stored on the stack, subroutines can be easily nested. The two types of instructions differ only in the way in which the subroutine address is determined and in the amount of program memory required for execution of the subroutine.

The CALL instruction uses the Extended Addressing modes (Direct, Register File Indirect, and Indexed) to specify the subroutine address. This permits simple calls with a fully specified address as well as more complex calls with a calculated address. Of the two types of instructions, the CALL instruction requires more program memory than the TRAP instructions. For example:

```
CALL @BITTEST
```

requires three bytes of memory: one byte for the opcode and two bytes for the subroutine address. If the subroutine call is required at six locations, 18 bytes are necessary to implement the CALLs. The equivalent task for the TRAP instruction requires only 8 bytes for six successive uses of the same TRAP, since only the opcode byte is necessary after the first use. Six of these 8 bytes are the TRAP opcodes and the other two bytes are the trap vector. The first use of the TRAP instruction requires one opcode byte plus the two bytes of the subroutine address which are located in the Trap Table. The next use and every subsequent use will only require one more byte as compared to the 3 bytes for every call. All the trap vectors are stored at the end of memory with the most significant byte of the trap subroutine stored in the lower numbered location. The exact address where the trap vector (which is the trap subroutine address) is stored is found from the following formula.

LSB of Address which contains the TRAP subroutine address = >FFFF - 2 x N where N is the TRAP number.

MSB of address = LSB - 1

The TRAP instructions (TRAPs 4-23) provide the most efficient means of invoking subroutines. Figure 6-19 illustrates an example of a subroutine call generated by a TRAP instruction.

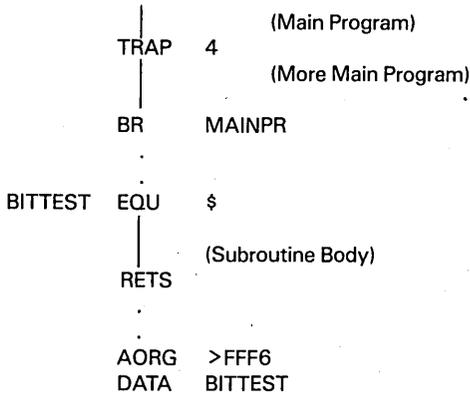


FIGURE 6-19 — EXAMPLE OF A SUBROUTINE CALL BY MEANS OF A TRAP INSTRUCTION

The Return-From Subroutine (RETS) instruction should be executed to pop the PC from the stack and restore program control to the instruction immediately following the CALL or TRAP instruction.

6.3.4 Multiplication And Shifting

The MPY instruction performs an 8-bit by 8-bit multiply with a 16-bit result that is stored in the A and B registers. The most significant byte (MSB) of the result is in A, and the least significant byte (LSB) is in B. The MPY instruction can also be used to perform multi-bit right or left shifts by using an immediate operand as the multiplier. For example:

```
MPY    %8,B
```

The above example takes the value of B and multiplies it by 8. After the instruction executes, B contains the previous value left-shifted three bits ($2^3 = 8$) with no fill bits. The A register contains the previous value's most significant 3 bits which gives a value equivalent to shifting the previous value right 5 bits ($8-3 = 5$) with no fill bits. Using this method it is possible to shift any 8 bit value left or right up to 8 bits. In most cases this is faster than the rotate instructions and almost always takes less program bytes. Table 6-12 lists the number of bits right- or left-shifted for a range of immediate multipliers.

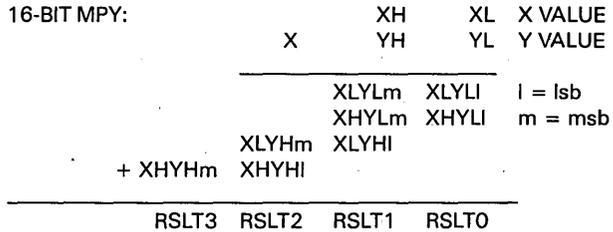
TABLE 6-12 — MULTI-BIT RIGHT OR LEFT SHIFTS BY IMMEDIATE MULTIPLY

IMMEDIATE MULTIPLIER	BITS RIGHT SHIFTED	BITS LEFT SHIFTED
2	7	1
4	6	2
8	5	3
16	4	4
32	3	5
64	2	6
128	1	7

NOTE: Rotate instructions may take less execution time than a Multiply instruction.

Multi-precision multiplications can be easily executed by breaking the multiplier and the multiplicand into scaled 8-bit quantities, as shown in Figure 6-20 (16 x 16 bit multiplication with a 32-bit result in R6-R9).

* 16 BIT MULTIPLICATION ROUTINE
 * MULTIPLIES THE 16 VALUE IN REGISTER PAIR R2,R3 BY THE
 * VALUE IN REGISTER PAIR R4,R5. THE RESULTS ARE STORED
 * IN R6, R7, R8, R9 AND THE A AND B REGISTERS ARE ALTERED
 *



XH EQU R2 Higher operand of X
 XL EQU R3 Lower operand of X
 YH EQU R4 Higher operand of Y
 YL EQU R5 Lower operand of Y
 RSLT3 EQU R6 Msb of the final result
 RSLT2 EQU R7
 RSLT1 EQU R8
 RSLT0 EQU R9 Lsb of the final result

*
 * MPY32 CLR RSLT2 Clear the present value
 * CLR RSLT3
 * MPY XL,YL Multiply lsb's
 * MOV B,RSLT0 Store lsb in result register 0
 * MOV A,RSLT1 Store msb in result register 1
 * MPY XH,YL Get XHYL
 * ADD R1,RSLT1 Add to existing result XLYL
 * ADC R0,RSLT2 Add carry if present
 * MPY XL,YH Multiply to get XLYH
 * ADD R1,RSLT1 Add to existing result XLYL + XHYL
 * ADC R0,RSLT2 Add to existing results and carry
 * ADC %0,RSLT3 Add if carry present
 * MPY XH,YH Multiply msb's
 * ADD R1,RSLT2 Add once again to the result reg
 * ADC R0,RSLT3 Do the final add to the result reg
 *
 *

FIGURE 6-20 — EXAMPLE OF A 16-BIT BY 16-BIT MULTIPLICATION SUBROUTINE

6.3.5 Branch Instruction

The branch instruction (BR) is used to unconditionally transfer program control to any desired location in the 64K byte memory space. The BR instruction supports direct, indexed, and indirect addressing. Direct addressing is used for simple "GOTO" programming. Indexed addressing allows table branches. This indexed branch technique is similar to the Pascal "CASE" statement, Program control is transferred to location CASE0 if the input is '0', to CASE1 if it is a '1', etc. This transferring method can implement up to 85 different cases. In the example below, indexed addressing is used to access a relative branch table:

```

JTABLE  MOVP  P4,A           Get data from A port ( Value ' 85 )
         ADD   A,B           Add twice to triple value
         ADD   A,B           Multiply it by 3 ( BR is 3 bytes long )
         BR    @CTABLE(B)    branch according to the A port value * 2

*
*
CTABLE  BR     @CASE0        If P4 = 0 do this branch
         BR     @CASE1        If P4 = 1 do this branch
         BR     @CASE2        If P4 = 2 do this branch

*
*

```

The branch instruction can also be used with indirect addressing in order to branch to a computed address. For example, suppose that a computed branch address has been constructed in R19 and R20. The desired program control transfer is made by:

```
BR      *R20
```

6.3.6 Interrupts

The number of interrupts and the hardware configuration for an TMS7000 family device is specified by each device in Section 2. The TMS7020, for example, has three interrupts in addition to RESET.

RESET and the interrupts are vectored through predetermined memory locations. RESET uses the "TRAP 0" vector which is stored at memory locations >FFFE - >FFFF. The interrupts also use the TRAP vector table with INT1 using the "TRAP 1" vector, etc. Thus, the "TRAP 2" instruction involves the same code as the interrupt INT2 (see Section 6.3.3).

The interrupts differ from the TRAPs in that they also push the status register value on the stack, clear the interrupt enable bit in the status register, and reset the corresponding interrupt flag bit. Thus the EINT instruction must be used if nested interrupts are desired. The return from interrupt (RETI) instruction restores the status register and the program counter, re-enabling interrupts.

Many interrupt service routines alter the status of key registers such as the A and B registers. These routines should use the stack to restore the machine state to the desired value. For example, the following interrupt routine performs an I/O driven table look-up. The A and B registers are used, but their values are saved and restored:

```

*
INT  PUSH  A           Store A and B registers on stack
      PUSH  B
      MOVP  P4,B       Get input from the A port
      LDA   @LOOKUP(B) Do a table lookup to get new value
      MOVP  A,P6       Output new value on B port
      POP   B           Restore A and B registers in the
      POP   A           reverse order that they were put on
      RETI              Back to main program.
*

```

Normally all interrupts are disabled during an interrupt service routine. If an interrupt needs to be able to occur while the processor is servicing another interrupt, then the interrupt enable bit in the status register should be set to a '1'. The number of interrupts that can be serviced at any one time is determined by the size of the stack which is always a maximum of 128 bytes because the stack resides in the register file. Since other registers and data will most likely share the same space, the stack size is usually much less. When doing nested interrupts, great care must be taken to avoid corrupting the data in the registers used by the most recent routine. If INT1 interrupts an ongoing INT1 service routine, then the registers used by the INT1 routine are used in two different contexts. If provisions are not made for these type situations, such as disabling all interrupts at critical times, then data errors will result.

Sometimes a program will have distinct parts which require different responses to the same interrupt call. Since the interrupt vector is always set in nonchangeable ROM, another method must be used to change the vector for each part. One way of accomplishing this is to store a second vector in a RAM register pair and let the first instruction in the interrupt routine execute an indirect branch on that register. The example below shows how this is done.

```

*          PROGRAM TO DEMONSTRATE MULTIPLE INTERRUPT SERVICE ROUTINE
*          LOCATIONS.
*          main program
MOVD    %SERVIC,R127    Put Int. 1 service routine
EINT                                address in register
IDLE                                turn on and wait for interrupts
MOVD    %SERV12,R127   change Int. 1 routine to SERV12.
*          . . . .
*
*          First interrupt 1 service routine
SERVIC  PUSH    A          Beginning of the Int. 1 service
      PUSH    B          routine for this part of the program
*          . . . .
*
*          Second interrupt 1 service routine
SERV12  PUSH    A          Start of another interrupt 1 service
      DEC    R4          routine
*          . . . .
*
INT1    BR      *R127      The entire Int. 1 service routine.
*                                Transfers control to the address which
*                                is in R127 and R126
*
*          Interrupt vector table at end of memory
AORG    >FFFC
DATA    INT1          Address of Interrupt 1 service routine
DATA    >F806        Reset vector Start of program.

```

The following routine is an example of a bubble type sorting program. This routine demonstrates the utility of the indexed mode of addressing. Tables up to 256 bytes in length can be sorted using the routine. Longer tables can be sorted using the indirect addressing mode.

```

*          150 BYTE BUBBLE SORT
*
FLAG    EQU     R2          'Swap has been made' flag
*
SORT    CLR     FLAG        Reset swap flag
      MOV     %149,B       Number of bytes to be sorted
LOOP1   LDA     @TABLE(B)   Look at entry in table
      CMPA   @TABLE-1(B)   Look at next lower byte
      JL    LOOP2         If lower skip to next value
      INC    FLAG         Entry is not lower, set swap flag
      PUSH  A            Store upper byte
      LDA   @TABLE-1(B)   Take lower byte
      STA   @TABLE(B)     Put where upper was
      POP  A            Get the old upper byte
      STA   @TABLE-1(B)   Put where the lower byte was
LOOP2   DJNZ  B,LOOP1     Loop until all the table is looked at
      BTJO  %>FF,FLAG,SORT If swap was made then resweep table
*                                If no swap was made, then table is done

```


7. DEVELOPMENT SUPPORT TOOLS

7.1 INTRODUCTION

With the introduction of the XDS* (Extended Development Support) concept of high performance support for the development needs of its customers, TI has taken a major step toward making it easier to use its microcomputers. This ease of use coupled with the high performance of the XDS tools will increase development productivity. Figure 7-1 shows the typical microprocessor development system.

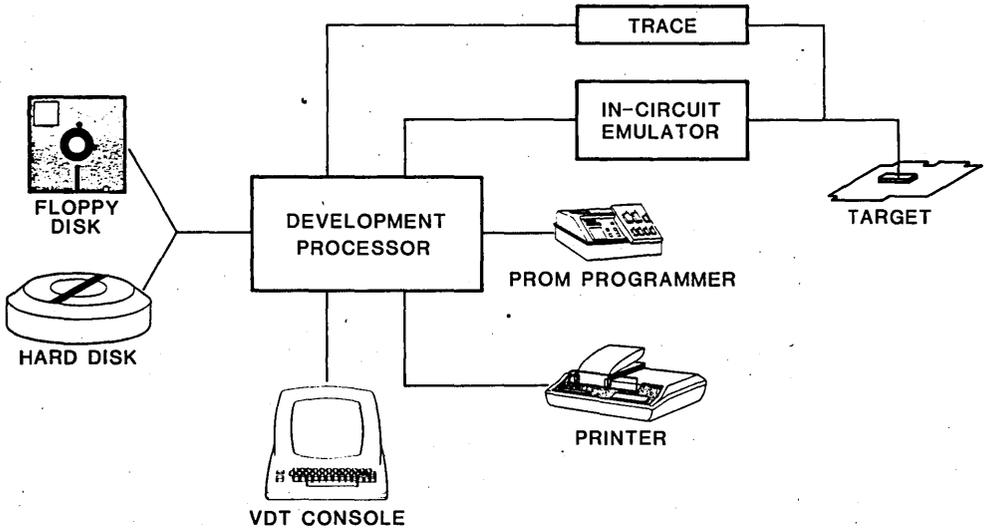


FIGURE 7-1 — TYPICAL MICROPROCESSOR DEVELOPMENT SYSTEM

* XDS is a registered trademark for Texas Instruments Incorporated, Dallas, Texas 75265. All rights are reserved.

As shown in Figure 7-2, the configuration for XDS development is different from the traditional development system configuration but results in the same functionality for the system developer. The ability to use the system tools that the developer is familiar with greatly enhances the productivity of the developer.

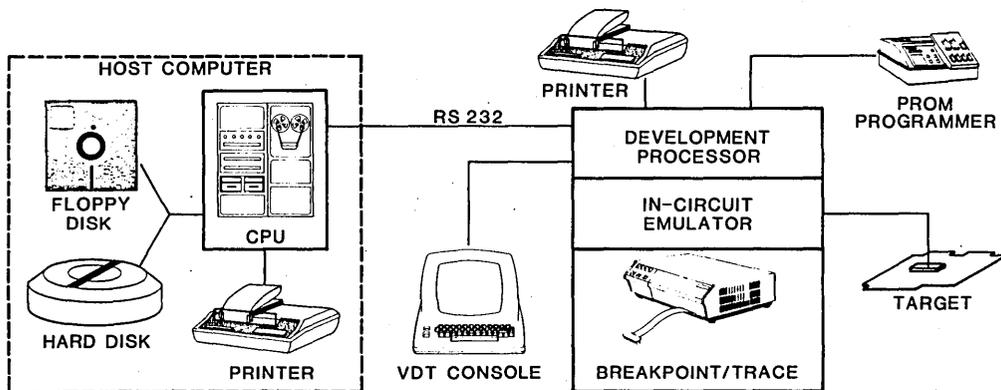


FIGURE 7-2 — TYPICAL XDS CONFIGURATION

7.1.1 XDS Concept

The XDS concept is centered around host independence that features a consistent development tool set for TI microcomputers and microprocessors. Included in the XDS concept are versatile Macro Assemblers pre-configured to run on a number of hosts. These Macro Assemblers include the necessary information to service the XDS workstations. The XDS workstations are powerful in-circuit emulators that include breakpoint and trace capabilities. As an option, intelligence can be added to provide XMPL* (High Level Debug Language) for increased target control.

The host-independent configuration of the XDS, coupled with a consistent set of development and debug tools lets the user select the TI processor best suited to solving his problem. Having a common set of tools available means the basic development format has to be learned only one time and then can be used with any member of the supported TI TMS7000 family.

XDS cross-assemblers and host interfaces are available for running under IBM 370 MVS and CMS operating systems, DEC VAX VMS operating system, and TI operating systems TX4, TX990, and DX10. This broad range of systems capability permits the development of software systems using pre-installed tools familiar to the user. This ever increasing range of operating systems supported allows development on many different hardware configurations (IBM 370, 3033, 43xx; DEC VAX 11; TI TMAM9000, FS990/4, FS990/10, DX10, and others) with more to come in the future (DEC PDP-11, IBM PC, TI PC, COMPAQ, and others). In addition, independent vendors offer support on a number of other systems (Intel MDS, Series II, Series III; CP/M based systems; etc.).

Emulation of a TI microcomputer is provided by the XDS unit using a RS232 link for interface with a variety of host systems. User supplied peripherals are also connected through similar RS232 links, thus creating a low-cost high-performance hardware/software development

* XMPL is a registered trademark for Texas Instruments Incorporated, Dallas, Texas 75265. All rights are reserved.

system. The XDS family of products supports RS232 downlink capabilities, in-circuit emulation, and target system debugging with breakpoint and trace capabilities. These capabilities enhance software development while executing real-time target system debugging.

7.1.2 Key Features

- Host independent (cross-assemblers available for IBM MVS and IBM CMS, VAX VMS, and TI DSG TX4, TX990, and DX10-with others planned)
- Provides support for TMS7000, TMS320, TMS9995, AND TMS99000 microprocessor families
- Real-time in-circuit emulation capability
- High performance at low cost
- User friendly hardware and software
- Easily expandable
- Convenient desk-top workstation, see Figure 7-3
- Allows integrated system level debug rather than just hardware or software

7.2 CROSS SUPPORT SOFTWARE PACKAGE

CrossWare* (Cross Support Software Package) is available to run on many hosts to support TI XDS development. TI CrossWare packages are available for the IBM MVS and CMS operating systems, DEC VMS operating system, and TI TX4, TX990, and DX10 operating systems. Support is available for Intel MDS 800, Intel Series II, Intel Series III, and CP/M based systems from independent vendors. See Section 8 on independent support. Future support is planned for the DEC PDP-11, IBM PC, TI Professional Computer and others.

The CrossWare packages come complete with a full featured macro assembler and a linkage editor to support modular software with link of the modules at link time rather than at assembly. This approach encourages writing of small modules and speeds the correction of program errors.

CrossWare documentation provides the installation information necessary for each specific host to implement the support package and support attachment of XDS hardware for target debug.

7.3 XDS HARDWARE

XDS hardware supports TMS7000 microcomputer system development utilizing a host independent approach. Currently, there are two product offerings available in the XDS hardware family. The Model 22 is a full featured, real-time in-circuit emulator offering hardware breakpoints and logic state trace capabilities. The Model 33 XDS offers all of the capabilities of the Model 22 with the added feature of built in intelligence running the high level target debug language XMPL (Extended Microprocessor Prototyping Language).

* CrossWare is a registered trademark for Texas Instruments Incorporated, Dallas, Texas 75265. All rights are reserved.

7.3.1 Model 22

The XDS Model 22 includes a chassis, card cage, power supply, fan, and a three board set consisting of an emulator, communications and memory expansion board, and a separate board for setting breakpoints and logic state tracing (see Figure 7-3).

The software written and developed on the host can be downloaded into the Model 22 emulator memory space through a standard RS232 EIA link. Further development and testing of target hardware and software is aided through the versatile and comprehensive debug monitor located in firmware, onboard the emulator. Over 65 commands are available (including HELP) to give the user complete control over the target system. Key among the 65 monitor commands is an assembler permitting almost any system to be used as an intelligent terminal and prepare the source text for assembly by the XDS box. The XDS Model 22 can perform full speed in-circuit emulation with breakpoint and trace capabilities.

Utilizing the hardware and software breakpoint commands and the logic state trace analyzer, a complete record of events can be examined to rapidly increase debugging efficiency and decrease development time. The user can select a range of memory addresses and I/O addresses to set valid breakpoints. The breakpoint/trace (B/T) board can breakpoint on any memory cycle, a memory read, a memory write, or an instruction acquisition. For I/O operations, the B/T board can breakpoint on any I/O cycle, I/O read, or I/O write, if the I/O address qualifications are met. A trace is provided to give a history of execution prior to the breakpoint. Trace samples are stored in the trace memory and can be read back after execution has been halted. The user can trace memory cycles and I/O cycles.

This cycle of using the host computer and the XDS Model 22 for testing produces a quick efficient way for target system development. After debugging is complete, EPROMs can be programmed using the host computer's PROM programming capabilities.

7.3.2 Model 33

The XDS Model 33 is one of the most advanced development support tools available on the market today. It includes the feature set of the XDS Model 22, and it presents a user-friendly, high-level interface and debug language for complete control of the target application system.

XMPL, a sophisticated, high level target debugging language, supports the previously mentioned TI microcomputers. The user interface presented by XMPL is screen oriented to maximize system use and offers a procedure oriented command system. By defining new screen formats and command processes, a collection of procedures can be supplied to support a wide variety of applications. XMPL gives the user a means of controlling emulator functions and communicating with the host system to gain access to mass storage and data generated on the host.



FIGURE 7-3 — THE XDS MODEL 22

7.3.3 Differences And Similarities - Model 22/Model 33

Table 7-1 provides the differences and similarities between the Model 22 and the Model 33 products.

TABLE 7-1 — HARDWARE CONFIGURATION DIFFERENCE MODEL 22 TO MODEL 33.

MODEL 22		
SLOT	BOARD	FUNCTION
7	-----	FUTURE EXPANSION
6	-----	MODEL 33 EXPANSION
5	-----	MODEL 33 EXPANSION
4	COMMUNICATIONS	COMM. WITH HOST
3	-----	TI EMULATOR SPACE
2	EMULATOR	IN-CIRCUIT EMULATION
1	BREAKPT/TRACE	BREAKPOINTS/TRACE
MODEL 33		
SLOT	BOARD	FUNCTION
7	-----	FUTURE EXPANSION
6	TM990/233	XMPL PROGRAM MEMORY
5	TM990/103	USER INTERFACE (HLI)
4	COMMUNICATIONS	COMM. WITH HOST
3	-----	TI EMULATOR SPACE
2	EMULATOR	IN-CIRCUIT EMULATION
1	BREAKPT/TRACE	BREAKPOINTS/TRACE

Model 22 XDS and Model 33 XDS are packaged in the same modular table top workstation. This workstation contains a dual motherboard providing for the interface between the emulation modules and the TM990 modules used to provide the intelligence for the XMPL language. Using this bus structure provides for easy upgrade and changing CPU support. A diagrammatic reference is provided in Table 7-1 to illustrate the similarities/differences in the Model 22 and the Model 33.

The memory map of the XDS for the TMS7000 family is extremely flexible. As shown in Figure 7-4, the map can be arranged in any practical configuration that the developer desires. This flexibility facilitates system level debug rather than just software or hardware debug.

Microprocessor	Memory Location Emulator : Memory Expansion
TMS70XX	64K bytes : (not used, all memory in emulator)

TMS7000: Allocated in 256 byte blocks, X blocks as on-chip ROM and Y blocks as off-chip memory, where $256(X + Y) = 64K$ bytes.

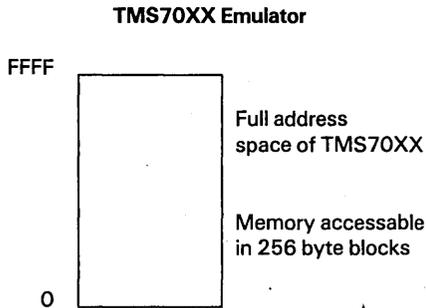


FIGURE 7-4 — MEMORY CONFIGURATION IN XDS/70 MODEL 22/33

7.3.4 XMPL

The XMPL high level debug language controls the emulation of target application programs by setting breakpoints, defining of data or address comparison events, data and address trace, as well as direct target system I/O and memory manipulation. These procedures are programmable in a high level language using integer and boolean mathematics. The Pascal constructs included are capable of repetitive sequences and decision making. Allowing a program to test and act upon target application events, XMPL reads emulator and trace conditions while viewed by the user through self-defined windows on the video screen.

The screen oriented user interface is designed to maximize system use. The user is encouraged to customize the interface to a particular application by writing a procedure to display target registers or memory pertinent to his application. The output of the procedure is displayed in a temporary window. If the user makes the window permanent, the information is updated whenever a command is entered and the emulator is not running. With very little effort the user has created a constant visible description of the state of his application.

XMPL supports multiple levels of user sophistication. This allows the experienced user to quickly enter commands and parameters all on one line, while the inexperienced user is helped by self prompting sequences which quickly direct the user to entering the required information. The flowchart in Figure 7-5 demonstrates the three levels.

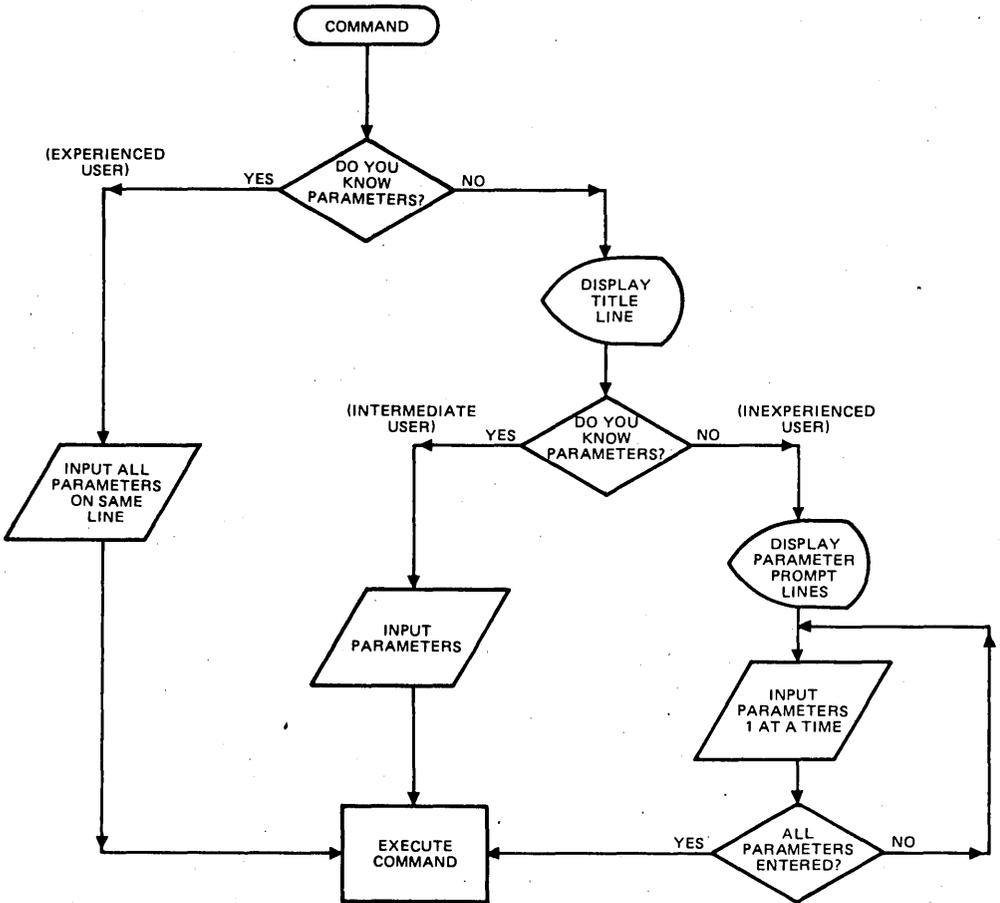


FIGURE 7-5 — LEVELS OF XEMPL INTERFACE

7.3.5 Breakpoint And Trace Functions

The XDS breakpoint and trace (B/T) board allows the user to set a hardware interrupt or breakpoint which halts emulator execution. Breakpoints can be set on I/O and/or memory operations with three simple monitor commands. The user can select a range of memory addresses and I/O addresses for valid breakpoints, or can select two separate memory addresses or two separate I/O addresses. The B/T board can breakpoint on any memory cycle; memory read, memory write, or an instruction acquisition. For I/O operations, the B/T board can breakpoint on any I/O cycle; I/O read, or I/O write if the I/O address qualifications are met.

The trace function is provided to give a history of execution prior to the breakpoint. It is used to analyze a set of signals based on addresses and commands. Trace samples are stored in trace memory and can be read back after execution has been halted. The user can trace both memory and I/O cycles including memory read, memory write, and instruction acquisitions or all memory cycles, and I/O read, I/O write, or any I/O cycle.

The trace memory can hold 2048 words by 48 bits of trace samples. The user is given the option of how many of these 2048 samples to take, or to keep wrapping around in trace memory, writing over the oldest trace sample with the newest trace sample.

7.3.6 Multiprocessing

With the ever increasing use of sophisticated designs of multiple microprocessor systems, there is need for multiprocessor development support. TI's XDS offers this multiprocessor support to debug up to 9 stations linked together in a daisy chained fashion. These systems can be any of the XDS supported TI processors (TMS7000 family, TMS320 family, TMS9995 family or the TMS99000). The XDS system is connected to the host computer via the RS232 port of the last XDS workstation. A single user CRT interface can control each of the workstations. The target system may be of any configuration of TI microprocessors that are supported by XDS. Each workstation may be utilized individually or the workstations can be grouped or subgrouped to synchronize control over the entire target system.

7.4 EVALUATION MODULES

The Texas Instruments RTC series of evaluation modules are designed for hands-on hardware evaluation of specific TI microcomputers. In addition, the RTC/EVM's can function as a limited feature, stand-alone development system for the family of parts that they support. To facilitate the evaluation/development functions, EVMs offer text editing, audio cassette interface, upload/download support, assembler, and EPROM programming utility.

The RTC/EVM7000, Figure 7-6, is designed to emulate the single-chip mode of the TMS7000. It does not support the expansion modes of the TMS7000 family of processors.

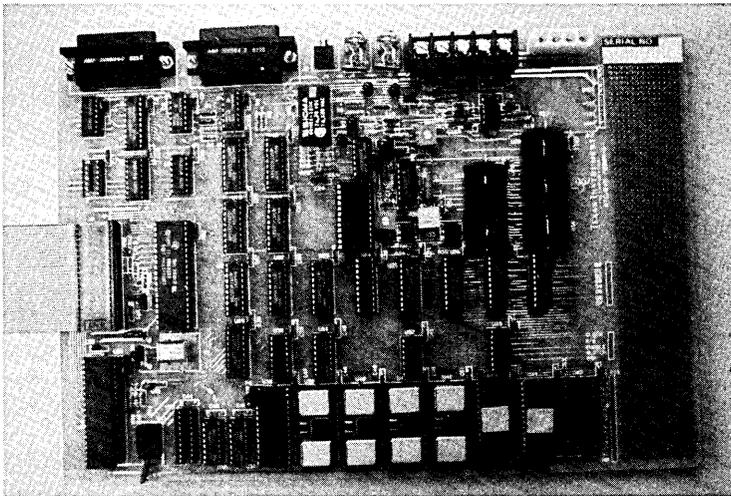


FIGURE 7-6 — THE RTC/EVM 7000 EVALUATION MODULE

7.4.1 TMS7000 EVM

The RTC/EVM7000 is a single board system capable of emulating the single-chip mode of operation of the TMS7000 family of microcomputers. There are two versions of the evaluation module: the RTC/EVM7000N-1 for NMOS members of the TMS7000 family and the RTC/EVM7000C-1 for CMOS versions. The EVM can stand alone as a development system, using the on-board text editor for creation of TMS7000 Assembly Language text files, and the audio cassette tape interface, with limited directory and file search capability as a mass storage media. A more productive environment can be accomplished by connecting the EVM to a resident host computer that is used to develop and save the text files and either using CrossWare to assemble them on the host or download the text files to the EVM for assembly by the on-board assembler. To support this and other possible configurations the EVM has two EIA RS232 ports.

The EVM firmware supports three ports in the operation of loading and dumping data (text, object code) for storage and/or display. Two of these ports conform to EIA RS232C and are called PORT 1 and PORT 2. The third port is the audio tape connection, PORT 3. The baud rates supported on PORTs 1 and 2 are 110 through 9600 baud.

The EVM comes equipped with eight 8K byte sockets for the entire 64K byte address space of the TMS7000. Currently, 16K bytes of the EPROM is devoted to the resident firmware (>C000 to >FFFF). User RAM is expanded in 8K byte increments, from 16K bytes to 32K bytes. Available to the user for addition of logic is a wire-wrap development area with all required signals provided and labeled.

To facilitate evaluation/development of a TMS7000 project, the EVM offers a limited feature emulation capability. The crystal frequency of the EVM can be tied to the target application through the emulation cable.

7.4.1.1 Operating System

The EVM operating system firmware resides in 16K bytes of EPROM and is divided into three functional areas:

- Debug monitor and EPROM programmer
- Assembler
- Text Editor

All the software is designed to interact with itself and the user to provide an easy to use development/evaluation tool.

The EPROM programmer provides control for:

- TMS2764 EPROMS
- TMS27128 EPROMS

During assembly/debug operations, the EVM RAM can be configured to emulate all TMS7000 family members and for the emulation of the 2K and 4K ROM version devices, allows assembly of text files directory from RAM.

7.5 PROTOTYPE COMPONENT

The SE70P161 is a prototyping component that Texas Instruments offers to support form factor evaluation of a TMS7000 target.

7.5.1 SE70P161 Description

The SE70P161 prototyping component is another member of the TMS7000 family of single-chip 8 bit microcomputers. The SE70P161 is pin compatible with the TMS7020, TMS7040, TMS70120, and TMS7041, and has the same instruction set as these devices. The SE70P161 can also be used to emulate CMOS members of the TMS7000 family, with the following limitations. Because the SE70P161 is an NMOS device, its logic levels are not CMOS compatible. Also, this device does not support the low-power modes of the CMOS devices such as HALT or Wake-up. Finally, INT1 on the SE70P161 is both latched and level triggered as in the NMOS devices, not just latched, as in the CMOS devices. Further details of these differences are provided in the sections which discuss the function.

The SE70P161 serves as a form fit and function component for the TMS7000 devices and provides the ability to verify in real-time the software written for all TMS7000 family members mentioned in the preceding paragraphs. This device uses standard 2764 or 27128 EPROMs. The EPROMs are located in a socket in the top of the 40 pin SE70P161. Refer to Table 7-2 for mapping information for the various EPROMs supported.

The SE70P161 is packaged so that an EPROM device can be plugged into the top of the package (piggy back). This two chip unit acts as an emulator of the TMS7020 (2K bytes of internal ROM space), the TMS7040/7041 (4K bytes of internal ROM space), and the TMS70120 (12K bytes of internal ROM space).

TABLE 7-2 — EPROM USE

EPROM TYPE	70XX ROM	70XX* START ADDRESS	27XX START ADDRESS
27128	16K Bytes	>C006	>0006
27128	12K Bytes	>D006	>1006
2764	8K Bytes	>E006	>0006
2764	4K Bytes	>F006	>1006
2764	2K Bytes	>F806	>1806

*NOTE: Texas Instruments reserves the first 6 bytes of ROM. For example addresses from >F000 to >F005 may not be defined by the user program for a TMS7040.

The SE70P161 is available in two versions. Both versions have fixed internal ROM space of 16K bytes (C000-FFFF), one with a divide by two clock generator and the other with a divide by four. Note that on the SE70P161, none of the 16K EPROM address space can be mapped as external addresses except in microprocessor mode.

7.5.1.1 Prototyping

System emulators such as the SE70P161 are only designed to be used in a prototype environment and as such are tested and supported for that purpose.

7.5.1.2 TMS7041 Prototyping

The SE70P161 serves as a prototyping component for the TMS7000 devices and provides the ability to verify in real-time software written for all TMS7000 family members mentioned in Section 4. This device uses standard TMS2764 or TMS27128 EPROMs. The EPROMs are located in a socket on top of a 40-pin dual-in-line package.

7.5.1.3 TMS7020/TMS7040/TMS70120 Prototyping

The SE70P161 system emulator can also be used as a TMS7020/TMS7040/TMS70120 prototype. In this case, P16 (Peripheral File location >010), must be cleared during the device initialization routine to prevent spurious interrupts from the unused serial port logic. One way to accomplish this is by coding MOVP %>00,P16 in the initialization routine.

7.5.1.4 SE70P161 Electrical Data

Reference Section 4.3 (SE70P161) for electrical specifications.

7.6 PHYSICAL AND ORDERING INFORMATION

7.6.1 CrossWare

PART NUMBER	DESCRIPTION	OPERATING SYSTEM
TMDS7040113-21	TI 990 DSDD	TX-5
TMDS7040123-06	TI 990 T50	DX10
TMDS7040123-08	TI 990 Tape	DX10
TMDS7040123-10	TI 990 DS10	DX10
TMDS7040123-22	TI 990 CD1400	DX10
TMDS7040133-03	TI 990 SSSD	TX-4
TMDS7040210-08	DEC VAX Tape	VMS 3.0
TMDS7040310-08	IBM Mainframe	MVS
TMDS7040320-08	IBM Mainframe	CMS

7.6.2 XDS Hardware

MICROCOMPUTER	XDS MODEL NO.	PART NO.
TMS7020, TMS7040 TMS7041, TMS70120	Model 22 Model 33	TMDS7062210 TMDS7063310

7.6.2.1 Physical Specifications

The XDS equipment is a professionally styled table top sized unit suitable for most work surfaces. The XDS Models 22 and 33 have an air inlet on each side of the unit and an air exhaust port on the rear of the unit. A minimum of five inches clearance must be maintained between the XDS and neighboring equipment on the sides and rear for proper air flow. Listed below are the dimension and clearance requirements.

DIMENSIONS	
Width =	17.0 Inches (43.2 CM)
Depth =	16.5 Inches (41.9 CM)
Height =	7.4 Inches (18.8 CM)
Target Cable =	18.0 Inches (46.0 CM)

CLEARANCE REQUIREMENTS	
Sides:	5 Inches Minimum (15.2 CM)
Back	5 Inches Minimum (15.2 CM)
Top	None Required
Front	None Required

7.6.3 Evaluation Modules

The RTC/EVM is available in two configurations. The first configuration RTC/EVM7000N-1 supports the NMOS versions of the TMS7000 single chip microcomputer family. The RTC/EVM7000C-1 is designed to support the CMOS members of the TMS7000 family. Listed below are the RTC/EVM part numbers.

RTC/EVM PN	DEVICES SUPPORTED
RTC/EVM7000N-1	7020/7040/70120/7041
RTC/EVM7000C-1	70C20/70C40

7.6.4 Warranty Services

A limited warranty covers the cost of parts and labor if any defects in materials or manufacturing methods require service within 90 days from the date of purchase from TI. The software license agreement and subscriber card must be completed and returned to TI before the license is in force.

All technical questions and requests for service for XDS development of hardware and software should be directed to the customer support lines at the nearest TI Regional Technology Center (See paragraph 1.4.2, Hotline Assistance).

Repair of XDS equipment is performed at the system level with chassis and all boards being returned to the Houston factory repair center.

8. INDEPENDENT SUPPORT

8.1 INTRODUCTION

The TMS7000 family of single chip microcomputers is supported by product offerings from a number of independent vendors. These support products take many forms, from cross assemblers that run on small systems to second sources for the TMS7000 components. Included in this section are a number of tools that augment the support provided by Texas Instruments. Inclusion of a product in this section does not constitute product endorsement on the part of Texas Instruments but merely an attempt at product awareness. The products listed here are representative of independent vendor supplied products and are not intended to be an all inclusive list of independent vendor supplied support tools.

8.2 PROCESSOR INNOVATIONS* - INTEL* BASED SUPPORT TOOLS

The XI* Core Cross-Development Package enhances an Intellec* system to provide stand-alone development facilities for designing with Texas Instruments' TMS320, TMS7000, TMS99XX, and TMS99XXX microprocessor families. The XI Core Cross-Development Package consists of an XI-90/30 CPU module, the XI Software System, and companion documentation.

The XI CPU module is a busmaster module which converts an Intellec system into a dual processor XI development station. The XI CPU module, when inserted in an available busmaster slot within the Intellec system chassis, coexists with the Intellec system's current 8080 family CPU module. The module is passive during microprocessor development sessions with Intel microprocessors and active for all XI microprocessor cross-development sessions for the TMS7000 family.

The XI Software System consists of Processor Innovations' XI cross-development operating system plus a companion set of target product development, debug, and firmware manufacturing utilities for execution on the XI system. The XI operating system is dedicated to microprocessor development cross-support and is functionally equivalent to the Texas Instruments' AMPLUS† operating system for TI development systems.

The XI Core Cross-Development Package has been designed to accommodate all currently available Intellec development systems.

8.2.1 XI Workstation Device Support

The XI Core Cross-Development Package converts an Intellec system into an XI workstation capable of supporting both Intel and Texas Instruments microprocessor development activity. An XI workstation, when used to design with Texas Instruments' microprocessors, supports the following Intellec and XI devices:

- Intellec dual disk drive system (single or double density)
- Intellec display terminal
- Intellec line printer

* Processor Innovations and XI are registered trademarks for Processor Innovations Corporation, Eatontown, N.J. 07724. Intel and Intellec are registered trademarks for Intel Corporation, Santa Clara, CA. 95051. All rights are reserved.

† AMPLUS is a registered trademark for Texas Instruments Incorporated, Dallas, Texas 75265. All rights are reserved.

- XI RS232C serial communications interface
- XI CRU expansion chassis interface

The XI RS232C interface is heavily used by the XI Software System to extend XI device support to include:

- Board level target systems, such as Texas Instruments' TMS7000 evaluation modules.
- RS232C based in-circuit emulation tools, such as TI's XDS* emulator instruments for:
 - TMS7000 8 bit single-chip microcomputer family
 - RTC/EVM evaluation modules
 - TMS320 32 bit processor family
 - TMS99000 16 bit processor family
 - TMS9995 16 bit microprocessor
 - RS232C PROM/EPROM programmers

The CRU expansion chassis interface is a high speed serial link used with separately-packaged XI software utilities to extend XI device support to include:

- TI CRU based in-circuit emulator systems for:
 - TMS7000 assembler and microassembler
 - TMS9900/9900-40
 - TMS9980A/9981
 - TMS9989
 - TMS9940
- TI CRU based logic state trace system
- CRU based PROM/EPROM programmer system

Additional development hardware support will be provided in later XI Software System releases and through the introduction of planned XI add-on packages.

8.2.2 Company To Contact

Processor Innovations Corp.
 P.O. Box L
 Eatontown, N.J. 07724

Phone (201) 542-6500

Contact - Marketing

8.2.3 Product Offerings

8.2.3.1 PIDS 1810-11

The XI-800 Core Cross-Development Package contains the hardware, software and documentaion necessary to upgrade an Intellec Model 800 to an XI-800 development workstation. This package contains an XI-90/30-20 CPU module, 4 floppy diskettes (2 single, 2 double density), and supporting documentation.

8.2.3.2 PIDS 1810-12

The XI-II Core Cross-Development Package contains the hardware, software and documentation necessary to upgrade an Intellec Series II/80 or Intellec Series II/85 system to an XI-II development workstation. This package contains an XI-90/30-30 CPU module, 2 floppy diskettes (1 single, 1 double density) and supporting documentation.

8.2.3.3 PIDS 1810-32

The XI TMS7000 family Macro Assembler Package contains the software and documentation necessary to add TMS7000 assembly language program translation capability to an XI development workstation. It contains two XI floppy diskettes (1 single, 1 double density) and supporting documentation.

8.3 ALLEN ASHLEY - CP/M* BASED SUPPORT TOOLS

Included in the Allen Ashley cross assembler series are cross assemblers for TI's TMS7000 family, TMS9900 family, and TMS320 family of processors. This series of cross assemblers allows any CP/M system to serve as a development station for single-chip microcomputers and microprocessors.

With minor exceptions the SYSTEM-TMS7 assembler features instruction mnemonics and syntax as defined by Texas Instruments. The SYSTEM-TMS7 includes the ASMB interactive assembler/editor, the MAKRO macro assembler, the EDIT text editor, a cross reference generator, and off-loading facilities.

The ASMB editor/assembler is intended for the creation, modification and test of program modules. ASMB includes a simple assembler, a line editor, and the facilities for saving and retrieving files from disk. Source code for ASMB is maintained in memory to eliminate the requirement for a separate edit cycle. The source language is assembled into object code directly into RAM for immediate testing. Program errors can be caught, repaired and re-assembled in seconds with ASMB. Validated program modules developed with ASMB can be saved on disk for input to the more powerful MAKRO disk assembler.

The MAKRO assembler includes full macro and conditional assembly features, as well as the ability to link a series of source files together during a single assembly. MAKRO reads the source code from disk and writes object code back to disk: all available memory is free for symbol tables and macro expansion. MAKRO is the vehicle by which the modules developed under ASMB can be collected together into a single program. MAKRO treats the disk as an extension of memory, and source files exceeding available memory size can be assembled.

* CP/M is a registered trademark for Digital Research Incorporated, Pacific Grove, CA. 93950. All rights are reserved.

EDIT is a full spectrum string oriented text editor which includes all the features required to create or modify source programs for the MAKRO assembler. Source programs on an input disk file are paged into a dynamic memory buffer, modified and written out to the output disk file. Commands include block move or delete, string search or change, and disk file merge. A single command reformats the line-oriented source file created under ASMB to the free-form source input of MAKRO.

Programs created with the development systems must be off-loaded to the target processor. Facilities are provided to implement the offload as a direct transfer from memory, via a byte stream over a CPU port, or via COM or HEX files. An off loader for HEX files is provided. Direct support for off loading to the XDS line of TI support tools is included.

8.3.1 Company To Contact

Allen Ashley, Inc.
395 Sierra Madre Villa
Pasadena, Ca. 91107

Phone (213) 793-5748

Contact - Marketing

8.3.2 Product Offerings

8.3.2.1 *CP/M Bases Development Software For The TMS7000 Family*

The SYSTEM-TMS7 is a total software package for the development of TMS7000 code on a CP/M based small microprocessor system complete with documentation and utilities.

The following formats can be supplied:

IBM PC Morrow Micro Decision
TRS-80 (TRSDOS) Mod III
Osborne I
Kaypro II
North Star - CP/M
Micropolis Mod II
Xerox 820
Standard 8" CP/M format (SSSD)

8.4 SEEQ*: SELF-ADAPTIVE EEROM

The SEEQ 72720 is a full function single-chip microcomputer, fabricated in N-channel Silicon Gate technology, which contains a 2K x 8 5V nonvolatile electrically erasable (EEROM) program memory. The program memory can be erased and programmed via the processor itself during normal program execution or can be programmed under control as if it were a standard 5V EEROM memory component. The EEROM can easily be expanded off-chip using the processor's Full Expansion Mode. External EEROM can be programmed with the same instruction used to alter on-chip EEROM.

* SEEQ, DiTrace, and Silicon Signature are registered trademarks for SEEQ Technology Incorporated, San Jose, CA. 95131. All rights are reserved.

A security lock mechanism is implemented in EEROM memory which allows the user's program to inhibit external access to its proprietary program code. Once activated this lock can be reset only by an external EEROM block clear operation which erases the entire program memory contents.

As with other EEROM devices which SEEQ manufactures, the 72720 has DiTrace* and Silicon Signature* features to facilitate production testing tracking. Each device is encoded with detailed processing and testing results which are stored in a special EEROM memory as it passes through the manufacturing cycle. Also stored is an unalterable identification code which contains information such as mask revision and EEROM programming parameters.

An EEROM Microcomputer member of the TMS7000 family is desirable because the availability of a single-chip microcomputer with nonvolatile program memory which can be altered under process control makes possible the design of low cost products with many new features:

- Self adaptive code for machines that learn as they perform their tasks.
- In-Circuit reprogrammability to eliminate product disassembly for firmware updates.
- Remote reprogrammability to eliminate service calls for firmware updates.
- Internally stored product history including factory test results, product configuration, revision level, and service records.
- Stored initialization parameters to eliminate front panel switches and automatically configure product for one or many users.
- Product usage and error logging to simplify maintenance and pinpoint product failure modes.
- Code and data security to protect proprietary programs and confidential data.

8.4.1 Company To Contact

SEEQ Technology Incorporated
1849 Fortune Drive
San Jose, California 95131

Phone (408) 942-1990

Contact - Marketing

* SEEQ, DiTrace, and Silicon Signature are registered trademarks for SEEQ Technology Incorporated, San Jose, CA. 95131. All rights are reserved.

9. QUALITY AND RELIABILITY

9.1 INTRODUCTION

Quality and reliability (Q&R) performance of Texas Instruments Programmable Products, which includes the TMS7000 family, relies upon systematic input from:

- Our customers
- Our total manufacturing operation from front end wafer fabrication through final shipping inspection
- Product quality and reliability monitoring

Our customers' perception of quality must be the governing criteria for judging performance, and this concept is the basis for Texas Instruments Corporate Quality Policy, which is as follows:

"For every product or service we offer we shall define the requirements that solve the customers' problems, and we shall conform to those requirements without exception."

The Programmable Products Division (PPD) has established aggressive internal quality and reliability goals for the TMS7000 series but is even more concerned with receiving continuing customer feedback to ensure user satisfaction. Customer perceived performance is the most important PPD Q&R measurement, though it is the last input received for any product delivery cycle.

9.2 AVERAGE OUTGOING QUALITY

PPD continually inspects its products prior to shipment for electrical and mechanical compliance to Data Manual or Customer Specifications. Discrepancies are analyzed and corrective actions are taken to achieve our internally established goals, which are as follows:

	4Q83	4Q84	4Q85
Electrical Testing	800PPM	400PPM	200PPM
Visual/Mechanical Inspection	800PPM	400PPM	200PPM

More significantly, PPD is currently working very closely with several customers to achieve comparable levels of performance, as measured by the customer in a system environment.

9.3 NEW PRODUCT AND MAJOR CHANGE RELIABILITY QUALIFICATION TESTING

As part of PPD’s normal process of introducing new products or making major changes, plastic packaged devices must demonstrate satisfactory performance in the following environments:

- 1000 hours, 125 °C Dynamic Operating Life Test
- 1000 hours, 150 °C Storage
- 1000 hours, 85 °C/85% Relative Humidity, Biased
- 1000 cycles, -65 °C to 150 °C Temperature Cycling
- 96 hours, Autoclave @ 15 PSI
- Electro Static Discharge resistance

Additional tests may also be performed when appropriate.

Failure mechanisms are identified through failure analysis, and corrective actions are implemented to provide continual performance improvements.

Current PPD goals for key performance environments are as follows:

	4Q83	4Q84	4Q85
Dynamic Operating Life Test Derated to 55 °C, .5EV, 60%UCL	100 FITS	60 FITS	50 FITS
85 °C/85% Relative Humidity, Biased % Failures	.8	.5	.3
Temperature Cycling % Failures	.3	.25	.1

Wherever possible PPD encourages customer cooperation/participation in achieving qualification certification for PPD. Joint customer/PPD qualifications have been achieved and provide an efficient approach to demonstrating required reliability performance for both PPD and the user.

9.4 RELIABILITY MONITORING

After products are initially qualified, representative product samples are tested on a quarterly basis to measure and verify performance to goals for the key performance environments:

- Operating life test
- 85 °C/85% relative humidity, Biased
- Temperature cycling
- Autoclave

Analysis of failures is performed to determine the need for design or manufacturing improvements.

9.5 TMS7000 Family Reliability Performance

A summary of recent performance data on this family of devices demonstrates the following results:

TABLE 9-1 – DYNAMIC LIFE TEST

DEVICE	TEMP °C	SAMPLE SIZE	FAILURES	DEVICE HOURS (.5eV @ 55 °C) Millions
7040	150	42	0	2.23
7040	125	687	0	10.70
7040	85	194	0	.85
7041	125	182	1	4.09
7041	95	78	0	.53
70120	125	455	0	10.23
		1638	1	28.63
(60% UCL) Failure Rate: 71 FITS MTTF: 1614 Years				

TABLE 9-2 – ENVIRONMENTAL TESTS

ENVIRONMENT	SAMPLE SIZE	FAILURES	% FAILURES
Biased 85°/85% RH, 1000 Hours	275	3	1.1
150 C Storage, 1000 Hours (Mil Std 883B)	220	1	0.5
Temp Cycle, -65°/150°, 1000 Cycles (Mil Std 883B)	401	2	0.5
Auto Clave, 96 hours	262	0	0
Cycled Biased Humidity, 1000 Hours	38	1	2.6
100 Temp Cycles -65°/150° + 500 Hours 85°/85% RH + 500 Hours 125°Dynamic Life Test	45	0	0
Solderability (Mil Std 883B)	48	0	0
Lead Fatigue (Mil Std 883B)	5	0	0
Salt Atmosphere (Mil Std 8833B)	10	0	0

PPD is committed to satisfying your quality and reliability requirements and invites comments and questions in supporting customer needs.

10. GENERAL INFORMATION

10.1 TMS7000 FAMILY DEVICES

10.1.1 Prototype And Production Flow

The TMS7000 family of masked ROM microcomputers are semi-custom devices with ROM tailored to the customer's application requirements. The semi-custom nature of these devices requires a standard, defined interface between the customer and the factory in the production of TMS7000 devices with on-chip ROM. Figure 10-1 shows this standard prototype/production flow for customer ROM receipt. The following sequential steps refer to the blocks in Figure 10-1.

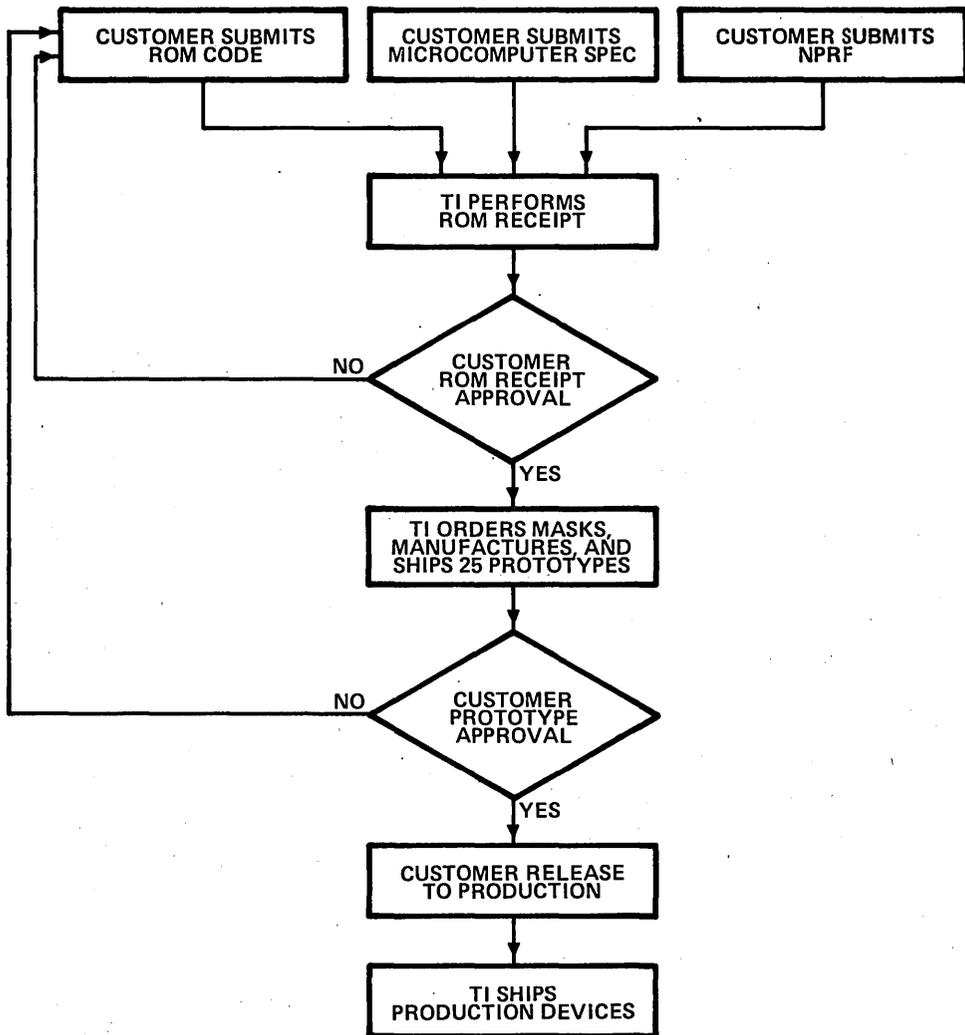


FIGURE 10-1 — PROTOTYPE AND PRODUCTION FLOW

- 1) For TI to accept the receipt of a customer ROM algorithm, each of the following three items must be received by the TI factory:
 - A. The customer completes and submits a New Products Release Form (NPRF) to TI describing the custom features of the device (e.g., customer information, prototype and production quantities and dates, any exceptions to standard electrical specifications, customer part numbers and symbolization, package type, etc). The NPRF is available from TIs' field sales engineers.
 - B. The customer submits a copy of the specification for the microcomputer in their system, including the functional description and electrical specification (including absolute maximum ratings, recommended operating conditions, and timing values).
 - C. When the customer has completed code development and after verification of this code with the development system, the standard TMS7000 tagged object code is submitted to the TI factory on an acceptable media for processing. These include:
 - Single-sided, single density floppy disks formatted by the 990/4 TXDX floppy disk operating system or the TX990 conversion utilities on hard-disk based AMPL systems.
 - Double-sided, double density floppy disks formatted by the TMAM9000 AMPLUS operating system.
 - Bulk Data Transfer from a Texas Instruments Regional Technology Center (RTC) to the TI Wilcrest facility to the DX990.
 - Coded EPROM devices (i.e., 2516, 2532, 2716, 2732)

The mask ROM codes should be sent to:

Texas Instruments Microcomputer Division
P.O. Box 1443, MS 6435
Houston, TX 77001

- 2) Code review and ROM receipt is performed on the customer's code and a manufacturing ROM code number is assigned to the customers algorithm. All future correspondence should indicate this number. The ROM receipt procedure reads the ROM code information, processes it, and reproduces the customers tagged ROM object code which is returned to the customer for verification of correct ROM receipt.
- 3) The customer then verifies that the ROM code received by TI is correct and that no information was misinterpreted in the transfer. The customer will then return written confirmation of correct ROM receipt verification or will re-submit the code for processing.
- 4) TI generates the prototype photomask, processes, manufactures, and tests 25 prototype devices for shipment to the customer. Limited quantities in addition to the initial 25 prototypes may also be purchased by the customer for use in customer evaluation.

NOTE

Texas Instruments recommends that prototype devices not be used in production system since their expected end-use failure rate is undefined but is predicted to be greater than standard qualified production.

All prototype devices are shipped against the following disclaimer:

"It is understood that, for expediency purposes, the initial 25 prototype devices (and any additional prototype devices purchased) were assembled on a prototype (i.e., non-production qualified) manufacturing line whose reliability has not been characterized. Therefore, the anticipated inherent reliability of these devices cannot be expressly defined."

- 5) The customer verifies the operation of these prototypes in the system and responds with either written customer prototype approval or disapproval.
- 6) With customer algorithm approval, the ROM code is released to production and TI will begin shipment of production devices according to customer's final specification and order requirements.

Two leadtimes are quoted in reference to the preceding flow:

- Prototype leadtime — elapsed time from the receipt of written ROM receipt verification to the delivery of 25 prototype devices.
- Production leadtime — elapsed time from the receipt of written customer prototype approval to delivery of production devices.

For the latest TMS7000 family leadtimes, contact the nearest TI field sales office.

10.1.2 Device Prefix Designators

To provide expeditious system evaluations by customers during the product development cycle, Texas Instruments assigns a prefix designator with four options: TMS, TMP, TMX, and SE.

TMX, TMP, and TMS are representative of the evolutionary stages of product development from engineering prototypes through fully qualified production devices. Figure 10-2 depicts this evolutionary development flowchart.

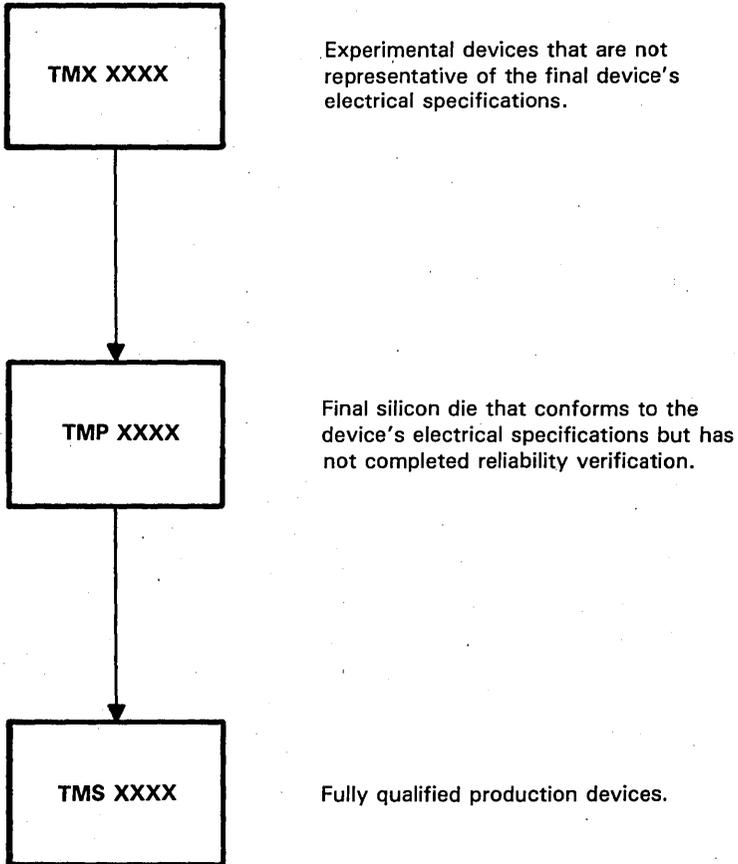


FIGURE 10-2 — DEVELOPMENT FLOWCHART

TMX devices are shipped against the following disclaimer:

- 1) Experimental product and its reliability has not been characterized.
- 2) Product is sold "as is".
- 3) Product is not warranted to be exemplary of final production version if or when released by Texas Instruments.

TMP devices are shipped against the following disclaimer:

- 1) Customer understands that the product purchased hereunder has not been fully characterized and the expectation of reliability cannot be defined; therefore, Texas Instruments standard warranty refers only to the device's specifications.

TMS devices have been fully characterized and the quality and reliability of the device has been fully demonstrated. Texas Instruments' standard warranty applies.

The SE prefix designation is given to the system evaluator devices used for prototyping purposes. Currently this designation applies only to the SE70P161 member of the TMS7000 family.

SE devices are shipped against the following disclaimer:

- 1) System evaluators and development tools are for use only in a prototype environment and not warranted for sale in the customer's application.

10.1.3 Clock Options

There are two clock options available on the NMOS TMS7000 family devices (TMS7000, TMS7020, TMS7040, TMS70120, TMS7001, TMS7041) for converting the external frequency to the internal machine cycle frequency, called Phi (ϕ). They are termed the divide by two (/2) or the divide by four (/4) clock options. These are mask options which means the option is finalized at the time of manufacture and is NOT changeable by software or hardware. If the divide by two clock option is chosen, the external frequency divided by 2 is the internal machine cycle frequency. A 5 MHz crystal would generate an internal machine cycle of 2.5 MHz with the divide by two option. If the divide by four clock option is chosen, the external clock is divided by 4 so that the same 5 MHz crystal would generate an internal machine cycle of 1.25 MHz. (In this example a 10 MHz crystal would be used to get a 2.5 MHz internal machine cycle.)

The divide by two clock option is recommended for use with crystals and the divide by four clock option use either a crystal or another external clock source. It is not recommended to use an external source to drive a divide by two device. If a crystal is used it is connected between pins XTAL1 and XTAL2. To improve the crystal waveform, 15 pF capacitors are connected between XTAL1 and ground and XTAL2 and ground. If an external clock source is used, it is connected to XTAL2 (also called CLKIN), while XTAL1 is left floating.

The selection of the divide by clock option for TMS7000 family members with on-chip ROM is designated by the customer in the New Products Release Form (see Section 10.1.1), while standard TMS7000 family members without on-chip ROM have this designation as part of their part number (see Section 10.1.5.2).

10.1.4 Reserved ROM Locations

TMS7000 family members with on-chip ROM have the first 6 byte locations reserved for TI use. Therefore these locations must not be used by the customer in the development of the ROM code. The user must remember this when performing development using the XDS emulator, the EVM, the SE70P161, or a TMS7000 family member without on-chip ROM. Table 10-1 depicts the valid ROM starting address for the common family members.

TABLE 10-1 – VALID ROM START ADDRESSES

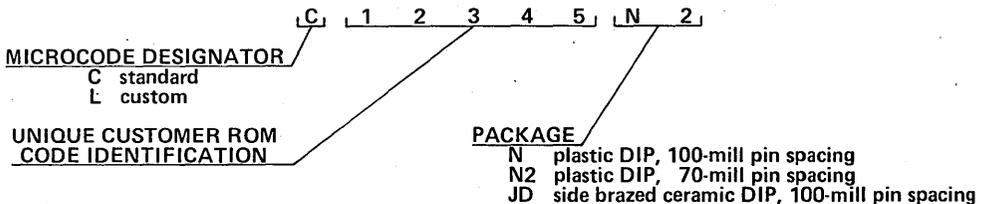
FAMILY MEMBER	ROM SPACE	VALID START ADDRESS
7020,70C20	2K Bytes	>F806
7040,7041,70C40	4K Bytes	>F006
70120	12K Bytes	>D006

10.1.5 Ordering Information

TMS7000 family devices can be divided into two categories for ordering information and symbolization, with the distinction being made on the presence (or absence) of on-chip ROM.

10.1.5.1 TMS7000 Family Members With On-Chip ROM

TMS7000 family members with on-chip ROM are semi-custom devices with the ROM mask programmed to the customer's requirements. These devices follow the prototyping and production flow outlined in Section 10.1.1. Since they are semi-custom devices, they receive a distinct identification as follows:



All packages are currently 40-pin, 600-mil dual-in-line packages (DIP). Refer to section 10.1.6 for complete package dimensions.

There are two types of symbolization for TMS7000 family members with on-chip ROM. These are:

- 1) TI standard symbolization
- 2) TI standard symbolization with customer part number.

	(a)	(b)	(c)	MEANING OF MARKINGS
line 1:		C12345N2	DBUA8327	(a) Texas Instruments trademark
line 2:	(d) ©1981 TI	©1983 TI	(f)	(b) Customer's ROM code
line 3:	(e) 24655	(g) Philippines		(c) Tracking mark and date code
				(d) TI microcode copyright
				(e) Lot code
				(f) Copyright of ROM code
				(g) Assembly site

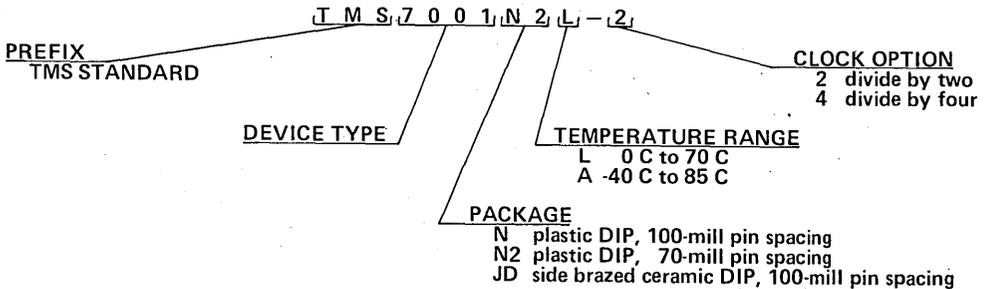
FIGURE 10-3 – TI STANDARD SYMBOLIZATION

	(a)	(g)		MEANING OF MARKINGS
line 1:		123456789012		(a) Texas Instruments trademark
line 2:	(d) ©1981 TI	(b) C12345N2	(c) DBUA8327	(b) Customer's ROM code
line 3:	(e) 24655	©1983 TI	(f)	(c) Tracking mark and date code
line 4:	(h) Philippines			(d) TI microcode copyright
				(e) Lot code
				(f) Copyright of ROM code
				(g) Customer part number
				(h) Assembly site

FIGURE 10-4 – TI STANDARD SYMBOLIZATION WITH CUSTOMER PART NUMBER

10.1.5.2 TMS7000 Family Members Without On-Chip ROM

TMS7000 family members without on-chip ROM are standard device types, and therefore have a standard identification as follows:



All packages are currently 40-pin, 600-mill dual-in-line packages (DIP). Refer to section 10.1.6 for complete package dimensions.

Examples of common TMS7000 family members without on-chip ROM are:

TMS7000NL-2
TMS7000NL-4

TMS7001NL-4
TMS7001N2L-2

The standard symbolization for these devices is shown in Figure 10-5.

	(a)	(b)	MEANINGS OF MARKINGS
line 1:		TMS7001NL-2	(a) Texas Instruments trademark
line 2:	(d) © 1981TI	DBUA8327 (c)	(b) Standard device number (c) Tracking mark and date code
line 3:	(e) 24655	(f) Philippines	(d) TI microcode copyright (e) Lot code (f) Assembly site

FIGURE 10-5 — TI STANDARD SYMBOLIZATION FOR DEVICES WITHOUT ON-CHIP ROM

10.1.6 Mechanical Data

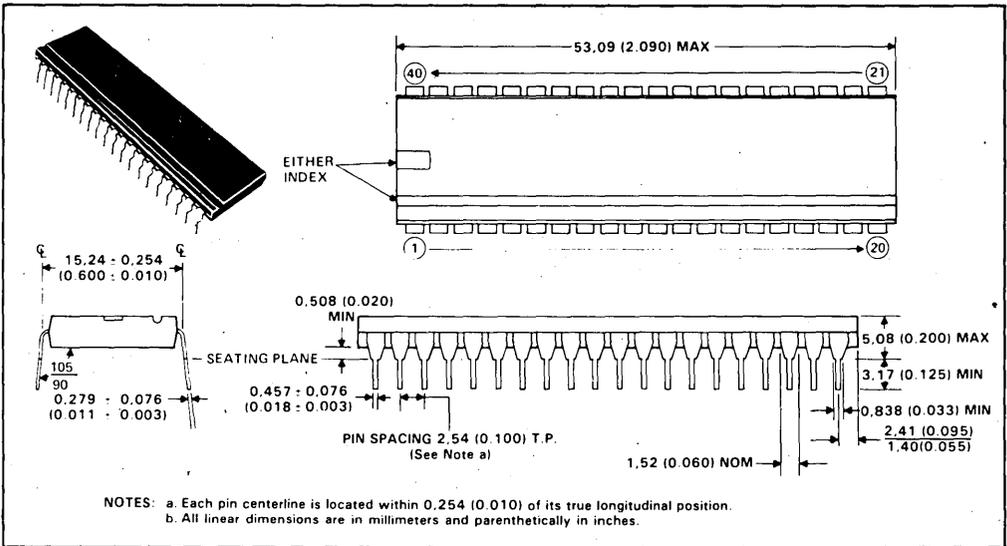
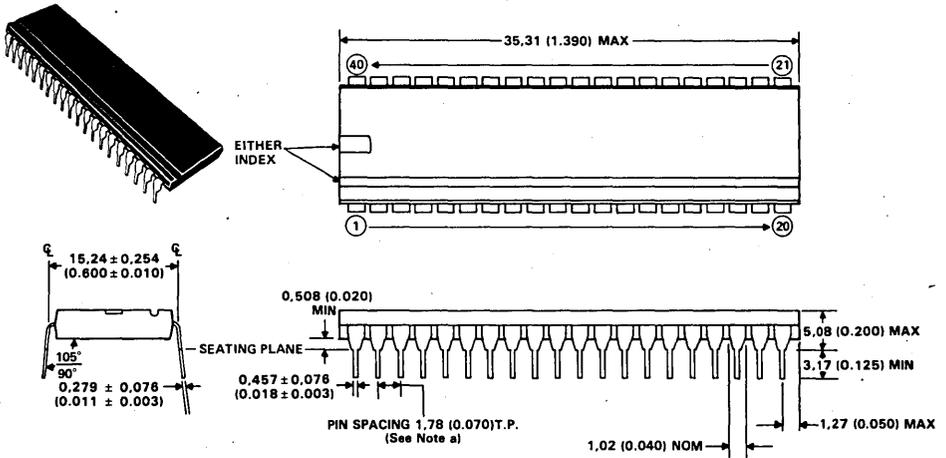


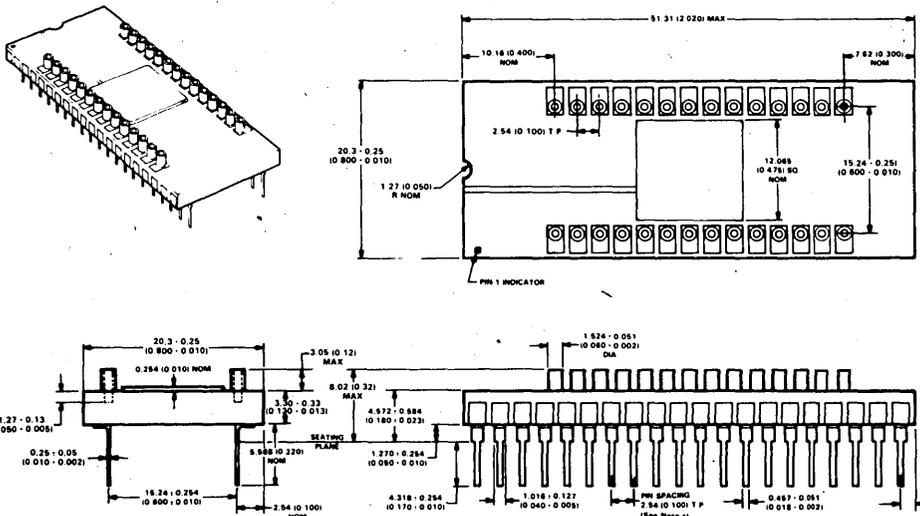
FIGURE 10-6 — 40 PIN PLASTIC PACKAGE, 100 MIL PIN SPACING
(TYPE N PACKAGE SUFFIX).

40-PIN PLASTIC PACKAGE (0.070 PIN SPACING)



NOTES: a. Each pin centerline is located within 0.254 (0.010) of its true longitudinal position.
b. All linear dimensions are in millimeters and parenthetically in inches.

FIGURE 10-7 — 40 PIN PLASTIC PACKAGE, 70 MIL PIN SPACING
(TYPE N2 PACKAGE SUFFIX).



Notes: a. Each pin centerline is located within 0.127 (0.005) of its true longitudinal position.
b. All dimensions are in millimeters and parenthetically in inches.

FIGURE 10-8 — 40 PIN CERAMIC PACKAGE, 100-MIL PIN SPACING
(TYPE JD PACKAGE SUFFIX).

10.2 DEVELOPMENT SUPPORT TOOLS

10.2.1 CrossWare

PART NUMBER	DESCRIPTION	OPERATING SYSTEM
TMDS7040113-21	TI 990 DSDD	TX-5
TMDS7040123-06	TI 990 T50	DX10
TMDS7040123-08	TI 990 Tape	DX10
TMDS7040123-10	TI 990 DS10	DX10
TMDS7040123-22	TI 990 CD1400	DX10
TMDS7040133-03	TI 990 SSSD	TX-4
TMDS7040210-08	DEC VAX Tape	VMS
TMDS7040310-08	IBM Mainframe	MVS
TMDS7040320-08	IBM Mainframe	CMS

10.2.2 XDS Hardware

PART NUMBER	XDS MODEL #	TMS7000 FAMILY SUPPORT
TMDS7062210	Model 22	TMS7020, TMS7040, TMS7041, TMS70120

10.2.3 Evaluation Modules

PART NUMBER	DEVICES SUPPORTED
RTC/EVM7000N-1	TMS7000, TMS7001, TMS7020, TMS7040, TMS7041, TMS70120
RTC/EVM7000C-1	TMS70C20, TMS70C40

10.3 TMS7000 FAMILY DOCUMENTATION

DOCUMENT NUMBER	DOCUMENT
-----------------	----------

TMS7000 FAMILY DATA MANUALS:

MP008A	TMS7000/7020/7040 8-BIT MICROCOMPUTER DATA MANUAL
SPNF002	TMS7000 PROGRAMMERS POCKET REFERENCE CARD
SPNV002	TMS7500 DATA ENCRYPTION DEVICE PRODUCT DESCRIPTION
SPNS004	TMS7500 DATA ENCRYPTION DEVICE PRELIMINARY DATA MANUAL

TMS7000 DATA SHEETS AND BROCHURES:

SPNS005	TMS7007 DATA SHEET
SPNB001	TMS7000 SALES BROCHURE
SPDS002	7000 XDS MODEL 22 DATA SHEET
SPDV002	7000 XDS MODEL 22 BROCHURE
SPNS006	SE70P161 DATA SHEET

TMS7000 FAMILY MICROCODE SUPPORT:

SPNV001	TMS7000 CUSTOM MICROCODING PRODUCT DESCRIPTION
MP061	TMS7000 FAMILY MICROARCHITECTURE USER'S GUIDE
SPNU001	TMS7000 MICROCODE DEVELOPMENT GUIDE
MP457	TMS7000 FAMILY MICROASSEMBLER USER'S GUIDE
MP459	TMS7000 MICROPROGRAMMERS REFERENCE CARD

TMS7000 FAMILY SOFTWARE SUPPORT:

SPNU002B	TMS7000 ASSEMBLY LANGUAGE PROGRAMMER'S GUIDE
MPB45	TMS7000 SOFTWARE DEVELOPMENT SYSTEM INTRODUCTION GUIDE
MPB52	TMS7000 SOFTWARE DEVELOPMENT SYSTEM INSTALLATION GUIDE
MPB10	TMS7000 IBM CROSS SUPPORT REFERENCE GUIDE
MPB53	TMS7000 VAX CROSS SUPPORT REFERENCE GUIDE
MPB38	TMS7000 EMULATOR INSTALLATION AND OPERATION GUIDE
MPB37	TMS7000 EMULATOR COMMAND LANGUAGE GUIDE

TMS7000 FAMILY APPLICATION NOTES:

SPNA001	INTERFACING TMS7000 TO PERIPHERAL AND MEMORY DEVICES
SPNA002	TMS7000 BUS ACTIVITY CHART
SPNA003	TMS7000 KEYBOARD INTERFACE APPLICATION REPORT
SPNU003	8051 — TMS7041 SYSTEM CONVERSION USER'S GUIDE

10.4 WORLDWIDE REGIONAL TECHNOLOGY CENTERS (RTC)

Atlanta
Texas Instruments, Inc.
3300 N.E. Expressway
Building 8
Atlanta, GA 30341
(404) 452-4682
(404) 452-4686 Hotline

Boston
Texas Instruments, Inc.
400-2 Totten Pond Rd.
Waltham, MA 02154
(617) 890-6671
(617) 890-4271 Hotline

Chicago
Texas Instruments, Inc.
515 W. Algonquin Rd.
Arlington Heights, IL
(312) 640-2909
(312) 228-6008 Hotline

Northern California
Texas Instruments, Inc.
5353 Betsy Ross Drive
Santa Clara, CA 95054
(408) 748-2220
(408) 980-0305 Hotline

Southern California
Texas Instruments, Inc.
17981 Cartwright Rd.
Irvine, CA 92714
(714) 660-8140
(714) 660-8164 Hotline

Dallas
Texas Instruments, Inc.
10001 E. Campbell Road
Richardson, TX 75081
(214) 680-5066
(214) 680-5096 Hotline

Bedford, England
Texas Instruments, LTD
Manton Lane
Bedford, MK41 7PA
0234 223000

Freising, West Germany
Texas Instruments Deutschland GmbH
Haggertystr. 1
8050 Freising
08161 800

Tokyo, Japan
Texas Instruments Japan
Aoyama Fuji Bldg.
6-12, Kita Aoyama 3 Chome
03-498-2111

Hannover, West Germany
Texas Instruments Deutschland GmbH
Kirchhorsterstr Str 2
3000 Hannover 51
0511/648021

WORLDWIDE REGIONAL TECHNOLOGY CENTERS (Concluded)

Stockholm, Sweden
Nordic Technology Center
Texas Instruments
Sverigefilialen
Norra, Hamnvagen 3
Box 39103
S-100 54 Stockholm, Sweden
8-235480

Cedex, France
Paris Technology Center
Texas Instruments
8-10 Avenue Morane Saulnier
Boite Postale 67
78141 Velizy-Villacoublay
Cedex, France
39-46-9712

Rieti, Italia
Rieti Technology Center
Semiconduttori
Italia S.P.A.
Viale Delle Scienze, 1
02015 Cittaducale
Rieti, Italia
746-6941

**APPENDIX A
INSTRUCTION EXECUTION TIMES**

A.1 INSTRUCTION EXECUTION TIMES

Each instruction of a TMS7000 family device requires from 1 to 4 bytes of program space. Execution time varies from 4 to 48 machine cycles with most instructions requiring less than 9 cycles to complete. Table A-1 summarizes the byte and machine cycle counts for each instruction. A variety of addressing modes are provided for each instruction, and the byte and cycle count for each is indicated. The form of the entries is byte count/cycle count. Table A-2 is the instruction opcode set.

TABLE A-1 — INSTRUCTION EXECUTION TIMES

OPERATION		ADDRESSING MODES								OTHER	NOTES
		A	B	RF	PF	@ lab	* RF	@ lab (B)			
ADC	B,--	1/5									
	RF,--	2/8	2/8	3/10							
	%iop,--	2/7	2/7	3/9							
ADD	B,--	1/5									
	RF,--	2/8	2/8	3/10							
	%iop,--	2/7	2/7	3/9							
AND	B,--	1/5									
	RF,--	2/8	2/8	3/10							
	%iop,--	2/7	2/7	3/9							
ANDP	A,--				2/10						
	B,--				2/9						
	%iop,--				3/11						
BTJO	B,--	2/7									(1)
	RF,--	3/10	3/10	4/12							
	%iop,--	3/9	3/9	4/11							
BTJOP	A,--				3/11						(1)
	B,--				3/10						
	%iop,--				4/12						
BTJZ	B,--	2/7									(1)
	RF,--	3/10	3/10	4/12							
	%iop,--	3/9	3/9	4/11							
BTJZP	A,--				3/11						(1)
	B,--				3/10						
	%iop,--				4/12						
BR	--					3/10	2/9	3/12			
CALL	--					3/14	2/13	3/16			
CLR	--	1/5	1/5	2/7							
CLRC	--									1/6	
CMP	B,--	1/5									
CMPA	RF,--	2/8	2/8	3/10							
	%iop,--	2/7	2/7	3/9							
	--					3/12	2/11	3/14			
DAC	B,--	1/7									
DEC	RF,--	2/10	2/10	3/12							
	%iop,--	2/9	2/9	3/11							
	--	1/5	1/5	2/7							
DECD	--	1/9	1/9	2/11							
DINT	--									1/5	
DJNZ	--	2/7	2/7	3/9							(1)
DSB	B,--	1/7									
	RF,--	2/10	2/10	3/12							
	%iop,--	2/9	2/9	3/11							
EINT	--									1/5	
IDLE	--									1/6+	
INC	--	1/5	1/5	2/7							
INV	--	1/5	1/5	2/7							
JMP	--									2/7	
Jcnd	lab									2/5	(1,2)
LDA	--					3/11	2/10	3/13			
LDSP	--									1/5	
MOV	A,--		1/6	2/8							
	B,--			2/7							
	RF,--	2/8	2/8	3/10							
	%iop,--	2/7	2/7	3/9							
	%iop(B),--			4/15							
MOVD	%iop,--			4/17							
	RF,--			3/14							
	A,--				2/10						
	B,--				2/9						
	%iop,--				3/11						
MOVP	PF,--	2/9	2/8								

TABLE A-1 — INSTRUCTION EXECUTION TIMES (CONTINUED)

OPERATION		ADDRESSING MODES							OTHER	NOTES
		A	B	RF	PF	@ lab	* RF	@ lab (B)		
MPY	B,---	1/44								
	RF,---	2/47	2/47	3/49						
	%iop,---	2/46	2/46	3/48						
NOP	---								1/4	
OR	B,---	1/5								
	RF,---	2/8	2/8	3/10						
	%iop,---	2/7	2/7	3/9						
ORP	A,---				2/10					
	B,---				2/9					
	%iop,---				3/11					
POP	---	1/6	1/6	2/8						
POP ST	---								1/6	
PUSH	---	1/6	1/6	2/8						
PUSH ST	---								1/6	
RETI	---								1/9	
RETS	---								1/7	
RL	---	1/5	1/5	2/7						
RLC	---	1/5	1/5	2/7						
RR	---	1/5	1/5	2/7						
RRC	---	1/5	1/5	2/7						
SBB	B,---	1/5								
	RF,---	2/8	2/8	3/10						
	%iop,---	2/7	2/7	3/9						
SETC	---								1/5	
STA	---					3/11	2/10	3/13		
STSP	---								1/6	
SUB	B,---	1/5								
	RF,---	2/8	2/8	3/10						
	%iop,---	2/7	2/7	3/9						
SWAP	---	1/8	1/8	2/10						
TSTA	---								1/6	
TSTB	---								1/5	
TRAP n	---								1/14	
XCHB	---	1/6	1/6	2/8						
XOR	B,---	1/5								
	RF,---	2/8	2/8	3/10						
	%iop,---	2/7	2/7	3/9						
XORP	A,---				2/10					
	B,---				2/9					
	%iop,---				3/11					

NOTES:

- (1) Add 2 to cycle count if branch is taken.
- (2) Conditional Jump Instructions (see Table 3-3).

NOTATION: Data Form — number of bytes/number of internal clock cycles.

- A A register
- B B register
- RF Register File number
- PF Peripheral File number
- lab Label
- iop Immediate operand

INSTRUCTION - OP CODE SET

	SINGLE OPERAND			DUAL OPERAND								PERIPHERAL				EXTENDED			OTH-ER	STATUS WORD			
	A	B	Rn	A,B	BA	Rn,A	%n,A	Rn,B		%nRn	A,Rn	B,Rn	A,Pn	Pn,A	B,Pn	Pn,B	%n,Pn	DIRECT	INDIRECT	INDEXED	OTHER	COND. BITS	INT. EN. BIT
ADC				69	19	29	39	59	49	79												X	
ADD				68	18	28	38	58	48	78												X	
AND				63	13	23	33	53	43	73												X	
ANDP												83	93	A3									
BTJO				66	16	26	36	56	46	76												X	
BTJOP												86	96	A6								X	
BTJZ				67	17	27	37	57	47	77												X	
BTJZP												87	97	A7								X	
BR																	8C	9C	AC				
CALL																	9E	9E	AE				
CLR	B5	C5	D5																			X	
CLRC																					80		X
CMP				6D	1D	2D	3D	5D	4D	7D												X	
CMPA																		8D	9D	AD			X
DAC				6E	1E	2E	3E	5E	4E	7E												X	
DEC	B2	C2	D2																				
DECD	B8	C8	D8																			X	
DINT																					06	X	X
DJNZ	BA	CA	DA																				
DSB				6F	1F	2F	3F	5F	4F	7F												X	
EINT																					05	X	X
IDLE																					01		
INC	B3	C3	D3																			X	
INV	B4	C4	D4																			X	
JMP																					E0		
JC/JHS																					E3		
JN/JLT																					E1		
JNC/JL																					E7		
JNZ/JNE																					E6		
JP/JGT																					E4		
JPZ/JGE																					E5		
JZ/JEQ																					E2		
LDA																	8A	9A	AA			X	
LDSP																					0D		X
MOV			C0	62	12	22	32	52	42	72	D0	D1									X		
MOVD																	88	98	A8			X	
MOVV													82	80	92	91	A2					X	
MPY				6C	1C	2C	3C	5C	4C	7C												X	
NOP																					00		
OR				64	14	24	34	54	44	74												X	
ORP												84	94	A4								X	
POP	B9	C9	D9																		08	X	
PUSH	B8	C8	D8																		0E	X	
RETI																					0B		
RETS																					0A		
RL	BE	CE	DE																			X	
RLC	BF	CF	DF																			X	
RR	BC	CC	DC																			X	
RRC	BD	CD	DD																			X	
SBB				6B	1B	2B	3B	5B	4B	7B												X	
SETC																					07	X	
STA																	8B	9B	AB			X	
STSP																					09		X
SUB			6A	1A	2A	3A	5A	4A	7A													X	
SWAP	B7	C7	D7																			X	
TSTA																					80	X	
TSTB																					C1	X	
TRAP																					E8-FF		
XCHB	B6	D6																				X	
XOR				65	15	25	35	55	45	75												X	
XORP												85	95	A5								X	

APPENDIX B TMS7000 BUS ACTIVITY CHART

The following tables describe the information present on the address and data buses during each cycle of each instruction. This information is useful to:

- 1) Document the contents of the address and data buses and control pins on a cycle by cycle basis.
- 2) Calculate instruction execution times.
- 3) Compare actual results to expected results.
- 4) Gain a better understanding of microcomputer operation.

The information on the address and data buses, as well as the control pins, can be externally monitored only when the device is in either the full expansion, peripheral expansion, microprocessor, or system emulator modes.

Because the TMS7000 is implemented using a microcoded architecture, the microcode that fetches the instructions and their data can be shared by many instructions. This allows the instruction set to be grouped according to the types of operands the instructions require and how they are fetched. The instruction set bus activity chart will be presented according to the different instruction groups supported. Each instruction group is based on one of the addressing modes supported by the TMS7000. The different addressing modes supported by the TMS7000 are as follows:

- 1) Double Operand Functions (DOPFUN). These instructions require 2 operands for execution. The instructions in this group are: ADC, SUB, SBB, MOV, AND, OR, XOR, BTJO, BTJZ, ADD, CMP, DAC, DSB, and MPY.
- 2) Miscellaneous Functions (MISCFUN). These instructions need no operands because the instruction function is implied in the opcode. Contained in this group are: NOP, IDLE, EINT, DINT, SETC, POP ST, STSP, RETS, RETI, LDSF, and PUSH SP.
- 3) Long Addressing Functions (LAFUN). This group of instructions requires a sixteen bit address which is used to address the entire 64K address range of the TMS7000. The instructions in this group are: LDA, STA, BR, CMPA, and CALL.
- 4) Single Operand Functions-Special (SOPFUNS). These instructions need 1 operand for execution. The instructions in this group are: DEC, INC, INV, CLR, XCHB, SWAP, MOV A,B, MOV A, RN, MOV B, RN, TSTA/CLRC, and TSTB.
- 5) Single Operand Functions - Normal(SOPFUNN). These instructions need one operand for execution. Because of the way CPU control is implemented and the number of supported single operand instructions, two groups of single operand functions are needed. The instructions that belong to this group are: PUSH, POP, DJNZ, DECD, RR, RRC, RL, and RLC.
- 6) Double Operand Functions, Peripheral (DOPFUNP). These instructions require two operands and interact with the TMS7000's peripheral ports. The instructions are: MOVP, ANDP, ORP, XORP, BTJOP, and BTJZP.

- 7) Move Double (MOVD). MOVD moves a register pair to a register pair.
- 8) Relative Jumps. These conditional and unconditional jumps alter program flow by adding or subtracting an 8 bit value from the program counter.
- 9) Traps (TRAP). This group of instructions is used to perform subroutine calls.

Each instruction's execution consists of three basic parts: instruction acquisition, operand addressing (addressing modes), and functional operation on the operands (functional modes). To construct the cycles required to execute any instruction, start with the instruction acquisition function as shown in Table B-2. These three cycles are needed to fetch the instruction opcode, increment the program counter, and pre-fetch the B register. Next, construct the number of addressing mode cycles needed to fetch the instruction's operands by looking up which instruction group the instruction belongs to in Table B-1 and then referencing that table (Tables B-3 through B-11). Each table consists of two parts: the addressing mode and the functional part. After the operand addressing cycles are found, the second half of the table will detail the cycles involved with the functional part of instruction execution. Add all these cycles together to obtain the bus activity present during that instruction's execution. As an example of this, Figure B-1 shows the execution steps involved with the instruction "ADD R5,R6".

ADDR MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
ALL INSTRUCTIONS	1	Opcode address	Irrelevant data	R
	2	Opcode address	Instruction Opcode	R
	3	B reg. address	B reg. contents	R

The first two cycles fetch the ADD instruction's opcode and increment the program counter. The third state prefetches the B register to speed up instructions that reference the B register. The addressing mode is entered next. This information comes from Table B-2.

ADDR MODE IS Rn, Rn	CYCLE	ADDRESS BUS	DATA BUS	R/W
MOV,AND,OR,XOR,BTJO, DAC,ADD,SUB,SBB,MPY, BTJZ,CMP,DSB	1	Opcode Address + 1	Irrelevant Data	R
	2	Opcode Address + 1	Rsrc address	R
	3	Rsrc address	Rsrc data	R
	4	Opcode address + 2	Irrelevant data	R
	5	Opcode address + 2	Rdest address	R
	6	Rdest address	Operand data	R

FIGURE B-1 – ADD R5, R6 EXAMPLE

The ADD instruction is a double operand function requiring two operands. Double operand functions are described in Table B-3. Cycles 1 and 2 of this mode read the "R5" operand address. Cycle 3 reads the register contents. Note that the internal register read (or a write) is a one cycle operation. All other reads/writes are two cycles long, requiring that the address bus be held stable for two complete machine cycles. Each machine cycle corresponds to one clock period of the CLKOUT signal (pin 2), starting with the rising edge of this signal. Cycles 4 and 5 read the Rdest address, "R6", which is where the resultant value will be left. Cycle 6 reads the contents of register R6. At this point, both operands are inside the CPU and the indicated function can be performed.

MACROINSTRUCTION	CYCLE	ADDRESS BUS	DATA BUS	\overline{RW}
ADD	1	Register address	Register data	W

The functional portion of the "ADD" instruction is detailed in the second half of Table B-3. The second half of each table describes the functional portion for each instruction group. Once both operands are inside the CPU, only one cycle is needed to perform the add operation. The result is written back to register R6 during this cycle. A total of 10 cycles was required to perform an "ADD R5, R6".

The instruction acquisition sequence is common to all instructions, and therefore is presented separately for clarity. Each addressing mode will be presented followed by the functional portion of each instruction's execution.

The tables are arranged in the following order:

<u>TABLE</u>	<u>TABLE CONTENTS</u>
B-1	Alphabetical Index Into Instruction Groups
B-2	Instruction Acquisition Functions
B-3	Double Operand Instructions
B-4	Miscellaneous Instructions
B-5	Long Addressing Instructions
B-6	Single Operand Instructions - Special
B-7	Single Operand Instructions - Normal
B-8	Double Operand Instructions - Peripheral
B-9	Move Double Instructions
B-10	Relative Jump Instructions
B-11	Trap Instructions
B-12	Reset Function

Each table will consist of two parts: the addressing mode portion and the functional portion.

Table B-1 is provided as an index into the rest of the tables. Table B-1 lists all standard TMS7000 instructions in alphabetical order with the corresponding addressing mode.

Each table indicates whether a read or a write is performed that cycle. The \overline{RW} signal will be high for reads and low (logic zero level) for writes. The memory control signals, ALATCH and ENABLE, are asserted during both reads and writes. Reference the memory interface timing diagrams contained in Section 4 of this manual for further information.

TABLE B-1 — ALPHABETICAL INDEX INTO INSTRUCTION GROUPS

INSTR	ADDR MODE	TABLE #	FUNCTION
ADC	DOPFUN	3	ADD WITH CARRY
ADD	DOPFUN	3	ADD
AND	DOPFUN	3	AND
ANDP	DOPFUNP	8	AND VALUE WITH PERIPHERAL PORT
BTJO	DOPFUN	3	TEST BIT AND JUMP IF ONE
BTJOP	DOPFUNP	8	TEST PERIPHERAL BIT & JUMP IF ONE
BTJZ	DOPFUN	3	TEST BIT AND JUMP IF ZERO
BTJZP	DOPFUNP	8	TEST PERIPHERAL BIT & JUMP IF ZERO
BR	LAFUN	5	LONG BRANCH
CALL	LAFUN	5	SUBROUTINE CALL
CLR	SOPFUNS	6	CLEAR
CLRC	SOPFUNS	6	CLEAR STATUS CARRY BIT
CMP	DOPFUN	3	COMPARE VALUE
CMPA	LAFUN	5	COMPARE VALUE WITH A REGISTER
DAC	DOPFUN	3	DECIMAL ADD WITH CARRY
DEC	SOPFUNS	6	DECREMENT VALUE
DECD	SOPFUNN	7	DECREMENT DOUBLE REGISTER PAIR
DINT	MISCFUN	4	DISABLE INTERRUPTS
DJNZ	SOPFUNN	7	DECREMENT AND JUMP IF NOT ZERO
DSB	DOPFUN	3	DECIMAL SUBTRACT
EINT	MISCFUN	4	ENABLE INTERRUPTS
IDLE	MISCFUN	4	IDLE (PC IS HELD UNCHANGED)
INC	SOPFUNS	6	INCREMENT
INV	SOPFUNS	6	INVERT
JMP	REL JUMPS	10	UNCONDITIONAL RELATIVE JUMP
Jcnd	REL JUMPS	10	CONDITIONAL RELATIVE JUMPS (JN/JLT,JZ/JEQ, JL,JC/JHS,JP/JGT,JPZ/JGE,JNZ/JNE,JNC)
LDA	LAFUN	5	LOAD A REGISTER FROM LONG ADDRESS
LDSP	MISCFUN	4	LOAD STACK POINTER
MOV	DOPFUN	3	MOVE A DATA VALUE
	SOPFUNS	6	
MOVD	MOVD	9	MOVE A 16 BIT VALUE TO REG. PAIR
MOVP	DOPFUNP	8	MOVE A DATA VALUE TO/FROM PORT
MPY	DOPFUN	3	MULTIPLY TWO 8 BIT VALUES
NOP	MISCFUN	4	NO OPERATION
OR	DOPFUN	3	OR TWO VALUES TOGETHER
ORP	DOPFUNP	8	OR PORT VALUE WITH ANOTHER VALUE
POP	SOPFUNN	7	POP A VALUE OFF THE STACK
POP ST	MISCFUN	4	POP STACK VALUE INTO STATUS REG.
PUSH	SOPFUNN	7	PUSH A VALUE ONTO THE STACK
PUSH ST	MISCFUN	4	PUSH STATUS REGISTER ONTO STACK
RETI	MISCFUN	4	RETURN FROM INTERRUPT
RETS	MISCFUN	4	RETURN FROM SUBROUTINE
RL	SOPFUNN	7	ROTATE LEFT
RLC	SOPFUNN	7	ROTATE LEFT THROUGH CARRY BIT
RR	SOPFUNN	7	ROTATE RIGHT
RRC	SOPFUNN	7	ROTATE RIGHT THROUGH CARRY BIT
SBB	DOPFUN	3	SUBTRACT WITH BORROW
SETC	MISCFUN	4	SET CARRY BIT

TABLE B-1 – ALPHABETICAL INDEX INTO INSTRUCTION GROUPS (CONTINUED)

INSTR	ADDR MODE	TABLE #	FUNCTION
STA	LAFUN	5	STORE A REGISTER TO LONG ADDRESS
STSP	MISCFUN	4	STORE STACK POINTER TO B REGISTER
SUB	DOPFUN	3	SUBTRACT
SWAP	SOPFUNS	6	SWAP NIBBLES OF AN 8 BIT VALUE
TSTA	SOPFUNS	6	TEST A REGISTER AND SET STATUS
TSTB	SOPFUNS	6	TEST B REGISTER AND SET STATUS
TRAP _n	TRAP	11	TRAP TO SUBROUTINE
XCHB	SOPFUNS	6	EXCHANGE VALUE WITH B REGISTER
XOR	DOPFUN	3	EXCLUSIVE OR
XORP	DOPFUNP	8	EXCLUSIVE OR WITH PERIPHERAL PORT

TABLE B-2 – INSTRUCTION ACQUISITION MODE - OPERATION CODE FETCH

ADDR MODE	CYCLE	ADDRESS BUS	DATA BUS	R \bar{W}
ALL INSTRUCTIONS	1	Opcode address	Irrelevant data	R
	2	Opcode address	Instruction Opcode	R
	If an interrupt is pending, go to interrupt code listed below			
	3	B reg. address	B reg. contents	R
Go to Addressing Mode (Tables 3 through 11)				

- NOTES:
1. This mode is executed for all instructions to fetch the instructions's operation code, or opcode.
 2. The B register is prefetched to speed up the execution of instructions that reference the B register.
 3. The Program Counter is incremented during cycles 1 and 2 of this mode.
 4. During cycle 2 an interrupt check is performed. If an interrupt is detected, cycle #3 is not executed. Control is passed immediately to the interrupt handling code shown below.

TABLE B-2 — INSTRUCTION ACQUISITION MODE-INTERRUPT HANDLING

ADDR MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
INTERRUPTS	1	Irrelevant data	Irrelevant data	—
	2	Irrelevant data	Irrelevant data	—
Jump to cycle number 5 if opcode was IDLE (>01). If it was an IDLE instruction, do not decrement PC because desired return is past the IDLE instruction.				
	3	Irrelevant data	Irrelevant data	—
	4	Irrelevant data	Irrelevant data	—
	5	SP register	Status register	W
	6	Irrelevant data	Irrelevant data	—
— Jump to Trap group at Table B-11 —				

- NOTES: 1. The Program Counter is decremented during cycles number 3 and 4. This is done because the instruction that the PC had pointed at has not been executed.
2. The status register is saved on the stack during cycle #5. When control is passed to the Trap group (at Table B-11) the program counters will be saved.

TABLE B-3 — DOUBLE OPERAND FUNCTIONS - ADDRESSING MODES

Instructions: ADD,ADC,AND,BTJO,BTJZ,CMP,DAC,DSB,MOV,MPY,OR,SBB,SUB,XOR

ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
Rn, A	1	Opcode Address + 1	Irrelevant Data	R
	2	Opcode Address + 1	Rn Address	R
	3	Rn Address	Rn data	R
	4	A register address	A register data	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
%n, A	1	Opcode Address + 1	Irrelevant Data	R
	2	Opcode Address + 1	Immediate value (%n)	R
	3	A register address	A register data	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
Rn, B	1	Opcode Address + 1	Irrelevant data	R
	2	Opcode Address + 1	Rn address	R
	3	Rn address	Rn data	R
	4	B register address	Operand data	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
Rn, Rn	1	Opcode Address + 1	Irrelevant Data	R
	2	Opcode Address + 1	Rsrc address	R
	3	Rsrc address	Rsrc data	R
	4	Opcode address + 2	Irrelevant data	R
	5	Opcode address + 2	Rdest address	R
	6	Rdest address	Rdest data	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
%n, B	1	Opcode Address + 1	Irrelevant Data	R
	2	Opcode Address + 1	Immediate data	R
	3	B register address	B reg. data	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
B, A	1	A register address	A register data	R
Go To Functional Modes For This Addressing Group				

TABLE B-3 — DOUBLE OPERAND FUNCTIONS - FUNCTIONAL MODES (CONTINUED)

Instructions: ADD,ADC,AND,BTJO,BTJZ,CMP,DAC,DSB,MOV,MPY,OR,SBB,SUB,XOR

ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R \bar{W}
%n, Rn	1	Opcode Address + 1	Irrelevant Data	R
	2	Opcode Address + 1	Immediate data	R
	3	Opcode Address + 2	Irrelevant data	R
	4	Opcode Address + 2	Rn address	R
	5	Rn address	Rn data	R
Go To Functional Modes For This Addressing Group				

MACROINSTRUCTION	CYCLE	ADDRESS BUS	DATA BUS	R \bar{W}
MOV	1	Register address	Register data	W
AND	1	Register address	Register data	W
OR	1	Register address	Register data	W
XOR	1	Register address	Register data	W
ADD	1	Register address	Register data	W
ADC	1	Register address	Register data	W
SUB	1	Register address	Register data	W
SBB	1	Register address	Register data	W
CMP	1	Irrelevant data	Irrelevant data	—
DAC	1	Register address	Register data	W
DAC	2	Register address	Register data	R
DAC	3	Register address	Register data	W
DSB	1	Register address	Register data	W
DSB	2	Register address	Register data	R
DSB	3	Register address	Register data	W
MPY	1	B reg. address	B reg. data	W
MPY	2	Irrelevant data	Irrelevant data	—
MPY	3	Irrelevant data	Irrelevant data	—
MPY	4	B reg. address	B reg. data	R
MPY	5	B reg. address	B reg. data	W
MPY	6	Irrelevant data	Irrelevant data	—
MPY	7	Irrelevant data	Irrelevant data	—
MPY	8	A reg. address	MSH mult. product	W
MPY	9	Irrelevant data	Irrelevant data	—
BTJO,BTJZ	1	Irrelevant data	Irrelevant data	—
BTJO,BTJZ	2	Opcode address + 1	Irrelevant data	R
BTJO,BTJZ	3	Opcode address + 1	Jump PC offset	R
BTJO,BTJZ	4	Opcode address + 1	Jump PC offset	R
BTJO,BTJZ	5	Irrelevant data	Irrelevant data	—
BTJO,BTJZ	6	Irrelevant data	Irrelevant data	—
BTJO,BTJZ	7	Irrelevant data	Irrelevant data	—
Jump To Instruction Acquisition Sequence				

TABLE B-3 — DOUBLE OPERAND FUNCTIONS - FUNCTIONAL MODES (CONTINUED)

- NOTES: 1. MPY - This microcode iterates to perform the multiply. The functional portion of the MPY instruction requires 40 states for execution.
2. BTJO,BTJZ - Not all states are executed. Either state 2 or 3 is executed but not both. The same applies to states 6 and 7.
3. Where referenced, Rsrc is the first operand listed and Rd is the second. The resultant value will be stored at the Rd address.

TABLE B-4 — MISCELLANEOUS FUNCTIONS - ADDRESSING MODES

Instructions: DINT,EINT,IDLE,LDSP,NOP,POP ST,PUSH ST,RETI,RETS,SETC,STSP

ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
	1	SP contents	Stack value	R
Go To Functional Modes For This Addressing Group				

TABLE B-4 — MISCELLANEOUS FUNCTIONS - FUNCTIONAL MODES

Instructions: DINT,EINT,IDLE,LDSP,NOP,POPST,PUSHST,RETI,RETS,SETC,STSP

MACROINSTRUCTION	CYCLE	ADDRESS BUS	DATA BUS	R/W
EINT	1	Irrelevant data	Irrelevant data	—
DINT	1	Irrelevant data	Irrelevant data	—
SETC	1	Irrelevant data	Irrelevant data	—
POP ST	1	SP contents	Stack data	R
POP ST	2	Irrelevant data	Irrelevant data	—
STSP	1	Irrelevant data	Irrelevant data	—
STSP	2	B reg. addr	SP contents	W
RETS	1	Irrelevant data	Irrelevant data	—
RETS	2	Register address	Register data	R
RETS	3	Irrelevant data	Irrelevant data	—
RETI	1	Irrelevant data	Irrelevant data	—
RETI	2	Register Address	Register data	R
RETI	3	Irrelevant data	Irrelevant data	—
RETI	4	SP contents	Register data	R
RETI	5	Irrelevant data	Irrelevant data	—
LDSP	1	Irrelevant data	Irrelevant data	—
PUSH ST	1	Irrelevant data	Irrelevant data	—
PUSH ST	2	SP contents	Status register	W
IDLE	1	Irrelevant data	Irrelevant data	—
IDLE	2	Irrelevant data	Irrelevant data	—
Jump To Instruction Acquisition Sequence				

- NOTE: 1. NOP does not have an execution state. From the addressing mode control is passed back to the instruction acquisition microcode.

TABLE B-5 — LONG ADDRESSING FUNCTIONS - ADDRESSING MODES

Instructions: BR, CALL, CMPA, LDA, STA

ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
@n	1	Opcode address + 1	Irrelevant data	R
	2	Opcode address + 1	MSH of long address	R
	3	Opcode address + 2	Irrelevant data	R
	4	Opcode address + 2	LSH of long address	R
	5	Irrelevant data	Irrelevant data	—
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
*Rn	1	Opcode address + 1	Irrelevant data	R
	2	Opcode address + 1	Rn address	R
	3	Rn address	LSH of long address	R
	4	Rn - 1 address	MSH of long address	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
@n(B)	1	Irrelevant data	Irrelevant data	—
	2	Opcode address + 1	Irrelevant data	R
	3	Opcode address + 1	MSH of long address	R
	4	Opcode address + 2	Irrelevant data	R
	5	Opcode address + 2	LSH of long address	R
	6	Irrelevant data	Irrelevant data	—
	7	Irrelevant data	Irrelevant data	—
Go To Functional Modes For This Addressing Group				

TABLE B-5 — LONG ADDRESSING FUNCTIONS - FUNCTIONAL MODES (CONTINUED)

Instructions: BR, CALL, CMPA, LDA, STA

MACROINSTRUCTION	CYCLE	ADDRESS BUS	DATA BUS	R/W
LDA	1	Operand address	Irrelevant data	R
LDA	2	Operand address	Operand data	R
LDA	3	A reg. address	Operand data	W
STA	1	A reg. address	A reg. contents	R
STA	2	Operand address	A reg. contents	W
STA	3	Operand address	A reg. contents	W
BR	1	Irrelevant data	Irrelevant data	—
BR	2	Irrelevant data	Irrelevant data	—
CMPA	1	Operand address	Irrelevant data	R
CMPA	2	Operand address	Operand data	R
CMPA	3	A reg. address	A reg. contents	R
CMPA	4	Irrelevant data	Irrelevant data	—
CALL	1	Irrelevant data	Irrelevant data	—
CALL	2	SP contents	PCH contents	W
CALL	3	Irrelevant data	Irrelevant data	—
CALL	4	SP + 1	PCL	W
CALL	5	Irrelevant data	Irrelevant data	—
CALL	6	Irrelevant data	Irrelevant data	—
Jump To Instruction Acquisition Sequence				

TABLE B-6 — SINGLE OPERAND FUNCTIONS, SPECIAL - ADDRESSING MODES

Instructions: CLR; DEC; INC; INV; MOV A,B; MOV A,RN; MOV B,RN;
SWAP; TSTA/CLRC; TSTB; XCHB;

ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
A	1	A register address	A reg. contents	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
B	1	B register address	B reg. contents	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
Rn	1	Opcode address + 1	Irrelevant data	R
	2	Opcode address + 1	Rn address	R
	3	Rn address	Rn data	R
Go To Functional Modes For This Addressing Group				

TABLE B-6 – SINGLE OPERAND FUNCTIONS, SPECIAL - FUNCTIONAL MODES

Instructions: CLR; DEC; INC; INV; MOV A,B; MOV A,Rn; MOV B,Rn; SWAP; TSTA/CLRC; TSTB; XCHB;

MACROINSTRUCTION	CYCLE	ADDRESS BUS	DATA BUS	R \bar{W}
DEC	1	Register address	Register Data	W
INC	1	Register address	Register Data	W
INV	1	Register address	Register Data	W
CLR	1	Register address	Register Data	W
XCHB	1	B reg. address	Register Data	W
XCHB	2	Register address	Register Data	W
SWAP	1	Irrelevant data	Irrelevant data	—
SWAP	2	Irrelevant data	Irrelevant data	—
SWAP	3	Irrelevant data	Irrelevant data	—
SWAP	4	Register address	Register data	W
MOV A,B	1	A reg. address	A reg. data	R
MOV A,B	2	B reg. address	A reg. data	W
MOV A,Rn	1	A reg. address	A reg. data	R
MOV A,Rn	2	Register address	A reg. data	W
MOV B,Rn	1	Register address	B reg. data	W
TSTA/CLRC	1	A reg. address	A reg. data	R
TSTA/CLRC	2	Register address	Register data	W
TSTB	1	B reg. address	Register data	W
Jump To Instruction Acquisition Sequence				

TABLE B-7 – SINGLE OPERAND FUNCTIONS, NORMAL - ADDRESSING MODES

Instructions: DECD, DJNZ, POP, PUSH, RL, RLC, RR, RRC

ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R \bar{W}
A	1	A reg. address	A reg. data	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R \bar{W}
B	1	B reg. address	B reg. data	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R \bar{W}
Rn	1	Opcode address + 1	Irrelevant data	R
	2	Opcode address + 1	Rn address	R
	3	Rn address	Rn data	R
Go To Functional Modes For This Addressing Group				

TABLE B-7 — SINGLE OPERAND FUNCTIONS, NORMAL - FUNCTIONAL MODES

Instructions: DECD, DJNZ, POP, PUSH, RL, RLC, RR, RRC

MACROINSTRUCTION	CYCLE	ADDRESS BUS	DATA BUS	R/W
PUSH	1	Irrelevant data	Irrelevant data	—
PUSH	2	SP contents	Register data	W
POP	1	SP contents	Register data	R
POP	2	Register data	Register data	W
RR	1	Register data	Register data	W
RRC	1	Register data	Register data	W
RL	1	Register data	Register data	W
RLC	1	Register data	Register data	W
DECD	1	Register data	Register data	W
DECD	2	Irrelevant data	Irrelevant data	—
DECD	3	Irrelevant data	Irrelevant data	—
DECD	4	Register address	Register data	R
DECD	5	Register address	Register data	W
DJNZ	1	Register address	Reg. data - 1	W
DJNZ	2	Opcode address + 1	Irrelevant data	R
		If result is not = 0, jump to state 4		
DJNZ	3	Opcode address + 1	Jump PC offset	R
		Jump to instruction acquisition sequence		
DJNZ	4	Opcode address + 1	Jump PC offset	R
DJNZ	5	Irrelevant data	Irrelevant data	—
		If jump PC offset is positive, jump to state 7		
DJNZ	6	Irrelevant data	Irrelevant data	—
		Jump To Instruction Acquisition Sequence		
DJNZ	7	Irrelevant data	Irrelevant data	—
		Jump To Instruction Acquisition Sequence		

TABLE B-8 — DOUBLE OPERAND FUNCTIONS, PERIPHERAL - ADDRESSING MODES

Instructions: ANDP, BTJOP, BTJZP, MOVP, ORP, XORP

ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R \overline{W}
A, Pn	1	A reg. address	A reg. data	R
	2	Opcode address + 1	Irrelevant data	R
	3	Opcode address + 1	Pn address	R
	4	Pn address	Irrelevant data	R
	5	Pn address	Pn data	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R \overline{W}
B, Pn	1	Opcode address + 1	Irrelevant data	R
	2	Opcode address + 1	Pn address	R
	3	Pn address	Irrelevant data	R
	4	Pn address	Pn data	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R \overline{W}
%n, Pn	1	Opcode address + 1	Irrelevant data	R
	2	Opcode address + 1	%n (immediate data)	R
	3	Opcode address + 2	Irrelevant data	R
	4	Opcode address + 2	Pn address	R
	5	Pn address	Irrelevant data	R
	6	Pn address	Pn data	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R \overline{W}
Pn, A	1	A reg. address	A reg. data	R
	2	Opcode address + 1	Irrelevant data	R
	3	Opcode address + 1	Pn address	R
	4	Pn address	Irrelevant data	R
	5	Pn address	Pn data	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R \overline{W}
Pn, B	1	Opcode address + 1	Irrelevant data	R
	2	Opcode address + 1	Pn address	R
	3	Pn address	Irrelevant data	R
	4	Pn address	Pn data	R
Go To Functional Modes For This Addressing Group				

- NOTES: 1. Addressing modes "A, Pn" and "Pn, A" fetch their operands the same way.
 2. Addressing modes "B, Pn" and "Pn, B" fetch their operands the same way.

TABLE B-8 — DOUBLE OPERAND FUNCTIONS, PERIPHERAL - FUNCTIONAL MODES

Instructions: ANDP, BTJOP, BTJZP, MOVP, ORP, XORP

MACROINSTRUCTION	CYCLE	ADDRESS BUS	DATA BUS	R/W
MOVP X, Pn	1	Pn address	Peripheral reg. data	W
MOVP X, Pn	2	Pn address	Peripheral reg. data	W
MOVP Pn, A	1	A reg. address	Register data	W
MOVP Pn, B	1	B reg. address	Register data	W
ANDP	1	Pn address	Peripheral reg. data	W
ANDP	2	Pn address	Peripheral reg. data	W
ORP	1	Pn address	Peripheral reg. data	W
ORP	2	Pn address	Peripheral reg. data	W
XORP	1	Pn address	Peripheral reg. data	W
XORP	2	Pn address	Peripheral reg. data	W
BTJOP	1	Irrelevant data	Irrelevant data	—
BTJOP	2	Opcode address + 1	Irrelevant data	R
		If bit tested is equal to a 1, jump to state 4		
BTJOP	3	Opcode address + 1	Jump PC offset	R
		Jump to instruction acquisition sequence		
BTJOP	4	Opcode address + 1	Jump PC offset	R
BTJOP	5	Irrelevant data	Irrelevant data	—
		If jump PC offset is positive, jump to state 7		
BTJOP	6	Irrelevant data	Irrelevant data	—
		Jump To Instruction Acquisition Sequence		
BTJOP	7	Irrelevant data	Irrelevant data	—
BTJZP	1	Irrelevant data	Irrelevant data	—
BTJZP	2	Opcode address + 1	Irrelevant data	R
		If bit tested is equal to a 0, jump to state 4		
BTJZP	3	Opcode address + 1	Jump PC offset	R
		Jump to instruction acquisition sequence		
BTJZP	4	Opcode address + 1	Jump PC offset	R
BTJZP	5	Irrelevant data	Irrelevant data	—
		If jump PC offset is positive, jump to state 7		
BTJZP	6	Irrelevant data	Irrelevant data	—
		Jump To Instruction Acquisition Sequence		
BTJZP	7	Irrelevant data	Irrelevant data	—
		Jump To Instruction Acquisition Sequence		

NOTE: 1. MOVP X, Pn - X is either the A or B register, or an 8 bit immediate value %n.

TABLE B-9 — MOVE DOUBLE - ADDRESSING MODES

Instructions: MOVD

ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
%n, Rn	1	Opcode address + 1	Irrelevant data	R
	2	Opcode address + 1	MSH of immed. data	R
	3	Opcode address + 2	Irrelevant data	R
	4	Opcode address + 2	LSH of immed. data	R
	5	Irrelevant data	Irrelevant data	—
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
Rn, Rn	1	Opcode address + 1	Irrelevant data	R
	2	Opcode address + 1	Rn source address	R
	3	Rn source address	Rn data - LSH	R
	4	Rn - 1 source addr.	Rn - 1 data - MSH	R
Go To Functional Modes For This Addressing Group				
ADDRESSING MODE	CYCLE	ADDRESS BUS	DATA BUS	R/W
%n(B), Rn	1	Irrelevant data	Irrelevant data	—
	2	Opcode address + 1	Irrelevant data	R
	3	Opcode address + 1	MSH of immed. data	R
	4	Opcode address + 2	Irrelevant data	R
	5	Opcode address + 2	LSH of immed. data	R
	6	Irrelevant data	Irrelevant data	—
	7	Irrelevant data	Irrelevant data	—
Go To Functional Modes For This Addressing Group				

TABLE B-9 — MOVE DOUBLE - FUNCTIONAL MODE

MACROINSTRUCTION	CYCLE	ADDRESS BUS	DATA BUS	R/W
MOVD	1	Irrelevant data	Irrelevant data	—
MOVD	2	Opcode address + 2/3	Irrelevant data	R
MOVD	3	Opcode address + 2/3	Destination Rn addr	R
MOVD	4	Irrelevant data	Irrelevant data	—
MOVD	5	Dest. Rn address	LSH register data	W
MOVD	6	Irrelevant data	Irrelevant data	—
MOVD	7	Dest. Rn-1 address	MSH register data	W
Jump To Instruction Acquisition Sequence				

NOTE: 1. MOVD - States 2 and 3 will be Opcode address + 2 for the "%n, Rn" and the "Rn, Rn" addressing modes. States 2 and 3 will be Opcode address + 3 for the "%n(B), Rn" addressing mode.

TABLE B-10 — RELATIVE JUMPS - ADDRESSING AND FUNCTIONAL MODES

Instructions: JMP,JN/JLT,JZ/JEQ,JC/JHS,JP/JGT,JPZ/JGE,JNZ/JNE,JNC,JL

RELATIVE JUMPS	CYCLE	ADDRESS BUS	DATA BUS	R/W
	1	Opcode address + 1	Irrelevant data	R
		If jump condition is true, jump to state 3		
	2	Opcode address + 1	Jump PC offset	R
		Jump To Instruction Acquisition Sequence		
	3	Opcode address + 1	Jump PC offset	R
	4	Irrelevant data	Irrelevant data	—
		If jump offset is positive go to state 6		
	5	Irrelevant data	Irrelevant data	—
		Jump To Instruction Acquisition Sequence		
	6	Irrelevant data	Irrelevant data	—
		Jump To Instruction Acquisition Sequence		

- NOTES: 1. Cycle 1 tests the jump condition. If the jump is true, go to state 3, else execute state 2 and return to the instruction acquisition sequence.
2. Cycle 4 tests whether the jump offset is positive or negative. If the jump offset is positive, go to state 6.

TABLE B-11 — TRAPS - ADDRESSING AND FUNCTIONAL MODES

Instructions: Trap 0 through Trap 23

TRAPS	CYCLE	ADDRESS BUS	DATA BUS	R/W
Trap 0 - 7 (Group A)	1	Irrelevant data	Irrelevant data	—
Trap 8 - 15 (Group B)	1	Irrelevant data	Irrelevant data	—
Trap 16 - 23 (Group C)	1	Irrelevant data	Irrelevant data	—
	2	Irrelevant data	Irrelevant data	—
	3	Addr. >FF00 + Opcode	Irrelevant data	R
	4	Addr. >FF00 + Opcode	LSH trap vector	R
	5	Addr. >FF00 + Opcode - 1	Irr. data	R
	6	Addr. >FF00 + Opcode - 1	MSH trap vector	R
	7	Sp contents	PCH contents	W
	8	Irrelevant data	Irrelevant data	—
	9	Sp + 1 contents	PCL contents	W
	10	Irrelevant data	Irrelevant data	—
	11	Irrelevant data	Irrelevant data	—
		Jump To Instruction Acquisition Sequence		

TABLE B-12 — RESET FUNCTION

RESET	CYCLE	ADDRESS BUS	DATA BUS	\overline{RW}
Reset Function	1	Irrelevant data	Irrelevant data	R
	2	Irrelevant data	Zeroes	—
	3	Address >0100	Zeroes	W
	4	Address >0100	Zeroes	W
Jump to Trap Group				

- NOTES:
1. read operation is done the first cycle even though the address and data buses contain irrelevant data. This read is done to protect memory in case a long write was in progress when the Reset action occurred.
 2. The write to address >0100 is done to disable all interrupts.
 3. The Stack Pointer is initialized to >01.
 4. The Program Counter is stored in the register pairs A and B.
 5. The Trap Group code will take the Reset vector from >FFFE and >FFFF and place it in the Program Counter.

The RESET function is initiated when the \overline{RESET} line of the TMS7000 device is held at a logic zero level for at least five clock cycles. The Reset function is active at a logic zero level, and occurs on pin 14 of the device. When an active signal is detected on the \overline{RESET} line the following sequence is entered immediately after the current machine cycle is done.

APPENDIX C TMS7500 DATA ENCRYPTION DEVICE

C.1 DESCRIPTION

The TMS7500 Data Encryption Device (DED)* is a peripheral device designed to perform the National Bureau of Standards (NBS) Data Encryption Standard (DES) algorithm as specified in the Federal Information Processing Standard (FIPS) Publication 46. The TMS7500 DED can be memory mapped on computer systems requiring the use of the Data Encryption Standard. The TMS7500 is a standard preprogrammed TMS7020 8-bit single-chip microcomputer using the standard microcoded instruction set. This allows the TMS7500 to be a very cost effective solution for low cost data encryption requirements. The device comes in a 40-pin package, requires a single 5 volt supply, and all I/O pins are TTL compatible (see Figure C-2). For more information on the TMS7500 refer to the TMS7500 Data Encryption Device Data Manual.

C.1.1 Typical Applications

The TMS7500 is particularly well suited for any system requiring the use of a low cost, medium speed data encryption device. It can easily keep up with the data rates required by most modems and terminals without sacrificing system performance. Some typical applications are:

- Computer to terminal communication links
- Home banking communication links
- Teller machines for banks
- Portable terminals
- Point of sale terminals
- Small business systems
- Trade market software protection
- Any system requiring a low cost, medium speed Data Encryption Device

* The products covered by this document (TMS7500) are within the group of electronic products that are wholly or partly of U. S. origin or technology, the export of which is subject to export license control by the U. S. Government. Therefore, prior to exportation, you are obligated to obtain the required export license from the U. S. Department of State. (Refer to Title 22, Code of Federal Regulations.)

C.1.2 Key Features

A number of key features, most of which are user programmable, enables the TMS7500 to enhance the flexibility of any system using data encryption. The device can store two keys at one time and operate in two of the standard data encryption modes. Some of the key features are highlighted below:

- Validated by the National Bureau of Standards
- Can store both a Master and an Active 64-bit key
- Active key can be encrypted or decrypted by the master key internally
- Electronic Codebook (ECB) or Cipher Feedback (CFB) modes of operation
- Dual 8-bit data bus operation possible; one for plain data, and one for cipher data
- Command register programmable from data bus or from external pins on chip
- Status is displayed on external pins and can be read from the data bus
- Clock source can be internal or external
- On-Chip clock uses crystal or ceramic resonator
- Maximum data rate of 3200 bits per second (ECB) or 400 bits per second (CFB) with 5 MHz clock (divide by 2 option) or 10 MHz clock (divide by 4 option)
- Single power source requirement (+ 5V)

C.2 PROCESSOR INTERFACE

All communications between a host processor and the TMS7500 can be handled through the main 8 bit data bus. The processor can access the command and status registers, both master and active key registers, and the 8 byte data buffer through this bus. An optional cipher data bus can be used to handle all encrypted data. The 7 bit read only status register provides the host processor with current status information such as:

- Key entered
- Key parity error
- Active key register is being accessed
- Encrypt or decrypt mode
- Electronic codebook or cipher feedback mode
- Initialization Vector loaded (for cipher feedback mode)

The five bit write only command register accepts several different commands from the processor, including the following commands:

- Reset the DED
- Enter an active key
- Enter active key and encrypt or decrypt under master key
- Encrypt or decrypt data
- Electronic codebook or cipher feedback operation

The master and active key registers are write only registers. This prevents the key value from ever being discovered once it is entered into the device. Another unique feature is that a new active key, when entered into the DED, can be encrypted or decrypted by the master key before it is stored into the active key register. This allows the user to send a new active key to the DED in encrypted or decrypted form for maximum security.

The 8 byte data buffer is used to handle all plain data and ciphered data sent to and read from the DED.

C.3 EXTERNAL COMMAND AND STATUS DISPLAY

The command and status registers may also be accessed from external pins. Status register contents are displayed at all times on six status display pins. The command register is accessible from five external command pins when the external command mode is enabled.

C.4 FUNCTIONAL BLOCK DIAGRAM

The functional block diagram of the TMS7500 DED in Figure C-1 illustrates an architecture organized around certain registers, buffers, and I/O buses which are all linked together through the data selectors. All of the necessary data path sequences through these selectors are determined by a 5-bit Command Register and 8 external Control-Handshake pins. The device status is stored in the Status Register and is also available on the Status Display Pins. The 64-bit key values and encryption data are passed along the 8-bit Main Data Bus and Cipher Data Bus.

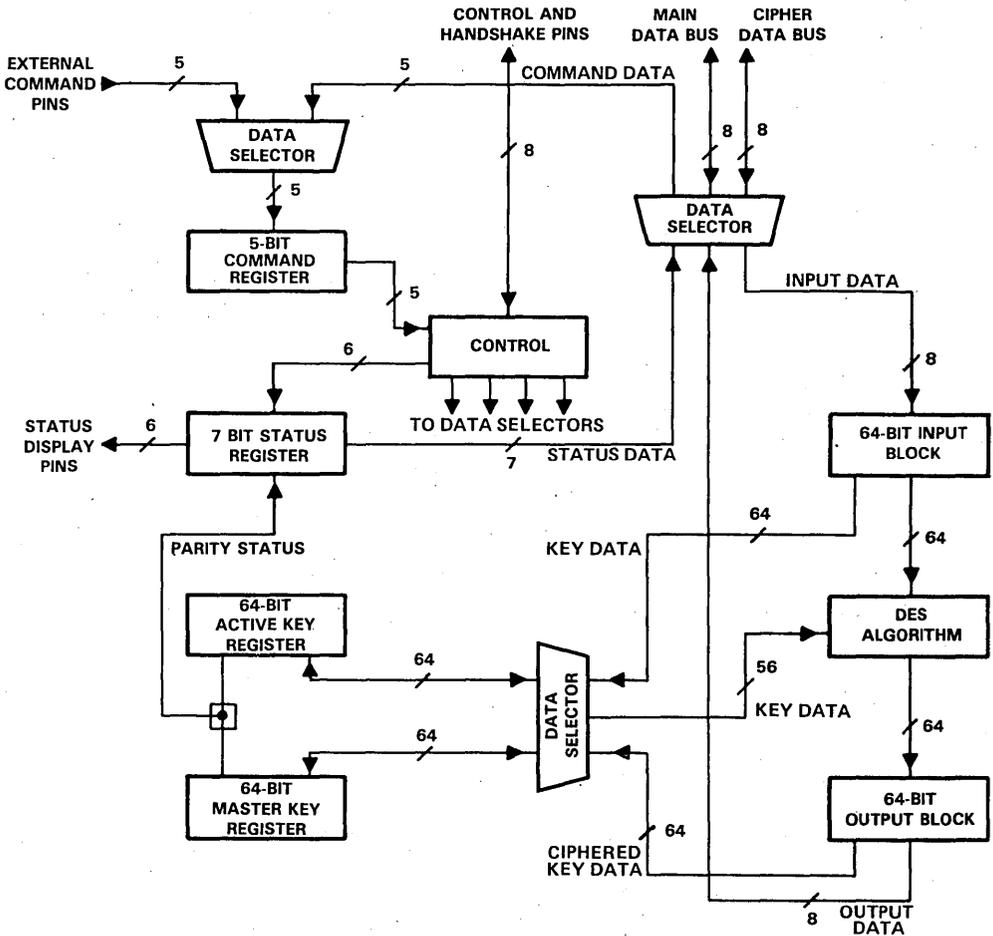
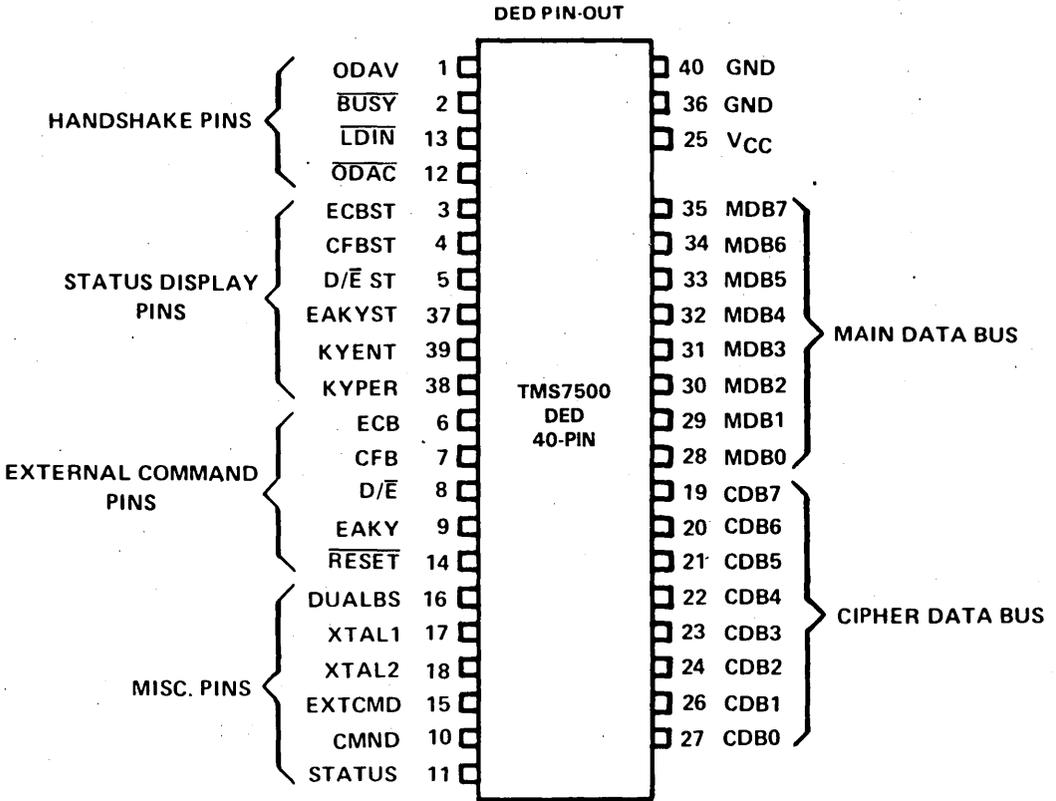


FIGURE C-1 – TMS7500 FUNCTIONAL BLOCK DIAGRAM

C.5 PIN-OUT AND PIN FUNCTION

Figure C-2 shows the TMS7500 pin-out. Following is a description of the pin functions.

FIGURE C-2 — TMS7500 DATA ENCRYPTION DEVICE PIN-OUT



C.5.1 Handshake Pins

SIGNATURE	PIN	I/O	DESCRIPTION
ODAV	1	O	Output Data Available-ODAV becomes active (high) when the DED has data available to be read on one of the data buses. After one read cycle, ODAV will go inactive (low).
$\overline{\text{BUSY}}$	2	O	Busy - $\overline{\text{BUSY}}$ becomes active (low) after $\overline{\text{LDIN}}$ is driven active (low), indicating that data was written to one of the data buses and is being stored by the DED. The DED will then set $\overline{\text{BUSY}}$ high. More data should not be fed to the DED until $\overline{\text{BUSY}}$ becomes inactive (high).
$\overline{\text{ODAC}}$	12	I	Output Data Accepted $\overline{\text{ODAC}}$ is made active (low) when a read cycle is executed from either the Main Data Bus or the Cipher Data Bus. This signal alerts the DED that output data has been read by the host processor. $\overline{\text{ODAC}}$ is ignored if the DED does not have output data available to be read.
$\overline{\text{LDIN}}$	13	I	Load Data In - $\overline{\text{LDIN}}$ is driven active (low) when a write cycle is executed to either the Main Data Bus or the Cipher Data Bus. When $\overline{\text{LDIN}}$ becomes active, the DED will activate $\overline{\text{BUSY}}$ and store the byte of data. $\overline{\text{LDIN}}$ is ignored if the DED is waiting for any output data to be read.

C.5.2 Status Display Pins

SIGNATURE	PIN	I/O	DESCRIPTION
ECBST	3	O	Electronic Codebook Status-ECBST reflects the logic level of the ECB bit in the Command Register.
CFBST	4	O	Cipher Feedback Status-CFBST reflects the logic level of the CFBST bit in the Status Register.
$\text{D}/\overline{\text{E}}\text{-ST}$	5	O	Decrypt/Encrypt Status- $\text{D}/\overline{\text{E}}\text{ST}$ reflects the logic level of the $\text{D}/\overline{\text{E}}\text{ST}$ bit in the Status Register.
EAKYST	37	O	Enter Active Key Status-EAKYST reflects the logic level of the EAKYST bit in the Status Register.
KYPER	38	O	Key Parity Error - KYPER reflects the logic level of the KYPER bit in the Status Register.
KYENT	39	O	Key Entered - KYENT reflects the logic level of the KYENT bit in the Status Register.

C.5.3 External Command Pins.

SIGNATURE	PIN	I/O	DESCRIPTION
ECB	6	I	Electronic Codebook - If the EXTCMD and CMND pins are active (high), the ECB bit in the Command Register will reflect the logic level of the ECB pin.
CFB	7	I	Cipher Feedback - If the EXTCMD and CMND pins are active (high), the CFB bit in the Command Register will reflect the logic level of the CFB pin.
D/ \bar{E}	8	I	Decrypt/Encrypt - If the EXTCMD and CMND pins are active (high), the D/ \bar{E} bit in the Command Register will reflect the logic level of the D/ \bar{E} pin.
EAKY	9	I	Enter Active Key - If the EXTCMD and CMND pins are active (high), the EAKY bit in the Command Register will reflect the logic level of the EAKY pin.
<u>RESET</u>	14	I	Reset - When active (low), the DED is reset, regardless of the logic level on any other pin. A reset will clear the Status Register, Status Display pins, Command Register, and both key registers. Both data buses will be in a high impedance (input) state. After the <u>RESET</u> pin is initiated, a delay time of at least 174 microseconds is required before any other commands can be given to the DED.

C.5.4 Cipher Data Bus Pins

SIGNATURE	PIN	I/O	DESCRIPTION
CDB 0-7	19-24, 26,27	I/O	Cipher Data Bus - When DUALBS is active 26-27 (high), the 8-bit Cipher Data Bus is used to pass all encrypted data to and from the DED. CDB7 is the most significant bit and CDB0 is the least significant bit. When DUALBS is inactive (low), the Cipher Data Bus is disabled and left in a high impedance state.

C.5.5 Main Data Bus Pins

SIGNATURE	PIN	I/O	DESCRIPTION
MDB0-7	28-35	I/O	I/O Main Data Bus - When DUALBS is inactive (low), the Main Data Bus is used to pass all data to and from the DED. When DUALBS is active (high), the Main Data Bus is used to pass only unencrypted data to and from the DED. MDB7 is the most significant bit and MDB0 is the least significant bit.

C.5.6 Miscellaneous Pins

SIGNATURE	PIN	I/O	DESCRIPTION
CMND	10	I	Command Register Update - When active (high), CMND will direct data to the Command Register. The source of command data is determined by the EXTCMD pin. When CMND is inactive (low), access to the Command Register is disabled.
STATUS	11	I	Read Status - When STATUS is active (high), the Status Register contents are available on the Main Data Bus (never on the Cipher Data Bus). The STATUS pin should be made inactive (low), before a read cycle is executed to get the status data from the bus.
EXTCMD	15	I	External Command - When active (high), all command data is received from the External Command Pins. When inactive (low), all command data is received from the Main Data Bus.
DUALBS	16	I	Dual Data Bus - When active (high), the DED will communicate on both the Main Data Bus and the Cipher Data Bus. When inactive (low), the DED will only communicate on the Main Data Bus.
XTAL1	17	I	Crystal Input 1 - Crystal input for internal clock oscillator. Leave open if an external clock source is used.
XTAL2	18	I	Crystal Input 2 - Crystal input for internal clock oscillator. Also input for an external clock source (divide by 4 only).
VCC	25	I	Power Source - Power supply source = + 5 V.
VSS	36,40	I	Power Ground - Power ground = 0 V. Both pins must be grounded.

DUALBS PIN	D / \bar{E} PIN	CIPHER DATA BUS	MAIN DATA BUS
0	0	NOT USED	READ/WRITE
0	1	NOT USED	READ/WRITE
1	0	READ FROM	WRITE TO
1	1	WRITE TO	READ FROM

FIGURE C-3 – DED DATA FLOW

C.6 STATUS AND COMMAND

The DED has two separate internal registers for command and status data. Most of the status data is also available on the Status Display Pins. The optional External Command Pins can be used to load the command register.

C.6.1 DED Status Register

The Status Register contains the operational status of the DED at all times. This is a read only register. All of the status bits, except for MSGST, are also available on the Status Display Pins. The DED status register contents can also be read from the Main Data Bus. This is done by setting the STATUS pin high and then low and doing a Read Cycle. A request for status can be initiated any time during DED operations, even during a DES calculation or when output data is available. It is important to note, however, that all other operations stop until the status byte is read from the DED. All status bits are true when equal to one. The Status Register is cleared when RESET or RESET2 is initiated. The following is the Status Register layout and bit descriptions.

STATUS REGISTER

7	6	5	4	3	2	1	0	
0	KYENT	MSGST	KYPER	EAKYST	D/E-ST	CFBST	ECBST	
MSB								LSB

KYENT - Key Entered - KYENT indicates when a key has been initially loaded into the DED after a RESET or RESET2 function. KYENT is cleared upon reset and is set to one after 8 bytes of key data have been loaded into the Master Key Register.

SGST - Message Start - MSGST indicates that the next 8 bytes loaded into the DED are to be used as an Initialization Vector (IV) for the CFB mode of operation. MSGST is set to one when the CFB mode has been initialized in the Command Register. It is cleared after 8 bytes are loaded in for an IV, upon a reset, or when any other mode of operation is entered.

- KYPER - Key Parity Error - KYPER indicates that a parity error was detected on the last key loaded into the DED. This is only to detect key parity errors and will not prevent continued operations of the DED. KYPER is set to one when parity errors are detected and will be cleared if the next key entered does not have a parity error or upon reset.
- EAKYST - Enter Active Key Status - EAKYST reflects the logic level of the EAKY bit in the Command Register.
- D/ \bar{E} ST - Decrypt / Encrypt Status - D/ \bar{E} ST reflects the logic level of the D/ \bar{E} bit in the Command Register.
- CFBST - Cipher Feedback Status - CFBST reflects the logic level of the CFB bit in the Command Register.
- ECBST - Electronic Codebook Status - ECBST reflects the logic level of the ECB bit in the Command Register.

C.6.2 DED Command Register

All DED operations are controlled from the Command Register. The Command Register is a write only register that can be accessed in two different ways, depending on the state of the EXTCMD pin. When EXTCMD is low, command data is received from the Main Data Bus. When EXTCMD is high, command data, except for RESET2, is received from the EAKY, D/E, CFB, and ECB pins on the External Command Bus. The RESET pin ignores the logic level of the EXTCMD pin and can be activated any time. In either case, the CMND pin determines when data will be directed to the Command Register. When EXTCMD is high and CMND is set high for a minimum of 42 μ s and then low, the data on four External Command Pins are latched into the lower nibble of the Command Register. When EXTCMD is low and CMND is held high, the next byte written to the Main Data Bus is moved into the Command Register. All command bits are true when high. The Command Register is cleared upon either a RESET or RESET2. The following is a layout of the Command Register and pin descriptions.

COMMAND REGISTER

	7	6	5	4	3	2	1	0
	RESET2	X	X	X	EAKY	D/E	CFB	ECB
MSB	(X = DON'T CARE)						LSB	

- RESET2 - Software Reset - when set to one, RESET2 will cause the DED to reset. The results are the same as if a reset occurred from the RESET pin.
- EAKY - Enter Active Key - when set to one, EAKY allows access to the Active Key Register from the Main Data Bus or the Cipher Data Bus.
- D/ \bar{E} - Decrypt / Encrypt - This bit can determine the direction of data flow as well as whether the DED will Encrypt or Decrypt. D/ \bar{E} is set to one for Decrypt and zero for Encrypt operations. This bit also affects the direction of data flow when the DED is using both data buses (DUALBS pin equal one).

- CFB - Cipher Feedback - When set to one, the DED will run in the Cipher Feedback mode of operation. EAKY and ECB should be equal to zero while in this mode. D/E is set for encrypting or decrypting.
- ECB - Electronic Codebook - When set to one, the DED will run in the Electronic Codebook mode of operation. CFB should be equal to zero while in this mode. EAKY could be a one since it uses ECB to encrypt or decrypt new active keys, but should be zero for normal ECB operation. D/E is set for encrypting or decrypting.

C.6.3 System Interface

Depending on system requirements, all or part of the TMS7500 I/O capability may be utilized. Figure C-4 is an example of using all of the TMS7500 I/O options. In this configuration two data buses are used as separated channels for clear data and ciphered data when the DUALBS pin connected to VCC. All ciphered data is passed along the Cipher Data Bus; all plain text data, master key, active key, and the Initialization Vector (IV) for the CFB mode is passed along the Main Data Bus. The two data buses can be memory mapped in separate locations of a single host microprocessor for added system security. They may also be hooked to two separate processors in a multiprocessor application.

With the EXTCMD pin connected to VCC, commands to the DED are received from external switches rather than from the Main Data Bus. A command is entered by setting the appropriate toggle switches and pushing the command button (see Figure C-4) to latch the bit pattern into the command register. The DED is reset by pushing the external reset button. The status is displayed constantly on LED indicators driven by the Status Display pins.

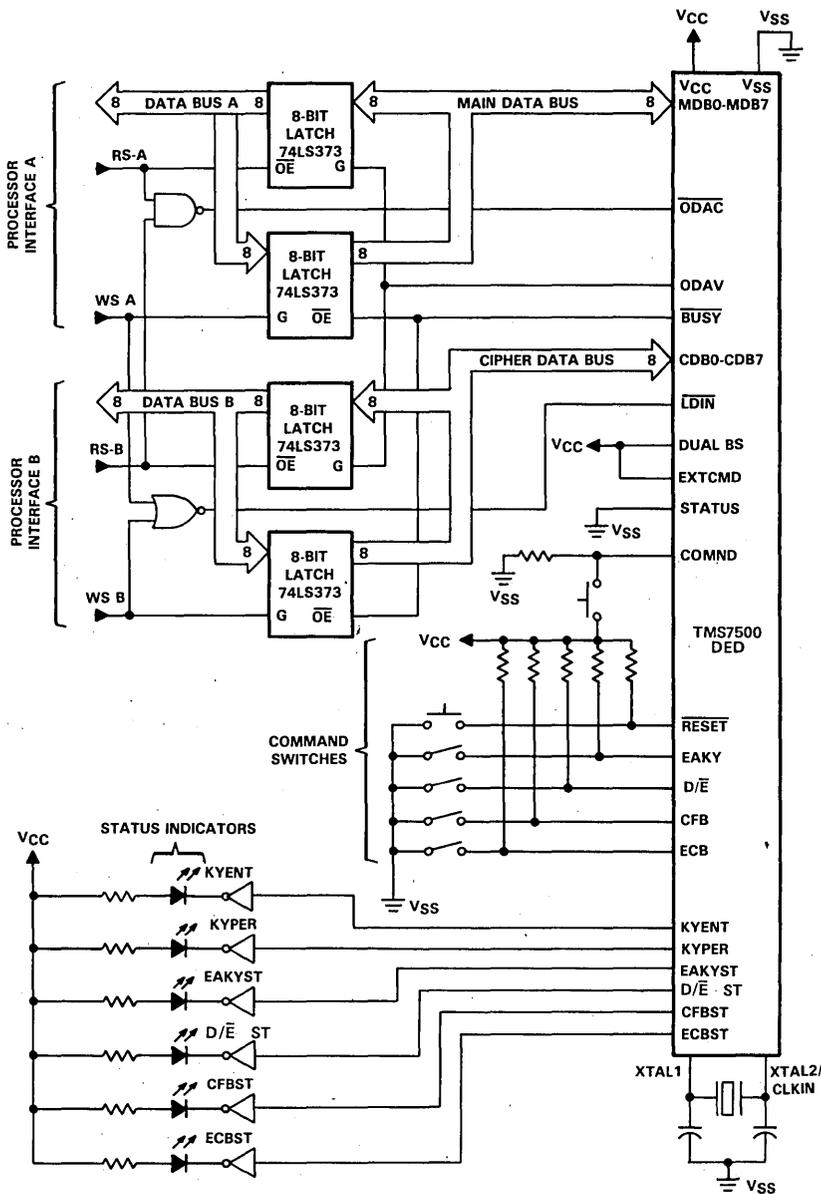


FIGURE C-4 - FULL TMS7500 I/O USAGE

APPENDIX D REFERENCES

D.1 HEXADECIMAL/DECIMAL CONVERSION TABLE

Table D-1 lists the hexadecimal/decimal conversion table. To convert a hexadecimal number to decimal, add the decimal equivalents for each of the four positions. To convert from decimal to hexadecimal, use the hex equivalents of the largest decimal numbers in each position that add up to the desired number. Begin summing the nearest MSB number that is less than (or equal to) the desired decimal number.

TABLE D-1 — HEXADECIMAL/DECIMAL CONVERSION TABLE

	0		3		4		7		8		11		12		15		
	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	DEC	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	1	4096	1	256	1	16	1	1	1	16	1	1	1	1	1	1	
	2	8192	2	512	2	32	2	2	2	32	2	2	2	2	2	2	
	3	12288	3	768	3	48	3	3	3	48	3	3	3	3	3	3	
	4	16384	4	1024	4	64	4	4	4	64	4	4	4	4	4	4	
	5	20480	5	1280	5	80	5	5	5	80	5	5	5	5	5	5	
	6	24576	6	1536	6	96	6	6	6	96	6	6	6	6	6	6	
	7	28672	7	1792	7	112	7	7	7	112	7	7	7	7	7	7	
	8	32768	8	2048	8	128	8	8	8	128	8	8	8	8	8	8	
	9	36864	9	2304	9	144	9	9	9	144	9	9	9	9	9	9	
	A	40960	A	2560	A	160	A	A	A	160	A	A	A	A	A	A	
	B	45056	B	2816	B	176	B	B	B	176	B	B	B	B	B	B	
	C	49152	C	3072	C	192	C	C	C	192	C	C	C	C	C	C	
	D	53248	D	3328	D	208	D	D	D	208	D	D	D	D	D	D	
	E	57344	E	3584	E	224	E	E	E	224	E	E	E	E	E	E	
	F	61440	F	3840	F	240	F	F	F	240	F	F	F	F	F	F	

ACRONYMS AND ABBREVIATIONS

ACK	Acknowledge
AMPL	Advanced Microprocessor Prototyping Laboratory
BCD	Binary Coded Decimal
BRKDT	Break Detect
CHAR	Character
CL	Capture Latch
CLK	Clock
COMM	Communication Mode
CPU	Central Processing Unit
CROM	Control ROM
DDR	Data Direction Register
EC	Event Counter
ENB	Enable
EPROM	Eraseable Programmable Read Only Memory
ER	Error Reset
FE	Framing Error
FLG	Flag
I/O	Input/Output
IAQ	Instruction Acquisition
IOCNTL	I/O Control Register
LSB	Least Significant Bit/Byte
MC	Mode Control
MOS	Metal Oxide Semiconductor
MSB	Most Significant Bit/Byte
MULTI	Multiprocessor Mode
OE	Overrun Error
PC	Program Counter
PE	Parity Error
PEN	Parity Enable
PEVEN	Parity Even
PF	Peripheral File
PL	Prescaler Latch
PLA	Programmable Logic Array
PWM	Pulse Width Modulation
R/W	Read/Write
RAM	Random Access Memory
RF	Register File
ROM	Read Only Memory
RTC	Real Time Clock or Regional Technology Center
RX	Receiver
RXBUF	Receiver Buffer
RXEN	Receiver Enable
RXRDY	Receiver Ready
SCAT	Strip Chip Architecture Topology
SCLK	Serial Clock
SCTL	Serial Control
SHF	Shift Register
SMODE	Serial Mode
SP	Stack Pointer
SSTAT	Serial Port Status Register
ST	Status Register

STOP	Stop Bit
TCTRL	Timer Control
TDATA	Timer Data
TL	Timer Latch
TTL	Transistor-Transistor Logic
TX	Transmitter
TXBUF	Transmitter Buffer
TXD	Transmit Data
TXE	Transmitter Empty
TXEN	Transmit Enable
TXRDY	Transmitter Ready
TXSHF	Transmitter Shift Register
UR	Software UART Reset
WU	Wake Up
WUT	Wake Up Temporary
XDS	Extended Development Support
XMPL	Extended Microprocessor Prototyping Laboratory
XTAL	Crystal

TI Sales Offices

ALABAMA: Huntsville, 500 Wynn Drive, Suite 514, Huntsville, AL 35805, (205) 837-7530.

ARIZONA: Phoenix, P.O. Box 35160, 8102 N. 23rd Ave., Suite A, Phoenix, AZ 85021, (602) 995-1007.

CALIFORNIA: El Segundo, 831 S. Douglas St., El Segundo, CA 90245, (213) 973-2571; Irvine, 17891 Carwright Rd., Irvine, CA 92714, (714) 660-1200; Sacramento, 1900 Point West Way, Suite 171, Sacramento, CA 95815, (916) 929-1521; San Diego, 4333 View Ridge Ave., Suite B, San Diego, CA 92123, (714) 278-9600; Santa Clara, 5353 Betsy Ross Dr., Santa Clara, CA 95054, (408) 980-9000; Woodland Hills, 21220 Erwin St., Woodland Hills, CA 91367, (213) 704-7759.

COLORADO: Denver, 9725 E. Hampden St., Suite 301, Denver, CO 80231, (303) 695-2800.

CONNECTICUT: Wallingford, 9 Barnes Industrial Park Rd., Barnes Industrial Park, Wallingford, CT 06492, (203) 269-0074.

FLORIDA: Clearwater, 2280 U.S. Hwy. 19 N., Suite 232, Clearwater, FL 33515, (813) 796-1926; Ft. Lauderdale, 2765 N.W. 62nd St., Ft. Lauderdale, FL 33309, (305) 973-8502; Maitland, 2601 Maitland Center Parkway, Maitland, FL 32751, (305) 646-9600.

GEORGIA: Atlanta, 3300 Northeast Expy., Building 9, Atlanta, GA 30341, (404) 452-4600.

ILLINOIS: Arlington Heights, 515 W. Algonquin, Arlington Heights, IL 60005, (312) 642-2934.

INDIANA: Ft. Wayne, 2020 Inwood Dr., Ft. Wayne, IN 46825, (219) 424-5174; Indianapolis, 2346 S. Lyburn, Suite J-400, Indianapolis, IN 46241, (317) 248-8555.

IOWA: Cedar Rapids, 173 Collins Rd. NE, Suite 200, Cedar Rapids, IA 52402, (319) 395-9550.

MARYLAND: Baltimore, 1 Rutherford Pl., 7133 Rutherford Rd., Baltimore, MD 21207, (301) 944-8600.

MASSACHUSETTS: Waltham, 504 Toren Pond Rd., Waltham, MA 02154, (617) 890-7400.

MICHIGAN: Farmington Hills, 33737 W. 12 Mile Rd., Farmington Hills, MI 48018, (313) 553-1500.

MINNESOTA: Edina, 7625 Parklawn, Edina, MN 55435, (612) 830-1600.

MISSOURI: Kansas City, 8080 Ward Pkwy., Kansas City, MO 64114, (816) 523-2500; St. Louis, 11861 Westline Industrial Drive, St. Louis, MO 63141, (314) 569-7600.

NEW JERSEY: Clark, 292 Terminal Ave. West, Clark, NJ 07066, (201) 574-9800.

NEW MEXICO: Albuquerque, 5907 Ace Nise, Suite E., Albuquerque, NM 87110, (505) 265-8491.

NEW YORK: East Syracuse, 6700 Old Collamer Rd., East Syracuse, NY 13057, (315) 463-9291; Endicott, 112 Nanticoke Ave., P.O. Box 618, Endicott, NY 13760, (607) 754-3900; Melville, 11 Huntington Quadrangle, Suite 3C10, P.O. Box 2936, Melville, NY 11747, (516) 464-6600; Poughkeepsie, 201 South Ave., Poughkeepsie, NY 12601, (914) 473-2900; Rochester, 1210 Jefferson Rd., Rochester, NY 14623, (716) 424-5400.

NORTH CAROLINA: Charlotte, 8 Woodlawn Green, Woodlawn Rd., Charlotte, NC 28210, (704) 527-0930; Raleigh, 3000 Highlands Blvd., Suite 118, Raleigh, NC 27625, (919) 876-2725.

OHIO: Beachwood, 23408 Commerce Park Rd., Beachwood, OH 44122, (216) 454-6100; Dayton, Kinley Bldg., 4124 Linden Ave., Dayton, OH 45432, (513) 258-3877.

OKLAHOMA: Tulsa, 3105 E. Skelly Dr., Suite 110, Tulsa, OK 74105, (918) 749-9547.

OREGON: Beaverton, 6700 SW 105th St., Suite 110, Beaverton, OR 97005, (503) 643-6758.

PENNSYLVANIA: Ft. Washington, 575 Virginia Dr., Ft. Washington, PA 19034, (215) 643-6450; Coropola, PA 15108, 420 Router Rd., 3 Airport Office Pk., (412) 771-8550.

TENNESSEE: Johnson City, P.O. Drawer 1255, Erwin Hwy., Johnson City, TN 37601, (615) 461-2191.

TEXAS: Austin, 12501 Research Blvd., P.O. Box 2909, Austin, TX 78723, (512) 250-7655; Dallas, P.O. Box 1087, Richardson, TX 75080; Houston, 9100 Southwest Hwy., Suite 237, Houston, TX 77036, (713) 778-6592; San Antonio, 1300 Central Park South, San Antonio, TX 78232, (512) 496-1779.

UTAH: Salt Lake City, 3672 West 2100 South, Salt Lake City, UT 84120, (801) 973-6310.

VIRGINIA: Fairfax, 3001 Prosperity, Fairfax, VA 22031, (703) 849-1400; Middlehian, 13711 Sutter's Mill Circle, Middlehian, VA 23113, (804) 744-1007.

WISCONSIN: Brookfield, 2025 Bishops Way, Suite 214, Brookfield, WI 53005, (414) 784-3040.

WASHINGTON: Redmond, 2723 152nd Ave., N.E. Bldg 6, Redmond, WA 98052, (206) 881-3080.

CANADA: Ottawa, 436 Mac Laren St., Ottawa, Canada, K2P0M8, (613) 233-1177; Richmond Hill, 280 Centre St. E., Richmond Hill L4C1B1, Ontario, Canada, (416) 884-9181; St. Laurent, Ville St. Laurent Quebec, 9460 Truro Canada Hwy. St. Laurent, Quebec, Canada H4S1R7, (514) 334-3635. **B**

INDIANA: Indianapolis, Arrow (317) 243-9353; Graham (317) 634-8202; Ft. Wayne, Graham (317) 423-3422.

IOWA: Arrow (319) 395-7220.

KANSAS: Kansas City, Component Specialties (913) 492-3555; Hall-Mark (913) 888-4747; Wichita, LCOMP (316) 265-9077.

MARYLAND: Arrow (301) 247-5200; Diplomat (301) 927-1226; Kierulf (617) 667-8311; Marshall (617) 272-8200; Milgray (301) 468-6400.

MASSACHUSETTS: Arrow (617) 933-8130; Diplomat (617) 429-4420; Kierulf (617) 667-8311; Marshall (617) 272-8200; Time (617) 933-8080.

MICHIGAN: Detroit, Arrow (313) 971-8220; Newark (313) 967-0600; Grand Rapids, Newbit (616) 243-0217; Arrow (616) 243-0912.

MINNESOTA: Arrow (612) 830-1800; Hall-Mark (612) 854-3233; Kierulf (612) 941-7500.

MISSOURI: Kansas City, LCOMP (816) 221-2400; St. Louis, Arrow (314) 567-8888; Hall-Mark (314) 291-5350; Kierulf (314) 739-8855.

NEW HAMPSHIRE: Arrow (603) 668-6968.

NEW JERSEY: Arrow (201) 575-5300, (609) 235-1900; Diplomat (201) 785-1830; General Radio (609) 964-8360; Hall-Mark (201) 575-4415, (609) 424-7300; JACO (201) 778-4722; Kierulf (201) 575-6750; Marshall (201) 881-0320; Milgray (609) 983-5010.

NEW MEXICO: Arrow (505) 243-4566; International Electronics (505) 345-8172.

NEW YORK: Long Island, Arrow (516) 231-1000; Diplomat (516) 454-6334; Hall-Mark (516) 737-0600; JACO (516) 273-5500; Marshall (516) 273-2424; Milgray (516) 546-5600, (603) 645-3986; Hall-Mark (516) 737-0600; Rochester, Arrow (716) 275-0300; Marshall (716) 235-7620; Rochester Radio Supply (716) 454-7800; Syracuse, Arrow (315) 452-1000; Diplomat (315) 652-5000; Marshall (607) 754-1570.

NORTH CAROLINA: Arrow (919) 876-3132, (919) 725-8711; Hall-Mark (919) 872-0712; Kierulf (919) 852-9440.

OHIO: Cincinnati, Graham (513) 772-1661; Hall-Mark (513) 563-5980; Cleveland, Arrow (216) 248-3990; Hall-Mark (216) 473-2907; Kierulf (216) 587-6558; Columbus, Hall-Mark (614) 891-4555; Dayton, Arrow (513) 435-5563; ESCO (513) 276-1133; Marshall (513) 236-8088.

OKLAHOMA: Arrow (918) 665-7700; Component Specialties (918) 664-2820; Hall-Mark (918) 665-3200; Kierulf (918) 252-7537.

OREGON: Kierulf (503) 641-9150; Wyle (503) 640-6000.

PENNSYLVANIA: Arrow (412) 856-7000, (215) 928-1800; General Radio (215) 922-7037; Hall-Mark (215) 355-7300.

TEXAS: Austin, Arrow (512) 835-4180; Component Specialties (512) 837-8922; Hall-Mark (512) 258-8848; Kierulf (512) 835-2090; Dallas, Arrow (214) 386-7000; Component Specialties (214) 357-5511; Hall-Mark (214) 341-1147; International Electronics (214) 233-9323; Kierulf (214) 343-2400; El Paso, International Electronics (915) 778-9761; Houston, Arrow (713) 491-4100; Component Specialties (713) 771-7237; Hall-Mark (713) 781-0020; Harrison Equipment (713) 879-2600; Kierulf (713) 530-7030.

UTAH: Diplomat (801) 486-4134; Kierulf (801) 973-6913; Wyle (801) 974-9953.

VIRGINIA: Arrow (804) 282-0413.

WASHINGTON: Arrow (206) 643-4800; Kierulf (206) 575-4420; Wyle (206) 453-8300.

WISCONSIN: Arrow (414) 784-6600; Hall-Mark (414) 761-3000; Kierulf (414) 784-8100.

CANADA: Calgary, Future (403) 259-6408; Varah (403) 230-1235; Hamilton, Varah (416) 561-9311; Montreal, CESCO (514) 753-5511; Future (514) 694-7710; Ottawa, CESCO (613) 226-6905; Future (613) 820-8313; Quebec City, CESCO (418) 687-4231; Toronto, CESCO (416) 663-0220; Future (416) 663-5563; Vancouver, Future (604) 438-5545; Varah (604) 873-3211; Winnipeg, Varah (204) 633-6190. **BB**

TI Distributors

ALABAMA: Hall-Mark (205) 837-8700.

ARIZONA: Phoenix, Kierulf (602) 243-4101; Marshall (602) 968-6181; Wyle (602) 249-2232; Tucson, Kierulf (602) 624-9966.

CALIFORNIA: Los Angeles/Orange County, Arrow (213) 701-7500, (714) 851-8961; IEC/JACO (714) 540-5600, (213) 998-2200; Kierulf (213) 725-0325, (714) 731-5711; Marshall (213) 999-5021, (213) 442-7204, (714) 556-6400, R.V. Weatherford (714) 634-9600, (213) 849-3451, (714) 623-1261; Wyle (213) 322-8100, (714) 641-1600, San Diego, Arrow (619) 565-4800; Kierulf (619) 278-2112; Marshall (619) 578-9600; R. V. Weatherford (619) 695-1200; Wyle (619) 565-9171; San Francisco Bay Area, Arrow (408) 745-6600; Kierulf (415) 968-6292; Marshall (408) 732-1100; Wyle (408) 727-2500; Santa Barbara, R. V. Weatherford (605) 965-8551.

COLORADO: Arrow (303) 758-2100; Kierulf (303) 790-4444; Wyle (303) 457-9953.

CONNECTICUT: Arrow (203) 265-7741; Diplomat (203) 797-9674; Kierulf (203) 265-1115; Marshall (203) 265-3822; Milgray (203) 795-0714.

FLORIDA: Ft. Lauderdale, Arrow (305) 776-7790; Diplomat (305) 971-7160; Hall-Mark (305) 971-9280; Kierulf (305) 652-6950; Orlando, Arrow (305) 725-1480; Diplomat (305) 729-4520; Hall-Mark (305) 855-0020; Milgray (305) 647-8747; Tampa, Diplomat (813) 443-4514; Hall-Mark (813) 576-8691; Kierulf (813) 576-1966.

GEORGIA: Arrow (404) 449-8252; Hall-Mark (404) 447-8000; Kierulf (404) 447-5252; Marshall (404) 923-5750.

ILLINOIS: Arrow (312) 397-3440; Diplomat (312) 995-1000; Hall-Mark (312) 860-3800; Kierulf (312) 640-0200; Newark (312) 638-4411.



Creating useful products and services for you.

TI Worldwide Sales Offices

ALABAMA: Huntsville, 500 Winn Drive, Suite 514, Huntsville, AL 35805, (205) 837-7530.

ARIZONA: Phoenix, P.O. Box 35160, 8102 N. 23rd Ave., Suite A, Phoenix, AZ 85021, (602) 995-1007.

CALIFORNIA: El Segundo, 831 S. Douglas St., El Segundo, CA 90245, (213) 973-2571; Irvine, 17891 Cornerwright Rd., Irvine, CA 92714, (714) 660-1200; Sacramento, 1900 Point West Way, Suite 171, Sacramento, CA 95815, (916) 929-1571; San Diego, 4333 Vene Ridge Ave., Suite B, San Diego, CA 92123, (714) 278-9600; Santa Clara, 5353 Berry Road Dr., Santa Clara, CA 95054, (408) 980-9000; Woodland Hills, 21220 Erwin St., Woodland Hills, CA 91367, (213) 704-7759.

COLORADO: Denver, 9725 E. Hampden St., Suite 301, Denver, CO 80231, (303) 695-2800.

CONNECTICUT: Wallingford, 9 Barnes Industrial Park Rd., Barnes Industrial Park, Wallingford, CT 06492, (203) 269-0074.

FLORIDA: Clearwater, 2280 U.S. Hwy. 19 N., Suite 232, Clearwater, FL 33515, (813) 796-1926; Ft. Lauderdale, 2765 N.W. 62nd St., Ft. Lauderdale, FL 33309, (305) 973-8502; Maitland, 2601 Maitland Center Parkway, Maitland, FL 32751, (305) 646-9600.

GEORGIA: Atlanta, 3300 Northeast Expy., Building 9, Atlanta, GA 30341, (404) 452-4600.

ILLINOIS: Arlington Heights, 515 W. Algonquin, Arlington Heights, IL 60005, (312) 660-2930.

INDIANA: Ft. Wayne, 2020 Inwood Dr., Ft. Wayne, IN 46805, (219) 424-5174; Indianapolis, 2366 S. Lynnhurst, Suite J-400, Indianapolis, IN 46241, (317) 248-8555.

IOWA: Cedar Rapids, 373 Collins Rd. NE, Suite 200, Cedar Rapids, IA 52402, (319) 995-9550.

MARYLAND: Baltimore, 1 Rutherford Pl., 7133 Rutherford Rd., Baltimore, MD 21207, (301) 944-8600.

MASSACHUSETTS: Waltham, 504 Totten Pond Rd., Waltham, MA 02154, (617) 890-7400.

MICHIGAN: Farmington Hills, 33737 W. 12 Mile Rd., Farmington Hills, MI 48018, (313) 551-1500.

MINNESOTA: Edina, 7625 Parklawn, Edina, MN 55435, (612) 830-1600.

MISSOURI: Kansas City, 8080 Ward Pkwy., Kansas City, MO 64114, (816) 523-2500; St. Louis, 11861 Westline Industrial Drive, St. Louis, MO 63141, (314) 569-7600.

NEW JERSEY: Clark, 292 Terminal Ave. West, Clark, NJ 07066, (201) 574-9800.

NEW MEXICO: Albuquerque, 5907 Alice NSE, Suite E., Albuquerque, NM 87110, (505) 265-8491.

NEW YORK: East Syracuse, 6700 Old Collamer Rd., East Syracuse, NY 13057, (315) 463-9291; Endicott, 112 Nianticola Ave., P.O. Box 618, Endicott, NY 13760, (607) 754-3900; Melville, 1 Huntington Quadrangle, Suite 3010, P.O. Box 2936, Melville, NY 11747, (516) 454-6000; Poughkeepsie, 201 South Ave., Poughkeepsie, NY 12601, (914) 473-2900; Rochester, 1210 Jefferson Rd., Rochester, NY 14623, (716) 424-5400.

NORTH CAROLINA: Charlotte, 8 Woodlawn Green, Woodlawn Rd., Charlotte, NC 28210, (704) 527-0930; Raleigh, 3000 Highlands Blvd., Suite 118, Raleigh, NC 27625, (919) 876-2725.

OHIO: Beachwood, 23408 Commerce Park Rd., Beachwood, OH 44122, (216) 464-6100; Dayton, Kingsley Bldg., 4124 Linden Ave., Dayton, OH 45432, (513) 258-3877.

OKLAHOMA: Tulsa, 3105 E. Skelly Dr., Suite 110, Tulsa, OK 74105, (918) 749-9547.

OREGON: Beaverton, 6700 SW 105th St., Suite 110, Beaverton, OR 97005, (503) 643-6758.

PENNSYLVANIA: Ft. Washington, 575 Virginia Rd., Ft. Washington, PA 19034, (215) 643-6450; Conrospolia, PA 15108, 420 Rouser Rd., 3 Airport Office Pk., (412) 711-8550.

TENNESSEE: Johnson City, P.O. Drawer 1255, Erwin Hwy., Johnson City, TN 37601, (615) 461-2191.

TEXAS: Austin, 12501 Research Blvd., P.O. Box 2909, Austin, TX 78733, (512) 250-7655; Dallas, P.O. Box 1087, Richardson, TX 75080; Houston, 9100 Southwest Freeway, Suite 237, Houston, TX 77036, (713) 778-6922; San Antonio, 1000 Central Park South, San Antonio, TX 78232, (512) 496-1779.

UTAH: Salt Lake City, 3672 West 2100 South, Salt Lake City, UT 84120, (801) 973-6310.

VIRGINIA: Fairfax, 3001 Prosperity, Fairfax, VA 22031, (703) 489-1400; Midlothian, 13711 Surreys Mill Circle, Midlothian, VA 23113, (804) 744-1007.

WISCONSIN: Brookfield, 205 Bishops Way, Suite 214, Brookfield, WI 53005, (414) 784-3040.

WASHINGTON: Redmond, 2723 152nd Ave. N.E. Bldg 3, Redmond, WA 98052, (206) 881-3080.

CANADA: Ottawa, 436 MacLaren St., Ottawa, Canada, K2P0M8, (613) 233-1177; Richmond Hill, 280 Centre St. E., Richmond Hill, ON L4B 1B1, Ontario, Canada, (416) 894-9181; St. Laurent, Ville St. Laurent Quebec, 9460 Trans Canada Hwy., St. Laurent, Quebec, Canada H4S1R7, (514) 334-3635. B

ARGENTINA: Texas Instruments Argentina S.A.I.C.F.: Esmeralda 130, 15th Floor, 1035 Buenos Aires, Argentina, 194-2963.

AUSTRALIA (& NEW ZEALAND): Texas Instruments Australia Ltd., 6-10 Talavera Rd., North Ryde (Sydney), New South Wales, Australia 2113, 02 + 887-1122; 5th Floor, 418 St. Kilda Road, Melbourne, Victoria, Australia 3004, 03 + 267-4677; 171 Philip Highway, Elizabeth, South Australia 5112, 08 + 255-2066.

AUSTRIA: Texas Instruments Ges.m.b.H.: Industriestrasse B/16, A-2345 Brunn-Gebrige, 2236-846210.

BELGIUM: Texas Instruments N.V. Belgium S.A.: Mercure Centre, Fekestraat 100, Rue de la Fusee, 1130 Brussels, Belgium, 02720-8000.

BRAZIL: Texas Instruments Eletronicos do Brasil Ltda.: Av. Pina Lima, 2003, 200 0 Andar—Pinheiros, Cep-01451 Sao Paulo, Brazil, 815-6166.

DENMARK: Texas Instruments A/S, Marielundvej 46E, DK-2730 Herlev, Denmark, 2-91 74 00.

FINLAND: Texas Instruments Finland OY: PL 56, 00510 Helsinki 51, Finland, (90) 7013133.

FRANCE: Texas Instruments France: Headquarters and Prod. Plant, BP 05, 06270 Villeneuve-Loubet, (93) 20-01-01, Paris Office, BP 67 8-10 Avenue Morane-Saulnier, 78141 Velizy-Villacoublay, (33) 946-97-12; Lion Sales Office, L'Orvee D'Ecally, Batiment B, Chemin de la Foresterie, 69130 Ecully, (7) 833-04-40; Strasbourg Sales Office, Le Sebastopol 3, Quai Kleber, 67055 Strasbourg Cedex, (88) 22-12-66; Rennes, 23-25 Rue du Puits Manager, 35100 Rennes, (99) 79-54-81; Toulouse Sales Office, Le Penipole—2, Chemin du Pigeonnier de La Cepiere, 31100 Toulouse, (61) 44-18-19; Marseille Sales Office, Noilly Paradis—146 Rue Paradis, 13006 Marseille, (91) 37-25-30.



TEXAS INSTRUMENTS

Creating useful products and services for you

GERMANY: Texas Instruments Deutschland GmbH: Hauptstrasse 1, D-8050 Freising, 08161-803; Kurt-Fuentermann 195/196, D-1000 Berlin 15, 030-8627365; III, Heegen 43/66b-Heilstrasse, D-4300 Essen, 0201-24250; Frankfurt/Allee 6-8, D-6236 Eschborn, 1, 06196-43074; Franzfurter Strasse 11, D-2020 Hamburg 76, 040-2201154; Kirchchornstrasse 2, D-3000 Hannover 51, 0511-648001; Annelstrasse 15, D-8000 Muenchen 81, 089-92341; Maybachstrasse 11, D-7302 Dietzenhofen 2/Nellingen, 0711-34030.

HONG KONG (& PEOPLES REPUBLIC OF CHINA): Texas Instruments Asia Ltd.: 8th Floor, World Shipping Ctr., Harbour City, 7 Canton Rd., Kowloon, Hong Kong, 3 + 722-1223.

IRELAND: Texas Instruments (Ireland) Limited: 25 St. Stephens Green, Dublin 2, Eire, 01 609222.

ITALY: Texas Instruments Semiconductor Italia Spa: Viale Delle Scienze, 1, 02015 Citraducate (Rieti), Italy, 0746 694-1; Via Salario KM 24 (Palazzo Coema), Montetronico Scalo (Rome), Italy, 06 9004395; Viale Europa, 38-44, 20093 Colongo Monzese (Milano), 02 2532541; Onno Sviners, 185, 1100 Torino, Italy, 011 774545; Via J. Barozzi, 6, 45100 Bologna, Italy, 051 355851.

JAPAN: Texas Instruments Asia Ltd.: 4F Aoyama Fuji Bldg., 6-12, Kita Aoyama 3-Chome, Minato-ku, Tokyo, Japan 107, (03) 498-2111; Osaka Branch, 5F Niakio Iwai Bldg., 30 Iwashiki 3-Chome, Higashi-ku, Osaka, Japan 541, 06-204-1881; Nagoya Branch, 7F Daini Toyota West Bldg., 10-27, Meieki 4-Chome, Nakamura-ku, Nagoya, Japan 450, 052-583-8691.

KOREA: Texas Instruments Supply Co.: Room 201, Kwangpoong Bldg., 24-1, Hwaryeong-Dong, Sungdong-ku, 133 Seoul, Korea, 02 + 464-6274/5.

MEXICO: Texas Instruments de Mexico S.A.: Poniente 116, No. 489, Colonia Vallejo, Mexico, D.F. 02300, 567-9200.

MIDDLE EAST: Texas Instruments: No. 13, 1st Floor Manal Bldg., Diplomatic Area, Manama, P.O. Box 26335, Bahrain, Arabian Gulf, 973-72 46 81.

NETHERLANDS: Texas Instruments Holland B.V., P.O. Box 12995, (Bollewijk) 1100 AZ Amsterdam, Zuid-Oost, Holland (020) 5602911.

NORWAY: Texas Instruments Norway A/S: Kr. Augustsgat. 13, Oslo 1, Norway, (2) 20 60 40.

PHILIPPINES: Texas Instruments Asia Ltd.: 14th Floor, Balapiano Bldg., 8747 Paseo de Roxas, Makati, Metro Manila, Philippines, 882465.

PORTUGAL: Texas Instruments Equipments Electronica (Portugal) Lda: Rua Eng. Frederico Ulrich, 2650 Moreira Da Maia, 4470 Maia, Portugal, 2-9481003.

SCOTLAND: Texas Instruments Limited: 126-128 George Street, Edinburgh, Scotland, EH1 2AN, 031 226 2691.

SINGAPORE (& INDIA, INDONESIA, MALAYSIA, THAILAND): Texas Instruments Asia Ltd.: P.O. Box 138, Unit #02-08, Block 6, Kolam Ayer Industrial Est., Kallang Sector, Singapore 1334, Republic of Singapore, 747-2353.

SPAIN: Texas Instruments Espana, S.A.: C/Jose Lazo Galdino No. 6, Madrid 16, 1458-14 58. C/Balmes, 89 Barcelona-8, 253 60 00/253 29 02.

SWEDEN: Texas Instruments International Trade Corporation (Sverigefilialen): Box 39103, 10054 Stockholm, Sweden, 08-235480.

SWITZERLAND: Texas Instruments, Inc. Riedstrasse 6, CH-8953 Dietikon (Zuerich) Switzerland, 1-740 2220.

TAIWAN: Texas Instruments Supply Co.: 10th Floor, Fushing Bldg., 71 Sung-Kiang Road, Taipei, Taiwan, Republic of China, 02 + 521-9321.

UNITED KINGDOM: Texas Instruments Limited: Manton Lane, Bedford, MK41 7PA, England, 0234 67466; 186 High Street, Slough, SL1 1JD, England, 0753 35545; St. James House, Wellington Road North, Stockport, SK4 2RT, England, 061 442-8448. BB

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES



TEXAS INSTRUMENTS

Creating useful products
and services for you.

April 1984
Revision A
1603481-9701
Printed in U.S.A.

SPND001A