

SONY®

Semiconductor IC

**Data Book
1986**

Microprocessors

SONY

Microprocessors

1986

SONY®

**Semiconductor Integrated Circuit Data Book
1986**

List of Model Names/
Index by Usage

1

Description

2

Microprocessors

3

Peripherals

4

Semiconductor Integrated Circuit Data Book 1986

SONY®

PREFACE

This is the 1986 version of the Sony semiconductor IC databook. This book covers all the semiconductor products manufactured and marketed by Sony.

In preparation of this databook, as much characteristic and application data as possible have been collected and added with a view of making this book a convenient reference for users of Sony products. If, however, you are dissatisfied with this book in any way, please write; we welcome suggestions and comments.

The Sony semiconductor IC databook has been edited to include only accurate and reliable data. However, because of technical improvements and other modifications the contents are subject to change without notice.

The circuit examples used in this book are for illustration of typical applications only; we are not responsible for any problems that may occur in the circuitry and patents of any third party if these examples are put in practice.

Package abbreviations

DIP : Dual Inline Package

MFP : Mini Flat Package (=Flat DIP)

QIP : Quad Inline Package (=Flat QUIP)

PGA : Pin Grid Array

SRK : Shrink Dual Inline Package

Contents

	Page
1. List of Model Names and Index by usage	(6)
2. IC Nomenclature	(7)
3. Precautions for IC Application	(8)
A) Absolute maximum ratings	
B) Protection against electrostatic breakdown	
C) Mounting method	
4. Quality Assurance and Reliability	(14)
5. Block Diagram	(18)
6. Data Sheet	(19)
Microprocessors	
Microprocessor peripherals	

1. List of Model Names and Index by Usage

Type	Page	Type	Page	Type	Page
CXQ70108	21	CXQ71054	182	CXQ71071	264
CXQ70116	71	CXQ71055	207	CXQ71086/ CXQ71087	137
CXQ71011	123	CXQ71059	230	CXQ71088	142
CXQ71051	150	CXQ71082/ CXQ71083	132		

■ Microprocessors

Type	Function	Page
CXQ70108	8-bit Microprocessor	21
CXQ70116	16-bit Microprocessor	71

■ Peripherals

Type	Function	Page
CXQ71011	Clock pulse Generator/driver	123
CXQ71082	8-bit latch (non invert)	132
CXQ71083	8-bit latch (invert)	132
CXQ71086	8-bit bus driver/receiver (non invert)	137
CXQ71087	8-bit bus driver/receiver (invert)	137
CXQ71088	System bus controller	142
CXQ71051	Serial Interface unit	150
CXQ71054	Programmable timer/counter	182
CXQ71055	Parallel interface unit	207
CXQ71059	Interrupt control unit	230
CXQ71071	DMA controller	264

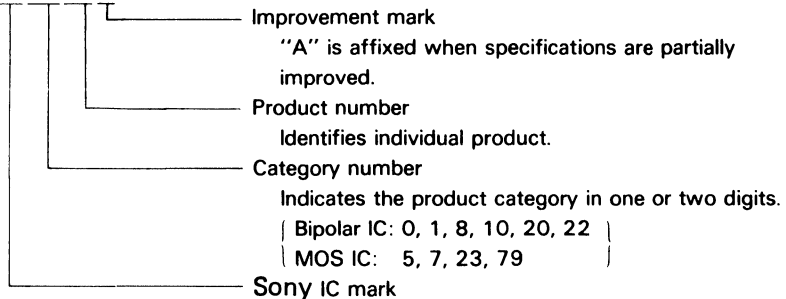
2. IC Nomenclature

1. Nomenclature of IC product name

Currently, both the conventional and new nomenclature systems are mixed in naming IC products.

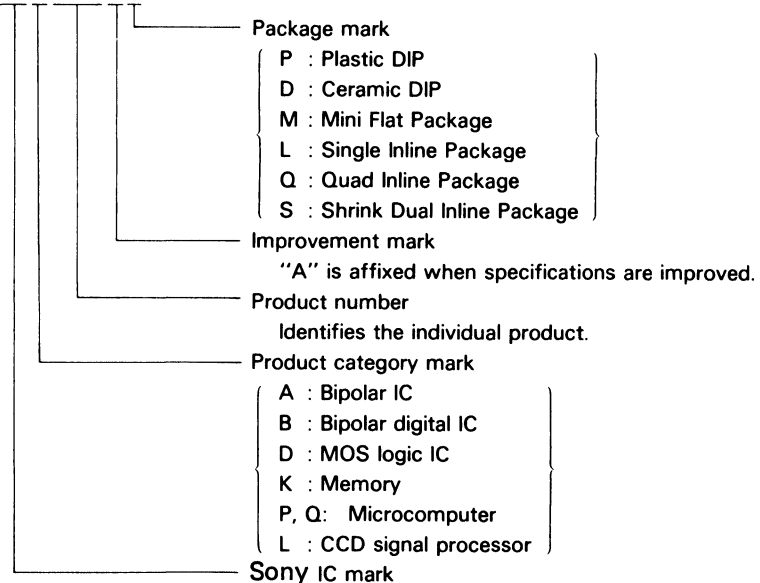
a) Conventional nomenclature system

[Example] CX20014A



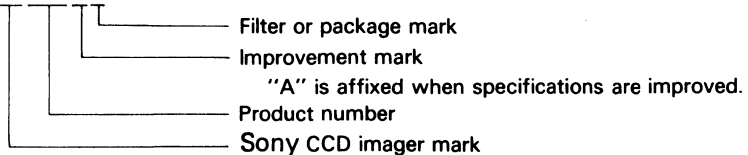
b) New nomenclature

[Example] CXA1001AP



2. Nomenclature for CCD image product name

[Example] ICX016AK



3. Precautions for IC Application

A) Absolute maximum ratings

The maximum ratings for semiconductor devices are normally specified by "absolute maximum ratings". The values shown in the maximum ratings table must never be exceeded even in a moment.

If the maximum rating is ever exceeded, device deterioration or damage will occur immediately. Then, even if the affected device can operate, the life will be considerably short.

Maximum rating must never be reached for two items at the same time.

IC maximum ratings

The following maximum ratings are used for ICs.

(1) Maximum power supply voltage V_{CC} (V_{DD})

The maximum voltage that can be applied between the power supply terminal and ground terminal.

This power supply voltage rating is directly related to the dielectric voltage of transistors in the internal circuit; the transistors may be destroyed if this voltage is exceeded.

(2) Allowable power dissipation P_D

The maximum power consumption allowed in IC

In the circuit design the absolute maximum ratings must not be exceeded, and it must be designed only after considering the worst situations among the following:

- Fluctuation in source voltage
- Scattering in the electrical characteristics of electrical parts (transistors, resistors, capacitors, etc.)
- Power dissipation in circuit adjustment
- Ambient temperature
- Fluctuation in input signal
- Abnormal pulses

If this allowable power dissipation is exceeded, electrical and thermal damage may result.

This value varies with the amount of IC integration in package types.

(3) Operating ambient temperature T_{opr}

The temperature range within which IC can operate satisfactorily.

Even if this temperature range is exceeded and some deterioration in operating characteristics is noted, the IC is not always damaged.

For some ICs, the electrical characteristics at $T_a=25^\circ\text{C}$ are not guaranteed even in this temperature range.

(4) Storage temperature T_{stg}

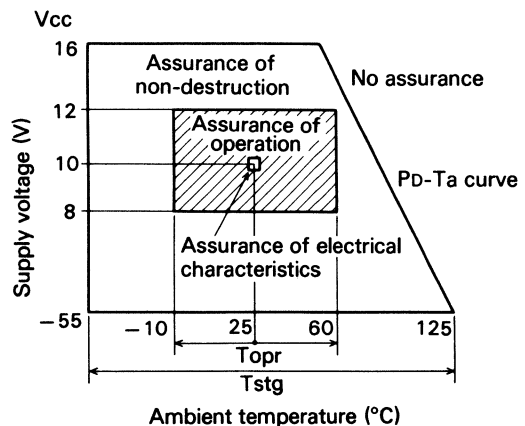
The temperature range for storing the IC which is not operating.

This temperature is restricted by the package material, and the intrinsic properties of the semiconductor.

(5) Other values

The input voltage V_{in} , output voltage V_{out} , input current I_{in} , output current I_{out} and other values may be specified in some IC's.

The relationship among these maximum ratings for IC is shown below.



B) Protection against electrostatic breakdown

There have been problems of electrostatic destruction of electronic devices since the 2nd World War. Their history is closely related to the advancement in the semiconductor devices; that is, with the development of semiconductor technology, new problems in electrostatic destruction have arisen. This situation, perhaps, can be understood by recalling the case of MOS FET.

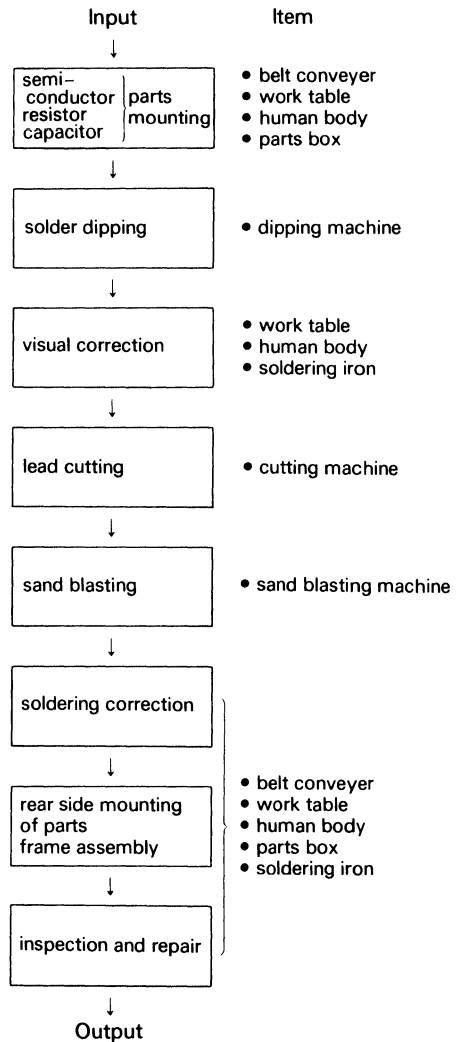
Today, the problem of electrostatic destruction is again drawing people's attention as we are entering the era of LSI and VLSI. Here are our suggestions for preventing electrostatic destruction in the device fabrication process.

Factors causing electrostatic generation in manufacture process

A number of dielectric materials are used in manufacture process. Friction of these materials with the substrate can generate static electricity which may destroy the semiconductor device.

Factors that can cause electrostatic destruction in the manufacture process are shown below:

Causes of electrostatic destruction of semiconductor parts in manufacture process



Handling precautions for preventing electrostatic destruction

Explained below are procedures that must be taken in fabrication for preventing the electrostatic destruction of semiconductor devices.

The following basic rules must be obeyed.

- ① Equalize potentials of terminals when transporting or storing.
- ② Equalize the potentials of the electric device, work bench, and operator's body that may come in contact with the semiconductor device.
- ③ Prepare an environment that does not generate static electricity.
One method is keeping relative humidity in the work room about 50%.

Operator

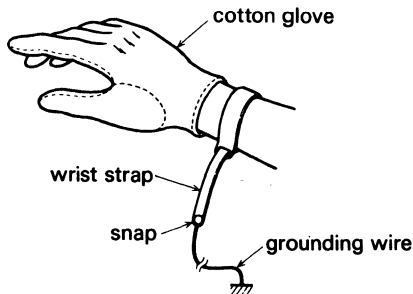
1) Clothes

Do not use nylon, rubber and other materials which easily generate static electricity. For clothes, use cotton, or antistatic-treated materials. Wear gloves during operation.

2) Grounding of operator's body

The operator should connect the specified wrist strap to his arm. If wrist strap cannot be used, then the operator should touch the grounding point with his hand, before handling any semiconductor device.

example of grounding band

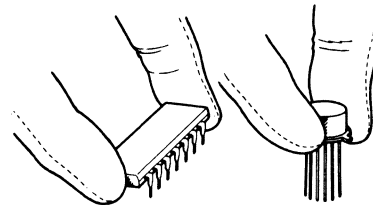


When using a copper wire for grounding, connect a 1M resistance in series near the hand for safety.

3) Handling of semiconductor device

Do not touch the lead. Touch the body of semiconductor device when holding. Limit the number of handling times to a minimum. Do not take the device out of the magazine or package box unless it is absolutely necessary.

holding of semiconductor device



DIP type

can type

Equipment and tools

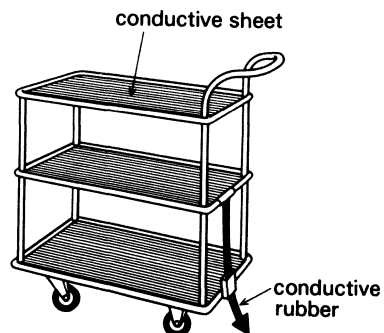
1) Grounding of equipment and tools

Ground the equipments and tools that are to be used. Check insulation beforehand to prevent leakage.

[Check point]

- measuring instrument
- conveyer
- electric deburr brush
- carrier
- solder dipping tank
- lead cutter
- shelves and racks

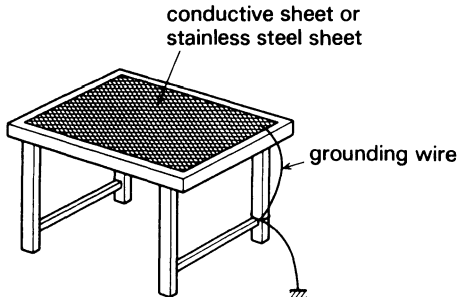
grounding of carrier



2) Grounding of work table

Ground the work table as illustrated. Do not put anything which can easily generate static electricity, such as foam styrol, on the work table.

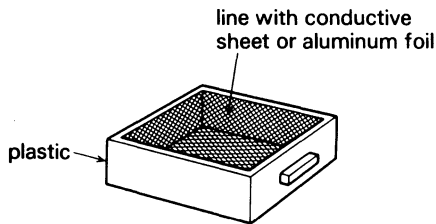
grounding of work table



3) Semiconductor device case

Use the metal case, or the antistatic plastic case (lined with conductive sheet or aluminum foil).

plastic case for semiconductor devices



4) Insertion of semiconductor device

Insert the semiconductor device in mounting process or on the belt conveyer. The insertion should be done on a conductive sheet, or a wood or metal carrier.

5) Operation in energized state

When the substrate is checked while energizing the substrate where the delicate semiconductor device is mounted, be sure to place the substrate on corrugated cardboard, wood, or on a metal carrier.

6) Other points of caution

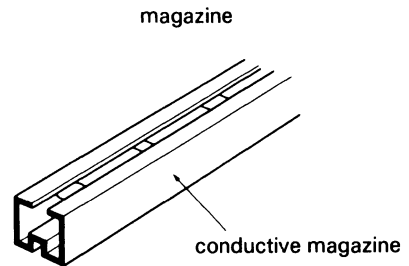
Take note of the kind of the brush material used for removing lead chips. Use metal or antistatic-treated plastic brushes.

Transporting, storing and packaging methods

1) Magazine

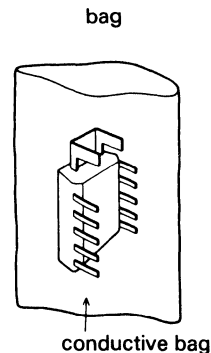
Use the metal, or antistatic-treated plastic IC magazines.

The plastic magazines used for shipping ICs are antistatic-treated, and they can be used for storing ICs.



2) Bag

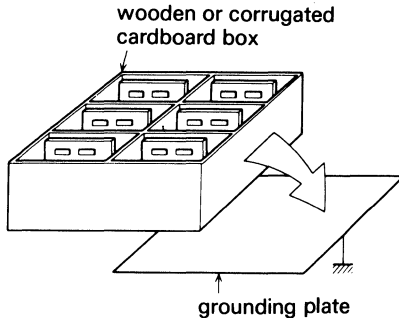
Use a conductive bag for keeping ICs. If use of a vinyl bag is unavoidable, be sure to wrap the IC with aluminum foil.



3) Handling of delivery box

The delivery box used for carrying substrates must be made of wood or corrugated cardboard. Do not use a vinyl chloride or acrylic delivery box, otherwise static electricity will be generated.

handling of delivery box



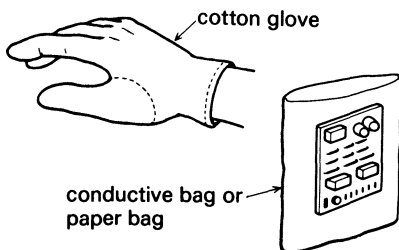
4) Treatment after vehicle transport

After truck transport, place the magazine, package box or delivery box on the grounded rack, work table, or concrete floor for discharging. Do not pull the delivery box for more than 1 meter except on a concrete or a wooden floor.

5) Handling of mounted substrates

Wear cotton gloves when handling. As far as possible, avoid touching soldered faces. When handling mounted substrates individually, be sure to use a conductive or paper bag. Do not use a polyethylene bag.

handling of mounted substrate



Soldering operation

1) Soldering iron

Use a soldering iron with a grounded metal part or a soldering iron whose insulation resistance after five minutes from energizing is greater than 10 M Ω (DC 500V).

2) Operation

After inserting the semiconductor device into the substrate, solder it as quickly as possible. Do not carry the substrate with the inserted semiconductor device by car.

3) Correction

When correcting parts (semiconductor device and CR parts) after solder-dipping, be sure to wear cotton gloves. Also, connect the grounding band to the arm, or touch the grounding point before operation.

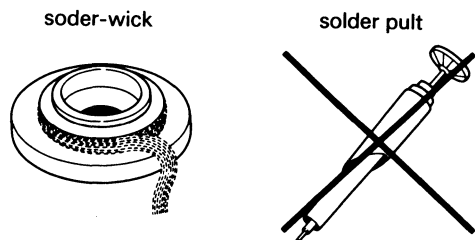
4) Manual soldering

Solder with wrist strap connected to the hand, or by touching the grounding point from time to time during operation.

5) Removing semiconductor device

Do not use the Solder-Pult when removing the semiconductor device. Use a Solder-wick or equivalent.

solder remover



6) Soldering work table

Use a grounded work table, corrugated cardboard, or wooden work table for soldering. Do not solder on foam styrol, vinyl, or decorative board.

C) Mounting method

Soldering and solderability

(1) Solderability guaranteed by JIS

JIS specifies solderability of an IC terminal (lead) in "JIS-C7022 Test Procedure A-2". An abstract of this standard follows:

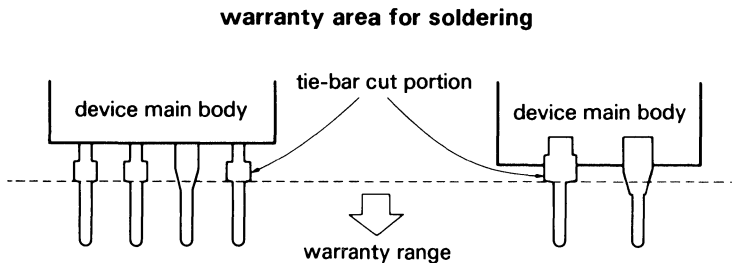
- Rosin flux must be used, and the terminal must be dipped in it for 5-10 seconds.
- H63A or equivalent solder must be used, and the terminal must be dipped in the solder which has been heated to $230^{\circ}\text{C} \pm 5^{\circ}\text{C}$ for 5 ± 1 seconds.
- Using a microscope, measure the area (%) deposited with solder. JIS specifies that more than 95% of the

total area should be coated with solder.

(2) Area for soldering warranty

Soldering is warranted for a specific portion of the terminal. The warranted portion is shown in the following figure.

The tie-bar cut portion also serves as a dam to prevent the sealing resin flowing out during device fabrication; it is cut off at the end of the process. Since the terminal is exposed at the cut-off end, the area for soldering is restricted. The portion near the resin is often covered with burrs when sealing with resin; it is not in the soldering warranty area.



Resistance to soldering heat

(1) Specification of JIS

JIS specifies the method for testing the resistance to soldering heat. This method is used for guaranteeing the IC resistance against thermal stresses by soldering. An abstract of this standard is as follows:

- Dip the device terminal only once for 10 ± 1 seconds in a solder bath of $260^{\circ}\text{C} \pm 5^{\circ}\text{C}$, or for $3_{-0}^{+0.5}$ seconds in a solder bath of $350^{\circ}\text{C} \pm 10^{\circ}\text{C}$, for a distance of up to 1 to 1.5 mm from the main body.

The temperature of $260^{\circ}\text{C} \pm 5^{\circ}\text{C}$ assumes the soldering with solder flow system, and the temperature $350^{\circ}\text{C} \pm 10^{\circ}\text{C}$ assumes soldering by soldering iron.

- Leave the device for more than two hours after dipping, then measure the device characteristics.
- Normally, the warranty is limited for 10 seconds at $260^{\circ}\text{C} \pm 5^{\circ}\text{C}$. The distance between the device main body and solder bath is assumed as 1.6 mm.

4. Quality Assurance and Reliability

Sony's Policy of Quality Assurance

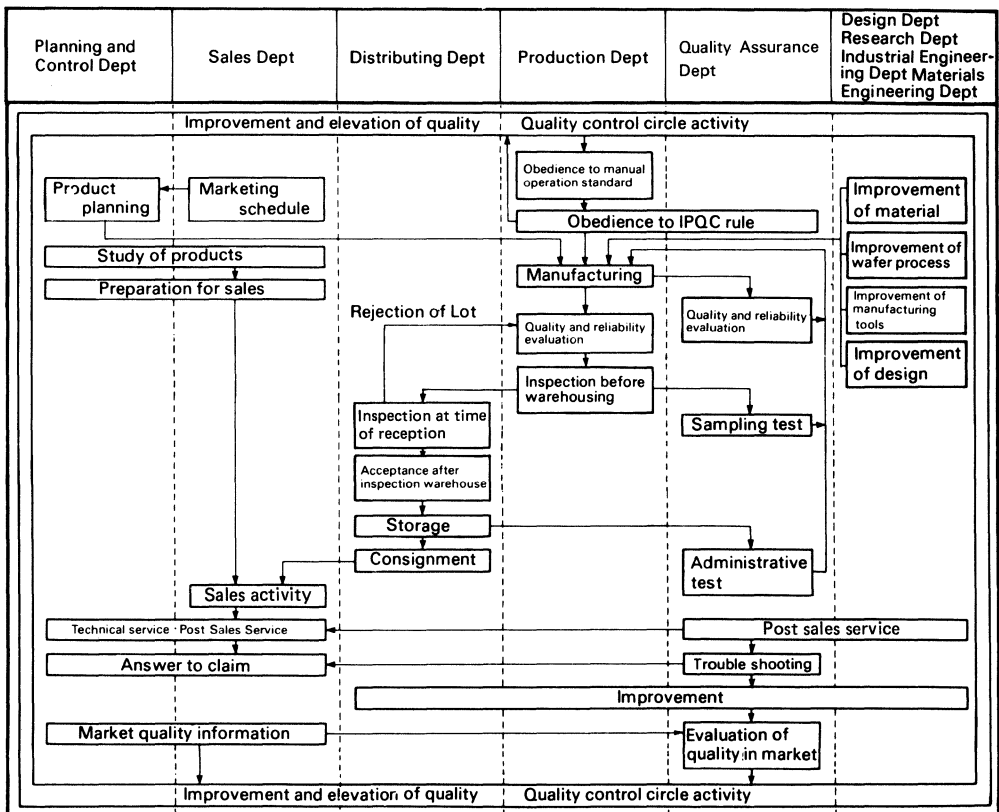
The Sony semiconductor embodies two fundamental ideas: "highest quality" and "lowest cost". There are the two key points for realizing these ideas.

One is the "quality" of men fabricating the semiconductor devices. The reliability of these people is reflected in the Sony products. Accordingly, Sony is making a continuous effort to raise the "quality" of people capable of manufacturing and fabricating Sony semiconductor devices.

The other point is a source management system combined with the concept of thorough quality design. With this system, higher quality products can be steadily manufactured through automation of device design, process design, and the fabrication process.

Sony is making constant efforts to supply the most economical and most useful products of very high quality for users.

Quality assurance system of semiconductor products



Quality assurance criteria and reliability test criteria

1) Quality assurance in shipping

Establishing quality in the design and in fabrication is essential to keep the quality and reliability levels of the semiconductor devices at a high level. This is done by the "Zero-defect" (ZD) movement. Further sampling checks, in units of shipping lot, is done on products that have been "total-inspected" at the final fabrication

stage, thus ensuring no defective items. This sampling inspection is done in accordance with MIL-STD-105D.

2) Reliability

The reliability test is done, periodically, to confirm reliability level.

Periodical reliability test

		Test Hour	LTPD (%)
Electrical characteristic test		In order to know the quality level, some types are selected and tested again.	
LIFE TEST	High temperature operation	Up to 1000 hr	10%
	High temperature storage	Up to 1000 hr	10%
	Low temperature storage	Up to 1000 hr	10%
	High temperature and high humidity storage	Up to 1000 hr	10%
	High humidity bias test	Up to 1000 hr	10%
	High temperature and high humidity with bias	Up to 500 hr	10%
	Pressure cooker	Up to 200 hr	10%
ENVIRONMENT TEST	Soldering heat resistance heat cycle	10 s	15%
	Heat cycle	10 cycle	15%
MECHANICAL TEST	Solderability	Japan Industrial	15%
	Lead strength	Standard (JIS)	15%
OTHER TESTS	if necessary test, are selected accordingly to JIS C7021, C7022, EIAJ SD121, IC121.		

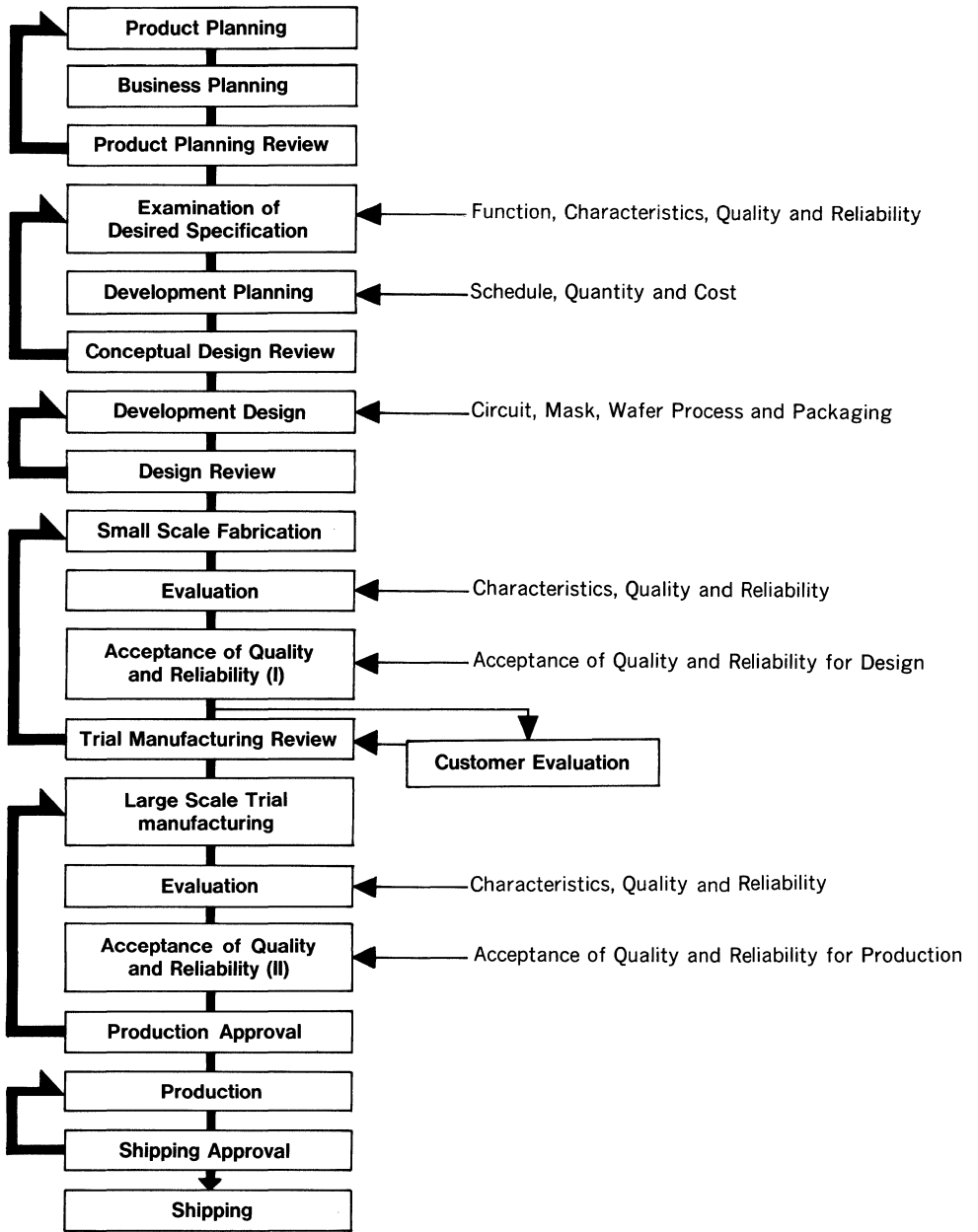
• These tests are selected by sampling standard.

These tests and Inspection data are useful not only to estimate quality in the market place but also as data to improve design and wafer processes.

Reliability test standard for acceptance of products

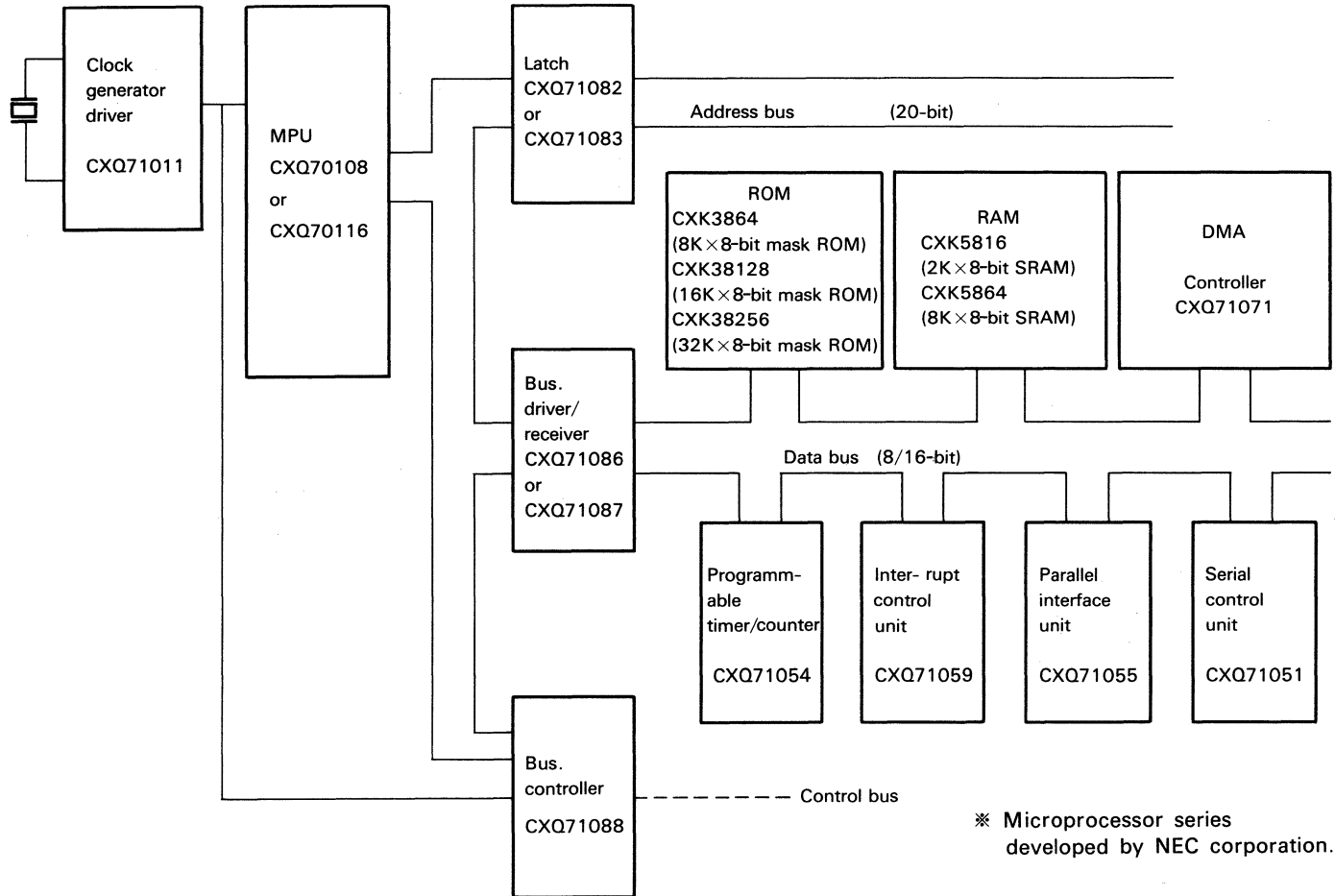
Item	Condition	Supply voltage	Testing time	LTPD(%)
High temperature operation	Ta = 125°C, 150°C	TYP	1000 hr	5%
High temperature with bias	Ta = 125°C, 150°C	TYP	1000 hr	5%
High temperature storage	Ta = 150°C		1000 hr	5%
Low temperature storage	Ta = -65°C		1000 hr	5%
High temperature and humidity storage	Ta = 85°C, 85%RH		1000 hr	5%
High temperature and High humidity with bias	Ta = 85°C, 85%RH	TYP (1 hr on/3 hr off)	1000 hr	5%
Pressure cooker	Ta = 121°C, 100%RH, 30 pounds per square inch		1000 hr	5%
Temperature cycle	Ta = -65°C to +150°C		100 C	10%
Heat shock	Ta = 0°C to +100°C		5 C	10%
Soldering heat resistance	Tsolder = 260°C		10 S	10%
Solderability	Tsolder = 230°C (Rosin type flux)		5 S	10%
Mechanical shock	X, Y, Z 1500G 0.5 ms half sine wave		3 times for each direction	10%
Vibration	X, Y, Z 20G 10~2000~10 Hz (4 min) sine wave vibration		16 minutes for each direction	10%
Constant acceleration	X, Y, Z 20,000 G centrifugal acceleration		1 minute for each direction	10%
Fall by gravity	Falling from the height of 75cm to maple plate by gravity		3 times	10%
Lead strength (Bend) (Pull)	Based on JIS			10%
Electrostatics strength	Device must be designed again, when electrostatic strength is below standard supplying surge voltage To each pin under the conditions of C = 200PF and Rs = 0Ω.			

From development to production



5. Block Diagram

Sony V Series* System Configuration



* Microprocessor series developed by NEC corporation.

Microprocessors

■ Microprocessors

Type	Function	Page
CXQ70108	8-bit Microprocessor	21
CXQ70116	16-bit Microprocessor	71

8-Bit Microprocessor

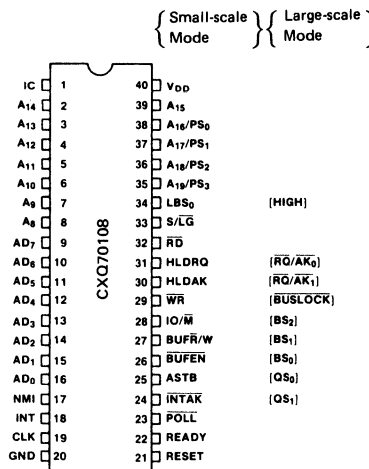
Description

The CXQ70108 is a CMOS 8-bit microprocessor with internal 16-bit architecture and an 8-bit external data bus. The CXQ70108 instruction set is a superset of the 8086/8088; however, mnemonics and execution times are different. The CXQ70108 additionally has a powerful instruction set including bit processing, packed BCD operations, and high-speed multiplication/division operations. The CXQ70108 can also emulate the functions of an 8080 and comes with a standby mode that significantly reduces power consumption. It is software-compatible with the CXQ70116 16-bit microprocessor.

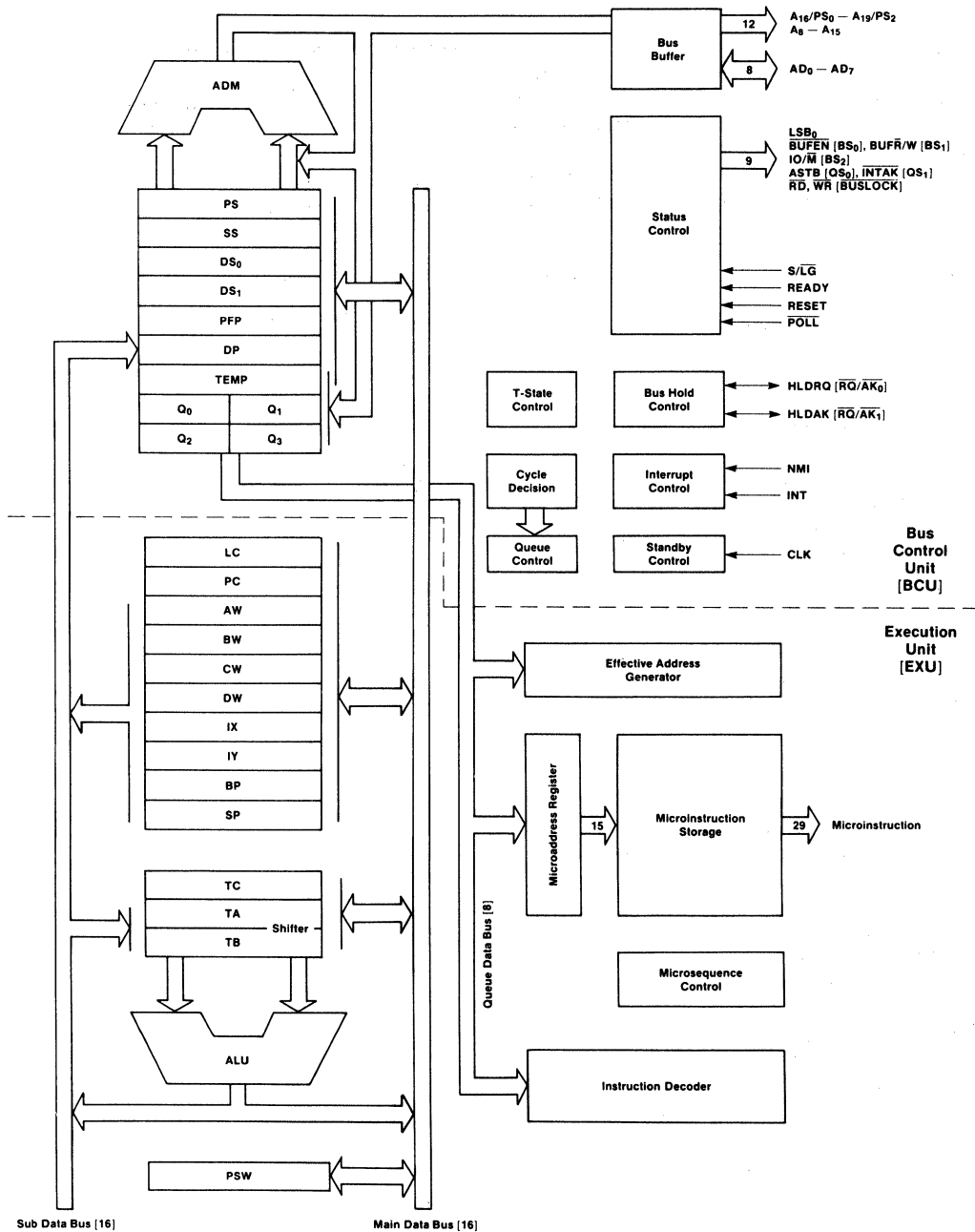
Features

- Minimum instruction execution time: 250 ns (at 8 MHz)
- Maximum addressable memory: 1 Mbytes
- Abundant memory addressing modes
- 14 × 16-bit register set
- 101 instructions
- Instruction set is a superset of 8086/8088 instruction set
- Bit, byte, word and block operations
- Bit field operation instructions
- Packed BCD operation instructions
- Multiplication/division instructions execution time: 2.4 μs to 7.1 μs (at 8 MHz)
- High-speed block transfer instructions: 1 Mbytes/s (at 8 MHz)
- High-speed calculation of effective addresses: 2 clock cycles in any addressing mode
- Maskable (INT) and nonmaskable (NMI) interrupt inputs
- IEEE-796 bus compatible interface
- 8080 emulation functions
- CMOS technology
- Low power consumption
- Standby function
- Single power supply
- 5 MHz or 8 MHz clock
- 40-pin Plastic/Ceramic DIP (600 mil)
- NEC μPD70108 (V20) compatible

Pin Configuration (Top View)



Block Diagram



Pin Identification

No.	Symbol	Direction	Function
1	IC*		Internally connected
2—8	A ₁₄ —A ₈	Out	Address bus, middle bits
9—16	AD ₇ —AD ₀	In/Out	Address/data bus
17	NMI	In	Nonmaskable interrupt input
18	INT	In	Maskable interrupt input
19	CLK	In	Clock input
20	GND		Ground
21	RESET	In	Reset input
22	READY	In	Ready input
23	$\overline{\text{POLL}}$	In	Poll input
24	$\overline{\text{INTAK}}$ (QS ₁)	Out	Interrupt acknowledge output (queue status bit 1 output)
25	ASTB (QS ₀)	Out	Address strobe output (queue status bit 0 output)
26	BUFEN (BS ₀)	Out	Buffer enable output (bus status bit 0 output)
27	BUF $\overline{\text{R}}$ /W (BS ₁)	Out	Buffer read/write output (bus status bit 1 output)
28	IO/ $\overline{\text{M}}$ (BS ₂)	Out	Access is I/O or memory (bus status bit 2 output)
29	$\overline{\text{WR}}$ ($\overline{\text{BUSLOCK}}$)	Out	Write strobe output (bus lock output)
30	HLD $\overline{\text{A}}$ K ($\overline{\text{RQ}}/\overline{\text{AK}}_1$)	Out (In/Out)	Hold acknowledge output, (bus hold request input/ acknowledge output 1)
31	HLD $\overline{\text{R}}$ Q ($\overline{\text{RQ}}/\overline{\text{AK}}_0$)	In (In/Out)	Hold request input (bus hold request input/acknowledge output 0)
32	$\overline{\text{RD}}$	Out	Read strobe output
33	S/ $\overline{\text{LG}}$	In	Small-scale/large-scale system input
34	LBS ₀ (HIGH)	Out	Latched bus status output 0 (always high in large-scale systems)
35—38	A ₁₉ /PS ₃ —A ₁₆ /PS ₀	Out	Address bus, high bits or processor status output
39	A ₁₅	Out	Address bus, bit 15
40	V _{DD}		Power supply

Notes: *IC should be connected to ground.

Where pins have different functions in small- and large-scale systems, the large-scale system pin symbol and function are in parentheses.

Unused input pins should be tied to ground or V_{DD} to minimize power dissipation and prevent the flow of potentially harmful currents.

Pin Functions

Some pins of the CXQ70108 have different functions according to whether the microprocessor is used in a small- or large-scale system. Other pins function the same way in either type of system.

A₁₅ — A₈ [Address Bus]

For small- and large-scale systems.

The CPU uses these pins to output the middle 8 bits of the 20-bit address data. They are three-state output and float to the high impedance during hold acknowledge.

AD₇ — AD₀ [Address/Data Bus]

For small- and large-scale systems.

The CPU uses these pins as the time-multiplexed address and data bus. They are active high. This bus contains the lower 8 bits of the 20-bit address during T₁ of the bus cycle and is used as an 8-bit data bus during T₂, T₃, and T₄ of the bus cycle.

Sixteen-bit data I/O is performed in two steps. The low byte is sent first, followed by the high byte. The address/data bus is a three-state bus and can be high or low during standby mode. The bus will float to the high impedance during hold and interrupt acknowledge.

NMI [Nonmaskable Interrupt]

For small- and large-scale systems.

This pin is used to input nonmaskable interrupt requests. NMI cannot be masked by software. This input is positive edge-triggered and can be sensed during any clock cycle. Actual interrupt processing begins, however, after completion of the instruction in progress.

The contents of interrupt vector 2 determine the starting address for the interrupt-servicing routine. Note that a hold request will be accepted even during NMI acknowledge.

This interrupt will cause the CXQ70108 to exit the standby mode.

INT [Maskable Interrupt]

For small- and large-scale systems.

This pin is a level-triggered interrupt request that can be masked by software.

INT is active high and is sensed during the last clock of the instruction. The interrupt will be accepted if the system is in interrupt enable state (if the interrupt enable flag IE is set). The CPU outputs the INTAK signal to inform external devices that the interrupt request has been granted.

If NMI and INT interrupts occur at the same time, NMI has higher priority than INT and INT cannot be accepted. A hold request will be accepted during INT acknowledge.

This interrupt causes the CXQ70108 to exit the standby mode.

CLK [Clock]

For small- and large-scale systems.

This pin is used for external clock input.

RESET [Reset]

For small- and large-scale systems.

This pin is used for the CPU reset signal. It is active high. Input of this signal has priority over all other operations. After the reset signal input returns low, the CPU begins execution of the program starting at address FFFF0H.

In addition to causing normal CPU start, RESET input will cause the CXQ70108 to exit the standby mode.

READY [Ready]

For small- and large-scale systems.

When the memory or I/O device being accessed cannot complete data read or write within the CPU basic access time, it can generate a CPU wait state (T_w) by setting this signal to inactive (low) and requesting a read/write cycle delay.

If the READY signal is active (high) during either T_3 or T_w state, the CPU will not generate a wait state.

POLL [Poll]

For small- and large-scale systems.

The CPU checks this input upon execution of the $\overline{\text{POLL}}$ instruction. If the input is low, then execution continues. If the input is high, the CPU will check the $\overline{\text{POLL}}$ input every five clock cycles until the input becomes low again.

The $\overline{\text{POLL}}$ and READY functions are used to synchronize CPU program execution with the operation of external devices.

 $\overline{\text{RD}}$ [Read Strobe]

For small- and large-scale systems.

The CPU outputs this strobe signal during data read from an I/O device or memory. The $\text{IO}/\overline{\text{M}}$ signal is used to select between I/O and memory. $\overline{\text{RD}}$ will be high during standby mode. It is three-state and floats to the high impedance during hold acknowledge.

 $\text{S}/\overline{\text{LG}}$ [Small/Large]

For small- and large-scale systems.

This signal determines the operation mode of the CPU. This signal is fixed either high or low. When this signal is high, the CPU will operate in small-scale system mode, and when low, in the large-scale system mode. A small-scale system will have at most one bus master such as a DMA controller device on the bus. A large-scale system can have more than one bus master accessing the bus as well as the CPU.

Pins 24 to 31 and pin 34 function differently depending on the operating mode of the CPU. Separate nomenclature is adopted for these signals in the two operation modes.

Pin No.	Function	
	$\text{S}/\overline{\text{LG}}\text{-high}$	$\text{S}/\overline{\text{LG}}\text{-low}$
24	$\overline{\text{INTAK}}$	QS_1
25	ASTB	QS_0
26	$\overline{\text{BUFEN}}$	BS_0
27	BUFR/W	BS_1
28	$\text{IO}/\overline{\text{M}}$	BS_2
29	$\overline{\text{WR}}$	$\overline{\text{BUSLOCK}}$
30	HLD $\overline{\text{AK}}$	$\overline{\text{RQ}}/\overline{\text{AK}}_1$
31	HLD $\overline{\text{RQ}}$	$\overline{\text{RQ}}/\overline{\text{AK}}_0$
34	LBS_0	Always high

 $\overline{\text{INTAK}}$ [Interrupt Acknowledge]

For small-scale systems.

The CPU generates the $\overline{\text{INTAK}}$ signal low when it accepts an INT signal.

The interrupting device synchronizes with this signal and outputs the interrupt vector to the CPU via the data bus ($\text{AD}_7 - \text{AD}_0$). $\overline{\text{INTAK}}$ will be high during standby mode.

ASTB [Address Strobe]

For small-scale systems.

The CPU outputs this strobe signal to latch address information at an external latch. ASTB will be low during standby mode.

BUFEN [Buffer Enable]

For small-scale systems.

It is used as the output enable signal for an external bidirectional buffer. The CPU generates this signal during data transfer operations with external memory or I/O devices or during input of an interrupt vector.

BUFEN will be high during standby mode. It is three-state and floats to the high impedance during hold acknowledge.

BUFR/W [Buffer Read/Write]

For small-scale systems.

The output of this signal determines the direction of data transfer with an external bidirectional buffer. A high output causes transmission from the CPU to the external device; a low signal causes data transfer from the external device to the CPU.

BUFR/W will be either high or low during standby mode. It is three-state and floats to the high impedance during hold acknowledge.

IO/M [IO/Memory]

For small-scale systems.

The CPU generates this signal to specify either I/O access or memory access. A high-level output specifies I/O and a low-level specifies memory.

IO/M will be either high or low during standby mode. It is three-state and floats to the high impedance during hold acknowledge.

WR [Write Strobe]

For small-scale systems.

The CPU generates this strobe signal during data write to an I/O device or memory. Selection of either I/O or memory is performed by the IO/M signal.

WR will be high during standby mode. It is three-state and floats to the high impedance during hold acknowledge.

HLDK [Hold Acknowledge]

For small-scale systems.

The HLDK signal is used to indicate that the CPU accepts the hold request signal (HLDRQ). When this signal is high, the address bus, address/data bus, and the control lines become high impedance.

HLDRQ [Hold Request]

For small-scale systems.

This input signal is used by external devices to request the CPU to release the address bus, address/data bus, and the control bus.

LBS₀ [Latched Bus Status 0]

For small-scale systems.

The CPU uses this signal along with the IO/M and BUFR/W signals to inform an external device what the current bus cycle is.

IO/ \bar{M}	BUFR/ \bar{W}	LBS ₀	Bus Cycle
0	0	0	Program fetch
0	0	1	Memory read
0	1	0	Memory write
0	1	1	Passive state
1	0	0	Interrupt acknowledge
1	0	1	I/O read
1	1	0	I/O write
1	1	1	Halt

A₁₉/PS₃ — A₁₆/PS₀ [Address Bus/Processor Status]

For small- and large-scale systems.

These pins are time-multiplexed to operate as an address bus and as processor status signals.

When used as the address bus, these pins are the high 4 bits of the 20-bit memory address. During I/O access, all 4 bits output data 0.

The processor status signals are provided for both memory and I/O use. PS₃ is always 0 in the native mode and 1 in 8080 emulation mode. The interrupt enable flag (IE) is pin on pin PS₂. Pins PS₁ and PS₀ indicate which memory segment is being accessed.

A ₁₇ /PS ₁	A ₁₆ /PS ₀	Segment
0	0	Data segment 1
0	1	Stack segment
1	0	Program segment
1	1	Data segment 0

A₁₉/PS₃ — A₁₆/PS₀ will be either high or low during standby mode. They are three-state and float to the high impedance during hold acknowledge.

QS₁, QS₀ [Queue Status]

For large-scale systems.

The CPU uses these signals to allow external devices, such as the floating-point arithmetic processor chip, to monitor the status of the internal CPU instruction queue.

QS ₁	QS ₀	Instruction Queue Status
0	0	NOP (queue does not change)
0	1	First byte of instruction
1	0	Flush queue
1	1	Subsequent bytes of instruction

The instruction queue status indicated by these signals is the status when the execution unit (EXU) accesses the instruction queue. The data output from these pins is therefore valid only for one clock cycle immediately following queue access. These status signals are provided so that the floating-point processor chip can monitor the CPU's program execution status and synchronize its operation with the CPU when control is passed to it by the FPO (Floating Point Operation) instructions.

QS₁, QS₀ will be low during standby mode.

BS₂ — BS₀ [Bus Status]

For large-scale systems.

The CPU uses these status signals to allow an external bus controller to monitor what the current bus cycle is.

The external bus controller decodes these signals and generates the control signals required to perform access of the memory or I/O device.

BS ₂	BS ₁	BS ₀	Bus Cycle
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Program fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive state

BS₂ — BS₀ will be high during standby mode. They are three-state and floats to the high impedance during hold acknowledge.

BUSLOCK [Bus Lock]

For large-scale systems.

The CPU uses this signal to secure the bus while executing the instruction immediately following the BUSLOCK prefix instruction. It is a status signal to the other bus masters in a multiprocessor system inhibiting them from using the system bus during this time.

The output of this signal is three-state and becomes high impedance during hold acknowledge. BUSLOCK is high during standby mode except if the HALT instruction has a BUSLOCK prefix.

 $\overline{RQ}/\overline{AK}_1$, $\overline{RQ}/\overline{AK}_0$ [Hold Request/Acknowledge]

For large-scale systems.

These pins function as bus hold request inputs (\overline{RQ}) and as bus hold acknowledge outputs (\overline{AK}). $\overline{RQ}/\overline{AK}_0$ has a higher priority than $\overline{RQ}/\overline{AK}_1$.

These pins have three-state outputs with on-chip pull-up resistors which keep the pin at high level when the output is high impedance.

V_{DD} [Power Supply]

For small- and large-scale systems.

This pin is used for the +5V power supply.

GND [Ground]

For small- and large-scale systems.

This pin is used for ground.

IC [Internally Connected]

This pin is used for tests performed at the factory by SONY. The CXQ70108 is used with this pin at ground potential.

Absolute Maximum Ratings

(Ta=+25°C)

Parameter	Symbol	Rating Value	Unit
Power supply voltage	V _{DD}	-0.5 to +0.7	V
Input voltage	V _I	-0.5 to V _{DD} +0.3	V
CLK input voltage	V _K	-0.5 to V _{DD} +1.0	V
Output voltage	V _O	-0.5 to V _{DD} +0.3	V
Power dissipation	P _{DMAX}	+0.5	W
Operating temperature	T _{opr}	-40 to +85	°C
Storage temperature	T _{stg}	-65 to +150	°C

Comment: Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of this specification.

Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

CXQ70108-5, Ta=-40°C to +85°C, V_{DD}=+5V±10%CXQ70108-8, Ta=-10°C to +70°C, V_{DD}=+5V±5%

Parameter	Symbol	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
Input voltage high	V _{IH}	2.2		V _{DD} +0.3	V	
Input voltage low	V _{IL}	-0.5		0.8	V	
CLK input voltage high	V _{KH}	3.9		V _{DD} +1.0	V	
CLK input voltage low	V _{KL}	-0.5		0.6	V	
Output voltage high	V _{OH}	0.7×V _{DD}			V	I _{OH} =-400 μA
Output voltage low	V _{OL}			0.4	V	I _{OL} =2.5 mA
Input leakage current high	I _{LIH}			10	μA	V _I =V _{DD}
Input leakage current low	I _{LIL}			-10	μA	V _I =0V
Output leakage current high	I _{LOH}			10	μA	V _O =V _{DD}
Output leakage current low	I _{LOL}			-10	μA	V _O =0V
Supply current	I _{DD}	70108-5 5 MHz	30	60	mA	Normal operation
			5	10	mA	Standby mode
		70108-8 8 MHz	45	80	mA	Normal Operation
			6	12	mA	Standby mode

Capacitance

(Ta=+25°C, V_{DD}=0V)

Parameter	Symbol	Limits		Unit	Test Conditions
		Min.	Max.		
Input capacitance	C _I		15	pF	f _c =1 MHz
I/O capacitance	C _{IO}		15	pF	Unmeasured pins returned to 0V

AC Characteristics

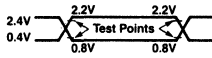
CXQ70108-5, $T_a = -40^\circ\text{C}$ to $+85^\circ\text{C}$, $V_{DD} = +5\text{V} \pm 10\%$
 CXQ70108-8, $T_a = -10^\circ\text{C}$ to $+70^\circ\text{C}$, $V_{DD} = +5\text{V} \pm 5\%$

Parameter	Symbol	CXQ70108-5		CXQ70108-8		Unit	Test Conditions
		Min.	Max.	Min.	Max.		
Small/Large Scale							
Clock cycle	tcyk	200	500	125	500	ns	
Clock pulse width high	tkKH	69		50		ns	$V_{KH} = 3.0\text{V}$
Clock pulse width low	tkKL	90		60		ns	$V_{KL} = 1.5\text{V}$
Clock rise time	tkR		10		8	ns	1.5V to 3.0V
Clock fall time	tkF		10		7	ns	3.0V to 1.5V
READY inactive setup to CLK ↓	tsRYLK	-8		-8		ns	
READY inactive hold after CLK ↑	thKRYH	30		20		ns	
READY active setup to CLK ↑	tsRYHK	tkKL-8		tkKL-8		ns	
READY active hold after CLK ↑	thKRYL	30		20		ns	
Data setup time to CLK ↓	tsDK	30		20		ns	
Data hold time after CLK ↓	thKD	10		10		ns	
NMI, INT, POLL setup time to CLK ↑	tsIK	30		15		ns	
RESET setup time to CLK ↑	tsRST	30		20		ns	
RESET hold time after CLK ↑	thRST	10		10		ns	
Input rise time (except CLK)	tIR		20		20	ns	0.8V to 2.2V
Input fall time (except CLK)	tIF		12		12	ns	2.2V to 0.8V
Output rise time	toR		20		20	ns	0.8V to 2.2V
Output fall time	toF		12		12	ns	2.2V to 0.8V
Small Scale							
Address delay time from CLK	tdKA	10	90	10	60	ns	$C_L = 100\text{ pF}$
Address hold time from CLK	thKA	10		10		ns	
PS delay time from CLK ↓	tdKP	10	90	10	60	ns	
PS float delay time from CLK ↑	tFKP	10	80	10	60	ns	
Address setup time to ASTB ↓	tsAST	tkKL-60		tkKL-30		ns	
Address float delay time from CLK ↓	tFA	thKA	80	thKA	60	ns	
ASTB ↑ delay time from CLK ↓	tdKSTH		80		50	ns	
ASTB ↓ delay time from CLK ↑	tdKSTL		85		55	ns	
ASTB width high	tSTST	tkKL-20		tkKL-10		ns	
Address hold time from ASTB ↓	thSTA	tkKH-10		tkKL-10		ns	

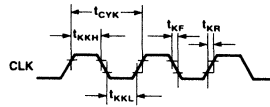
Parameter	Symbol	CXQ70108-5		CXQ70108-8		Unit	Test Conditions
		Min.	Max.	Min.	Max.		
Control delay time from CLK	tDKCT	10	110	10	65	ns	C _L =100 pF
Address float to \overline{RD} ↓	tAFRL	0		0		ns	
\overline{RD} ↓ delay time from CLK ↓	tDKRL	10	165	10	80	ns	
\overline{RD} ↑ delay time from CLK ↓	tDKRH	10	150	10	80	ns	
Address delay time from \overline{RD} ↑	tDRHA	t _{CYK} -45		t _{CYK} -40		ns	
\overline{RD} width low	t _{RR}	2t _{CYK} -75		2t _{CYK} -50		ns	
Data output delay time from CLK ↓	tDKD	10	90	10	60	ns	
Data float delay time from CLK ↓	tFKD	10	80	10	60	ns	
\overline{WR} width low	t _{WW}	2t _{CYK} -60		2t _{CYK} -40		ns	
HLDRO setup time to CLK ↑	tSHQK	35		20		ns	
HLDK delay time from CLK ↓	tDKHA	10	160	10	100	ns	
Large Scale							
Address delay time from CLK	tDKA	10	90	10	60	ns	C _L =100 pF
Address hold time from CLK	tHKA	10		10		ns	
PS delay time from CLK ↓	tDKP	10	90	10	60	ns	
PS float delay time from CLK ↑	tFKP	10	80	10	60	ns	
Address float delay time from CLK ↓	tFKA	tHKA	80	tHKA	60	ns	
Address delay time from \overline{RD} ↑	tDRHA	t _{CYK} -45		t _{CYK} -40		ns	
ASTB ↑ delay time from BS ↓	tDBST		15		15	ns	
BS ↓ delay time from CLK ↑	tDKBL	10	110	10	60	ns	
BS ↑ delay time from CLK ↓	tDKBH	10	130	10	65	ns	
\overline{RD} ↓ delay time from address float	tDAFRL	0		0		ns	
\overline{RD} ↓ delay time from CLK ↓	tDKRL	10	165	10	80	ns	
\overline{RD} ↑ delay time from CLK ↓	tDKRH	10	150	10	80	ns	
\overline{RD} width low	t _{RR}	2t _{CYK} -75		2t _{CYK} -50		ns	
Data output delay time from CLK ↓	tDKD	10	90	10	60	ns	
Data float delay time from CLK ↓	tFKD	10	80	10	60	ns	
\overline{AK} delay time from CLK ↓	tDKAK		70		50	ns	
\overline{RQ} setup time to CLK ↑	tSRQK	20		10		ns	
\overline{RQ} hold time after CLK ↑	tHKRQ	40		30		ns	

Timing Waveforms

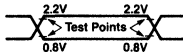
AC Test input Waveform [Except CLK]



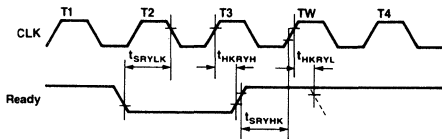
Clock Timing



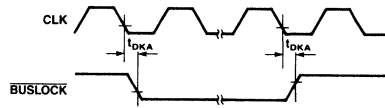
AC Output Test Points



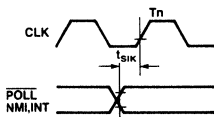
Wait [Ready] Timing



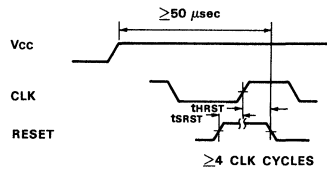
BUSLOCK Output Timing



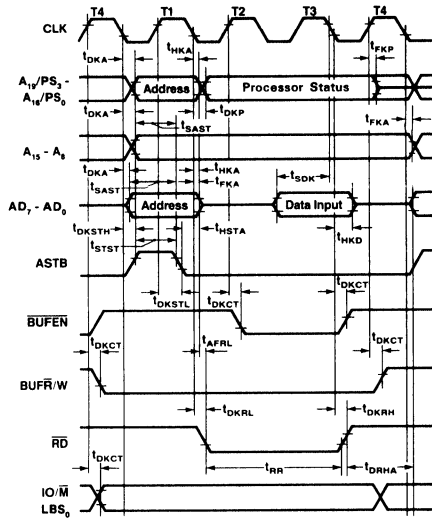
POLL, NMI, INT Input Timing



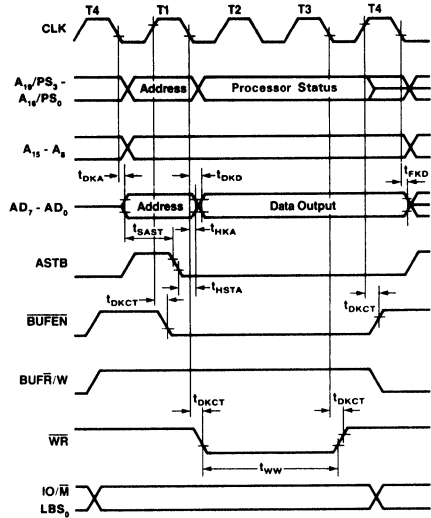
RESET Timing



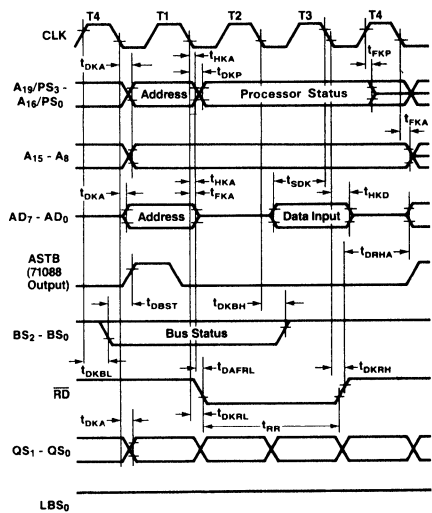
Read Timing [Small Scale]



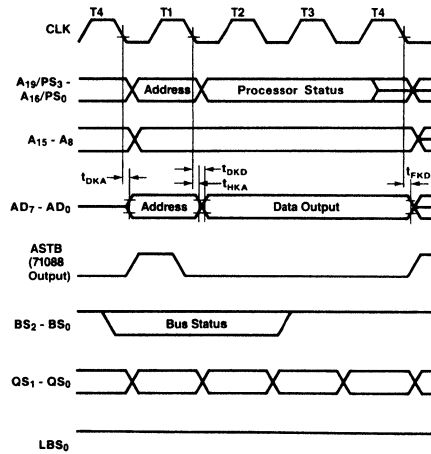
Write Timing [Small Scale]



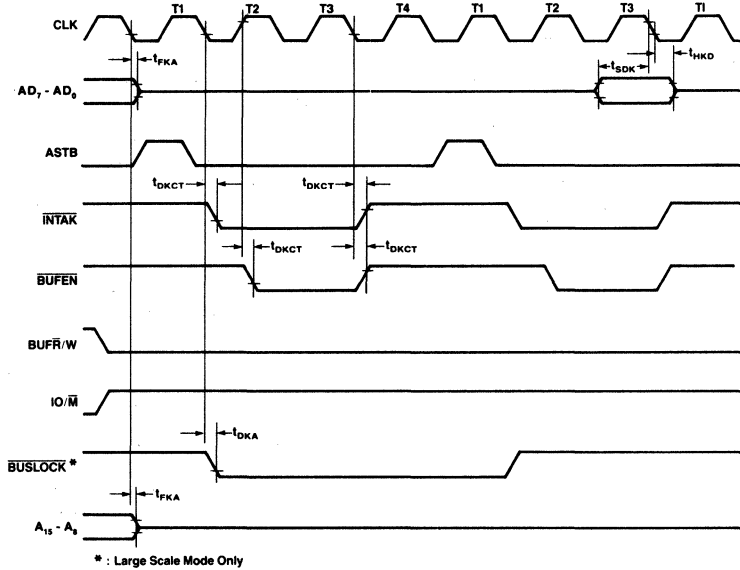
Read Timing [Large Scale]



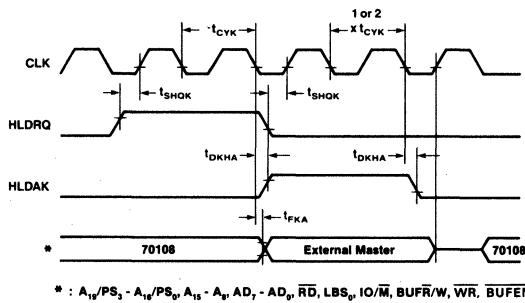
Write Timing [Large Scale]



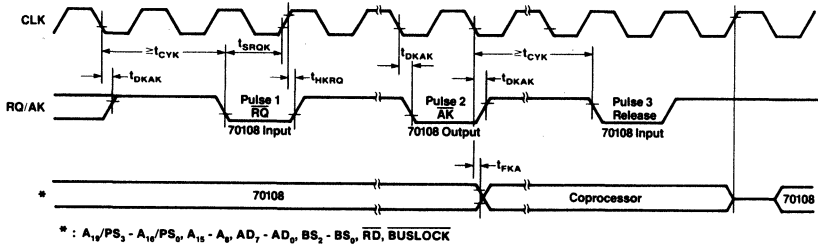
Interrupt Acknowledge Timing



Hold Request/Acknowledge Timing [Small Scale]



Bus Request/Acknowledge Timing [Large Scale]



Register Configuration

Program Counter [PC]

The program counter is a 16-bit binary counter that contains the segment offset address of the next instruction which the EXU is to execute.

The PC increments each time the microprogram fetches an instruction from the instruction queue. A new location value is loaded into the PC each time a branch, call, return, or break instruction is executed. At this time, the contents of the PC are the same as the Prefetch Pointer (PFP).

Prefetch Pointer [PFP]

The prefetch pointer (PFP) is 16-bit binary counter which contains a segment offset which is used to calculate a program memory address that the bus control unit (BCU) uses to prefetch the next byte for the instruction queue. The contents of PFP are an offset from the PS (Program Segment) register.

The PFP is incremented each time the BCU prefetches an instruction from the program memory. A new location will be loaded into the PFP whenever a branch, call, return, or break instruction is executed. At that time the contents of the PFP will be the same as those of the PC (Program Counter).

Segment Registers [PS, SS, DSo, and DS₁]

The memory addresses accessed by the CXQ70108 are divided into 64K-byte logical segments. The starting (base) address of each segment is specified by a segment register, and the offset from this starting address is specified by the contents of another register or by the effective address.

These are the four types of segment registers used.

Segment Register	Default Offset
PS (Program Segment)	PFP
SS (Stack Segment)	SP, effective address
DSo (Data Segment 0)	IX, effective address
DS ₁ (Data Segment 1)	IY

General-Purpose Registers [AW, BW, CW, and DW]

There are four 16-bit general-purpose registers. Each one can be used as one 16-bit register or as two 8-bit registers by dividing them into their high and low bytes (AH, AL, BH, BL, CH, CL, DH, DL).

Each register is also used as a default register for processing specific instructions. The default assignments are:

- AW: Word multiplication/division, word I/O, BCD rotation, data conversion, translation
- AL: Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation
- AH: Byte multiplication/division
- BW: Translation
- CW: Loop control branch, repeat prefix
- CL: Shift instructions, rotation instructions, BCD operations
- DW: Word multiplication/division, indirect addressing I/O

Pointers [SP, BP] and index Registers [IX, IY]

These registers serve as base pointers or index registers when accessing the memory using based addressing, indexed addressing, or based indexed addressing.

These registers can also be used for data transfer and arithmetic and logical operations in the same manner as the general-purpose registers. They cannot be used as 8-bit registers.

Also, each of these registers acts as a default register for specific operations. The default assignments are:

- SP: Stack operations
- IX: Block transfer (source), BCD string operations
- IY: Block transfer (destination), BCD string operations

Program Status Word [PSW]

The program status word consists of the following six status and four control flags.

- | | |
|------------------------|-------------------------|
| Status Flags | Control Flags |
| • V (Overflow) | • MD (Mode) |
| • S (Sign) | • DIR (Direction) |
| • Z (Zero) | • IE (Interrupt Enable) |
| • AC (Auxiliary Carry) | • BRK (Break) |
| • P (Parity) | |
| • CY (Carry) | |

When the PSW is pushed on the stack, the word images of the various flags are as shown here.

PSW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M	1	1	1	V	D	I	B	S	Z	0	A	0	P	1	C
D					I	E	R				C				Y
					R		K								

The status flags are set and reset depending upon the result of each type of instruction executed. Instructions are provided to set, reset, and complement the CY flag directly. Other instructions set and reset the control flags and control the operation of the CPU.

High-Speed Execution of Instructions

This section highlights the major architectural features that enhance the performance of the CXQ70108.

- Dual data bus in EXU
- Effective address generator
- 16/32-bit temporary registers/shifters (TA, TB)
- 16-bit loop counter
- PC and PFP

Dual Data Bus Method

To reduce the number of processing steps for instruction execution, the dual data bus method has been adopted for the CXQ70108 (figure 1). The two data buses (the main data bus and the subdata bus) are both 16 bits wide. For addition/subtraction and logical and comparison operations, processing time has been speeded up some 30% over single-bus systems.

Fig. 1. Dual Data Buses

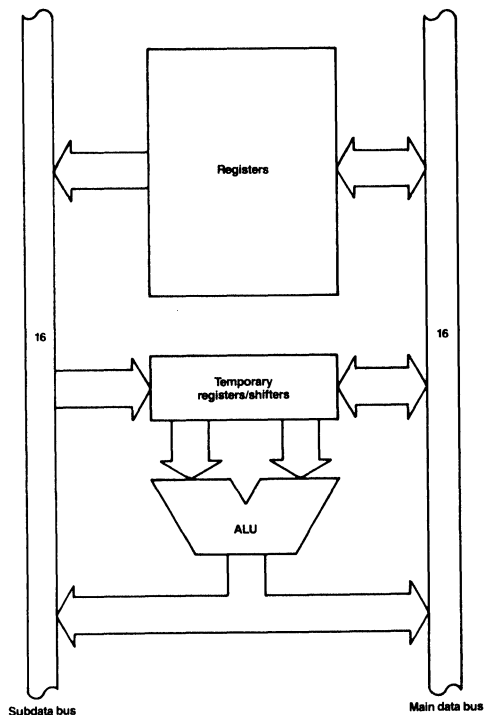
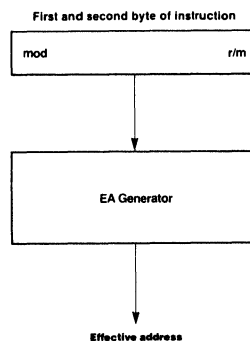


Fig. 2. Effective Address Generator

**Example**

ADD AW, BW	;AW ← AW + BW
Single Bus	Dual Bus
Step 1 TA ← AW	TA ← AW, TB ← BW
Step 2 TB ← BW	AW ← TA + TB
Step 3 AW ← TA + TB	

Effective address Generator

This circuit (figure 2) performs high-speed processing to calculate effective addresses for accessing memory.

Calculating an effective address by the microprogramming method normally requires 5 to 12 clock cycles. This circuit requires only two clock cycles for addresses to be generated for any addressing mode. Thus, processing is several times faster.

16/32-Bit Temporary Registers/Shifters [TA, TB]

These 16-bit temporary registers/shifters (TA, TB) are provided for multiplication/division and shift/rotation instructions.

These circuits have decreased the execution time of multiplication/division instructions. In fact, these instructions can be executed about four times faster than with the microprogramming method.

TA + TB: 32-bit temporary register/shifter for multiplication and division instructions.

TB: 16-bit temporary register/shifter for shift/rotation instructions.

Loop Counter [LC]

This counter is used to count the number of loops for a primitive block transfer instruction controlled by a repeat prefix instruction and the number of shifts that will be performed for a multiple bit shift/rotation instruction.

The processing performed for a multiple bit rotation of a register is shown below. The average speed is approximately doubled over the microprogram method.

Example

RORC AW, CL ; CL = 5

Microprogram method

8 + (4 × 5) = 28 clocks

LC method

7 + 5 = 12 clocks

Program Counter and Prefetch Pointer [PC and PFP]

The CXQ70108 microprocessor has a program counter (PC), which addresses the program memory location of the instruction to be executed next, and a prefetch pointer (PFP), which addresses the program memory location to be accessed next. Both functions are provided in hardware. A time saving of several clocks is realized for branch, call, return, and break instruction execution, compared with microprocessors that have only one instruction pointer.

Enhanced Instructions

In addition to the 8088/86 instructions, the CXQ70108 has the following enhanced instructions.

Instruction	Function
PUSH imm	Pushes immediate data onto stack
PUSH R	Pushes 8 general registers onto stack
POP imm	Pops immediate data from stack
POP R	Pops 8 general registers from stack
MUL imm	Executes 16-bit multiply of register or memory contents by immediate data
SHL imm8 SHR imm8 SHRA imm8 ROL imm8 ROR imm8 ROLC imm8 RORC imm8	Shifts/rotates register or memory by immediate value
CHKIND	Checks array index against designated boundaries
INM	Moves a string from an I/O port to memory
OUTM	Moves a string from memory to an I/O port
PREPARE	Allocates an area for a stack frame and copies previous frame pointers
DISPOSE	Frees the current stack frame on a procedure exit

Enhanced Stack Operation Instructions**PUSH imm/POP imm**

These instructions allow immediate data to be pushed onto or popped from the stack.

PUSH R/POP R

These instructions allow the contents of the eight general registers to be pushed onto or popped from the stack with a single instruction.

Enhanced Multiplication Instructions**MUL reg16, imm16/MUL mem16, imm16**

These instructions allow the contents of a register or memory location to be 16-bit multiplied by immediate data.

Enhanced Shift and Rotate Instructions**SHL reg, imm8/SHR reg, imm8/SHRA reg, imm8**

These instructions allow the contents of a register to be shifted by the number of bits defined by the immediate data.

ROL reg, imm8/ROR reg, imm8/ROLC reg, imm8/RORC reg, imm8

These instructions allow the contents of a register to be rotated by the number of bits defined by the immediate data.

Check Array Boundary Instruction**CHKIND reg16, mem32**

This instruction is used to verify that index values pointing to the elements of an array data structure are within the defined range. The lower limit of the array should be in memory location mem32, the upper limit in mem32 + 2. If the index value in reg16 is not between these limits when CHKIND is executed, a BRK 5 will occur. This causes a jump to the location in interrupt vector 5.

Block I/O Instructions**OUTM DW, src-block/INM dst-block, DW**

These instructions are used to output or input a string to or from memory, when preceded by a repeat prefix.

Stack Frame Instructions**PREPARE imm16, imm8**

This instruction is used to generate the stack frames required by block-structured languages, such as PASCAL and Ada. The stack frame consists of two areas. One area has a pointer that points to another frame which has variables that the current frame can access. The other is a local variable area for the current procedure.

DISPOSE

This instruction releases the last stack frame generated by the PREPARE instruction. It returns the stack and base pointers to the values they had before the PREPARE instruction was used to call a procedure.

Unique Instructions

In addition to the 8088/86 instructions and the enhanced instructions, the CXQ70108 has the following unique instructions.

Instruction	Function
INS	Insert bit field
EXT	Extract bit field
ADD4S	Adds packed decimal strings
SUB4S	Subtracts one packed decimal string from another
CMP4S	Compares two packed decimal strings
ROL4	Rotates one BCD digit left through AL lower 4 bits
ROR4	Rotates one BCD digit right through AL lower 4 bits
TEST1	Tests a specified bit and sets/resets Z flag
NOT1	Inverts a specified bit
CLR1	Clears a specified bit
SET1	Sets a specified bit
REPC	Repeats next instruction until CY flag is cleared
REPNC	Repeats next instruction until CY flag is set
FPO2	Additional floating point processor call

Variable Length Bit Field Operation Instructions

This category has two instructions: INS (Insert Bit Field) and EXT (Extract Bit Field). These instructions are highly effective for computer graphics and high-level languages. They can, for example, be used for data structures such as packed arrays and record type data used in PASCAL.

INS reg8, reg8/INS reg8, imm4

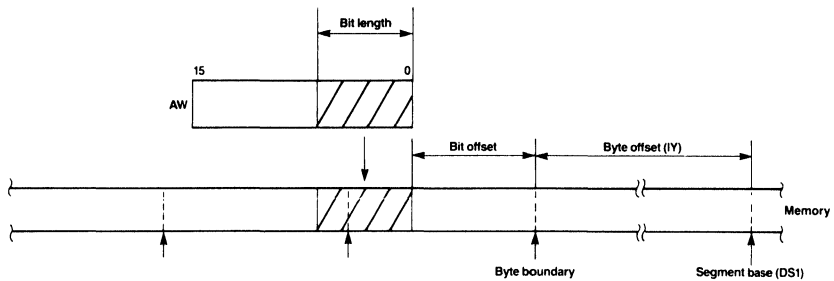
This instruction (figure 3) transfers low bits from the 16-bit AW register (the number of bits is specified by the second operand) to the memory location specified by the segment base (DS₁ register) plus the byte offset (IY register). The starting bit position within this byte is specified as an offset by the lower 4-bits of the first operand.

After each complete data transfer, the IY register and the register specified by the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may specify the number of bits transferred (second operand). Because the maximum transferable bit length is 16-bits, only the lower 4-bits of the specified register (00H to 0FH) will be valid.

Bit field data may overlap the byte boundary of memory.

Fig. 3. Bit Field Insertion



EXT reg8, reg8/EXT reg8, imm4

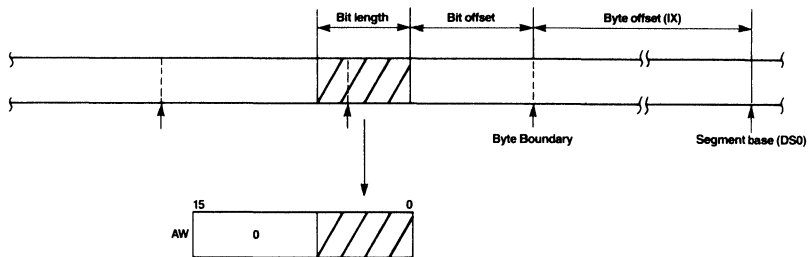
This instruction (figure 4) loads to the AW register the bit field data whose bit length is specified by the second operand of the instruction from the memory location that is specified by the DS0 segment register (segment base), the IX index register (byte offset), and the lower 4-bits of the first operand (bit offset).

After the transfer is complete, the IX register and the lower 4-bits of the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may be specified for the second operand. Because the maximum transferrable bit length is 16 bits, however, only the lower 4-bits of the specified register (OH to OFH) will be valid.

Bit field data may overlap the byte boundary of memory.

Fig. 4. Bit Field Extraction



Packed BCD Operation Instructions

The instructions described here process packed BCD data either as strings (ADD4S, SUB4S, CMP4S) or byte-format operands (ROR4, ROL4). Packed BCD strings may be from 1 to 255 digits in length.

When the number of digits is even, the zero and carry flags will be set according to the result of the operation. When the number of digits is odd, the zero and carry flags may not be set correctly in this case, (CL = odd), the zero flag will not be set unless the upper 4 bits of the highest byte are all zero. The carry flag will not be set unless there is a carry out of the upper 4 bits of the highest byte. When CL is odd, the contents of the upper 4 bits of the highest byte of the result are undefined.

ADD4S

This instruction adds the packed BCD string addressed by the IX index register to the packed BCD string addressed by the IY index register, and stores the result in the string addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the result of the operation will affect the carry flag (CY) and zero flag (Z).

$$\text{BCD string (IY, CL)} \leftarrow \text{BCD string (IY, CL)} + \text{BCD string (IX, CL)}$$

SUB4S

This instruction subtracts the packed BCD string addressed by the IX index register from the packed BCD string addressed by the IY index register, and stores the result in the string addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the result of the operation will affect the carry flag (CY) and zero flag (Z).

$$\text{BCD string (IY, CL)} \leftarrow \text{BCD string (IY, CL)} - \text{BDC String (IX, CL)}$$

CMP4S

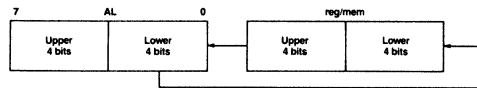
This instruction performs the same operation as SUB4S except that the result is not stored and only carry flags (CY) and zero flag (Z) are affected.

$$\text{BCD string (IY, CL)} - \text{BCD string (IX, CL)}$$

ROL4

This instruction (figure 5) treats the byte data of the register or memory directly specified by the instruction byte as BCD data and uses the lower 4-bits of the AL register (AL) to rotate that data one BCD digit to the left.

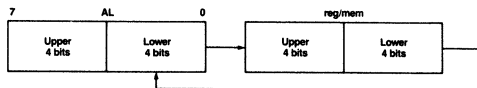
Fig. 5. BCD Rotate Left (ROL4)



ROR4

This instruction (figure 6) treats the byte data of the register or memory directly specified by the instruction byte as BCD data and uses the lower 4-bits of the AL register (AL) to rotate that data one BCD digit to the right.

Fig. 6. BCD Rotate Right (ROR4)



Bit Manipulation Instructions

TEST1

This instruction tests a specific bit in a register or memory location. If the bit is 1, the Z flag is reset to 0. If the bit is 0, the Z flag is set to 1.

NOT1

This instruction inverts a specific bit in a register or memory location.

CLR1

This instruction clears a specific bit in a register or memory location.

SET1

This instruction sets a specific bit in a register or memory location.

Repeat Prefix Instructions

REPC

This instruction causes the CXQ70108 to repeat the following primitive block transfer instruction until the CY flag becomes cleared or the CW register becomes zero.

REPNC

This instruction causes the CXQ70108 to repeat the following primitive block transfer instruction until the CY flag becomes set or the CW register becomes zero.

Floating Point Instruction

FPO2

This instruction is in addition to the 8088/86 floating point instruction, FPO1. These instructions are covered in a later section.

Mode Operation Instructions

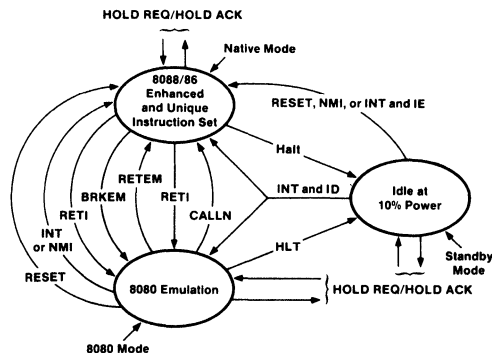
The CXQ70108 has two operating modes (figure 7). One is the native mode which executes 8088/86, enhanced and unique instructions. The other is the 8080 emulation mode in which the instruction set of the 8080 is emulated. A mode flag (MD) is provided to select between these two modes. Native mode is selected when MD is 1 and emulation mode when MD is 0. MD is set and reset, directly and indirectly, by executing the mode manipulation instructions.

Two instructions are provided to switch operation from the native mode to the emulation mode and back: BRKEM (Break for Emulation), and RETEM (Return from Emulation).

Two instructions are used to switch from the emulation mode to the native mode and back: CALLN (Call Native Routine), and RETI (Return from Interrupt).

The system will return from the 8080 emulation mode to the native mode when the RESET signal is present, or when an external interrupt (NMI or INT) is present.

Fig. 7. Operating Modes



BRKEM imm8

This is the basic instruction used to start the 8080 emulation mode. This instruction operates exactly the same as the BRK instruction, except that BRKEM resets the mode flag (MD) to 0. PSW, PS, and PC are saved to the stack. MD is then reset and the interrupt vector specified by the operand imm8 of this command is loaded into PS and PC.

The instruction codes of the interrupt processing routine jumped to are then fetched. Then the CPU executes these codes as 8080 instructions.

In 8080 emulation mode, registers and flags of the 8080 are performed by the following registers and flags of the CXQ70108.

	8080	CXQ70108
Registers:	A	AL
	B	CH
	C	CL
	D	DH
	E	DL
	H	BH
	L	BL
	SP	BP
	PC	PC
Flags:	C	CY
	Z	Z
	S	S
	P	P
	AC	AC

In the native mode, SP is used for the stack pointer. In the 8080 emulation mode this function is performed by BP.

This use of independent stack pointers allows independent stack areas to be secured for each mode and keeps the stack of one of the modes from being destroyed by an erroneous stack operation in the other mode.

The SP, IX, IY and AH registers and the four segment registers (PS, SS, DSo, and DS₁) used in the native mode are not affected by operations in 8080 emulation mode.

In the 8080 emulation mode, the segment register for instructions is determined by the PS register (set automatically by the interrupt vector) and the segment register for data is the DSo register (set by the programmer immediately before the 8080 emulation mode is entered).

RETEM [no operand]

When RETEM is executed in 8080 emulation mode (interpreted by the CPU as a 8080 instruction), the CPU restores PS, PC and PSW (as it would when returning from an interrupt processing routine), and returns to the native mode. At the same time, the contents of the mode flag (MD) which was saved to the stack by the BRKEM instruction, is restored to MD = 1. The CPU is set to the native mode.

CALLN imm8

This instruction makes it possible to call the native mode subroutines from the 8080 emulation mode. To return from subroutine to the emulation mode, the RETI instruction is used.

The processing performed when this instruction is executed in the 8080 emulation mode (it is interpreted by the CPU as 8080 instruction), is similar to that performed when a BRK instruction is executed in the

native mode. The imm8 operand specifies an interrupt vector type. The contents of PS, PC, and PSW are pushed on the stack and an MD flag value of 0 is saved. The mode flag is set to 1 and the interrupt vector specified by the operand is loaded into PS and PC.

RETI [no operand]

This is a general-purpose instruction used to return from interrupt routines entered by the BRK instruction or by an external interrupt in the native mode. When this instruction is executed at the end of a subroutine entered by the execution of the CALLN instruction, the operation that restores PS, PC, and PSW is exactly the same as the native mode execution. When PSW is restored, however, the 8080 emulation mode value of the mode flag (MD) is restored, the CPU is set in emulation mode, and all subsequent instructions are interpreted and executed as 8080 instructions.

RETI is also used to return from an interrupt procedure initiated by an NMI or INT interrupt in the emulation mode.

Floating Point Operation Chip Instructions

FPO1 fp-op, mem/FPO2 fp-op, mem

These instructions are used for the external floating point processor. The floating point operation is passed to the floating point processor when the CPU fetches one of these instructions. From this point the CPU performs only the necessary auxiliary processing (effective address calculation, generation of physical addresses, and start-up of the memory read cycle).

The floating point processor always monitors the instructions fetched by the CPU. When it interprets one as an instruction to itself, it performs the appropriate processing. At this time, the floating point processor chip uses either the address alone or both the address and read data of the memory read cycle executed by the CPU. This difference in the data used depends on which of these instructions is executed.

Note: During the memory read cycle initiated by the CPU for FPO1 or FPO2 execution, the CPU does not accept any read data on the data bus from memory. Although the CPU generates the memory address, the data is used by the floating point processor.

Interrupt Operation

The interrupts used in the CXQ70108 can be divided into two types: interrupts generated by external interrupt requests and interrupts generated by software processing. These are the classifications.

External Interrupts

- (a) NMI input (nonmaskable)
- (b) INT input (maskable)

Software Processing

As the result of instruction execution

- When a divide error occurs during execution of the DIV or DIVU instruction
- When a memory-boundary-over error is detected by the CHKIND instruction

Conditional break instruction

- When $V = 1$ during execution of the BRKV instruction

Unconditional break instructions

- 1-byte break instruction: BRK3
- 2-byte break instruction: BRK imm8

Flag processing

- When stack operations are used to set the BRK flag

8080 Emulation mode instructions

- BRKEM imm8
- CALLN imm8

Interrupt Vectors

Starting addresses for interrupt processing routines are either determined automatically by a single location of the interrupt vector table or selected each time interrupt processing is entered.

The interrupt vector table is shown in figure 8. The table uses 1 K bytes of memory addresses 000H to 3FFH and can store starting address data for a maximum of 256 vectors (4 bytes per vector).

The corresponding interrupt sources for vectors 0 to 5 are predetermined and vectors 6 to 31 are reserved. These vectors consequently cannot be used for general applications.

The BRKEM instruction and CALLN instruction (in the emulation mode) and the INT input are available for general applications for vectors 32 to 255.

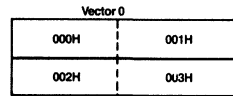
A single interrupt vector is made up of 4 bytes (figure 9). The 2 bytes in the low addresses of memory are loaded into PC as the offset, and the high 2 bytes are loaded into PS as the base address. The bytes are combined in reverse order. The lower-order bytes in the vector become the most significant bytes in the PC and PS, and the higher-order bytes become the least significant bytes.

Fig. 8. Interrupt Vector Table

000H	Vector 0	Divide Error	Dedicated
004H	Vector 1	Break Flag	
008H	Vector 2	NMI Input	
00CH	Vector 3	BRK 3 Instruction	
010H	Vector 4	BRKV Instruction	
014H	Vector 5	CHKIND Instruction	Reserved
018H	Vector 6		
01CH			
01EH			
01FH			
07CH	Vector 31		General Use
080H	Vector 32		
084H			
088H			
08CH			
3FCH	Vector 255		

• BRK imm8 Instruction
• BRKEM Instruction
• INT Input [External]
• CALLN Instruction

Fig. 9. Interrupt Vector 0



PS ← (003H, 002H)
PC ← (001H, 000H)

Based on this format, the contents of each vector should be initialized at the beginning of the program.

The basic steps to jump to an interrupt processing routine are now shown.

(SP — 1, SP — 2) ← PSW

(SP — 3, SP — 4) ← PS

(SP — 5, SP — 6) ← PC

SP ← SP — 6

IE ← 0, BRK ← 0, MD ← 1

PS ← vector high bytes

PC ← vector low bytes

Standby Function

The CXQ70108 has a standby mode to reduce power consumption during program wait states. This mode is set by the HALT instruction in both the native and the emulation mode.

In the standby mode, the internal clock is supplied only to those circuits related to functions required to release this mode and bus hold control functions. As a result, power consumption can be reduced to 1/10 the level of normal operation in either native or emulation mode.

The standby mode is released by inputting a RESET signal or an external interrupt (NMI, INT).

The bus hold function is effective during standby mode. The CPU returns to standby mode when the bus hold request is removed.

During standby mode, all control outputs are disabled and the address/data bus will be either high or low.

Instruction Set

The following tables briefly describe the CXQ70108's instruction set.

- Operation and Operand Types — defines abbreviations used in the Instruction Set table.
- Flag Operations — defines the symbols used to describe flag operations.
- Memory Addressing — shows how mem and mod combinations specify memory addressing modes.
- Selection of 8- and 16-Bit Registers — shows how reg and W select a register when mod = 111.
- Selection of Segment Registers — shows how sreg selects a segment register.
- Instruction Set — shows the instruction mnemonics, their effect, their operation codes the number of bytes in the instruction, the number of clocks required for execution, and the effect on the CXQ70108 flags.

Operation and Operand Types

Identifier	Description
reg	8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
dmem	8- or 16-bit direct memory location
mem	8- or 16-bit memory location
mem8	8-bit memory location
mem16	16-bit memory location
mem32	32-bit memory location
imm	Constant (0 to FFFFH)
imm16	Constant (0 to FFFFH)
imm8	Constant (0 to FFH)
imm4	Constant (0 to FH)
imm3	Constant (0 to 7)
acc	AW or AL register
sreg	Segment register
src-table	Name of 256-byte translation table

Identifier	Description
src-block	Name of block addressed by the IX register
dst-block	Name of block addressed by the IY register
near-proc	Procedure within the current program segment
far-proc	Procedure located in another program segment
near-label	Label in the current program segment
short-label	Label between -128 and +127 bytes from the end of instruction
far-label	Label in another program segment
memptr16	Word containing the offset of the memory location within the current program segment to which control is to be transferred
memptr32	Double word containing the offset and segment base address of the memory location to which control is to be transferred
regptr16	16-bit register containing the offset of the memory location within the program segment to which control is to be transferred
pop-value	Number of bytes of the stack to be discarded (0 to 64K bytes, usually even addresses)
fp-op	Immediate data to identify the instruction code of the external floating point operation
R	Register set
W	Word/byte field (0 to 1)
reg	Register field (000 to 111)
mem	Memory field (000 to 111)
mod	Mode field (00 to 10)
S:W	When S:W=01 or 11, data=16 bits. At all other times, data=8 bits.
X, XXX, YYY, ZZZ	Data to identify the instruction code of the external floating point arithmetic chip
AW	Accumulator (16 bits)
AH	Accumulator (high byte)
AL	Accumulator (low byte)
BW	BW register (16 bits)
CW	CW register (16 bits)
CL	CW register (low byte)
DW	DW register (16 bits)
SP	Stack pointer (16 bits)
PC	Program counter (16 bits)
PSW	Program status word (16 bits)
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)

Identifier	Description
PS	Program segment register (16 bits)
SS	Stack segment register (16 bits)
DS ₀	Data segment 0 register (16 bits)
DS ₁	Data segment 1 register (16 bits)
AC	Auxiliary carry flag
CY	Carry flag
P	Parity flag
S	Sign flag
Z	Zero flag
DIR	Direction flag
IE	Interrupt enable flag
V	Overflow flag
BRK	Break flag
MD	Mode flag
(. . .)	Values in parentheses are memory contents
disp	Displacement (8 or 16 bits)
ext-disp8	16-bit displacement (sign-extension byte+8-bit displacement)
temp	Temporary register (8/16/32 bits)
tmpcy	Temporary carry flag (1 bit)
seg	Immediate segment data (16 bits)
offset	Immediate offset data (16 bits)
←	Transfer direction
+	Addition
−	Subtraction
×	Multiplication
÷	Division
%	Modulo
AND	Logical product
OR	Logical sum
XOR	Exclusive logical sum
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value

Flag Operations

Identifier	Description
(blank)	No change
0	Cleared to 0
1	Set to 1
X	Set or cleared according to the result
U	Undefined
R	Value saved earlier is restored

Memory Addressing

mem	mod		
	00	01	10
000	BW + IX	BW + IX + disp8	BW + IX + disp16
001	BW + IY	BW + IY + disp8	BW + IY + disp16
010	BP + IX	BP + IX + disp8	BP + IX + disp16
011	BP + IY	BP + IY + disp8	BP + IY + disp16
100	IX	IX + disp8	IX + disp16
101	IY	IY + disp8	IY + disp16
110	Direct address	BP + disp8	BP + disp16
111	BW	BW + disp8	BW + disp16

Selection of 8-and 16-Bit Registers (mod 11)

reg	W=0	W=1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Selection of Segment Registers

sreg	
00	DS ₁
01	PS
10	SS
11	DS ₀

The table on the following pages shows the instruction set.

AT "No. of Clocks," for instructions referencing memory operands, the left side of the slash (/) is the number of clocks for byte operands and the right side is for word operands. For conditional control transfer instructions, the left side of the slash (/) is the number of clocks if a control transfer takes place. The right side is the number of clocks when no control transfer or branch occurs. Some instructions show a range of clock times, separated by a hyphen. The execution time of these instructions varies from the minimum value to the maximum, depending on the operands involved.

"No. of Clocks" includes these times:

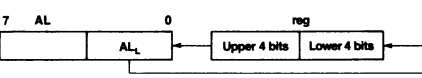
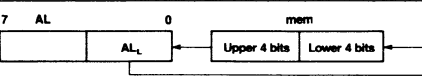
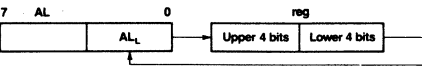
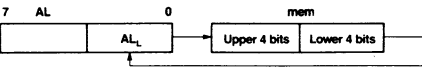
- Decoding
- Effective address generation
- Operand fetch
- Execution

It assumes that the instruction bytes have been prefetched.

Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Data Transfer Instructions																										
MOV	reg, reg	reg ← reg	1	0	0	0	1	0	1	W	1	1	reg	reg	2	2										
	mem, reg	(mem) ← reg	1	0	0	0	1	0	0	W	mod	reg	mem	9/13	2-4											
	reg, mem	reg ← (mem)	1	0	0	0	1	0	1	W	mod	reg	mem	11/15	2-4											
	mem, imm	(mem) ← imm	1	1	0	0	0	1	1	W	mod	0	0	0	mem	11/15	3-6									
	reg, imm	reg ← imm	1	0	1	1	W	reg	4	2-3																
	acc, dmem	When W = 0 AL ← (dmem) When W = 1 AH ← (dmem + 1), AL ← (dmem)	1	0	1	0	0	0	0	W	10/14	3														
	dmen, acc	When W = 0 (dmem) ← AL When W = 1 (dmem + 1) ← AH, (dmem) ← AL	1	0	1	0	0	0	1	W	9/13	3														
	sreg, reg16	sreg ← reg16 sreg : SS, DS0, DS1	1	0	0	0	1	1	1	0	1	1	0	sreg	reg	2	2									
	sreg, mem16	sreg ← (mem16) sreg : SS, DS0, DS1	1	0	0	0	1	1	1	0	mod	0	sreg	mem	11/15	2-4										
	reg16, sreg	reg16 ← sreg	1	0	0	0	1	1	0	0	1	1	0	sreg	reg	2	2									
	mem16, sreg	(mem16) ← sreg	1	0	0	0	1	1	0	0	mod	0	sreg	mem	10/14	2-4										
	DS0, reg16, mem32	reg16 ← (mem32) DS0 ← (mem32 + 2)	1	1	0	0	0	1	0	1	mod	reg	mem	18/26	2-4											
	DS1, reg16, mem32	reg16 ← (mem32) DS1 ← (mem32 + 2)	1	1	0	0	0	1	0	0	mod	reg	mem	18/26	2-4											
AH, PSW	AH ← S, Z, x, AC, x, P, x, CY	1	0	0	1	1	1	1	1	2	1	x	x	x	x	x										
PSW, AH	S, Z, x, AC, x, P, x, CY ← AH	1	0	0	1	1	1	1	0	3	1	x	x	x	x	x										
LDEA	reg16, mem16 reg16 ← mem16	1	0	0	0	1	1	0	1	mod	reg	mem	4	2-4												
TRANS	src-table AL ← (BW + AL)	1	1	0	1	0	1	1	1	9	1															
XCH	reg, reg	reg ↔ reg	1	0	0	0	0	1	1	W	1	1	reg	reg	3	2										
	mem, reg or reg, mem	(mem) ↔ reg	1	0	0	0	0	1	1	W	mod	reg	mem	16/24	2-4											
	AW, reg16 or reg16, AW	AW ↔ reg16	1	0	0	1	0	reg	2	1																
Repeat Prefixed																										
REPC		While CW ≠ 0, the following primitive block transfer instruction is executed and CW is decremented (-1). If there is a waiting interrupt, it is processed. When CY ≠ 1, exit the loop.	0	1	1	0	0	1	0	1	2	1														
REPNC		While CW ≠ 0, the following primitive block transfer instruction is executed and CW is decremented (-1). If there is a waiting interrupt, it is processed. When CY ≠ 0, exit the loop.	0	1	1	0	0	1	0	0	2	1														

Mnemonic	Operand	Operation	Operation Code	No. of Clocks	No. of Bytes	Flags																			
			7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z														
Repeat Prefixed (cont)																									
REP REPE REPZ		While $CW \neq 0$, the following primitive block transfer instruction is executed and CW is decremented (-1). If there is a waiting interrupt, it is processed. If the primitive block transfer instruction is $CMPBK$ or $CMPM$ and $Z \neq 1$, exit the loop.	1 1 1 1 0 0 1 1	2	1																				
REPNE REPNZ		While $CW \neq 0$, the following primitive block transfer instruction is executed and CW is decremented (-1). If there is a waiting interrupt, it is processed. If the primitive block transfer instruction is $CMPBK$ or $CMPM$ and $Z \neq 0$, exit the loop.	1 1 1 1 0 0 1 0	2	1																				
Primitive Block Transfer Instructions																									
MOVBK	dst-block, src-block	When $W = 0$ ($IY \leftarrow (IX)$) DIR = 0: $IX \leftarrow IX + 1, IY \leftarrow IY + 1$ DIR = 1: $IX \leftarrow IX - 1, IY \leftarrow IY - 1$ When $W = 1$ ($IY + 1, IY$) $\leftarrow (IX + 1, IX)$ DIR = 0: $IX \leftarrow IX + 2, IY \leftarrow IY + 2$ DIR = 1: $IX \leftarrow IX - 2, IY \leftarrow IY - 2$	1 0 1 0 0 1 0 W	11 + 8n 11 + 16n	1																				
CMPBK	src-block, dst-block	When $W = 0$ ($IX - (IY)$) DIR = 0: $IX \leftarrow IX + 1, IY \leftarrow IY + 1$ DIR = 1: $IX \leftarrow IX - 1, IY \leftarrow IY - 1$ When $W = 1$ ($IX + 1, IX$) $- (IY + 1, IY)$ DIR = 0: $IX \leftarrow IX + 2, IY \leftarrow IY + 2$ DIR = 1: $IX \leftarrow IX - 2, IY \leftarrow IY - 2$	1 0 1 0 0 1 1 W	7 + 14n 7 + 22n	1	x	x	x	x	x	x														
CMPM	dst-block	When $W = 0$ $AL - (IY)$ DIR = 0: $IY \leftarrow IY + 1$; DIR = 1: $IY \leftarrow IY - 1$ When $W = 1$ $AW - (IY + 1, IY)$ DIR = 0: $IY \leftarrow IY + 2$; DIR = 1: $IY \leftarrow IY - 2$	1 0 1 0 1 1 1 W	7 + 10n 7 + 14n	1	x	x	x	x	x	x														
LDM	src-block	When $W = 0$ $AL \leftarrow (IX)$ DIR = 0: $IX \leftarrow IX + 1$; DIR = 1: $IX \leftarrow IX - 1$ When $W = 1$ $AW \leftarrow (IX + 1, IX)$ DIR = 0: $IX \leftarrow IX + 2$; DIR = 1: $IX \leftarrow IX - 2$	1 0 1 0 1 1 0 W	7 + 9n 7 + 13n	1																				
STM	dst-block	When $W = 0$ (IY) $\leftarrow AL$ DIR = 0: $IY \leftarrow IY + 1$; DIR = 1: $IY \leftarrow IY - 1$ When $W = 1$ ($IY + 1, IY$) $\leftarrow AW$ DIR = 0: $IY \leftarrow IY + 2$; DIR = 1: $IY \leftarrow IY - 2$	1 0 1 0 1 0 1 W	7 + 4n 7 + 8n	1																				
Bit Field Transfer Instructions																									
INS	reg8, reg8	16-Bit field $\leftarrow AW$	0 0 0 0 1 1 1 1 0 0 1 1 0 0 0 1 1 1 reg reg	35-133	3																				
	reg8, imm4	16-Bit field $\leftarrow AW$	0 0 0 0 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 reg	75-103	4																				

Mnemonic	Operand	Operation	Operation Code													No. of Clocks	No. of Bytes	Flags								
			7	6	5	4	3	2	1	0	7	6	5	4	3			2	1	0	AC	CY	V	P	S	Z
Bit Field Transfer Instructions (cont)																										
EXT	reg8, reg8	AW ← 16-Bit field	0	0	0	0	1	1	1	1	0	0	1	1	0	0	1	1	34-59	3						
	reg8, imm4	AW ← 16-Bit field	1	1			reg		reg										25-52	4						
I/O Instructions																										
IN	acc, imm8	When W = 0 AL ← (imm8) When W = 1 AH ← (imm8 + 1), AL ← (imm8)	1	1	1	0	0	1	0										9/13	2						
	acc, DW	When W = 0 AL ← (DW) When W = 1 AH ← (DW + 1), AL ← (DW)	1	1	1	0	1	1	0										8/12	1						
OUT	imm8, acc	When W = 0 (imm8) ← AL When W = 1 (imm8 + 1) ← AH, (imm8) ← AL	1	1	1	0	0	1	1										8/12	2						
	DW, acc	When W = 0 (DW) ← AL When W = 1 (DW + 1) ← AH, (DW) ← AL	1	1	1	0	1	1	1										8/12	1						
Primitive I/O Instructions																										
INM	dst-block, DW	When W = 0 (IY) ← (DW) DIR = 0: IY ← IY + 1; DIR = 1: IY ← IY - 1	0	1	1	0	1	1	0	W	9 + 8n	1							9 + 16n							
		When W = 1 (IY + 1, IY) ← (DW + 1, DW) DIR = 0: IY ← IY + 2; DIR = 1: IY ← IY - 2																								
OUTM	DW, src-block	When W = 0 (DW) ← (IX) DIR = 0: IX ← IX + 1; DIR = 1: IX ← IX - 1	0	1	1	0	1	1	1	W	9 + 8n	1							9 + 16n							
		When W = 1 (DW + 1, DW) ← (IX + 1, IX) DIR = 0: IX ← IX + 2; DIR = 1: IX ← IX - 2										n: number of transfers														
Addition/Subtraction Instructions																										
ADD	reg, reg	reg ← reg + reg	0	0	0	0	0	0	1	W	1	1	reg	reg	2	2	x	x	x	x	x	x				
	mem, reg	(mem) ← (mem) + reg	0	0	0	0	0	0	0	W	mod	reg	mem	16/24	2-4	x	x	x	x	x	x					
	reg, mem	reg ← reg + (mem)	0	0	0	0	0	0	1	W	mod	reg	mem	11/15	2-4	x	x	x	x	x	x					
	reg, imm	reg ← reg + imm	1	0	0	0	0	0	S	W	1	1	0	0	reg	4	3-4	x	x	x	x	x	x			
	mem, imm	(mem) ← (mem) + imm	1	0	0	0	0	0	S	W	mod	0	0	0	mem	18/26	3-6	x	x	x	x	x	x			
	acc, imm	When W = 0 AL ← AL + imm When W = 1 AW ← AW + imm	0	0	0	0	0	1	0	W	4	2-3	x	x	x	x	x	x								
ADDC	reg, reg	reg ← reg + reg + CY	0	0	0	1	0	0	1	W	1	1	reg	reg	2	2	x	x	x	x	x	x				
	mem, reg	(mem) ← (mem) + reg + CY	0	0	0	1	0	0	0	W	mod	reg	mem	16/24	2-4	x	x	x	x	x	x					
	reg, mem	reg ← reg + (mem) + CY	0	0	0	1	0	0	1	W	mod	reg	mem	11/15	2-4	x	x	x	x	x	x					
	reg, imm	reg ← reg + imm + CY	1	0	0	0	0	0	S	W	1	1	0	1	reg	4	3-4	x	x	x	x	x	x			
	mem, imm	(mem) ← (mem) + imm + CY	1	0	0	0	0	0	S	W	mod	0	1	0	mem	18/26	3-6	x	x	x	x	x	x			

Mnemonic	Operand	Operation	Operation Code										No. of Clocks	No. of Bytes	Flags															
			7	6	5	4	3	2	1	0	7	6			5	4	3	2	1	0	AC	CY	V	P	S	Z				
Addition/Subtraction Instructions (cont)																														
ADDC	acc, imm	When W = 0 AL ← AL + imm + CY When W = 1 AW ← AW + imm + CY	0	0	0	1	0	1	0	W													4	2-3	x	x	x	x	x	x
SUB	reg, reg	reg ← reg - reg	0	0	1	0	1	0	1	W	1	1	reg	reg	2	2	x	x	x	x	x	x								
	mem, reg	(mem) ← (mem) - reg	0	0	1	0	1	0	0	W	mod	reg	mem	16/24	2-4	x	x	x	x	x	x									
	reg, mem	reg ← reg - (mem)	0	0	1	0	1	0	1	W	mod	reg	mem	11/15	2-4	x	x	x	x	x	x									
	reg, imm	reg ← reg - imm	1	0	0	0	0	0	S	W	1	1	1	0	1	reg	4	3-4	x	x	x	x	x	x						
	mem, imm	(mem) ← (mem) - imm	1	0	0	0	0	0	S	W	mod	1	0	1	mem	18/26	3-6	x	x	x	x	x	x							
acc, imm	When W = 0 AL ← AL - imm When W = 1 AW ← AW - imm	0	0	1	0	1	1	0	W													4	2-3	x	x	x	x	x	x	
SUBC	reg, reg	reg ← reg - reg - CY	0	0	0	1	1	0	1	W	1	1	reg	reg	2	2	x	x	x	x	x	x								
	mem, reg	(mem) ← (mem) - reg - CY	0	0	0	1	1	0	0	W	mod	reg	mem	16/24	2-4	x	x	x	x	x	x									
	reg, mem	reg ← reg - (mem) - CY	0	0	0	1	1	0	1	W	mod	reg	mem	11/15	2-4	x	x	x	x	x	x									
	reg, imm	reg ← reg - imm - CY	1	0	0	0	0	0	S	W	1	1	0	1	1	reg	4	3-4	x	x	x	x	x	x						
	mem, imm	(mem) ← (mem) - imm - CY	1	0	0	0	0	0	S	W	mod	0	1	1	mem	18/26	3-6	x	x	x	x	x	x							
acc, imm	When W = 0 AL ← AL - imm - CY When W = 1 AW ← AW - imm - CY	0	0	0	1	1	1	0	W													4	2-3	x	x	x	x	x	x	
BCD Operation Instructions																														
ADD4S		dst BCD string ← dst BCD string + src BCD string	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	7 + 19n	2	u	x	u	u	u	x	
SUB4S		dst BCD string ← dst BCD string - src BCD string	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	1	0	0	0	7 + 19n	2	u	x	u	u	u	x	
CMP4S		dst BCD string - src BCD string	0	0	0	0	1	1	1	1	0	0	1	0	0	1	1	0	0	0	0	7 + 19n	2	u	x	u	u	u	x	
		n: number of BCD numerals divided by 2																												
ROL4	reg8		0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	0	0	0	25	3							
	mem8		0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	0	0	0	28	3-5							
ROR4	reg8		0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0	1	0	0	29	3							
	mem8		0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0	1	0	0	33	3-5							

Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags				
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S
Increment/Decrement Instructions (cont)																									
INC	reg8	reg8 ← reg8 + 1	1	1	1	1	1	1	1	1	0	1	1	0	0	0	reg	2	2	x	x	x	x	x	
	mem	(mem) ← (mem) + 1	1	1	1	1	1	1	1	W	mod	0	0	0	0	mem	16/24	2-4	x	x	x	x	x		
	reg16	reg16 ← reg16 + 1	0	1	0	0	0	reg	2	1	x	x	x	x	x										
DEC	reg8	reg8 ← reg8 - 1	1	1	1	1	1	1	1	0	1	1	0	0	1	reg	2	2	x	x	x	x	x		
	mem	(mem) ← (mem) - 1	1	1	1	1	1	1	1	W	mod	0	0	1	mem	16/24	2-4	x	x	x	x	x			
	reg16	reg16 ← reg16 - 1	0	1	0	0	1	reg	2	1	x	x	x	x	x										
Multiplication Instructions																									
MULU	reg8	AW ← AL x reg8 AH = 0: CY ← 0, V ← 0 AH ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	1	1	1	0	0	reg	21-22	2	u	x	x	u	u		
	mem8	AW ← AL x (mem8) AH = 0: CY ← 0, V ← 0 AH ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	mod	1	0	0	mem	27-28	2-4	u	x	x	u	u			
	reg16	DW, AW ← AW x reg16 DW = 0: CY ← 0, V ← 0 DW ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	1	1	1	0	0	reg	29-30	2	u	x	x	u	u		
	mem16	DW, AW ← AW x (mem16) DW = 0: CY ← 0, V ← 0 DW ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	mod	1	0	0	mem	39-40	2-4	u	x	x	u	u			
MUL	reg8	AW ← AL x reg8 AH = AL sign expansion: CY ← 0, V ← 0 AH ≠ AL sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	1	1	1	0	1	reg	33-39	2	u	x	x	u	u		
	mem8	AW ← AL x (mem8) AH = AL sign expansion: CY ← 0, V ← 0 AH ≠ AL sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	mod	1	0	1	mem	39-45	2-4	u	x	x	u	u			
	reg16	DW, AW ← AW x reg16 DW = AW sign expansion: CY ← 0, V ← 0 DW ≠ AW sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	1	1	1	0	1	reg	41-47	2	u	x	x	u	u		
	mem16	DW, AW ← AW x (mem16) DW = AW sign expansion: CY ← 0, V ← 0 DW ≠ AW sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	mod	1	0	1	mem	51-57	2-4	u	x	x	u	u			
	reg16, (reg16,) imm8	reg16 ← reg16 x imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	1	1	1	1	reg	reg	28-34	3	u	x	x	u	u				
	reg16, mem16, imm8	reg16 ← (mem16) x imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	1	1	mod	reg	mem	38-44	3-5	u	x	x	u	u					

Mnemonic	Operand	Operation	Operation Code														No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2			1	0	AC	CY	V	P
Multiplication Instructions (cont)																								
MUL	reg16, (reg16,) imm16	reg16 ← reg16 x imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	0	1	1	1	reg	reg	36-42	4	u	x	x	u	u	u		
	reg16, mem16, imm16	reg16 ← (mem16) x imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	0	1	mod	reg	mem	46-52	4-6	u	x	x	u	u	u			
Unsigned Division Instructions																								
DIVU	reg8	temp ← AW When temp ÷ reg8 > FFH (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg8, AL ← temp ÷ reg8	1	1	1	1	0	1	1	0	1	1	1	1	0	reg	19	2	u	u	u	u	u	u
	mem8	temp ← AW When temp ÷ (mem8) > FFH (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem8), AL ← temp ÷ (mem8)	1	1	1	1	0	1	1	0	mod	1	1	0	mem	25	2-4	u	u	u	u	u	u	
	reg16	temp ← AW When temp ÷ reg16 > FFFFH (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg16, AL ← temp ÷ reg16	1	1	1	1	0	1	1	1	1	1	1	1	0	reg	25	2	u	u	u	u	u	u
	mem16	temp ← AW When temp ÷ (mem16) > FFFFH (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem16), AL ← temp ÷ (mem16)	1	1	1	1	0	1	1	1	mod	1	1	0	mem	35	2-4	u	u	u	u	u	u	
Signed Division Instructions																								
DIV	reg8	temp ← AW When temp ÷ reg8 > 0 and temp ÷ reg8 > 7FH or temp ÷ reg8 < 0 and temp ÷ reg8 ≤ 0-7FH - 1 (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg8, AL ← temp ÷ reg8	1	1	1	1	0	1	1	0	1	1	1	1	1	reg	29-34	2	u	u	u	u	u	u

Mnemonic	Operand	Operation	Operation Code													No. of Clocks	No. of Bytes	Flags				
			7	6	5	4	3	2	1	0	7	6	5	4	3			2	1	0	AC	CY
Signed Division Instructions (cont)																						
DIV	mem8	temp ← AW When temp÷(mem8)>0 and temp÷(mem8)>7FH or temp ÷ (mem8) < 0 and temp ÷ (mem8) ≤ 0-7FH - 1 (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem8), AL ← temp ÷ (mem8)	1 1 1 1 0 1 1 0	mod	1 1 1 1	mem	35-40	2-4	u	u	u	u	u	u	u							
	reg16	temp ← AW When temp÷reg16>0 and temp÷reg16>7FFFH or temp ÷ reg16 < 0 and temp ÷ reg16 ≤ 0-7FFFH - 1 (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg16, AL ← temp ÷ reg 16	1 1 1 1 0 1 1 1	1 1 1 1	reg	38-43	2	u	u	u	u	u	u	u								
	mem16	temp ← AW When temp÷(mem16)>0 and temp÷(mem16)>7FFFH or temp ÷ (mem16) < 0 and temp ÷ (mem16) ≤ 0-7FFFH - 1 (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem16), AL ← temp ÷ (mem16)	1 1 1 1 0 1 1 1	mod	1 1 1 1	mem	48-53	2-4	u	u	u	u	u	u	u							
BCD Complement Instructions																						
ADJBA		When (AL AND OFH) > 9 or AC = 1, AL ← AL + 6, AH ← AH + 1, AC ← 1, CY ← AC, AL ← AL AND OFH	0 0 1 1 0 1 1 1				3	1	x	x	u	u	u	u								
ADJ4A		When (AL AND OFH) > 9 or AC = 1, AL ← AL + 6, CY ← CY OR AC, AC ← 1, When AL > 9FH, or CY = 1 AL ← AL + 60H, CY ← 1	0 0 1 0 0 1 1 1				3	1	x	x	u	x	x	x								
ADJBS		When (AL AND OFH) > 9 or AC = 1, AL ← AL - 6, AH ← AH - 1, AC ← 1, CY ← AC, AL ← AL AND OFH	0 0 1 1 1 1 1 1				7	1	x	x	u	u	u	u								
ADJ4S		When (AL AND OFH) > 9 or AC = 1, AL ← AL - 6, CY ← CY OR AC, AC ← 1 When AL > 9FH or CY = 1 AL ← AL - 60H, CY ← 1	0 0 1 0 1 1 1 1				7	1	x	x	u	x	x	x								

Mnemonic	Operand	Operation	Operation Code	No. of Clocks	No. of Bytes	Flags					
			7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Data Conversion Instructions											
CVTBD		AH ← AL ÷ 0AH, AL ← AL % 0AH	1 1 0 1 0 1 0 0 0 0 0 0 0 1 0 1 0	15	2	u	u	u	x	x	x
CVTDB		AH ← 0, AL ← AH x 0AH + AL	1 1 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0	7	2	u	u	u	x	x	x
CVTBW		When AL < 80H, AH ← 0, all other times AH ← FFH	1 0 0 1 1 0 0 0	2	1						
CVTWL		When AL < 8000H, DW ← 0, all other times DW ← FFFFH	1 0 0 1 1 0 0 1	4-5	1						
Comparison Instructions											
CMP	reg, reg	reg - reg	0 0 1 1 1 0 1 W 1 1 reg reg	2	2	x	x	x	x	x	x
	mem, reg	(mem) - reg	0 0 1 1 1 0 0 W mod reg mem	11/15	2-4	x	x	x	x	x	x
	reg, mem	reg - (mem)	0 0 1 1 1 0 1 W mod reg mem	11/15	2-4	x	x	x	x	x	x
	reg, imm	reg - imm	1 0 0 0 0 0 S W 1 1 1 1 1 reg	4	3-4	x	x	x	x	x	x
	mem, imm	(mem) - imm	1 0 0 0 0 0 S W mod 1 1 1 mem	13/17	3-6	x	x	x	x	x	x
	acc, imm	When W = 0, AL - imm When W = 1, AW - imm	0 0 1 1 1 1 0 W	4	2-3	x	x	x	x	x	x
Complement Instructions											
NOT	reg	reg ← $\overline{\text{reg}}$	1 1 1 1 0 1 1 W 1 1 0 1 0 reg	2	2						
	mem	(mem) ← $\overline{(\text{mem})}$	1 1 1 1 0 1 1 W mod 0 1 0 mem	16/24	2-4						
NEG	reg	reg ← $\overline{\text{reg}} + 1$	1 1 1 1 0 1 1 W 1 1 0 1 1 reg	2	2	x	x	x	x	x	x
	mem	(mem) ← $\overline{(\text{mem})} + 1$	1 1 1 1 0 1 1 W mod 0 1 1 mem	16/24	2-4	x	x	x	x	x	x
Logical Operation Instructions											
TEST	reg, reg	reg AND reg	1 0 0 0 0 1 0 W 1 1 reg reg	2	2	u	0	0	x	x	x
	mem, reg or reg, mem	(mem) AND reg	1 0 0 0 0 1 0 W mod reg mem	10/14	2-4	u	0	0	x	x	x
	reg, imm	reg AND imm	1 1 1 1 0 1 1 W 1 1 0 0 0 reg	4	3-4	u	0	0	x	x	x
	mem, imm	(mem) AND imm	1 1 1 1 0 1 1 W mod 0 0 0 mem	11/15	3-6	u	0	0	x	x	x
	acc, imm	When W = 0, AL AND imm8 When W = 1, AW AND imm8	1 0 1 0 1 0 0 W	4	2-3	u	0	0	x	x	x
	AND	reg, reg	reg ← reg AND reg	0 0 1 0 0 0 1 W 1 1 reg reg	2	2	u	0	0	x	x
mem, reg		(mem) ← (mem) AND reg	0 0 1 0 0 0 0 W mod reg mem	16/24	2-4	u	0	0	x	x	x
reg, mem		reg ← reg AND (mem)	0 0 1 0 0 0 1 W mod reg mem	11/15	2-4	u	0	0	x	x	x
reg, imm		reg ← reg AND imm	1 0 0 0 0 0 0 W 1 1 1 0 0 reg	4	3-4	u	0	0	x	x	x
mem, imm		(mem) ← (mem) AND imm	1 0 0 0 0 0 0 W mod 1 1 0 mem	18/26	3-6	u	0	0	x	x	x
acc, imm		When W = 0, AL ← AL AND imm8 When W = 1, AW ← AW AND imm16	0 0 1 0 0 1 0 W	4	2-3	u	0	0	x	x	x

Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags				
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S
Logical Operation Instructions (cont)																									
OR	reg, reg	reg ← reg OR reg	0	0	0	0	1	0	1	W	1	1	reg	reg	2	2	u	0	0	x	x	x			
	mem, reg	(mem) ← (mem) OR reg	0	0	0	0	1	0	0	W	mod	reg	mem	16/24	2-4	u	0	0	x	x	x				
	reg, mem	reg ← reg OR (mem)	0	0	0	0	1	0	1	W	mod	reg	mem	11/15	2-4	u	0	0	x	x	x				
	reg, imm	reg ← reg OR imm	1	0	0	0	0	0	0	W	1	1	0	0	1	reg	4	3-4	u	0	0	x	x	x	
	mem, imm	(mem) ← (mem) OR imm	1	0	0	0	0	0	0	W	mod	0	0	1	mem	18/26	3-6	u	0	0	x	x	x		
	acc, imm	When W = 0, AL ← AL OR imm8 When W = 1, AW ← AW OR imm16	0	0	0	0	1	1	0	W	4	2-3	u	0	0	x	x	x							
XOR	reg, reg	reg ← reg XOR reg	0	0	1	1	0	0	1	W	1	1	reg	reg	2	2	u	0	0	x	x	x			
	mem, reg	(mem) ← (mem) XOR reg	0	0	1	1	0	0	0	W	mod	reg	mem	16/24	2-4	u	0	0	x	x	x				
	reg, mem	reg ← reg XOR (mem)	0	0	1	1	0	0	1	W	mod	reg	mem	11/15	2-4	u	0	0	x	x	x				
	reg, imm	reg ← reg XOR imm	1	0	0	0	0	0	0	W	1	1	1	1	0	reg	4	3-4	u	0	0	x	x	x	
	mem, imm	(mem) ← (mem) XOR imm	1	0	0	0	0	0	0	W	mod	1	1	0	mem	18/26	3-6	u	0	0	x	x	x		
	acc, imm	When W = 0, AL ← AL XOR imm8 When W = 1, AW ← AW XOR imm16	0	0	1	1	0	1	0	W	4	2-3	u	0	0	x	x	x							
Bit Operation Instructions																									
TEST1	reg8, CL	reg8 bit no. CL = 0: Z ← 1 reg8 bit no. CL = 1: Z ← 0	2nd byte*				3rd byte*																		
	mem8, CL	(mem8) bit no. CL = 0: Z ← 1 (mem8) bit no. CL = 1: Z ← 0	0	0	0	1	0	0	0	0	1	1	0	0	0	reg	3	3	u	0	0	u	u	x	
	reg16, CL	reg16 bit no. CL = 0: Z ← 1 reg16 bit no. CL = 1: Z ← 0	0	0	0	1	0	0	0	1	1	1	0	0	0	reg	3	3	u	0	0	u	u	x	
	mem16, CL	(mem16) bit no. CL = 0: Z ← 1 (mem16) bit no. CL = 1: Z ← 0	0	0	0	1	0	0	0	1	mod	0	0	0	mem	16	3-5	u	0	0	u	u	x		
	reg8, imm3	reg8 bit no. imm3 = 0: Z ← 1 reg8 bit no. imm3 = 1: Z ← 0	0	0	0	1	1	0	0	0	1	1	0	0	0	reg	4	4	u	0	0	u	u	x	
	mem8, imm3	(mem8) bit no. imm3 = 0: Z ← 1 (mem8) bit no. imm3 = 1: Z ← 0	0	0	0	1	1	0	0	0	mod	0	0	0	mem	13	4-6	u	0	0	u	u	x		
	reg16, imm4	reg16 bit no. imm4 = 0: Z ← 1 reg16 bit no. imm4 = 1: Z ← 0	0	0	0	1	1	0	0	1	1	1	0	0	0	reg	4	4	u	0	0	u	u	x	
mem16, imm4	(mem16) bit no. imm4 = 0: Z ← 1 (mem16) bit no. imm4 = 1: Z ← 0	2nd byte*				3rd byte*																			
*Note: First byte = 0FH																									

Mnemonic	Operand	Operation	Operation Code	No. of Clocks	No. of Bytes	Flags				
			7 6 5 4 3 2 1 0			7 6 5 4 3 2 1 0	AC	CY	V	P
Bit Operation Instructions (cont)										
NOT1	reg8, CL	reg8 bit no. CL ← $\overline{\text{reg8 bit no. CL}}$	<div style="display: flex; justify-content: space-around;"> 2nd byte* 3rd byte* </div> 0 0 0 1 0 1 1 0 1 1 0 0 0 reg	4	3					
	mem8, CL	(mem8) bit no. CL ← $\overline{\text{(mem8) bit no. CL}}$	0 0 0 1 0 1 1 0 mod 0 0 0 mem	18	3-5					
	reg16, CL	reg16 bit no. CL ← $\overline{\text{reg16 bit no. CL}}$	0 0 0 1 0 1 1 1 1 1 0 0 0 reg	4	3					
	mem16, CL	(mem16) bit no. CL ← $\overline{\text{(mem16) bit no. CL}}$	0 0 0 1 0 1 1 1 mod 0 0 0 mem	26	3-5					
	reg8, imm3	reg8 bit no. imm3 ← $\overline{\text{reg8 bit no. imm3}}$	0 0 0 1 1 1 1 0 1 1 0 0 0 reg	5	4					
	mem8, imm3	(mem8) bit no. imm3 ← $\overline{\text{(mem8) bit no. imm3}}$	0 0 0 1 1 1 1 0 mod 0 0 0 mem	19	4-6					
	reg16, imm4	reg16 bit no. imm4 ← $\overline{\text{(reg16) bit no. imm4}}$	0 0 0 1 1 1 1 1 1 1 0 0 0 reg	5	4					
	mem16, imm4	(mem16) bit no. imm4 ← $\overline{\text{(mem16) bit no. imm4}}$	<div style="display: flex; justify-content: space-around;"> 2nd byte* 3rd byte* </div> 0 0 0 1 1 1 1 1 mod 0 0 0 mem *Note: First byte = 0FH	27	4-6					
CY	CY ← $\overline{\text{CY}}$	1 1 1 1 0 1 0 1	2	1			x			
CLR1	reg8, CL	reg8 bit no. CL ← 0	<div style="display: flex; justify-content: space-around;"> 2nd byte* 3rd byte* </div> 0 0 0 1 0 0 1 0 1 1 0 0 0 reg	5	3					
	mem8, CL	(mem8) bit no. CL ← 0	0 0 0 1 0 0 1 0 mod 0 0 0 mem	14	3-5					
	reg16, CL	reg16 bit no. CL ← 0	0 0 0 1 0 0 1 1 1 1 0 0 0 reg	5	3					
	mem16, CL	(mem16) bit no. CL ← 0	0 0 0 1 0 0 1 1 mod 0 0 0 mem	22	3-5					
	reg8, imm3	reg8 bit no. imm3 ← 0	0 0 0 1 1 0 1 0 1 1 0 0 0 reg	6	4					
	mem8, imm3	(mem8) bit no. imm3 ← 0	0 0 0 1 1 0 1 0 mod 0 0 0 mem	15	4-6					
	reg16, imm4	reg16 bit no. imm4 ← 0	0 0 0 1 1 0 1 1 1 1 0 0 0 reg	6	4					
	mem16, imm4	(mem16) bit no. imm4 ← 0	<div style="display: flex; justify-content: space-around;"> 2nd byte* 3rd byte* </div> 0 0 0 1 1 0 1 1 mod 0 0 0 mem *Note: First byte = 0FH	27	4-6					
	CY	CY ← 0	1 1 1 1 1 0 0 0	2	1			0		
DIR	DIR ← 0	1 1 1 1 1 1 0 0	2	1						

Mnemonic	Operand	Operation	Operation Code													No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3			2	1	0	AC	CY	V
Bit Operation Instructions (cont)																							
SET1	reg8, CL	reg8 bit no. CL ← 1	0	0	0	1	0	1	0	0	1	1	0	0	0	reg	4	3					
	mem8, CL	(mem8) bit no. CL ← 1	0	0	0	1	0	1	0	0	mod	0	0	0	mem	13	3-5						
	reg16, CL	reg16 bit no. CL ← 1	0	0	0	1	0	1	0	1	1	1	0	0	reg	4	3						
	mem16, CL	(mem16) bit no. CL ← 1	0	0	0	1	0	1	0	1	mod	0	0	0	mem	21	3-5						
	reg8, imm3	reg8 bit no. imm3 ← 1	0	0	0	1	1	1	0	0	1	1	0	0	reg	5	4						
	mem8, imm3	(mem8) bit no. imm3 ← 1	0	0	0	1	1	1	0	0	mod	0	0	0	mem	14	4-6						
	reg16, imm4	reg16 bit no. imm4 ← 1	0	0	0	1	1	1	0	1	1	1	0	0	reg	5	4						
	mem16, imm4	(mem16) bit no. imm4 ← 1	0	0	0	1	1	1	0	1	mod	0	0	0	mem	22	4-6						
			2nd byte*				3rd byte*																
			*Note: First byte = 0FH																				
	CY	CY ← 1	1	1	1	1	1	0	J	1							2	1		1			
	DIR	DIR ← 1	1	1	1	1	1	0	1							2	1						
Shift Instructions																							
SHL	reg, 1	CY ← MSB of reg, reg ← reg x 2 When MSB of reg ≠ CY, V ← 1 When MSB of reg = CY, V ← 0	1	1	0	1	0	0	0	W	1	1	1	0	0	reg	2	2	u	x	x	x	x
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 When MSB of (mem) ≠ CY, V ← 1 When MSB of (mem) = CY, V ← 0	1	1	0	1	0	0	0	W	mod	1	0	0	mem	16/24	2-4	u	x	x	x	x	
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← MSB of reg, reg ← reg x 2, temp ← temp - 1	1	1	0	1	0	0	1	W	1	1	1	0	0	reg	7 + n	2	u	x	u	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← MSB of (mem), (mem) ← (mem) x 2, temp ← temp - 1	1	1	0	1	0	0	1	W	mod	1	0	0	mem	19/27 + n	2-4	u	x	u	x	x	
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← MSB of reg, reg ← reg x 2, temp ← temp - 1	1	1	0	0	0	0	0	W	1	1	1	0	0	reg	7 + n	3	u	x	u	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← MSB of (mem), (mem) ← (mem) x 2, temp ← temp - 1	1	1	0	0	0	0	0	W	mod	1	0	0	mem	19/27 + n	3-5	u	x	u	x	x	
			n: number of shifts																				
SHR	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2 When MSB of reg ≠ bit following MSB of reg: V ← 1 When MSB of reg = bit following MSB of reg: V ← 0	1	1	0	1	0	0	0	W	1	1	1	0	1	reg	2	2	u	x	x	x	x

Mnemonic	Operand	Operation	Operation Code														No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2			1	0	AC	CY	V	P
Shift Instructions (cont)																								
SHR	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2 When MSB of (mem) ≠ bit following MSB of (mem): V ← 1 When MSB of (mem) = bit following MSB of (mem): V ← 0	1	1	0	1	0	0	0	W	mod	1	0	1	mem	16/24	2-4	u	x	x	x	x	x	
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp - 1	1	1	0	1	0	0	0	W	1	1	1	0	1	reg	7 + n	2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp - 1	1	1	0	1	0	0	1	W	mod	1	0	1	mem	19/27 + n	2-4	u	x	u	x	x	x	
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp - 1	1	1	0	0	0	0	0	W	1	1	1	0	1	reg	7 + n	3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp - 1 n: number of shifts	1	1	0	0	0	0	0	W	mod	1	0	1	mem	19/27 + n	3-5	u	x	u	x	x	x	
SHRA	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2, V ← 0 MSB of operand does not change	1	1	0	1	0	0	0	W	1	1	1	1	1	reg	2	2	u	x	0	x	x	x
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2, V ← 0, MSB of operand does not change	1	1	0	1	0	0	0	W	mod	1	1	1	mem	16/24	2-4	u	x	0	x	x	x	
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp - 1 MSB of operand does not change	1	1	0	1	0	0	1	W	1	1	1	1	1	reg	7 + n	2	u	x	u	x	x	x
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp - 1 MSB of operand does not change	1	1	0	1	0	0	1	W	mod	1	1	1	mem	19/27 + n	2-4	u	x	u	x	x	x	
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp - 1 MSB of operand does not change	1	1	0	0	0	0	0	W	1	1	1	1	1	reg	7 + n	3	u	x	u	x	x	x
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp - 1 MSB of operand does not change n: number of shifts	1	1	0	0	0	0	0	W	mod	1	1	1	mem	19/27 + n	3-5	u	x	u	x	x	x	

Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags				
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S
Rotation Instructions																									
ROL	reg, 1	CY ← MSB of reg, reg ← reg x 2 + CY MSB of reg ≠ CY: V ← 1 MSB of reg = CY: V ← 0	1	1	0	1	0	0	0	W	1	1	0	0	0	reg	2	2		x	x				
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 + CY MSB of (mem) ≠ CY: V ← 1 MSB of (mem) = CY: V ← 0	1	1	0	1	0	0	0	W	mod	0	0	0	0	mem	16/24	2-4		x	x				
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← MSB of reg, reg ← reg x 2 + CY temp ← temp - 1	1	1	0	1	0	0	1	W	1	1	0	0	0	reg	7 + n	2		x	u				
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← MSB of (mem), (mem) ← (mem) x 2 + CY temp ← temp - 1	1	1	0	1	0	0	1	W	mod	0	0	0	0	reg	19/27 + n	2-4		x	u				
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← MSB of reg, reg ← reg x 2 + CY temp ← temp - 1	1	1	0	0	0	0	0	W	1	1	0	0	0	reg	7 + n	3		x	u				
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← MSB of (mem), (mem) ← (mem) x 2 + CY temp ← temp - 1	1	1	0	0	0	0	0	W	mod	0	0	0	0	mem	19/27 + n	3-5		x	u				
n: number of shifts																									
ROR	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2 MSB of reg ← CY MSB of reg ≠ bit following MSB of reg: V ← 1 MSB of reg = bit following MSB of reg: V ← 0	1	1	0	1	0	0	0	W	1	1	0	0	1	reg	2	2		x	x				
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← CY MSB of (mem) ≠ bit following MSB of (mem): V ← 1 MSB of (mem) = bit following MSB of (mem): V ← 0	1	1	0	1	0	0	0	W	mod	0	0	0	1	mem	16/24	2-4		x	x				
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY temp ← temp - 1	1	1	0	1	0	0	1	W	1	1	0	0	1	reg	7 + n	2		x	u				
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← CY temp ← temp - 1	1	1	0	1	0	0	1	W	mod	0	0	0	1	mem	19/27 + n	2-4		x	u				
n: number of shifts																									

Mnemonic	Operand	Operation	Operation Code	No. of Clocks	No. of Bytes	Flags				
			7 6 5 4 3 2 1 0			7 6 5 4 3 2 1 0	AC	CY	V	P
Rotation Instructions (cont)										
ROR	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY temp ← temp - 1	1 1 0 0 0 0 0 W 1 1 0 0 1 reg	7 + n	3		x	u		
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← LSB of (mem), (mem) ← (mem) ÷ 2 temp ← temp - 1	1 1 0 0 0 0 0 W mod 0 0 1 mem	19/27 + n	3-5		x	u		
n: number of shifts										
Rotate Instruction										
ROLC	reg, 1	tmpcy ← CY, CY ← MSB of reg reg ← reg x 2 + tmpcy MSB of reg = CY: V ← 0 MSB of reg ≠ CY: V ← 1	1 1 0 1 0 0 0 W 1 1 0 1 0 reg	2	2		x	x		
	mem, 1	tmpcy ← CY, CY ← MSB of (mem) (mem) ← (mem) x 2 + tmpcy MSB of (mem) = CY: V ← 0 MSB of (mem) ≠ CY: V ← 1	1 1 0 1 0 0 0 W mod 0 1 0 mem	16/24	2-4		x	x		
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation: tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy temp ← temp - 1	1 1 0 1 0 0 1 W 1 1 0 1 0 reg	7 + n	2		x	u		
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation: tmpcy ← CY, CY ← MSB of (mem), (mem) ← (mem) x 2 + tmpcy temp ← temp - 1	1 1 0 1 0 0 1 W mod 0 1 0 mem	19/27 + n	2-4		x	u		
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy temp ← temp - 1	1 1 0 0 0 0 0 W 1 1 0 1 0 reg	7 + n	3		x	u		
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: tmpcy ← CY, CY ← MSB of (mem) (mem) ← (mem) x 2 + tmpcy temp ← temp - 1	1 1 0 0 0 0 0 W mod 0 1 0 mem	19/27 + n	3-5		x	u		
n: number of shifts										

Mnemonic	Operand	Operation	Operation Code	No. of Clocks	No. of Bytes	Flags					
			7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Rotate Instructions (cont)											
RORC	reg, 1	tmpcy ← CY, CY ← LSB of reg reg ← reg ÷ 2, MSB of reg ← tmpcy MSB of reg ≠ bit following MSB of reg: V ← 1 MSB of reg = bit following MSB of reg: V ← 0	1 1 0 1 0 0 0 W 1 1 1 0 1	reg	2	2		x	x		
	mem, 1	tmpcy ← CY, CY ← LSB of (mem) (mem) ← (mem) ÷ 2, MSB of (mem) ← tmpcy MSB of (mem) ≠ bit following MSB of (mem): V ← 1 MSB of (mem) = bit following MSB of (mem): V ← 0	1 1 0 1 0 0 0 W mod 0 1 1	mem	16/24	2-4		x	x		
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation: tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← tmpcy, temp ← temp - 1	1 1 0 1 0 0 1 W 1 1 0 1 1	reg	7 + n	2		x	u		
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation: tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← tmpcy, temp ← temp - 1	1 1 0 1 0 0 1 W mod 0 1 1	mem	19/27+n	2-4		x	u		
	reg, imm8	temp ← imm8, while temp ≠ 0 repeat this operation: tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2 MSB of reg ← tmpcy, temp ← temp - 1	1 1 0 0 0 0 0 W 1 1 0 1 1	reg	7 + n	3		x	u		
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← tmpcy, temp ← temp - 1	1 1 0 0 0 0 0 W mod 0 1 1	mem	19/27+n	3-5		x	u		
n: number of shifts											
Subroutine Control Instructions											
CALL	near-proc	(SP - 1, SP - 2) ← PC, SP ← SP - 2 PC ← PC + disp	1 1 1 0 1 0 0 0		20	3					
	regptr16	(SP - 1, SP - 2) ← PC, SP ← SP - 2 PC ← regptr16	1 1 1 1 1 1 1 1 1 1 0 1 0	reg	18	2					
	memptr16	(SP - 1, SP - 2) ← PC, SP ← SP - 2 PC ← (memptr16)	1 1 1 1 1 1 1 1 1 mod 0 1 0	mem	31	2-4					
	far-proc	(SP - 1, SP - 2) ← PS, (SP - 3, SP - 4) ← PC SP ← SP - 4, PS ← seg, PC ← offset	1 0 0 1 1 0 1 0		29	5					
	memptr32	(SP - 1, SP - 2) ← PS, (SP - 3, SP - 4) ← PC SP ← SP - 4, PS ← (memptr32 + 2), PC ← (memptr32)	1 1 1 1 1 1 1 1 1 mod 0 1 1	mem	47	2-4					

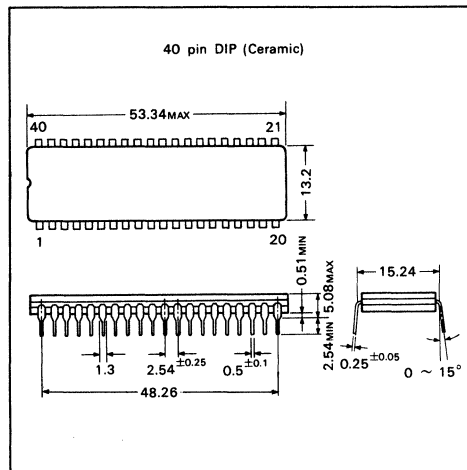
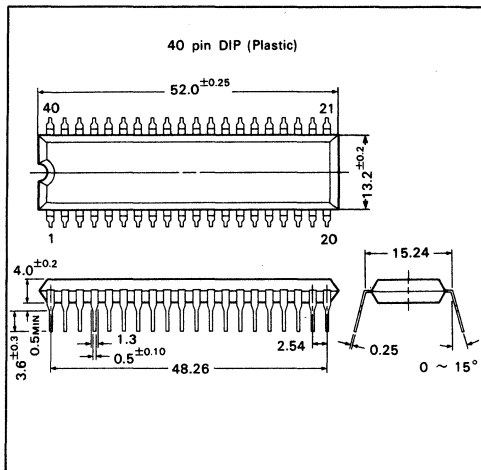
Mnemonic	Operand	Operation	Operation Code	No. of Clocks	No. of Bytes	Flags					
			7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Subroutine Control Instructions (cont)											
RET		$PC \leftarrow (SP + 1, SP), SP \leftarrow SP + 2$	1 1 0 0 0 0 0 1 1	19	1						
	pop-value	$PC \leftarrow (SP + 1, SP)$ $SP \leftarrow SP + 2, SP \leftarrow SP + \text{pop-value}$	1 1 0 0 0 0 0 1 0	24	3						
		$PC \leftarrow (SP + 1, SP), PS \leftarrow (SP + 3, SP + 2)$ $SP \leftarrow SP + 4$	1 1 0 0 1 0 1 1	29	1						
	pop-value	$PC \leftarrow (SP + 1, SP), PS \leftarrow (SP + 3, SP + 2)$ $SP \leftarrow SP + 4, SP \leftarrow SP + \text{pop-value}$	1 1 0 0 1 0 1 0	32	3						
Stack Manipulation Instructions											
PUSH	mem16	$(SP - 1, SP - 2) \leftarrow (\text{mem16}), SP \leftarrow SP - 2$	1 1 1 1 1 1 1 1 mod 1 1 0 mem	26	2-4						
	reg16	$(SP - 1, SP - 2) \leftarrow \text{reg16}, SP \leftarrow SP - 2$	0 1 0 1 0 reg	12	1						
	sreg	$(SP - 1, SP - 2) \leftarrow \text{sreg}, SP \leftarrow SP - 2$	0 0 0 sreg 1 1 0	12	1						
	PSW	$(SP - 1, SP - 2) \leftarrow \text{PSW}, SP \leftarrow SP - 2$	1 0 0 1 1 1 0 0	12	1						
	R	Push registers on the stack	0 1 1 0 0 0 0 0	67	1						
	imm	$(SP - 1, SP - 2) \leftarrow \text{imm}, SP \leftarrow SP - 2,$ When S = 1, sign extension	0 1 1 0 1 0 S 0	11/ 12	2-3						
POP	mem16	$(\text{mem16}) \leftarrow (SP + 1, SP), SP \leftarrow SP + 2$	1 0 0 0 1 1 1 1 mod 0 0 0 mem	25	2-4						
	reg16	$\text{reg16} \leftarrow (SP + 1, SP), SP \leftarrow SP + 2$	0 1 0 1 1 reg	12	1						
	sreg	$\text{sreg} \leftarrow (SP + 1, SP)$ sreg : SS, DS0, DS1 $SP \leftarrow SP + 2$	0 0 0 sreg 1 1 1	12	1						
	PSW	$\text{PSW} \leftarrow (SP + 1, SP), SP \leftarrow SP + 2$	1 0 0 1 1 1 0 1	12	1	R	R	R	R	R	
	R	Pop registers from the stack	0 1 1 0 0 0 0 1	75	1						
PREPARE	imm16, imm8	Prepare new stack frame	1 1 0 0 1 0 0 0 *: imm8 = 0: 16 imm8 ≥ 1: 23 + 16 (imm8 - 1)	*	4						
DISPOSE		Dispose of stack frame	1 1 0 0 1 0 0 1	10	1						
Branch Instruction											
BR	near-label	$PC \leftarrow PC + \text{disp}$	1 1 1 0 1 0 0 1	12	3						
	short-label	$PC \leftarrow PC + \text{ext-disp8}$	1 1 1 0 1 0 1 1	12	2						
	regptr16	$PC \leftarrow \text{regptr16}$	1 1 1 1 1 1 1 1 1 1 1 0 0 reg	11	2						
	memptr16	$PC \leftarrow (\text{memptr16})$	1 1 1 1 1 1 1 1 mod 1 0 0 mem	24	2-4						
	far-label	$PS \leftarrow \text{seg}, PC \leftarrow \text{offset}$	1 1 1 0 1 0 1 0	15	5						
	memptr32	$PS \leftarrow (\text{memptr32} + 2), PC \leftarrow (\text{memptr32})$	1 1 1 1 1 1 1 1 mod 1 0 1 mem	35	2-4						

Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Conditional Branch Instructions																										
BV	short-label	if V = 1, PC ← PC + ext-disp8	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	14/4	2							
BNV	short-label	if V = 0, PC ← PC + ext-disp8	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	14/4	2							
BC, BL	short-label	if CY = 1, PC ← PC + ext-disp8	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	14/4	2							
BNC, BNL	short-label	if CY = 0, PC ← PC + ext-disp8	0	1	1	1	0	0	0	1	1	0	0	0	0	0	0	14/4	2							
BE, BZ	short-label	if Z = 1, PC ← PC + ext-disp8	0	1	1	1	0	1	0	0	0	0	0	0	0	0	0	14/4	2							
BNE, BNZ	short-label	if Z = 0, PC ← PC + ext-disp8	0	1	1	1	0	1	0	1	0	1	0	0	0	0	0	14/4	2							
BNH	short-label	if CY OR Z = 1, PC ← PC + ext-disp8	0	1	1	1	0	1	1	0	0	0	0	0	0	0	0	14/4	2							
BH	short-label	if CY OR Z = 0, PC ← PC + ext-disp8	0	1	1	1	0	1	1	1	0	1	1	0	0	0	0	14/4	2							
BN	short-label	if S = 1, PC ← PC + ext-disp8	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	14/4	2							
BP	short-label	if S = 0, PC ← PC + ext-disp8	0	1	1	1	1	0	0	0	1	0	0	0	0	0	0	14/4	2							
BPE	short-label	if P = 1, PC ← PC + ext-disp8	0	1	1	1	1	0	1	0	0	0	0	0	0	0	0	14/4	2							
BPO	short-label	if P = 0, PC ← PC + ext-disp8	0	1	1	1	1	0	1	1	0	0	0	0	0	0	0	14/4	2							
BLT	short-label	if S XOR V = 1, PC ← PC + ext-disp8	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	14/4	2							
BGE	short-label	if S XOR V = 0, PC ← PC + ext-disp8	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	14/4	2							
BLE	short-label	if (S XOR V) OR Z = 1, PC ← PC + ext-disp8	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	14/4	2							
BGT	short-label	if (S XOR V) OR Z = 0, PC ← PC + ext-disp8	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	14/4	2							
DBNZNE	short-label	CW ← CW - 1 if Z = 0 and CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	14/5	2							
DBNZE	short-label	CW ← CW - 1 if Z = 1 and CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	0	0	1	0	0	0	0	0	0	14/5	2							
DBNZ	short-label	CW ← CW - 1 if CW ≠ 0, PC ← PC + ext-disp8	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	13/5	2							
BCWZ	short-label	if CW = 0, PC ← PC + ext-disp8	1	1	1	0	0	0	0	1	1	0	0	0	0	0	0	13/5	2							
Interrupt Instructions																										
BRK	3	(SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS, (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0 PS ← (15, 14), PC ← (13, 12)	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	50	1							
	imm8 (≠ 3)	(SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS, (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0 PC ← (n x 4, + 1, n x 4) PS ← (n x 4 + 3, n x 4 + 2) n = imm8	1	1	0	0	1	1	0	1	0	0	0	0	0	0	0	50	2							

Mnemonic	Operand	Operation	Operation Code	No. of Clocks	No. of Bytes	Flags					
			7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Interrupt Instructions (cont)											
BRKV		When V = 1 (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS, (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0 PS ← (19, 18), PC ← (17, 16)	1 1 0 0 1 1 1 0	52/3	1						
RETI		PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2), PSW ← (SP + 5, SP + 4), SP ← SP + 6	1 1 0 0 1 1 1 1	39	1	R	R	R	R	R	
CHKIND	reg16, mem32	When (mem32) > reg16 or (mem32 + 2) < reg16 (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS, (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (23, 22), PC ← (21, 20)	0 1 1 0 0 0 1 0 mod reg mem	73-76 26	2-4						
BRKEM	imm8	(SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS, (SP - 5, SP - 6) ← PC, SP ← SP - 6 MD ← 0, PC ← (n x 4 + 1, n x 4) PS ← (n x 4 + 3, n x 4 + 2), n = imm8	0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1	50	3						
CPU Control Instructions											
HALT		CPU Halt	1 1 1 1 0 1 0 0	2	1						
BUSLOCK		Bus Lock Prefix	1 1 1 1 0 0 0 0	2	1						
FP01	fp-op	No Operation	1 1 0 1 1 X X X 1 1 Y Y Y Z Z Z	2	2						
	fp-op, mem	data bus ← (mem)	1 1 0 1 1 X X X mod Y Y Y mem	15	2-4						
FP02	fp-op	No Operation	0 1 1 0 0 1 1 X 1 1 Y Y Y Z Z Z	2	2						
	fp-op, mem	data bus ← (mem)	0 1 1 0 0 1 1 X mod Y Y Y mem	15	2-4						
POLL		Poll and wait n: number of times POLL pin is sampled	1 0 0 1 1 0 1 1	2 + 5n	1						
NOP		No Operation	1 0 0 1 0 0 0 0	3	1						
DI		IE ← 0	1 1 1 1 1 0 1 0	2	1						
EI		IE ← 1	1 1 1 1 1 0 1 1	2	1						
8080 Mode Instructions											
RETEM		PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2), PSW ← (SP + 5, SP + 4), SP ← SP + 6	1 1 1 0 1 1 0 1 1 1 1 1 1 1 0 1	39	2	R	R	R	R	R	
CALLN	imm8	(SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS, (SP - 5, SP - 6) ← PC, SP ← SP - 6 MD ← 1, PC ← (n x 4 + 1, n x 4) PS ← (n x 4 + 3, n x 4 + 2), n = imm8	1 1 1 0 1 1 0 1 1 1 1 0 1 1 0 1	58	3						

Package Outline

Unit: mm



16-Bit Microprocessor

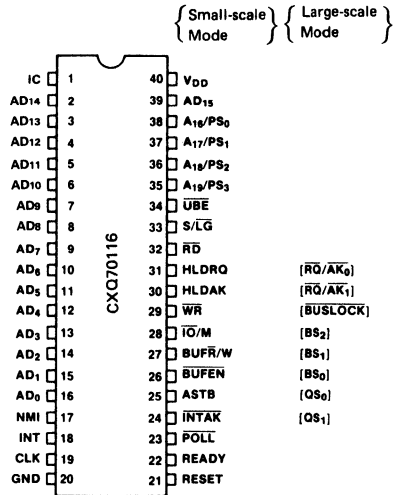
Description

The CXQ70116 is a CMOS 16-bit microprocessor with internal 16-bit architecture and a 16-bit external data bus. The CXQ70116 instruction set is a superset of the 8086/8088; however, mnemonics and execution times are different. The CXQ70116 additionally has a powerful instruction set including bit processing, packed BCD operations, and high-speed multiplication/division operations. The CXQ70116 can also emulate the functions of an 8080 and comes with a standby mode that significantly reduces power consumption. It is software-compatible with the CXQ70108 microprocessor.

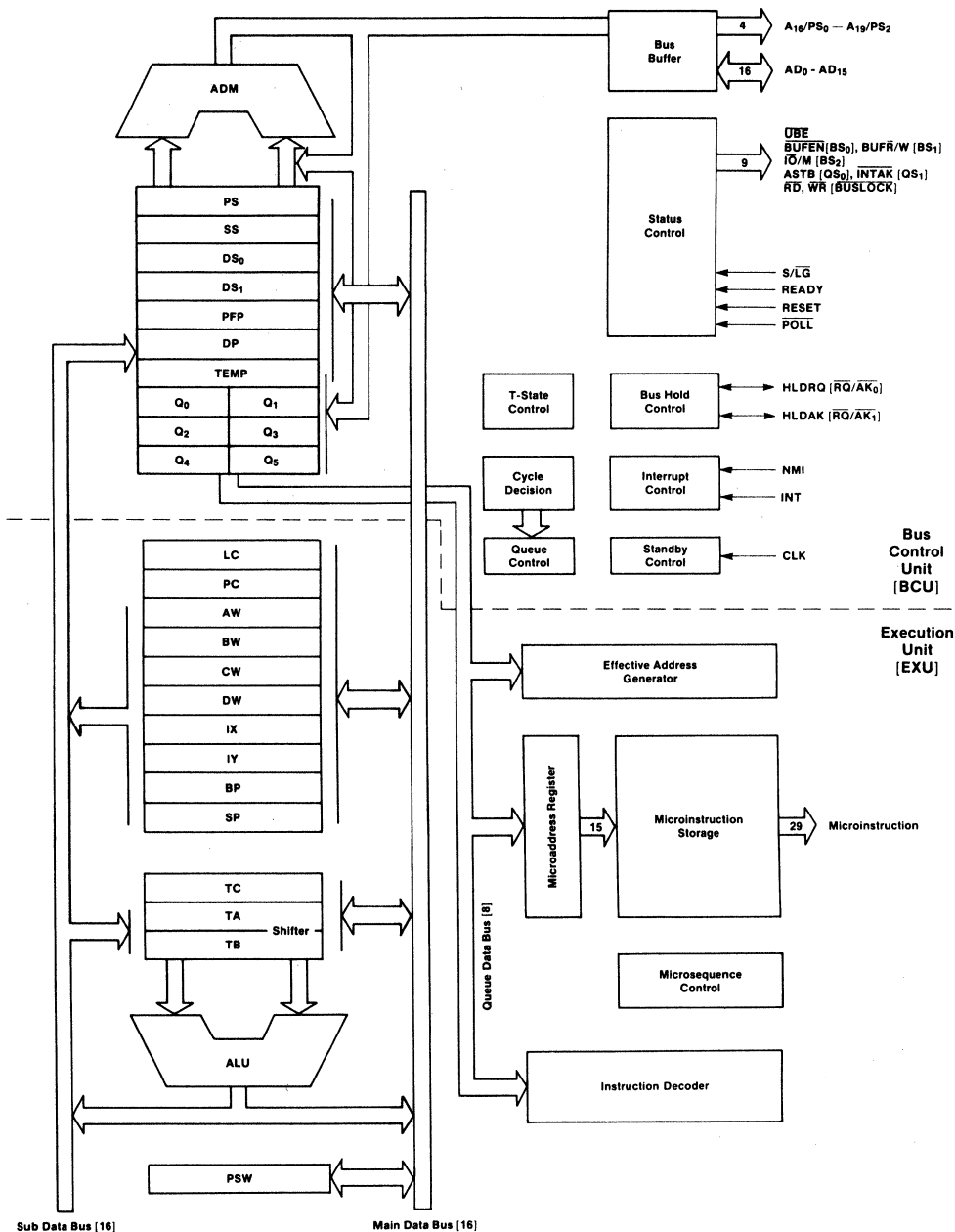
Features

- Minimum instruction execution time: 250 ns (at 8 MHz)
- Maximum addressable memory: 1 Mbytes
- Abundant memory addressing modes
- 14 × 16-bit register set
- 101 instructions
- Instruction set is a superset of 8086/8088 instruction set
- Bit, byte, word, and block operations
- Bit field operation instructions
- Packed BCD operation instructions
- Multiplication/division instructions execution time: 2.4 μs to 7.1 μs (at 8 MHz)
- High-speed block transfer instructions: 2 Mbytes/s (at 8 MHz)
- High-speed calculation of effective addresses: 2 clock cycles in any addressing mode
- Maskable (INT) and nonmaskable (NMI) interrupt inputs
- IEEE-796 bus compatible interface
- 8080 emulation functions
- CMOS technology
- Low power consumption
- Standby function
- Single power supply
- 5-MHz or 8-MHz clock
- 40-pin Plastic/Ceramic DIP (600 mil)
- NEC μPD70116 (V30) compatible

Pin Configuration (Top View)



Block Diagram



Pin Identification

No.	Symbol	Direction	Function
1	IC*		Internally connected
2—16	AD ₁₄ —AD ₀	In/Out	Address/data bus
17	NMI	In	Nonmaskable interrupt input
18	INT	In	Maskable interrupt input
19	CLK	In	Clock input
20	GND		Ground
21	RESET	In	Reset input
22	READY	In	Ready input
23	$\overline{\text{POLL}}$	In	Poll input
24	$\overline{\text{INTAK}}$ (QS ₁)	Out	Interrupt acknowledge output (queue status bit 1 output)
25	ASTB (QS ₀)	Out	Address strobe output (queue status bit 0 output)
26	$\overline{\text{BUFEN}}$ (BS ₀)	Out	Buffer enable output (bus status bit 0 output)
27	$\overline{\text{BUF}}\overline{\text{R}}/\overline{\text{W}}$ (BS ₁)	Out	Buffer read/write output (bus status bit 1 output)
28	$\overline{\text{IO}}/\overline{\text{M}}$ (BS ₂)	Out	Access is I/O or memory (bus status bit 2 output)
29	$\overline{\text{WR}}$ (BUSLOCK)	Out	Write strobe output (bus lock output)
30	HLD $\overline{\text{AK}}$ ($\overline{\text{RQ}}/\overline{\text{AK}}_1$)	Out (In/Out)	Hold acknowledge output, (bus hold request input/ acknowledge output 1)
31	HLD $\overline{\text{RQ}}$ ($\overline{\text{RQ}}/\overline{\text{AK}}_0$)	In (In/Out)	Hold request input (bus hold request input/acknowledge output 0)
32	$\overline{\text{RD}}$	Out	Read strobe output
33	S/ $\overline{\text{LG}}$	In	Small-scale/large-scale system input
34	$\overline{\text{UBE}}$	Out	Upper byte enable
35—38	A ₁₉ /PS ₃ —A ₁₆ /PS ₀	Out	Address bus, high bits or processor status output
39	AD ₁₅	In/Out	Address/data bus, bit 15
40	V _{DD}		Power supply

Notes: *IC should be connected to ground.

Where pins have different functions in small- and large-scale systems, the large-scale system pin symbol and function are in parentheses.

Unused input pins should be tied to ground or V_{DD} to minimize power dissipation and prevent the flow of potentially harmful currents.

Pin Functions

Some pins of the CXQ70116 have different functions according to whether the microprocessor is used in a small- or large-scale system. Other pins function the same way in either type of system.

AD₁₅ — AD₀ [Address/Data Bus]

For small- and large-scale systems.

AD₁₅ — AD₀ are the time-multiplexed address and data bus. They are active high. This bus contains the lower 16 bits of the 20-bit address during T₁ of the bus cycle. It is used as a 16-bit data bus during T₂, T₃, and T₄ of the bus cycle.

The address/data bus is a three-state bus and can be high or low during standby mode. The bus will float to the high impedance during hold and interrupt acknowledge.

NMI [Nonmaskable Interrupt]

For small- and large-scale systems.

This pin is used to input nonmaskable interrupt requests. NMI cannot be masked by software. This input is positive edge-triggered and can be sensed during any clock cycle. Actual interrupt processing begins, however, after completion of the instruction in progress.

The contents of interrupt vector 2 determine the starting address for the interrupt-servicing routine. Note that a hold request will be accepted even during NMI acknowledge.

This interrupt will cause the CXQ70116 to exit the standby mode.

INT [Maskable Interrupt]

For small- and large-scale systems.

This pin is a level-triggered interrupt request that can be masked by software.

INT is active high and is sensed during the last clock of the instruction. The interrupt will be accepted if the system is in interrupt enable state (if the interrupt enable flag IE is set). The CPU outputs the INTAK signal to inform external devices that the interrupt request has been granted.

If NMI and INT interrupts occur at the same time, NMI has higher priority than INT and INT cannot be accepted. A hold request will be accepted during INT acknowledge.

This interrupt causes the CXQ70116 to exit the standby mode.

CLK [Clock]

For small- and large-scale systems.

This pin is used for external clock input.

RESET [Reset]

For small- and large-scale systems.

This pin is used for the CPU reset signal. It is active high. Input of this signal has priority over all other operations. After the reset signal input returns low, the CPU begins execution of the program starting at address FFFF0H.

In addition to causing normal CPU start, RESET input will cause the CXQ70116 to exit the standby mode.

READY [Ready]

For small- and large-scale systems.

When the memory or I/O device being accessed cannot complete data read or write within the CPU basic access time, it can generate a CPU wait state (T_w) by setting this signal to inactive (low) and requesting a read/write cycle delay.

If the READY signal is active (high) during either T₃ or T_w state, the CPU will not generate a wait state.

POLL [Poll]

For small- and large-scale systems.

The CPU checks this input upon execution of the POLL instruction. If the input is low, then execution continues. If the input is high, the CPU will check the POLL input every five clock cycles until the input becomes low again.

The POLL and READY functions are used to synchronize CPU program execution with the operation of external devices.

RD [Read Strobe]

For small- and large-scale systems.

The CPU outputs this strobe signal during data read from an I/O device or memory. The IO/M signal is used to select between I/O and memory. RD will be high during standby mode. It is three-state and floats to the high impedance during hold acknowledge.

S/LG [Small/Large]

For small- and large-scale systems.

This signal determines the operation mode of the CPU. This signal is fixed either high or low. When this signal is high, the CPU will operate in small-scale system mode, and when low, in the large-scale system mode. A small-scale system will have at most one bus master such as a DMA controller device on the bus. A large-scale system can have more than one bus master accessing the bus as well as the CPU.

Pins 24 to 31 function differently depending on the operating mode of the CPU. Separate nomenclature is adopted for these signals in the two operational modes.

Pin No.	Function	
	S/LG-high	S/LG-low
24	INTAK	QS ₁
25	ASTB	QS ₀
26	BUFEN	BS ₀
27	BUFR/W	BS ₁
28	IO/M	BS ₂
29	WR	BUSLOCK
30	HLD _{AK}	RQ/AK ₁
31	HLD _{RQ}	RQ/AK ₀

INTAK [Interrupt Acknowledge]

For small-scale systems.

The CPU generates the INTAK signal low when it accepts an INT signal.

The interrupting device synchronizes with this signal and outputs the interrupt vector to the CPU via the data bus (AD₇ — AD₀). INTAK will be high during standby mode.

ASTB [Address Strobe]

For small-scale systems.

The CPU outputs this strobe signal to latch address information at an external latch. ASTB will be low during standby mode.

$\overline{\text{BUFEN}}$ [Buffer Enable]

For small-scale systems.

It is used as the output enable signal for an external bidirectional buffer. The CPU generates this signal during data transfer operations with external memory or I/O devices or during input of an interrupt vector.

$\overline{\text{BUFEN}}$ will be high during standby mode. It is three-state and floats to the high impedance during hold acknowledge.

 $\overline{\text{BUFR/W}}$ [Buffer Read/Write]

For small-scale systems.

The output of this signal determines the direction of data transfer with an external bidirectional buffer. A high output causes transmission from the CPU to the external device; a low signal causes data transfer from the external device to the CPU.

$\overline{\text{BUFR/W}}$ will be either high or low during standby mode. It is three-state and floats to the high impedance during hold acknowledge.

 $\overline{\text{IO/M}}$ [IO/Memory]

For small-scale systems.

The CPU generates this signal to specify either I/O access or memory access. A low-level output specifies I/O and a high-level specifies memory.

$\overline{\text{IO/M}}$ will be either high or low during standby mode. It is three-state and floats to the high impedance during hold acknowledge.

 $\overline{\text{WR}}$ [Write Strobe]

For small-scale systems.

The CPU generates this strobe signal during data write to an I/O device or memory. Selection of either I/O or memory is performed by the $\overline{\text{IO/M}}$ signal.

$\overline{\text{WR}}$ will be high during standby mode. It is three-state and floats to the high impedance during hold acknowledge.

 HLD\!AK [Hold Acknowledge]

For small-scale systems.

The HLD\!AK signal is used to indicate that the CPU accepts the hold request signal (HLD\!RQ). When this signal is high, the address bus, address/data bus, and the control lines become high impedance.

 HLD\!RQ [Hold Request]

For small-scale systems.

This input signal is used by external devices to request the CPU to release the address bus, address/data bus, and the control bus.

 $\overline{\text{UBE}}$ [Upper Byte Enable]

For small- and large-scale systems.

$\overline{\text{UBE}}$ indicates the use of the upper eight bits (AD_{15} — AD_8) of the address/data bus during a bus cycle. This signal is active low during T1 for read, write, and interrupt acknowledge cycles when AD_{15} — AD_8 are to be used. Bus cycles in which $\overline{\text{UBE}}$ is active are shown in the following table.

Type of Bus Operation	\overline{UBE}	AD_0	Number of Bus Cycle
Word at even address	0	0	1
Word at odd address	0 1	1* 0**	2
Byte at even address	1	0	1
Byte at odd address	0	1	1

Notes: *First bus cycle

**Second bus cycle

\overline{UBE} is low continuously during the interrupt acknowledge state. It will be high during standby mode. It is three-state and floats to the high impedance during hold acknowledge.

A19/PS3 — A16/PS0 [Address Bus/Processor Status]

For small- and large-scale systems.

These pins are time-multiplexed to operate as an address bus and as processor status signals.

When used as the address bus, these pins are the high 4 bits of the 20-bit memory address. During I/O access, all 4 bits output data 0.

The processor status signals are provided for both memory and I/O use. PS3 is always 0 in the native mode and 1 in 8080 emulation mode. The interrupt enable flag (IE) is output on pin PS2. Pins PS1 and PS0 indicate which memory segment is being accessed.

A17/PS1	A16/PS0	Segment
0	0	Data segment 1
0	1	Stack segment
1	0	Program segment
1	1	Data segment 0

A19/PS3 — A16/PS0 will be either high or low during standby mode. They are three-state and float to the high impedance during hold acknowledge.

QS1, QS0 [Queue Status]

For large-scale systems.

The CPU uses these signals to allow external devices, such as the floating-point arithmetic processor chip, to monitor the status of the internal CPU instruction queue.

QS1	QS0	Instruction Queue Status
0	0	NOP (queue does not change)
0	1	First byte of instruction
1	0	Flush queue
1	1	Subsequent bytes of instruction

The instruction queue status indicated by these signals is the status when the execution unit (EXU) accesses the instruction queue. The data output from these pins is therefore valid only for one clock cycle immediately following queue access. These status signals are provided so that the floating-point processor chip can monitor the CPU's program execution status and synchronize its operation with the CPU when control is passed to it by the FPO (Floating Point Operation) instructions.

QS1, QS0 will be low during standby mode.

BS₂ — BS₀ [Bus Status]

For large-scale systems.

The CPU uses these status signals to allow an external bus controller to monitor what the current bus cycle is.

The external bus controller decodes these signals and generates the control signals required to perform access of the memory or I/O device.

BS ₂	BS ₁	BS ₀	Bus Cycle
0	0	0	Interrupt acknowledge
0	0	1	I/O read
0	1	0	I/O write
0	1	1	Halt
1	0	0	Program fetch
1	0	1	Memory read
1	1	0	Memory write
1	1	1	Passive state

BS₂ — BS₀ will be high during standby mode. They are three-state and float to the high impedance during hold acknowledge.

BUSLOCK [Bus Lock]

For large-scale systems.

The CPU uses this signal to secure the bus while executing the instruction immediately following the BUSLOCK prefix instruction. It is a status signal to the other bus masters in a multiprocessor system inhibiting them from using the system bus during this time.

The output of this signal is three-state and becomes high impedance during hold acknowledge. BUSLOCK is high during standby mode except if the HALT instruction has a BUSLOCK prefix.

RQ/AK₁, RQ/AK₀ [Hold Request/Acknowledge]

For large-scale systems.

These pins function as bus hold request inputs (\overline{RQ}) and as bus hold acknowledge outputs (\overline{AK}). $\overline{RQ/AK_0}$ has a higher priority than $\overline{RQ/AK_1}$.

These pins have three-state outputs with on-chip pull-up resistors which keep the pin at high level when the output is high impedance.

V_{DD} [Power Supply]

For small- and large-scale systems.

This pin is used for the +5V power supply.

GND [Ground]

For small- and large-scale systems.

This pin is used for ground.

IC [Internally Connected]

This pin is used for tests performed at the factory by SONY. The CXQ70116 is used with this pin at ground potential.

Absolute Maximum Ratings

(Ta=+25°C)

Parameter	Symbol	Rating Value	Unit
Power supply voltage	V _{DD}	-0.5 to +7.0	V
Input voltage	V _I	-0.5 to V _{DD} +0.3	V
CLK input voltage	V _K	-0.5 to V _{DD} +1.0	V
Output voltage	V _O	-0.5 to V _{DD} +0.3	V
Power dissipation	P _{DMAX}	+0.5	W
Operating temperature	T _{opr}	-40 to +85	°C
Storage temperature	T _{stg}	-65 to +150	°C

Comment: Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of this specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

CXQ70116-5, Ta=-40°C to +85°C, V_{DD}=+5V±10%CXQ70116-8, Ta=-10°C to +70°C, V_{DD}=+5V±5%

Parameter	Symbol	Limits			Unit	Test Conditions
		Min.	Typ.	Max.		
Input voltage high	V _{IH}	2.2		V _{DD} +0.3	V	
Input voltage low	V _{IL}	-0.5		0.8	V	
CLK input voltage high	V _{KH}	3.9		V _{DD} +1.0	V	
CLK input voltage low	V _{KL}	-0.5		0.6	V	
Output voltage high	V _{OH}	0.7×V _{DD}			V	I _{OH} =-400 μA
Output voltage low	V _{OL}			0.4	V	I _{OL} =2.5 mA
Input leakage current high	I _{LIH}			10	μA	V _I =V _{DD}
Input leakage current low	I _{LIL}			-10	μA	V _I =0V
Output leakage current high	I _{LOH}			10	μA	V _O =V _{DD}
Output leakage current low	I _{LOL}			-10	μA	V _O =0V
Supply current	I _{DD}	70116-5 5 MHz	30	60	mA	Normal operation
			5	10	mA	Standby mode
		70116-8 8 MHz	45	80	mA	Normal Operation
			6	12	mA	Standby mode

Capacitance

(Ta=+25°C, V_{DD}=0V)

Parameter	Symbol	Limits		Unit	Test Conditions
		Min.	Max.		
Input capacitance	C _I		15	pF	f _c =1 MHz
I/O capacitance	C _{IO}		15	pF	Unmeasured pins returned to 0V

AC Characteristics

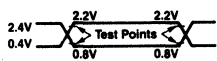
CXQ70116-5, $T_a = -40^\circ\text{C}$ to $+85^\circ\text{C}$, $V_{DD} = +5\text{V} \pm 10\%$
 CXQ70116-8, $T_a = -10^\circ\text{C}$ to $+70^\circ\text{C}$, $V_{DD} = +5\text{V} \pm 5\%$

Parameter	Symbol	CXQ70116-5		CXQ70116-8		Unit	Test Conditions
		Min.	Max.	Min.	Max.		
Small/Large Scale							
Clock cycle	tcyk	200	500	125	500	ns	
Clock pulse width high	tkkh	69		50		ns	$V_{KH} = 3.0\text{V}$
Clock pulse width low	tkkl	90		60		ns	$V_{KL} = 1.5\text{V}$
Clock rise time	tkr		10		8	ns	1.5V to 3.0V
Clock fall time	tkf		10		7	ns	3.0V to 1.5V
READY inactive setup to CLK ↓	tsrylk	-8		-8		ns	
READY inactive hold after CLK ↑	thkryh	30		20		ns	
READY active setup to CLK ↑	tsryhk	tkkl-8		tkkl-8		ns	
READY active hold after CLK ↑	thkryl	30		20		ns	
Data setup time to CLK ↓	tsdk	30		20		ns	
Data hold time after CLK ↓	thkd	10		10		ns	
NMI, INT, POLL setup time to CLK ↑	tsik	30		15		ns	
RESET setup time to CLK ↑	tsrst	30		20		ns	
RESET hold time to CLK ↑	thrst	10		10		ns	
Input rise time (except CLK)	tir		20		20	ns	0.8V to 2.2V
Input fall time (except CLK)	tif		12		12	ns	2.2V to 0.8V
Output rise time	tor		20		20	ns	0.8V to 2.2V
Output fall time	tof		12		12	ns	2.2V to 0.8V
Small Scale							
Address delay time from CLK	tdka	10	90	10	60	ns	$C_L = 100\text{ pF}$
Address hold time from CLK	thka	10		10		ns	
PS delay time from CLK ↓	tdkp	10	90	10	60	ns	
PS float delay time from CLK ↑	tfkp	10	80	10	60	ns	
Address setup time to ASTB ↓	tsast	tkkl-60		tkkl-30		ns	
Address float delay time from CLK ↓	tfka	thka	80	thka	60	ns	
ASTB ↑ delay time from CLK ↓	tdksth		80		50	ns	
ASTB ↓ delay time from CLK ↑	tdkstl		85		55	ns	
ASTB width high	tstst	tkkl-20		tkkl-10		ns	
Address hold time from ASTB ↓	thsta	tkkh-10		tkkl-10		ns	

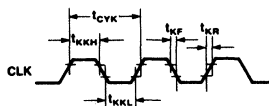
Parameter	Symbol	CXQ70116-5		CXQ70116-8		Unit	Test Conditions
		Min.	Max.	Min.	Max.		
Control delay time from CLK	tdKCT	10	110	10	65	ns	CL=100 pF
Address float to \overline{RD} ↓	tAFRL	0		0		ns	
\overline{RD} ↓ delay time from CLK ↓	tdKRL	10	165	10	80	ns	
\overline{RD} ↑ delay time from CLK ↓	tdKRH	10	150	10	80	ns	
Address delay time from \overline{RD} ↑	tDRHA	$t_{CYK}-45$		$t_{CYK}-40$		ns	
\overline{RD} width low	tRR	$2t_{CYK}-75$		$2t_{CYK}-50$		ns	
Data output delay time from CLK ↓	tdKD	10	90	10	60	ns	
Data float delay time from CLK ↓	tFKD	10	80	10	60	ns	
\overline{WR} width low	tWW	$2t_{CYK}-60$		$2t_{CYK}-40$		ns	
HLDRO setup time to CLK ↑	tSHQK	35		20		ns	
HLDAR delay time from CLK ↓	tdKHA	10	160	10	100	ns	
Large Scale							
Address delay time from CLK	tdKA	10	90	10	60	ns	CL=100 pF
Address hold time from CLK	thKA	10		10		ns	
PS delay time from CLK ↓	tdKP	10	90	10	60	ns	
PS float delay time from CLK ↑	tFKP	10	80	10	60	ns	
Address float delay time from CLK ↓	tFKA	thKA	80	thKA	60	ns	
Address delay time from \overline{RD} ↑	tDRHA	$t_{CYK}-45$		$t_{CYK}-40$		ns	
ASTB ↑ delay time from BS ↓	tDBST		15		15	ns	
BS ↓ delay time from CLK ↑	tdKBL	10	110	10	60	ns	
BS ↑ delay time from CLK ↓	tdKBH	10	130	10	65	ns	
\overline{RD} ↓ delay time from address float	tDAFRL	0		0		ns	
\overline{RD} ↓ delay time from CLK ↓	tdKRL	10	165	10	80	ns	
\overline{RD} ↑ delay time from CLK ↓	tdKRH	10	150	10	80	ns	
\overline{RD} width low	tRR	$2t_{CYK}-75$		$2t_{CYK}-50$		ns	
Data output delay time from CLK ↓	tdKD	10	90	10	60	ns	
Data float delay time from CLK ↓	tFKD	10	80	10	60	ns	
\overline{AK} delay time from CLK ↓	tdKAK		70		50	ns	
\overline{RQ} setup time to CLK ↑	tSRQK	20		10		ns	
\overline{RQ} hold time after CLK ↑	thKRQ	40		30		ns	

Timing Waveforms

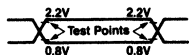
AC Test Input Waveform [Except CLK]



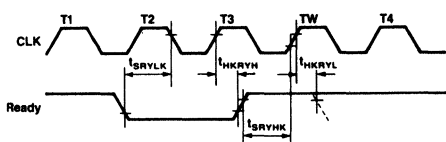
Clock Timing



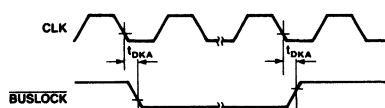
AC Output Test Points



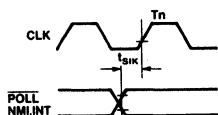
Wait [Ready] Timing



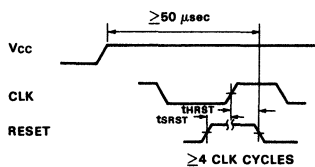
BUSLOCK Output Timing



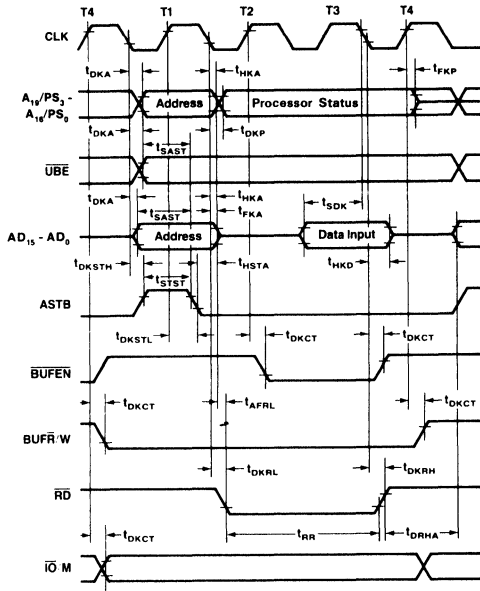
POLL, NMI, INT input Timing



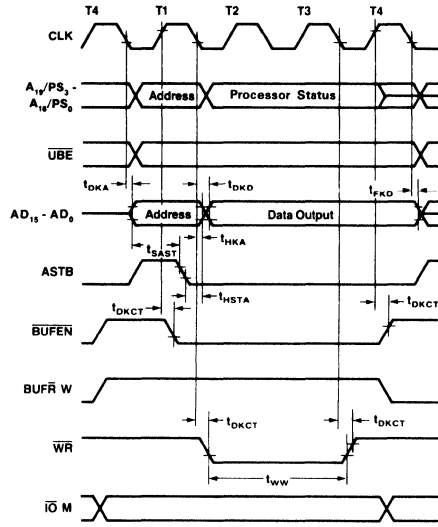
RESET Timing



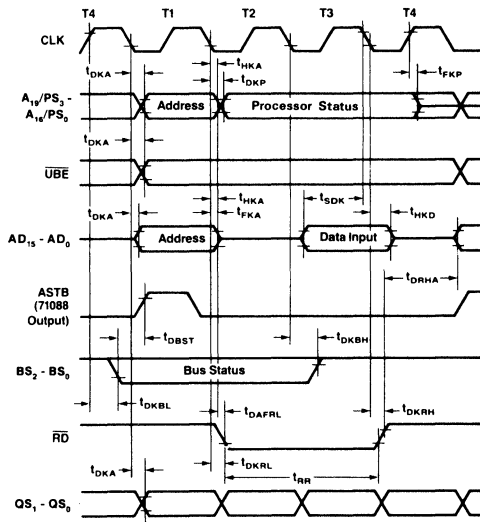
Read Timing [Small Scale]



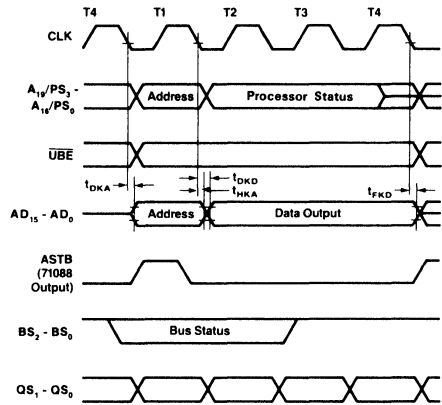
Write Timing [Small Scale]



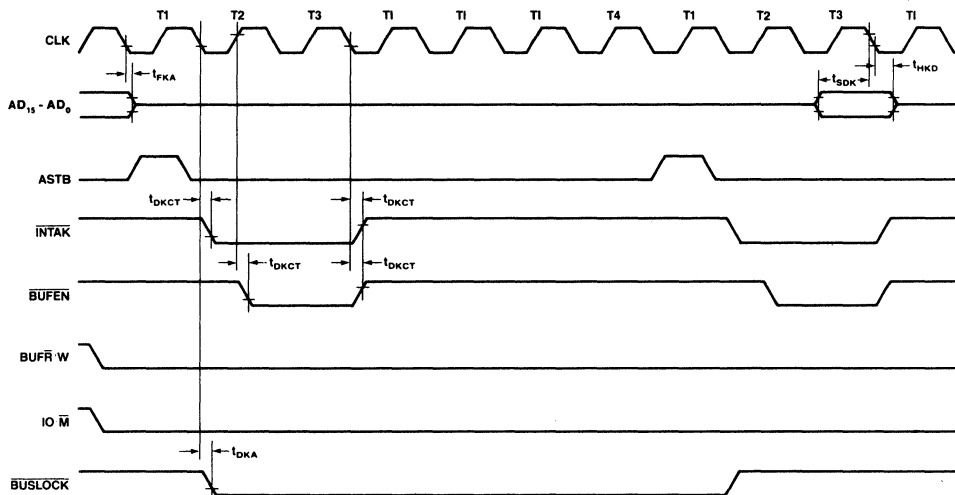
Read Timing [Large Scale]



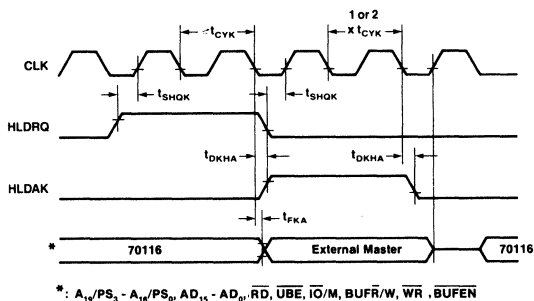
Write Timing [Large Scale]



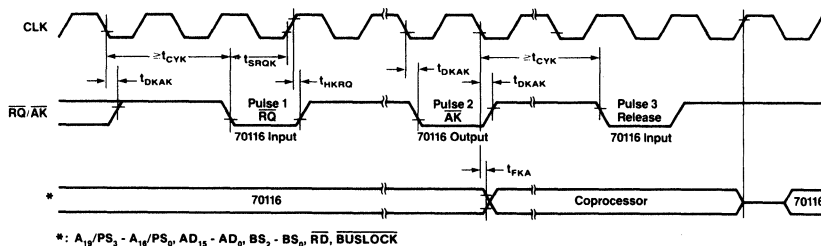
Interrupt Acknowledge Timing



Hold Request/Acknowledge Timing [Small Scale]



Bus Request/Acknowledge Timing [Large Scale]



Register Configuration

Program Counter [PC]

The program counter is a 16-bit binary counter that contains the segment offset address of the next instruction which the EXU is to execute.

The PC increments each time the microprogram fetches an instruction from the instruction queue. A new location value is loaded into the PC each time a branch, call, return, or break instruction is executed. At this time, the contents of the PC are the same as the Prefetch Pointer (PFP).

Prefetch Pointer [PFP]

The prefetch pointer (PFP) is a 16-bit binary counter which contains a segment offset which is used to calculate a program memory address that the bus control unit (BCU) uses to prefetch the next word for the instruction queue. The contents of PFP are an offset from the PS (Program Segment) register.

The PFP is incremented each time the BCU prefetches an instruction from the program memory. A new location will be loaded into the PFP whenever a branch, call, return, or break instruction is executed. At that time the contents of the PFP will be the same as those of the PC (Program Counter).

Segment Registers [PS, SS, DSo, and DS1]

The memory addresses accessed by the CXQ70116 are divided into 64K-byte logical segments. The starting (base) address of each segment is specified by a segment register, and the offset from this starting address is specified by the contents of another register or by the effective address.

These are the four types of segment registers used.

Segment Register	Default Offset
PS (Program Segment)	PFP
SS (Stack Segment)	SP, effective address
DSo (Data Segment 0)	IX, effective address
DS1 (Data Segment 1)	IY

General-Purpose Registers [AW, BW, CW, and DW]

There are four 16-bit general-purpose registers. Each one can be used as one 16-bit register or as two 8-bit registers by dividing them into their high and low bytes (AH, AL, BH, BL, CH, CL, DH, DL).

Each register is also used as a default register for processing specific instructions. The default assignments are:

AW: Word multiplication/division, word I/O, BCD rotation, data conversion, translation

AL: Byte multiplication/division, byte I/O, BCD rotation, data conversion, translation

AH: Byte multiplication/division

BW: Translation

CW: Loop control branch, repeat prefix

CL: Shift instructions, rotation instructions, BCD operations

DW: Word multiplication/division, indirect addressing I/O

Pointers [SP, BP] and Index Registers [IX, IY]

These registers serve as base pointers or index registers when accessing the memory using based addressing, indexed addressing, or based indexed addressing.

These registers can also be used for data transfer and arithmetic and logical operations in the same manner as the general-purpose registers. They cannot be used as 8-bit registers.

Also, each of these registers acts as a default register for specific operations. The default assignments are:

SP: Stack operations

IX: Block transfer (source), BCD string operations

IY: Block transfer (destination), BCD string operations

Program Status Word [PSW]

The program status word consists of the following six status and four control flags.

- | | |
|--|---|
| Status Flags <ul style="list-style-type: none"> • V (Overflow) • S (Sign) • Z (Zero) • AC (Auxiliary Carry) • P (Parity) • CY (Carry) | Control Flags <ul style="list-style-type: none"> • MD (Mode) • DIR (Direction) • IE (Interrupt Enable) • BRK (Break) |
|--|---|

When the PSW is pushed on the stack, the word images of the various flags are as shown here.

PSW

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M	1	1	1	V	D	I	B	S	Z	0	A	0	P	1	C
D					I	E	R				C				Y
					R		K								

The status flags are set and reset depending upon the result of each type of instruction executed. Instructions are provided to set, reset, and complement the CY flag directly. Other instructions set and reset the control flags and control the operation of the CPU.

High-Speed Execution of Instructions

This section highlights the major architectural features that enhance the performance of the CXQ70116.

- Dual data bus in EXU
- Effective address generator
- 16/32-bit temporary registers/shifters (TA, TB)
- 16-bit loop counter
- PC and PFP

Dual Data Bus Method

To reduce the number of processing steps for instruction execution, the dual data bus method has been adopted for the CXQ70116 (figure 1). The two data buses (the main data bus and the subdata bus) are both 16 bits wide. For addition/subtraction and logical and comparison operations, processing time has been speeded up some 30% over single-bus systems.

Fig. 1. Dual Data Buses

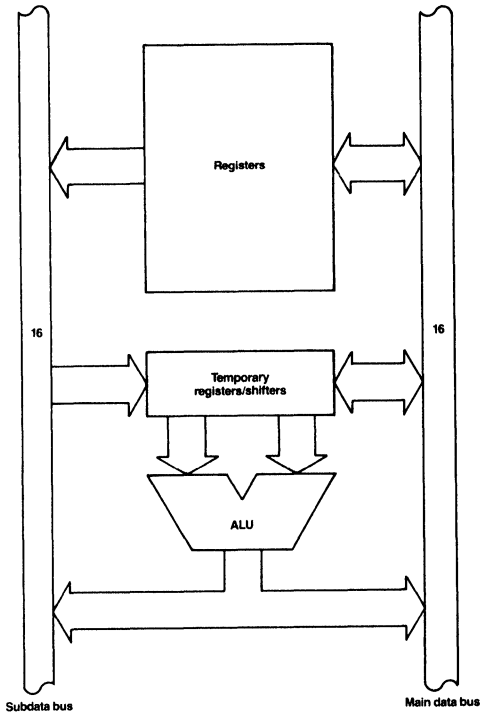
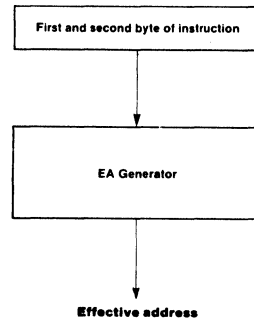


Fig. 2. Effective Address Generator

**Example**

ADD AW, BW	;AW ← AW + BW
Single Bus	Dual Bus
Step 1 TA ← AW	TA ← AW, TB ← BW
Step 2 TB ← BW	AW ← TA + TB
Step 3 AW ← TA + TB	

Effective Address Generator

This circuit (figure 2) performs high-speed processing to calculate effective addresses for accessing memory.

Calculating an effective address by the microprogramming method normally requires 5 to 12 clock cycles. This circuit requires only two clock cycles for addresses to be generated for any addressing mode. Thus, processing is several times faster.

16/32-Bit Temporary Registers/Shifters [TA, TB]

These 16-bit temporary registers/shifters (TA, TB) are provided for multiplication/division and shift/rotation instructions.

These circuits have decreased the execution time of multiplication/division instructions. In fact, these instructions can be executed about four times faster than with the microprogramming method.

TA + TB: 32-bit temporary register/shifter for multiplication and division instructions.

TB: 16-bit temporary register/shifter for shift/rotation instructions.

Loop Counter [LC]

This counter is used to count the number of loops for a primitive block transfer instruction controlled by a repeat prefix instruction and the number of shifts that will be performed for a multiple bit shift/rotation instruction.

The processing performed for a multiple bit rotation of a register is shown below. The average speed is approximately doubled over the microprogram method.

Example

RORC AW,CL ; CL = 5

Microprogram method

$8 + (4 \times 5) = 28$ clocks

LC method

$7 + 5 = 12$ clocks

Program Counter and Prefetch Pointer [PC and PFP]

The CXQ70116 microprocessor has a program counter (PC), which addresses the program memory location of the instruction to be executed next, and a prefetch pointer (PFP), which addresses the program memory location to be accessed next. Both functions are provided in hardware. A time saving of several clocks is realized for branch, call, return, and break instruction execution, compared with microprocessors that have only one instruction pointer.

Enhanced Instructions

In addition to the 8088/86 instructions, the CXQ70116 has the following enhanced instructions.

Instruction	Function
PUSH imm	Pushes immediate data onto stack
PUSH R	Pushes 8 general registers onto stack
POP imm	Pops immediate data onto stack
POP R	Pops 8 general registers from stack
MUL imm	Executes 16-bit multiply of register or memory contents by immediate data
SHL imm8 SHR imm8 SHRA imm8 ROL imm8 ROR imm8 ROLC imm8 RORC imm8	Shifts/rotates register or memory by immediate value
CHKIND	Checks array index against designated boundaries
INM	Moves a string from an I/O port to memory
OUTM	Moves a string from memory to an I/O port
PREPARE	Allocates an area for a stack frame and copies previous frame pointers
DISPOSE	Frees the current stack frame on a procedure exit

Enhanced Stack Operation Instructions

PUSH imm/POP imm

These instructions allow immediate data to be pushed onto or popped from the stack.

PUSH R/POP R

These instructions allow the contents of the eight general registers to be pushed onto or popped from the stack with a single instruction.

Enhanced Multiplication Instructions

MUL reg16, imm16/MUL mem16, imm16

These instructions allow the contents of a register or memory location to be 16-bit multiplied by immediate data.

Enhanced Shift and Rotate Instructions

SHL reg, imm8/SHR reg, imm8/SHRA reg, imm8

These instructions allow the contents of a register to be shifted by the number of bits defined by the immediate data.

ROL reg, imm8/ROR reg, imm8/ROLC reg, imm8/RORC reg, imm8

These instructions allow the contents of a register to be rotated by the number of bits defined by the immediate data.

Check Array Boundary Instruction

CHKIND reg16, mem32

This instruction is used to verify that index values pointing to the elements of an array data structure are within the defined range. The lower limit of the array should be in memory location mem32, the upper limit in mem32 + 2. If the index value in reg16 is not between these limits when CHKIND is executed, a BRK 5 will occur. This causes a jump to the location in interrupt vector 5.

Block I/O Instructions

OUTM DW, src-block/INM dst-block, DW

These instructions are used to output or input a string to or from memory, when preceded by a repeat prefix.

Stack Frame Instructions

PREPARE imm16, imm8

This instruction is used to generate the stack frames required by block-structures languages, such as PASCAL and Ada. The stack frame consists of two area. One area has a pointer that points to another frame which has variables that the current frame can access. The other is a local variable area for the current procedure.

DISPOSE

This instruction releases the last stack frame generated by the PREPARE instruction. It returns the stack and base pointers to the values they had before the PREPARE instruction was used to call a procedure.

Unique Instructions

In addition to the 8088/86 instructions and the enhanced instructions, the CXQ70116 has the following unique instructions.

Instruction	Function
INS	Insert bit field
EXT	Extract bit field
ADD4S	Adds packed decimal strings
SUB4S	Subtracts one packed decimal string from another
CMP4S	Compares two packed decimal strings
ROL4	Rotates one BCD digit left through AL lower 4 bits
ROR4	Rotates one BCD digit right through AL lower 4 bits
TEST1	Tests a specified bit and sets/resets Z flag
NOT1	Inverts a specified bit
CLR1	Clears a specified bit
SET1	Sets a specified bit
REPC	Repeats next instruction until CY flag is cleared
REPMC	Repeats next instruction until CY flag is set
FPO2	Additional floating point processor call

Variable Length Bit Field Operation Instructions

This category has two instructions: INS (Insert Bit Field) and EXT (Extract Bit Field). These instructions are highly effective for computer graphics and high-level languages. They can, for example, be used for data structures such as packed arrays and record type data used in PASCAL.

INS reg8, reg8/INS reg8, imm4

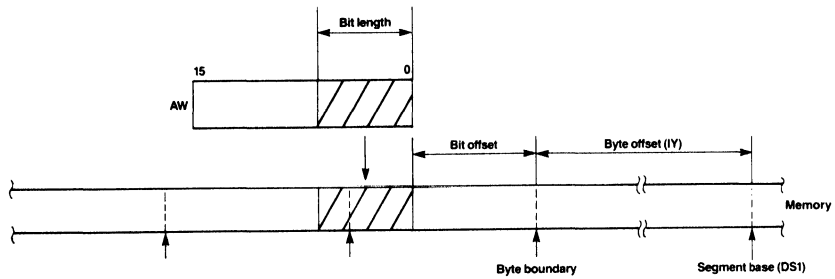
This instruction (figure 3) transfers low bits from the 16-bit AW register (the number of bits is specified by the second operand) to the memory location specified by the segment base (DS₁ register) plus the byte offset (IY register). The starting bit position within this byte is specified as an offset by the lower 4-bits of the first operand.

After each complete data transfer, the IY register and the register specified by the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may specify the number of bits transferred (second operand). Because the maximum transferable bit length is 16-bits, only the lower 4-bits of the specified register (00H to 0FH) will be valid.

Bit field data may overlap the byte boundary of memory.

Fig. 3. Bit Field Insertion



EXT reg8, reg8/EXT reg8, imm4

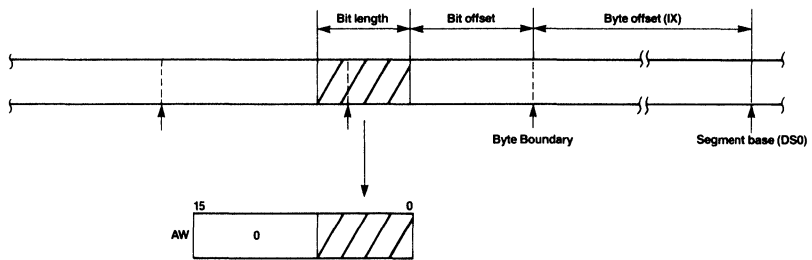
This instruction (figure 4) loads to the AW register the bit field data whose bit length is specified by the second operand of the instruction from the memory location that is specified by the DS0 segment register (segment base), the IX index register (byte offset), and the lower 4-bits of the first operand (bit offset).

After the transfer is complete, the IX register and the lower 4-bits of the first operand are automatically updated to point to the next bit field.

Either immediate data or a register may be specified for the second operand. Because the maximum transferrable bit length is 16 bits, however, only the lower 4-bits of the specified register (0H to 0FH) will be valid.

Bit field data may overlap the byte boundary of memory.

Fig. 4. Bit Field Extraction



Packed BCD Operation Instructions

The instructions described here process packed BCD data either as strings (ADD4S, SUB4S, CMP4S) or byte-format operands (ROR4, ROL4). Packed BCD strings may be from 1 to 255 digits in length.

When the number of digits is even, the zero and carry flags will be set according to the result of the operation. When the number of digits is odd, the zero and carry flags may not be set correctly in this case, (CL = odd), the zero flag will not be set unless the upper 4 bits of the highest byte are all zero. The carry flag will not be set unless there is a carry out of the upper 4 bits of the highest byte. When CL is odd, the contents of the upper 4 bits of the highest byte of the result are undefined.

ADD4S

This instruction adds the packed BCD string addressed by the IX index register to the packed BCD string addressed by the IY index register, and stores the result in the string addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the result of the operation will affect the carry flag (CY) and zero flag (Z).

$$\text{BCD string (IY, CL)} \leftarrow \text{BCD string (IY, CL)} + \text{BCD string (IX, CL)}$$

SUB4S

This instruction subtracts the packed BCD string addressed by the IX index register from the packed BCD string addressed by the IY index register, and stores the result in the string addressed by the IY register. The length of the string (number of BCD digits) is specified by the CL register, and the result of the operation will affect the carry flag (CY) and zero flag (Z).

$$\text{BCD string (IY, CL)} \leftarrow \text{BCD string (IY, CL)} - \text{BCD String (IX, CL)}$$

CMP4S

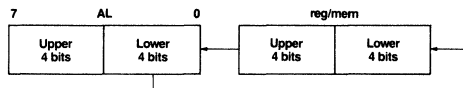
This instruction performs the same operation as SUB4S except that the result is not stored and only carry flags (CY) and zero flag (Z) are affected.

$$\text{BCD string (IY, CL)} - \text{BCD string (IX, CL)}$$

ROL4

This instruction (figure 5) treats the byte data of the register or memory directly specified by the instruction byte as BCD data and uses the lower 4 bits of the AL register (AL) to rotate that data one BCD digit to the left.

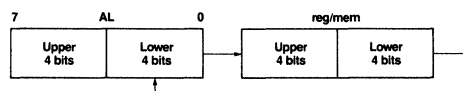
Fig. 5. BCD Rotate Left (ROL4)



ROR4

This instruction (figure 6) treats the byte data of the register or memory directly specified by the instruction byte as BCD data and uses the lower 4 bits of the AL register (AL) to rotate that data one BCD digit to the right.

Fig. 6. BCD Rotate Right (ROR4)



Bit Manipulation Instructions

TEST1

This instruction tests a specific bit in a register or memory location. If the bit is 1, the Z flag is reset to 0. If the bit is 0, the Z flag is set to 1.

NOT1

This instruction inverts a specific bit in a register or memory location.

CLR1

This instruction clears a specific bit in a register or memory location.

SET1

This instruction sets a specific bit in a register or memory location.

Repeat Prefix Instructions**REPC**

This instruction causes the CXQ70116 to repeat the following primitive block transfer instruction until the CY flag becomes cleared or the CW register becomes zero.

REPNC

This instruction causes the CXQ70116 to repeat the following primitive block transfer instruction until the CY flag becomes set or the CW register becomes zero.

Floating Point Instruction**FPO2**

This instruction is in addition to the 8088/86 floating point instruction, FPO1. These instructions are covered in a later section.

Mode Operation Instructions

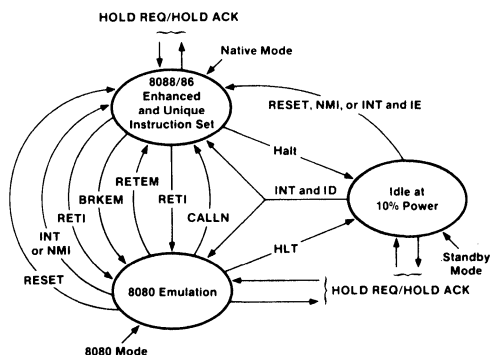
The CXQ70116 has two operating modes (figure 7). One is the native mode which executes 8088/86, enhanced and unique instructions. The other is the 8080 emulation mode in which the instruction set of the 8080 is emulated. A mode flag (MD) is provided to select between these two modes. Native mode is selected when MD is 1 and emulation mode when MD is 0. MD is set and reset, directly and indirectly, by executing the mode manipulation instructions.

Two instructions are provided to switch operation from the native mode to the emulation mode and back: BRKEM (Break for Emulation), and RETEM (Return from Emulation).

Two instructions are used to switch from the emulation mode to the native mode and back: CALLN (Call Native Routine), and RETI (Return from Interrupt).

The system will return from the 8080 emulation mode to the native mode when the RESET signal is present, or when an external interrupt (NMI or INT) is present.

Fig. 7. Operating Modes



BRKEM imm8

This is the basic instruction used to start the 8080 emulation mode. This instruction operates exactly the same as the BRK instruction, except that BRKEM resets the mode flag (MD) to 0. PSW, PS and PC are saved to the stack. MD is then reset and the interrupt vector specified by the operand imm8 of this command is loaded into PS and PC.

The instruction codes of the interrupt processing routine jumped to are then fetched. Then the CPU executes these codes as 8080 instructions.

In 8080 emulation mode, registers and flags of the 8080 are performed by the following registers and flags of the CXQ70116.

	8080	CXQ70116
Registers:	A	AL
	B	CH
	C	CL
	D	DH
	E	DL
	H	BH
	L	BL
	SP	BP
	PC	PC
Flags:	C	CY
	Z	Z
	S	S
	P	P
	AC	AC

In the native mode, SP is used for the stack pointer. In the 8080 emulation mode this function is performed by BP.

This use of independent stack pointers allows independent stack areas to be secured for each mode and keeps the stack of one of the modes from being destroyed by an erroneous stack operation in the other mode.

The SP, IX, IY and AH registers and the four segment registers (PS, SS, DSo, and DS_i) used in the native mode are not affected by operations in 8080 emulation mode.

In the 8080 emulation mode, the segment register for instructions is determined by the PS register (set automatically by the interrupt vector) and the segment register for data is the DSo register (set by the programmer immediately before the 8080 emulation mode is entered).

RETEM [no operand]

When RETEM is executed in 8080 emulation mode (interpreted by the CPU as 8080 instruction), the CPU restores PS, PC, and PSW (as it would when returning from an interrupt processing routine), and returns to the native mode. At the same time, the contents of the mode flag (MD) which was saved to the stack by the BRKEM instruction, is restored to MD = 1. The CPU is set to the native mode.

CALLN imm8

This instruction makes it possible to call the native mode subroutines from the 8080 emulation mode. To return from subroutine to the 8080 emulation mode, the RETI instruction is used.

The processing performed when this instruction is executed in the 8080 emulation mode (it is interpreted by the CPU as 8080 instruction), is similar to that performed when a BRK instruction is executed in the

native mode. The imm8 operand specifies an interrupt vector type. The contents of PS, PC, and PSW are pushed on the stack and an MD flag value of 0 is saved. The mode flag is set to 1 and the interrupt vector specified by the operand is loaded into PS and PC.

RETI [no operand]

This is a general-purpose instruction used to return from interrupt routines entered by the BRK instruction or by an external interrupt in the native mode. When this instruction is executed at the end of a subroutine entered by the execution of the CALLN instruction, the operation that restores PS, PC, and PSW is exactly the same as the native mode execution. When PSW is restored, however, the 8080 emulation mode value of the mode flag (MD) is restored, the CPU is set in emulation mode, and all subsequent instructions are interpreted and executed as 8080 instructions.

RETI is also used to return from an interrupt procedure initiated by an NMI or INT interrupt in the emulation mode.

Floating Point Operation Chip Instructions

FPO1 fp-op, mem/FPO2 fp-op, mem

These instructions are used for the external floating point processor. The floating point operation is passed to the floating point processor when the CPU fetches one of these instructions. From this point the CPU performs only the necessary auxiliary processing (effective address calculation, generation of physical addresses, and start-up of the memory read cycle).

The floating point processor always monitors the instructions fetched by the CPU. When it interprets one as an instruction to itself, it performs the appropriate processing. At this time, the floating point processor chip uses either the address alone or both the address and read data of the memory read cycle executed by the CPU. This difference in the data used depends on which of these instructions is executed.

Note: During the memory read cycle initiated by the CPU for FPO1 or FPO2 execution, the CPU does not accept any read data on the data bus from memory. Although the CPU generates the memory address, the data is used by the floating point processor.

Interrupt Operation

The interrupts used in the CXQ70116 can be divided into two types: interrupts generated by external interrupt requests and interrupts generated by software processing. These are the classifications.

External interrupts

- (a) NMI input (nonmaskable)
- (b) INT input (maskable)

Software processing

As the result of instruction execution

- When a divide error occurs during execution of the DIV or DIVU instruction
- When a memory-boundary-over error is detected by the CHKIND instruction

Conditional break instruction

- When V = 1 during execution of the BRKV instruction

Unconditional break instructions

- 1-byte break instruction: BRK3
- 2-byte break instruction: BRK imm8

Flag processing

- When stack operations are used to set the BRK flag

8080 Emulation mode instructions

- BRKEM imm8
- CALLN imm8

Interrupt vectors

Starting addresses for interrupt processing routines are either determined automatically by a single location of the interrupt vector table or selected each time interrupt processing is entered.

The interrupt vector table is shown in figure 8. The table uses 1 K bytes of memory addresses 000H to 3FFH and can store starting address data for a maximum of 256 vectors (4 bytes per vector).

The corresponding interrupt sources for vectors 0 to 5 are predetermined and vectors 6 to 31 are reserved. These vectors consequently cannot be used for general applications.

The BRKEM instruction and CALLN instruction (in the emulation mode) and the INT input are available for general applications for vectors 32 to 255.

A single interrupt vector is made up of 4 bytes (figure 9). The 2 bytes in the low addresses of memory are loaded into PC as the offset, and the high 2 bytes are loaded into PS as the base address. The bytes are combined in reverse order. The lower-order bytes in the vector become the most significant bytes in the PC and PS, and the higher-order bytes become the least significant bytes.

Fig. 8. Interrupt Vector Table

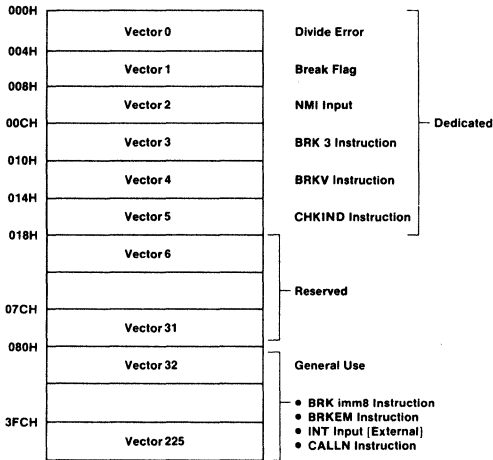
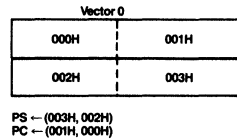


Fig. 9. Interrupt Vector 0



Based on this format, the contents of each vector should be initialized at the beginning of the program. The basic steps to jump to an interrupt processing routine are now shown.

- (SP -1, SP -2) ← PSW
- (SP -3, SP -4) ← PS
- (SP -5, SP -6) ← PC
- SP ← SP -6
- IE ← 0, BRK ← 0, MD ← 1
- PS ← vector high bytes
- PC ← vector low bytes

Standby Function

The CXQ70116 has a standby mode to reduce power consumption during program wait states. This mode is set by the HALT instruction in both the native and the emulation mode.

In the standby mode, the internal clock is supplied only to those circuits related to functions required to release this mode and bus hold control functions. As a result, power consumption can be reduced to 1/10 the level of normal operation in either native or emulation mode.

The standby mode is released by inputting a RESET signal or an external interrupt (NMI, INT).

The bus hold function is effective during standby mode. The CPU returns to standby mode when the bus hold request is removed.

During standby mode, all control outputs are disabled and the address/data bus will be either high or low.

Instruction Set

The following tables briefly describe the CXQ70116's instruction set.

- Operation and Operand Types — defines abbreviations used in the Instruction Set table.
- Flag Operations — defines the symbols used to describe flag operations.
- Memory Addressing — shows how mem and mod combinations specify memory addressing modes.
- Selection of 8- and 16-Bit Registers — shows how reg and W select a register when mod = 111.
- Selection of Segment Registers — shows how sreg selects a segment register.
- Instruction Set — shows the instruction mnemonics, their effect, their operation codes the number of bytes in the instruction, the number of clocks required for execution, and the effect on the CXQ70116 flags.

Operation and Operand Types

Identifier	Description
reg	8- or 16-bit general-purpose register
reg8	8-bit general-purpose register
reg16	16-bit general-purpose register
dmem	8- or 16-bit direct memory location
mem	8- or 16-bit memory location
mem8	8-bit memory location
mem16	16-bit memory location
mem32	32-bit memory location
imm	Constant (0 to FFFFH)
imm16	Constant (0 to FFFFH)
imm8	Constant (0 to FFH)
imm4	Constant (0 to FH)
imm3	Constant (0 to 7)
acc	AW or AL register
sreg	Segment register
src-table	Name of 256-byte translation table

Identifier	Description
src-block	Name of block addressed by the IX register
dst-block	Name of block addressed by the IY register
near-proc	Procedure within the current program segment
far-proc	Procedure located in another program segment
near-label	Label in the current program segment
short-label	Label between -128 and +127 bytes from the end of instruction
far-label	Label in another program segment
memptr16	Word containing the offset of the memory location within the current program segment to which control is to be transferred
memptr32	Double word containing the offset and segment base address of the memory location to which control is to be transferred
regptr16	16-bit register containing the offset of the memory location within the program segment to which control is to be transferred
pop-value	Number of bytes of the stack to be discarded (0 to 64K bytes, usually even addresses)
fp-op	Immediate data to identify the instruction code of the external floating point operation
R	Register set
W	Word/byte field (0 to 1)
reg	Register field (000 to 111)
mem	Memory field (000 to 111)
mod	Mode field (00 to 10)
S:W	When S:W=01 or 11, data=16 bits. At all other times, data=8 bits.
X, XXX, YYY, ZZZ	Data to identify the instruction code of the external floating point arithmetic chip
AW	Accumulator (16 bits)
AH	Accumulator (high byte)
AL	Accumulator (low byte)
BW	BW register (16 bits)
CW	CW register (16 bits)
CL	CW register (low byte)
DW	DW register (16 bits)
SP	Stack pointer (16 bits)
PC	Program counter (16 bits)
PSW	Program status word (16 bits)
IX	Index register (source) (16 bits)
IY	Index register (destination) (16 bits)

Identifier	Description
PS	Program segment register (16 bits)
SS	Stack segment register (16 bits)
DS ₀	Data segment 0 register (16 bits)
DS ₁	Data segment 1 register (16 bits)
AC	Auxiliary carry flag
CY	Carry flag
P	Parity flag
S	Sign flag
Z	Zero flag
DIR	Direction flag
IE	Interrupt enable flag
V	Overflow flag
BRK	Break flag
MD	Mode flag
(. . .)	Values in parentheses are memory contents
disp	Displacement (8 or 16 bits)
ext-disp8	16-bit displacement (sign-extension byte +8-bit displacement)
temp	Temporary register (8/16/32 bits)
tmpcy	Temporary carry flag (1 bit)
seg	Immediate segment data (16 bits)
offset	Immediate offset data (16 bits)
←	Transfer direction
+	Addition
-	Subtraction
×	Multiplication
÷	Division
%	Modulo
AND	Logical product
OR	Logical sum
XOR	Exclusive logical sum
XXH	Two-digit hexadecimal value
XXXXH	Four-digit hexadecimal value

Flag Operations

Identifier	Description
(blank)	No change
0	Cleared to 0
1	Set to 1
X	Set or cleared according to the result
U	Undefined
R	Value saved earlier is restored

Memory Addressing

mem	mod		
	00	01	10
000	BW + IX	BW + IX + disp8	BW + IX + disp16
001	BW + IY	BW + IY + disp8	BW + IY + disp16
010	BP + IX	BP + IX + disp8	BP + IX + disp16
011	BP + IY	BP + IY + disp8	BP + IY + disp16
100	IX	IX + disp8	IX + disp16
101	IY	IY + disp8	IY + disp16
110	Direct address	BP + disp8	BP + disp16
111	BW	BW + disp8	BW + disp16

Selection of 8-and 16-Bit Registers (mod 11)

reg	W=0	W=1
000	AL	AW
001	CL	CW
010	DL	DW
011	BL	BW
100	AH	SP
101	CH	BP
110	DH	IX
111	BH	IY

Selection of Segment Registers

sreg	
00	DS _i
01	PS
10	SS
11	DS _o

The table on the following pages shows the instruction set.

At "No. of Clocks," for instructions referencing memory operands, the left side of the slash (/) is the number of clocks for byte operands or word operands of an even address, and the right side is for word operands of an odd address. For conditional control transfer instructions, the left side of the slash (/) is the number of clocks if a control transfer takes place. The right side is the number of clocks when no control transfer or branch occurs. Some instructions show a range of clock times, separated by a hyphen. The execution time of these instructions varies from the minimum value to the maximum, depending on the operands involved.

Note: Add four clocks to these times for each word transfer made to an odd address.

"No. of Clocks" includes these times:

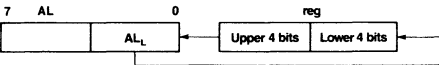
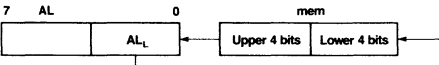
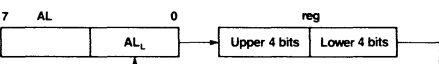
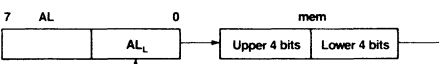
- Decoding
- Effective address generation
- Operand fetch
- Execution

It assumes that the instruction bytes have been prefetched.

Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Data Transfer Instructions																										
MOV	reg, reg	reg ← reg	1	0	0	0	1	0	1	W	1	1	reg	reg	2	2										
	mem, reg	(mem) ← reg	1	0	0	0	1	0	0	W	mod	reg	mem	9/13	2-4											
	reg, mem	reg ← (mem)	1	0	0	0	1	0	1	W	mod	reg	mem	11/15	2-4											
	mem, imm	(mem) ← imm	1	1	0	0	0	1	1	W	mod	0	0	0	mem	11/15	3-6									
	reg, imm	reg ← imm	1	0	1	1	W	reg	4	2-3																
	acc, dmem	When W = 0 AL ← (dmem) When W = 1 AH ← (dmem + 1), AL ← (dmem)	1	0	1	0	0	0	0	W	10/14	3														
	dmem, acc	When W = 0 (dmem) ← AL When W = 1 (dmem + 1) ← AH, (dmem) ← AL	1	0	1	0	0	0	1	W	9/13	3														
	sreg, reg16	sreg ← reg16 sreg : SS, DS0, DS1	1	0	0	0	1	1	1	0	1	1	0	sreg	reg	2	2									
	sreg, mem16	sreg ← (mem16) sreg : SS, DS0, DS1	1	0	0	0	1	1	1	0	mod	0	sreg	mem	11/15	2-4										
	reg16, sreg	reg16 ← sreg	1	0	0	0	1	1	0	0	1	1	0	sreg	reg	2	2									
	mem16, sreg	(mem16) ← sreg	1	0	0	0	1	1	0	0	mod	0	sreg	mem	10/14	2-4										
	DS0, reg16, mem32	reg16 ← (mem32) DS0 ← (mem32 + 2)	1	1	0	0	0	1	0	1	mod	reg	mem	18/26	2-4											
	DS1, reg16, mem32	reg16 ← (mem32) DS1 ← (mem32 + 2)	1	1	0	0	0	1	0	0	mod	reg	mem	18/26	2-4											
AH, PSW	AH ← S, Z, x, AC, x, P, x, CY	1	0	0	1	1	1	1	1	2	1	x	x	x	x	x										
PSW, AH	S, Z, x, AC, x, P, x, CY ← AH	1	0	0	1	1	1	1	0	3	1	x	x	x	x	x										
LDEA	reg16, mem16	reg16 ← mem16	1	0	0	0	1	1	0	1	mod	reg	mem	4	2-4											
TRANS	src-table	AL ← (BW + AL)	1	1	0	1	0	1	1	1	9	1														
XCH	reg, reg	reg ↔ reg	1	0	0	0	0	1	1	W	1	1	reg	reg	3	2										
	mem, reg or reg, mem	(mem) ↔ reg	1	0	0	0	0	1	1	W	mod	reg	mem	16/24	2-4											
	AW, reg16 or reg16, AW	AW ↔ reg16	1	0	0	1	0	reg	2	1																
Repeat Prefixed																										
REPC		While CW = 0, the following primitive block transfer instruction is executed and CW is decremented (- 1). If there is a waiting interrupt, it is processed. When CY = 1, exit the loop.	0	1	1	0	0	1	0	1	2	1														
REPNC		While CW = 0, the following primitive block transfer instruction is executed and CW is decremented (- 1). If there is a waiting interrupt, it is processed. When CY = 0, exit the loop.	0	1	1	0	0	1	0	0	2	1														

Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Repeat Prefixed (cont)																										
REP REPE REPZ		While $CW \neq 0$, the following primitive block transfer instruction is executed and CW is decremented (-1). If there is a waiting interrupt, it is processed. If the primitive block transfer instruction is $CMPBK$ or $CMPM$ and $Z \neq 1$, exit the loop.	1 1 1 1 0 0 1 1																2	1						
REPNE REPZ		While $CW \neq 0$, the following primitive block transfer instruction is executed and CW is decremented (-1). If there is a waiting interrupt, it is processed. If the primitive block transfer instruction is $CMPBK$ or $CMPM$ and $Z \neq 0$, exit the loop.	1 1 1 1 0 0 1 0																2	1						
Primitive Block Transfer Instructions																										
MOV BK	dst-block, src-block	When $W = 0$ ($IY \leftarrow IX$) DIR = 0: $IX \leftarrow IX + 1, IY \leftarrow IY + 1$ DIR = 1: $IX \leftarrow IX - 1, IY \leftarrow IY - 1$ When $W = 1$ ($IY + 1, IY \leftarrow IX + 1, IX$) DIR = 0: $IX \leftarrow IX + 2, IY \leftarrow IY + 2$ DIR = 1: $IX \leftarrow IX - 2, IY \leftarrow IY - 2$	1 0 1 0 0 1 0 W																$11 + 8n$	1						
CMP BK	dst-block, src-block	When $W = 0$ ($IX - (IY)$) DIR = 0: $IX \leftarrow IX + 1, IY \leftarrow IY + 1$ DIR = 1: $IX \leftarrow IX - 1, IY \leftarrow IY - 1$ When $W = 1$ ($IX + 1, IX - (IY + 1, IY)$) DIR = 0: $IX \leftarrow IX + 2, IY \leftarrow IY + 2$ DIR = 1: $IX \leftarrow IX - 2, IY \leftarrow IY - 2$	1 0 1 0 0 1 1 W																$7 + 14n$	1	x	x	x	x	x	x
CMP M	dst-block	When $W = 0$ $AL - (IY)$ DIR = 0: $IY \leftarrow IY + 1; DIR = 1: IY \leftarrow IY - 1$ When $W = 1$ $AW - (IY + 1, IY)$ DIR = 0: $IY \leftarrow IY + 2; DIR = 1: IY \leftarrow IY - 2$	1 0 1 0 1 1 1 W																$7 + 10n$	1	x	x	x	x	x	x
LDM	src-block	When $W = 0$ $AL \leftarrow IX$ DIR = 0: $IX \leftarrow IX + 1; DIR = 1: IX \leftarrow IX - 1$ When $W = 1$ $AW \leftarrow IX + 1, IX$ DIR = 0: $IX \leftarrow IX + 2; DIR = 1: IX \leftarrow IX - 2$	1 0 1 0 1 1 0 W																$7 + 9n$	1						
STM	dst-block	When $W = 0$ ($IY \leftarrow AL$) DIR = 0: $IY \leftarrow IY + 1; DIR = 1: IY \leftarrow IY - 1$ When $W = 1$ ($IY + 1, IY \leftarrow AW$) DIR = 0: $IY \leftarrow IY + 2; DIR = 1: IY \leftarrow IY - 2$	1 0 1 0 1 0 1 W																$7 + 4n$	1						
Bit Field Transfer Instructions																										
INS	reg8, reg8	16-Bit field $\leftarrow AW$	0 0 0 0 1 1 1 1 0 0 1 1 0 0 0 1 1 1 reg reg																$31-117$ $/35-133$	3						
	reg8, imm4	16-Bit field $\leftarrow AW$	0 0 0 0 1 1 1 1 0 0 1 1 1 0 0 1 1 1 0 0 0 reg																$67-87$ $/75-103$	4						

Mnemonic	Operand	Operation	Operation Code	No. of Clocks	No. of Bytes	Flags				
			7 6 5 4 3 2 1 0			7 6 5 4 3 2 1 0	AC	CY	V	P
Bit Field Transfer Instructions (cont)										
EXT	reg8, reg8	AW ← 16-Bit field	0 0 0 0 1 1 1 1 0 0 1 1 0 0 1 1 1 1 reg reg	26-55 /34-59	3					
	reg8, imm4	AW ← 16-Bit field	0 0 0 0 1 1 1 1 0 0 1 1 1 0 1 1 1 1 0 0 0 reg	21-44 /25-52	4					
I/O Instructions										
IN	acc, imm8	When W = 0 AL ← (imm8) When W = 1 AH ← (imm8 + 1), AL ← (imm8)	1 1 1 0 0 1 0 W	9/13	2					
	acc, DW	When W = 0 AL ← (DW) When W = 1 AH ← (DW + 1), AL ← (DW)	1 1 1 0 1 1 0 W	8/12	1					
OUT	imm8, acc	When W = 0 (imm8) ← AL When W = 1 (imm8 + 1) ← AH, (imm8) ← AL	1 1 1 0 0 1 1 W	8/12	2					
	DW, acc	When W = 0 (DW) ← AL When W = 1 (DW + 1) ← AH, (DW) ← AL	1 1 1 0 1 1 1 W	8/12	1					
Primitive I/O Instructions										
INM	dst-block, DW	When W = 0 (IY) ← (DW) DIR = 0: IY ← IY + 1; DIR = 1: IY ← IY - 1 When W = 1 (IY + 1, IY) ← (DW + 1, DW) DIR = 0: IY ← IY + 2; DIR = 1: IY ← IY - 2	0 1 1 0 1 1 0 W	9 + 8n	1					
OUTM	DW, src-block	When W = 0 (DW) ← (IX) DIR = 0: IX ← IX + 1; DIR = 1: IX ← IX - 1 When W = 1 (DW + 1, DW) ← (IX + 1, IX) DIR = 0: IX ← IX + 2; DIR = 1: IX ← IX - 2	0 1 1 0 1 1 1 W n: number of transfers	9 + 8n	1					
Addition/Subtraction Instructions										
ADD	reg, reg	reg ← reg + reg	0 0 0 0 0 0 1 W 1 1 reg reg	2	2	x	x	x	x	x
	mem, reg	(mem) ← (mem) + reg	0 0 0 0 0 0 0 W mod reg mem	16/24	2-4	x	x	x	x	x
	reg, mem	reg ← reg + (mem)	0 0 0 0 0 0 1 W mod reg mem	11/15	2-4	x	x	x	x	x
	reg, imm	reg ← reg + imm	1 0 0 0 0 0 S W 1 1 0 0 0 reg	4	3-4	x	x	x	x	x
	mem, imm	(mem) ← (mem) + imm	1 0 0 0 0 0 S W mod 0 0 0 mem	18/26	3-6	x	x	x	x	x
	acc, imm	When W = 0 AL ← AL + imm When W = 1 AW ← AW + imm	0 0 0 0 0 1 0 W	4	2-3	x	x	x	x	x
ADDC	reg, reg	reg ← reg + reg + CY	0 0 0 1 0 0 1 W 1 1 reg reg	2	2	x	x	x	x	x
	mem, reg	(mem) ← (mem) + reg + CY	0 0 0 1 0 0 0 W mod reg mem	16/24	2-4	x	x	x	x	x
	reg, mem	reg ← reg + (mem) + CY	0 0 0 1 0 0 1 W mod reg mem	11/15	2-4	x	x	x	x	x
	reg, imm	reg ← reg + imm + CY	1 0 0 0 0 0 S W 1 1 0 1 0 reg	4	3-4	x	x	x	x	x
	mem, imm	(mem) ← (mem) + imm + CY	1 0 0 0 0 0 S W mod 0 1 0 mem	18/26	3-6	x	x	x	x	x

Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags						
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z	
Addition/Subtraction Instructions (cont)																											
ADDC	acc, imm	When W = 0 AL ← AL - imm - CY When W = 1 AW ← AW - imm - CY	0	0	0	1	0	1	0	1	0	W	4	2-3	x	x	x	x	x	x							
SUB	reg, reg	reg ← reg - reg	0	0	1	0	1	0	1	0	1	W	2	2	x	x	x	x	x	x							
	mem, reg	(mem) ← (mem) - reg	0	0	1	0	1	0	0	W	16/24	2-4	x	x	x	x	x	x									
	reg, mem	reg ← reg - (mem)	0	0	1	0	1	0	1	W	11/15	2-4	x	x	x	x	x	x									
	reg, imm	reg ← reg - imm	1	0	0	0	0	0	S	W	1	1	1	0	1	4	3-4	x	x	x	x	x	x				
	mem, imm	(mem) ← (mem) - imm	1	0	0	0	0	0	S	W	mod	1	0	1	18/26	3-6	x	x	x	x	x	x					
	acc, imm	When W = 0 AL ← AL - imm When W = 1 AW ← AW - imm	0	0	1	0	1	1	0	W	4	2-3	x	x	x	x	x	x									
SUBC	reg, reg	reg ← reg - reg - CY	0	0	0	1	1	0	1	W	2	2	x	x	x	x	x	x									
	mem, reg	(mem) ← (mem) - reg - CY	0	0	0	1	1	0	0	W	16/24	2-4	x	x	x	x	x	x									
	reg, mem	reg ← reg - (mem) - CY	0	0	0	1	1	0	1	W	11/15	2-4	x	x	x	x	x	x									
	reg, imm	reg ← reg - imm - CY	1	0	0	0	0	0	S	W	1	1	0	1	1	4	3-4	x	x	x	x	x	x				
	mem, imm	(mem) ← (mem) - imm - CY	1	0	0	0	0	0	S	W	mod	0	1	1	18/26	3-6	x	x	x	x	x	x					
	acc, imm	When W = 0 AL ← AL + imm + CY When W = 1 AW ← AW + imm + CY	0	0	0	1	1	1	0	W	4	2-3	x	x	x	x	x	x									
BCD Operation Instructions																											
ADD4S		dst BCD string ← dst BCD string + src BCD string	0	0	0	0	1	1	1	1	0	0	1	0	0	0	0	0	7+19n	2	u	x	u	u	u	x	
SUB4S		dst BCD string ← dst BCD string - src BCD string	0	0	0	0	1	1	1	1	0	0	1	0	0	0	1	0	7+19n	2	u	x	u	u	u	x	
CMP4S		dst BCD string - src BCD string	0	0	0	0	1	1	1	1	0	0	1	0	0	1	1	0	7+19n	2	u	x	u	u	u	x	
n: number of BCD numerals divided by 2																											
ROL4	reg8		0	0	0	0	1	1	1	1	0	0	1	0	0	1	0	0	0	25	3						
	mem8		0	0	0	0	1	1	1	1	0	0	1	0	1	0	0	0	0	28	3-5						
ROR4	reg8		0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0	29	3							
	mem8		0	0	0	0	1	1	1	1	0	0	1	0	1	0	1	0	33	3-5							

Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags				
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S
Increment/Decrement Instructions (cont)																									
INC	reg8	reg8 ← reg8 + 1	1	1	1	1	1	1	1	1	0	1	1	0	0	0	reg	2	2	x	x	x	x	x	
	mem	(mem) ← (mem) + 1	1	1	1	1	1	1	1	1	W	mod	0	0	0	0	mem	16/24	2-4	x	x	x	x	x	
	reg16	reg16 ← reg16 + 1	0	1	0	0	0	0	reg	2	1	x	x	x	x	x									
DEC	reg8	reg8 ← reg8 - 1	1	1	1	1	1	1	1	0	1	1	0	0	1	reg	2	2	x	x	x	x	x		
	mem	(mem) ← (mem) - 1	1	1	1	1	1	1	1	1	W	mod	0	0	1	mem	16/24	2-4	x	x	x	x	x		
	reg16	reg16 ← reg16 - 1	0	1	0	0	1	reg	2	1	x	x	x	x	x										
Multiplication Instructions																									
MULU	reg8	AW ← AL x reg8 AH = 0: CY ← 0, V ← 0 AH ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	1	1	1	0	0	reg	21-22	2	u	x	x	u	u	u	
	mem8	AW ← AL x (mem8) AH = 0: CY ← 0, V ← 0 AH ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	mod	1	0	0	mem	27-28	2-4	u	x	x	u	u	u		
	reg16	DW, AW ← AW x reg16 DW = 0: CY ← 0, V ← 0 DW ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	1	1	1	0	0	reg	29-30	2	u	x	x	u	u	u	
	mem16	DW, AW ← AW x (mem16) DW = 0: CY ← 0, V ← 0 DW ≠ 0: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	mod	1	0	0	mem	35-36 /39-40	2-4	u	x	x	u	u	u		
MUL	reg8	AW ← AL x reg8 AH = AL sign expansion: CY ← 0, V ← 0 AH ≠ AL sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	1	1	1	0	1	reg	33-39	2	u	x	x	u	u	u	
	mem8	AW ← AL x (mem8) AH = AL sign expansion: CY ← 0, V ← 0 AH ≠ AL sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	0	mod	1	0	1	mem	39-45	2-4	u	x	x	u	u	u		
	reg16	DW, AW ← AW x reg16 DW = AW sign expansion: CY ← 0, V ← 0 DW ≠ AW sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	1	1	1	0	1	reg	41-47	2	u	x	x	u	u	u	
	mem16	DW, AW ← AW x (mem16) DW = AW sign expansion: CY ← 0, V ← 0 DW ≠ AW sign expansion: CY ← 1, V ← 1	1	1	1	1	0	1	1	1	mod	1	0	1	mem	47-53 /51-57	2-4	u	x	x	u	u	u		
	reg16, (reg16,) imm8	reg16 ← reg16 x imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	1	1	1	1	1	reg	reg	28-34	3	u	x	x	u	u	u		
	reg16, mem16, imm8	reg16 ← (mem16) x imm8 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0	1	1	0	1	0	1	1	mod	reg	mem	34-40 /38-44	3-5	u	x	x	u	u	u				

Mnemonic	Operand	Operation	Operation Code	No. of Clocks	No. of Bytes	Flags					
			7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Multiplication Instructions (cont)											
MUL	reg16, (reg16.) imm16	reg16 ← reg16 x imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0 1 1 0 1 0 0 1 1 1 reg reg	36-42	4	u	x	x	u	u	u
	reg16, mem16, imm16	reg16 ← (mem16) x imm16 Product ≤ 16 bits: CY ← 0, V ← 0 Product > 16 bits: CY ← 1, V ← 1	0 1 1 0 1 0 0 1 mod reg mem	42-48 /46-52	4-6	u	x	x	u	u	u
Unsigned Division Instructions											
DIVU	reg8	temp ← AW When temp ÷ reg8 > FFH (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg8, AL ← temp ÷ reg8	1 1 1 1 0 1 1 0 1 1 1 1 0 reg	19	2	u	u	u	u	u	u
	mem8	temp ← AW When temp ÷ (mem8) > FFH (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem8), AL ← temp ÷ (mem8)	1 1 1 1 0 1 1 0 mod 1 1 0 mem	25	2-4	u	u	u	u	u	u
	reg16	temp ← AW When temp ÷ reg16 > FFFFH (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg16, AL ← temp ÷ reg16	1 1 1 1 0 1 1 1 1 1 1 1 1 0 reg	25	2	u	u	u	u	u	u
	mem16	temp ← AW When temp ÷ (mem16) > FFFFH (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem16), AL ← temp ÷ (mem16)	1 1 1 1 0 1 1 1 mod 1 1 0 mem	31/35	2-4	u	u	u	u	u	u
Signed Division Instructions											
DIV	reg8	temp ← AW When temp ÷ reg8 > 0 and temp ÷ reg8 > 7FH or temp ÷ reg8 < 0 and temp ÷ reg8 ≤ 0 - 7FH - 1 (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg8, AL ← temp ÷ reg8	1 1 1 1 0 1 1 0 1 1 1 1 1 reg	29-34	2	u	u	u	u	u	u

Mnemonic	Operand	Operation	Operation Code	No. of Clocks	No. of Bytes	Flags					
			7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Signed Division Instructions (cont)											
DIV	mem8	temp ← AW When temp ÷ (mem8) > 0 and temp ÷ (mem8) > 7FH or temp ÷ (mem8) < 0 and temp ÷ (mem8) ≤ 0 - 7FH - 1 (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem8), AL ← temp ÷ (mem8)	1 1 1 1 0 1 1 0 mod 1 1 1 mem	35-40	2-4	u	u	u	u	u	u
	reg16	temp ← AW When temp ÷ reg16 > 0 and temp ÷ reg16 > 7FFFH or temp ÷ reg16 < 0 and temp ÷ reg16 ≤ 0 - 7FFFH - 1 (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % reg16, AL ← temp ÷ reg 16	1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 reg	38-43	2	u	u	u	u	u	u
	mem16	temp ← AW When temp ÷ (mem16) > 0 and temp ÷ (mem16) > 7FFFH or temp ÷ (mem16) < 0 and temp ÷ (mem16) ≤ 0 - 7FFFH - 1 (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (3, 2), PC ← (1, 0) All other times AH ← temp % (mem16), AL ← temp ÷ (mem16)	1 1 1 1 0 1 1 1 mod 1 1 1 mem	44-49 /48-53	2-4	u	u	u	u	u	u
BCD Complement Instructions											
ADJBA		When (AL AND 0FH) > 9 or AC = 1, AL ← AL + 6, AH ← AH + 1, AC ← 1, CY ← AC, AL ← AL AND 0FH	0 0 1 1 0 1 1 1	3	1	x	x	u	u	u	u
ADJ4A		When (AL AND 0FH) > 9 or AC = 1, AL ← AL + 6, CY ← CY OR AC, AC ← 1, When AL > 9FH, or CY = 1 AL ← AL + 60H, CY ← 1	0 0 1 0 0 1 1 1	3	1	x	x	u	x	x	x
ADJBS		When (AL AND 0FH) > 9 or AC = 1, AL ← AL - 6, AH ← AH - 1, AC ← 1, CY ← AC, AL ← AL AND 0FH	0 0 1 1 1 1 1 1	7	1	x	x	u	u	u	u
ADJ4S		When (AL AND 0FH) > 9 or AC = 1, AL ← AL - 6, CY ← CY OR AC, AC ← 1 When AL > 9FH or CY = 1 AL ← AL - 60H, CY ← 1	0 0 1 0 1 1 1 1	7	1	x	x	u	x	x	x

Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags						
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z	
Data Conversion Instructions																											
CVTBD		AH ← AL ÷ 0AH, AL ← AL % 0AH	1	1	0	1	0	1	0	0	0	0	0	0	0	1	0	1	0	15	2	u	u	u	x	x	x
CVTDB		AH ← 0, AL ← AH x 0AH + AL	1	1	0	1	0	1	0	1	0	0	0	0	1	0	1	0	7	2	u	u	u	x	x	x	
CVTBW		When AL < 80H, AH ← 0, all other times AH ← FFH	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	2	1							
CVTWL		When AL < 8000H, DW ← 0, all other times DW ← FFFFH	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	4-5	1							
Comparison Instructions																											
CMP	reg, reg	reg - reg	0	0	1	1	1	0	1	W	1	1	reg	reg	2	2	x	x	x	x	x	x					
	mem, reg	(mem) - reg	0	0	1	1	1	0	0	W	mod	reg	mem	11/15	2-4	x	x	x	x	x	x						
	reg, mem	reg - (mem)	0	0	1	1	1	0	1	W	mod	reg	mem	11/15	2-4	x	x	x	x	x	x						
	reg, imm	reg - imm	1	0	0	0	0	0	S	W	1	1	1	1	reg	4	3-4	x	x	x	x	x	x				
	mem, imm	(mem) - imm	1	0	0	0	0	0	S	W	mod	1	1	1	mem	13/17	3-6	x	x	x	x	x	x				
acc, imm	When W = 0, AL - imm When W = 1, AW - imm	0	0	1	1	1	1	0	W	4	2-3	x	x	x	x	x	x										
Complement Instructions																											
NOT	reg	reg ← $\overline{\text{reg}}$	1	1	1	1	0	1	1	W	1	1	0	1	0	reg	2	2									
	mem	(mem) ← $\overline{(\text{mem})}$	1	1	1	1	0	1	1	W	mod	0	1	0	mem	16/24	2-4										
NEG	reg	reg ← $\overline{\text{reg}} + 1$	1	1	1	1	0	1	1	W	1	1	0	1	1	reg	2	2	x	x	x	x	x	x			
	mem	(mem) ← $\overline{(\text{mem})} + 1$	1	1	1	1	0	1	1	W	mod	0	1	1	mem	16/24	2-4	x	x	x	x	x	x				
Logical Operation Instructions																											
TEST	reg, reg	reg AND reg	1	0	0	0	0	1	0	W	1	1	reg	reg	2	2	u	0	0	x	x	x					
	mem, reg or reg, mem	(mem) AND reg	1	0	0	0	0	1	0	W	mod	reg	mem	10/14	2-4	u	0	0	x	x	x						
	reg, imm	reg AND imm	1	1	1	1	0	1	1	W	1	1	0	0	0	reg	4	3-4	u	0	0	x	x	x			
	mem, imm	(mem) AND imm	1	1	1	1	0	1	1	W	mod	0	0	0	mem	11/15	3-6	u	0	0	x	x	x				
	acc, imm	When W = 0, AL AND imm8 When W = 1, AW AND imm8	1	0	1	0	1	0	0	W	4	2-3	u	0	0	x	x	x									
AND	reg, reg	reg ← reg AND reg	0	0	1	0	0	0	1	W	1	1	reg	reg	2	2	u	0	0	x	x	x					
	mem, reg	(mem) ← (mem) AND reg	0	0	1	0	0	0	0	W	mod	reg	mem	16/24	2-4	u	0	0	x	x	x						
	reg, mem	reg ← reg AND (mem)	0	0	1	0	0	0	1	W	mod	reg	mem	11/15	2-4	u	0	0	x	x	x						
	reg, imm	reg ← reg AND imm	1	0	0	0	0	0	0	W	1	1	1	0	0	reg	4	3-4	u	0	0	x	x	x			
	mem, imm	(mem) ← (mem) AND imm	1	0	0	0	0	0	0	W	mod	1	1	0	mem	18/26	3-6	u	0	0	x	x	x				
	acc, imm	When W = 0, AL ← AL AND imm8 When W = 1, AW ← AW AND imm16	0	0	1	0	0	1	0	W	4	2-3	u	0	0	x	x	x									

Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags										
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z					
Logical Operation Instructions (cont)																															
OR	reg, reg	reg ← reg OR reg	0	0	0	0	1	0	1	W	1	1	reg	reg	2	2	u	0	0	x	x	x									
	mem, reg	(mem) ← (mem) OR reg	0	0	0	0	1	0	0	W	mod	reg	mem	16/24	2-4	u	0	0	x	x	x										
	reg, mem	reg ← reg OR (mem)	0	0	0	0	1	0	1	W	mod	reg	mem	11/15	2-4	u	0	0	x	x	x										
	reg, imm	reg ← reg OR imm	1	0	0	0	0	0	0	W	1	1	0	1	reg	4	3-4	u	0	0	x	x	x								
	mem, imm	(mem) ← (mem) OR imm	1	0	0	0	0	0	0	W	mod	0	0	1	mem	18/26	3-6	u	0	0	x	x	x								
	acc, imm	When W = 0, AL ← AL OR imm8 When W = 1, AW ← AW OR imm16	0	0	0	0	1	1	0	W	4	2-3	u	0	0	x	x	x													
XOR	reg, reg	reg ← reg XOR reg	0	0	1	1	0	0	1	W	1	1	reg	reg	2	2	u	0	0	x	x	x									
	mem, reg	(mem) ← (mem) XOR reg	0	0	1	1	0	0	0	W	mod	reg	mem	16/24	2-4	u	0	0	x	x	x										
	reg, mem	reg ← reg XOR (mem)	0	0	1	1	0	0	1	W	mod	reg	mem	11/15	2-4	u	0	0	x	x	x										
	reg, imm	reg ← reg XOR imm	1	0	0	0	0	0	0	W	1	1	1	1	0	reg	4	3-4	u	0	0	x	x	x							
	mem, imm	(mem) ← (mem) XOR imm	1	0	0	0	0	0	0	W	mod	1	1	0	mem	18/26	3-6	u	0	0	x	x	x								
	acc, imm	When W = 0, AL ← AL XOR imm8 When W = 1, AW ← AW XOR imm16	0	0	1	1	0	1	0	W	4	2-3	u	0	0	x	x	x													
Bit Operation Instructions																															
TEST1	reg8, CL	reg8 bit no. CL = 0: Z ← 1 reg8 bit no. CL = 1: Z ← 0	2nd byte*				3rd byte*																								
	mem8, CL	(mem8) bit no. CL = 0: Z ← 1 (mem8) bit no. CL = 1: Z ← 0	0	0	0	1	0	0	0	0	1	1	0	0	0	reg	3	3	u	0	0	u	u	x							
	reg16, CL	reg16 bit no. CL = 0: Z ← 1 reg16 bit no. CL = 1: Z ← 0	0	0	0	1	0	0	0	1	1	1	0	0	0	reg	3	3	u	0	0	u	u	x							
	mem16, CL	(mem16) bit no. CL = 0: Z ← 1 (mem16) bit no. CL = 1: Z ← 0	0	0	0	1	0	0	0	1	mod	0	0	0	0	mem	12/16	3-5	u	0	0	u	u	x							
	reg8, imm3	reg8 bit no. imm3 = 0: Z ← 1 reg8 bit no. imm3 = 1: Z ← 0	0	0	0	1	1	0	0	0	1	1	0	0	0	reg	4	4	u	0	0	u	u	x							
	mem8, imm3	(mem8) bit no. imm3 = 0: Z ← 1 (mem8) bit no. imm3 = 1: Z ← 0	0	0	0	1	1	0	0	0	mod	0	0	0	0	mem	13	4-6	u	0	0	u	u	x							
	reg16, imm4	reg16 bit no. imm4 = 0: Z ← 1 reg16 bit no. imm4 = 1: Z ← 0	0	0	0	1	1	0	0	1	1	1	0	0	0	reg	4	4	u	0	0	u	u	x							
mem16, imm4	(mem16) bit no. imm4 = 0: Z ← 1 (mem16) bit no. imm4 = 1: Z ← 0	0	0	0	1	1	0	0	1	mod	0	0	0	0	mem	13/17	4-6	u	0	0	u	u	x								
			2nd byte*				3rd byte*																								
			*Note: First byte = 0FH																												

Mnemonic	Operand	Operation	Operation Code								No. of Clocks	No. of Bytes	Flags											
			7	6	5	4	3	2	1	0			7	6	5	4	3	2	1	0	AC	CY	V	P
Bit Operation Instructions (cont)																								
NOT1	reg8, CL	reg8 bit no. CL ← reg8 bit no. CL	2nd byte*				3rd byte*																	
	mem8, CL	(mem8) bit no. CL ← (mem8) bit no. CL	0	0	0	1	0	1	1	0	1	1	0	0	0	reg	4	3						
	reg16, CL	reg16 bit no. CL ← reg16 bit no. CL	0	0	0	1	0	1	1	1	1	1	0	0	0	reg	4	3						
	mem16, CL	(mem16) bit no. CL ← (mem16) bit no. CL	0	0	0	1	0	1	1	1	1	mod	0	0	0	mem	18/26	3-5						
	reg8, imm3	reg8 bit no. imm3 ← reg8 bit no. imm3	0	0	0	1	1	1	1	0	1	1	0	0	0	reg	5	4						
	mem8, imm3	(mem8) bit no. imm3 ← (mem8) bit no. imm3	0	0	0	1	1	1	1	0	mod	0	0	0	mem	19	4-6							
	reg16, imm4	reg16 bit no. imm4 ← (reg16) bit no. imm4	0	0	0	1	1	1	1	1	1	1	0	0	0	reg	5	4						
	mem16, imm4	(mem16) bit no. imm4 ← (mem16) bit no. imm4	0	0	0	1	1	1	1	1	mod	0	0	0	mem	19/27	4-6							
			2nd byte*				3rd byte*																	
			*Note: First byte = 0FH																					
	CY	CY ← $\bar{C}Y$	1	1	1	1	0	1	0	1						2	1						x	
CLR1	reg8, CL	reg8 bit no. CL ← 0	2nd byte*				3rd byte*																	
	mem8, CL	(mem8) bit no. CL ← 0	0	0	0	1	0	0	1	0	1	1	0	0	0	reg	5	3						
	reg16, CL	reg16 bit no. CL ← 0	0	0	0	1	0	0	1	1	1	1	0	0	0	reg	5	3						
	mem16, CL	(mem16) bit no. CL ← 0	0	0	0	1	0	0	1	1	mod	0	0	0	mem	14/22	3-5							
	reg8, imm3	reg8 bit no. imm3 ← 0	0	0	0	1	1	0	1	0	1	1	0	0	0	reg	6	4						
	mem8, imm3	(mem8) bit no. imm3 ← 0	0	0	0	1	1	0	1	0	mod	0	0	0	mem	15	4-6							
	reg16, imm4	reg16 bit no. imm4 ← 0	0	0	0	1	1	0	1	1	1	1	0	0	0	reg	6	4						
	mem16, imm4	(mem16) bit no. imm4 ← 0	0	0	0	1	1	0	1	1	mod	0	0	0	mem	15/27	4-6							
			2nd byte*				3rd byte*																	
			*Note: First byte = 0FH																					
	CY	CY ← 0	1	1	1	1	1	0	0	0						2	1						0	
	DIR	DIR ← 0	1	1	1	1	1	1	0	0						2	1							

Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags								
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z			
Bit Operation Instructions (cont)																													
SET1	reg8, CL	reg8 bit no. CL ← 1	2nd byte*				3rd byte*				4	3																	
			0	0	0	1	0	1	0	0			1	1	0	0	0	reg											
	mem8, CL	(mem8) bit no. CL ← 1	0	0	0	1	0	1	0	0	mod	0	0	0	0	mem	13	3-5											
	reg16, CL	reg16 bit no. CL ← 1	0	0	0	1	0	1	0	1	1	1	0	0	0	reg	4	3											
	mem16, CL	(mem16) bit no. CL ← 1	0	0	0	1	0	1	0	1	mod	0	0	0	0	mem	13/21	3-5											
	reg8, imm3	reg8 bit no. imm3 ← 1	0	0	0	1	1	1	0	0	1	1	0	0	0	reg	5	4											
	mem8, imm3	(mem8) bit no. imm3 ← 1	0	0	0	1	1	1	0	0	mod	0	0	0	0	mem	14	4-6											
	reg16, imm4	reg16 bit no. imm4 ← 1	0	0	0	1	1	1	0	1	1	1	0	0	0	reg	5	4											
	mem16, imm4	(mem16) bit no. imm4 ← 1	2nd byte*				3rd byte*				14/22	4-6																	
		0	0	0	1	1	1	0	1	mod			0	0	0	0	mem												
			2nd byte*				3rd byte*																						
			*Note: First byte = 0FH																										
CY		CY ← 1	1	1	1	1	1	0	0	1												2	1						
DIR		DIR ← 1	1	1	1	1	1	1	0	1												2	1						
Shift Instructions																													
SHL	reg, 1	CY ← MSB of reg, reg ← reg x 2 When MSB of reg ≠ CY, V ← 1 When MSB of reg = CY, V ← 0	0	1	0	0	0	W	1	1	1	0	0	reg	2	2	u	x	x	x	x	x							
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 When MSB of (mem) ≠ CY, V ← 1 When MSB of (mem) = CY, V ← 0	1	1	0	1	0	0	0	W	mod	1	0	0	mem	16/24	2-4	u	x	x	x	x	x						
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← MSB of reg, reg ← reg x 2, temp ← temp - 1	1	1	0	1	0	0	1	W	1	1	1	0	0	reg	7+n	2	u	x	u	x	x	x					
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← MSB of (mem), (mem) ← (mem) x 2, temp ← temp - 1	1	1	0	1	0	0	1	W	mod	1	0	0	mem	19/27 + n	2-4	u	x	u	x	x	x						
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← MSB of reg, reg ← reg x 2, temp ← temp - 1	1	1	0	0	0	0	0	W	1	1	1	0	0	reg	7+n	3	u	x	u	x	x	x					
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← MSB of (mem), (mem) ← (mem) x 2, temp ← temp - 1	1	1	0	0	0	0	0	W	mod	1	0	0	mem	19/27 + n	3-5	u	x	u	x	x	x						
			n: number of shifts																										
SHR	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2 When MSB of reg ≠ bit following MSB of reg: V ← 1 When MSB of reg = bit following MSB of reg: V ← 0	1	1	0	1	0	0	0	W	1	1	1	0	1	reg	2	2	u	x	x	x	x	x					

Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags					
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z
Shift Instructions (cont)																										
SHR	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2 When MSB of (mem) ≠ bit following MSB of (mem): V ← 1 When MSB of (mem) = bit following MSB of (mem): V ← 0	1	1	0	1	0	0	0	0	W	mod	1	0	1	mem	16/24	2-4	u	x	x	x	x	x		
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp - 1	1	1	0	1	0	0	0	0	W	1	1	1	0	1	reg	7+n	2	u	x	u	x	x	x	
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp - 1	1	1	0	1	0	0	1	W	mod	1	0	1	mem	19/27 + n	2-4	u	x	u	x	x	x			
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp - 1	1	1	0	0	0	0	0	0	W	1	1	1	0	1	reg	7+n	3	u	x	u	x	x	x	
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp - 1 n: number of shifts	1	1	0	0	0	0	0	0	W	mod	1	0	1	mem	19/27 + n	3-5	u	x	u	x	x	x		
SHRA	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2, V ← 0 MSB of operand does not change	1	1	0	1	0	0	0	W	1	1	1	1	1	reg	2	2	u	x	0	x	x	x		
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2, V ← 0, MSB of operand does not change	1	1	0	1	0	0	0	W	mod	1	1	1	mem	16/24	2-4	u	x	0	x	x	x			
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp - 1 MSB of operand does not change	1	1	0	1	0	0	1	W	1	1	1	1	1	reg	7+n	2	u	x	u	x	x	x		
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp - 1 MSB of operand does not change	1	1	0	1	0	0	1	W	mod	1	1	1	mem	19/27 + n	2-4	u	x	u	x	x	x			
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← LSB of reg, reg ← reg ÷ 2, temp ← temp - 1 MSB of operand does not change	1	1	0	0	0	0	0	0	W	1	1	1	1	1	reg	7+n	3	u	x	u	x	x	x	
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← LSB of (mem), (mem) ← (mem) ÷ 2, temp ← temp - 1 MSB of operand does not change n: number of shifts	1	1	0	0	0	0	0	0	W	mod	1	1	1	mem	19/27 + n	3-5	u	x	u	x	x	x		

Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags				
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S
Rotation Instructions																									
ROL	reg, 1	CY ← MSB of reg, reg ← reg x 2 + CY MSB of reg ≠ CY: V ← 1 MSB of reg = CY: V ← 0	1	1	0	1	0	0	0	W	1	1	0	0	0	reg	2	2		x	x				
	mem, 1	CY ← MSB of (mem), (mem) ← (mem) x 2 + CY MSB of (mem) ≠ CY: V ← 1 MSB of (mem) = CY: V ← 0	1	1	0	1	0	0	0	W	mod	0	0	0	0	mem	16/24	2-4		x	x				
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← MSB of reg, reg ← reg x 2 + CY temp ← temp - 1	1	1	0	1	0	0	1	W	1	1	0	0	0	reg	7 + n	2		x	u				
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← MSB of (mem), (mem) ← (mem) x 2 + CY temp ← temp - 1	1	1	0	1	0	0	1	W	mod	0	0	0	0	reg	19/27 + n	2-4		x	u				
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← MSB of reg, reg ← reg x 2 + CY temp ← temp - 1	1	1	0	0	0	0	0	W	1	1	0	0	0	reg	7 + n	3		x	u				
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← MSB of (mem), (mem) ← (mem) x 2 + CY temp ← temp - 1 n: number of shifts	1	1	0	0	0	0	0	W	mod	0	0	0	0	mem	19/27 + n	3-5		x	u				
ROR	reg, 1	CY ← LSB of reg, reg ← reg ÷ 2 MSB of reg ← CY MSB of reg ≠ bit following MSB of reg: V ← 1 MSB of reg = bit following MSB of reg: V ← 0	1	1	0	1	0	0	0	W	1	1	0	0	1	reg	2	2		x	x				
	mem, 1	CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← CY MSB of (mem) ≠ bit following MSB of (mem): V ← 1 MSB of (mem) = bit following MSB of (mem): V ← 0	1	1	0	1	0	0	0	W	mod	0	0	0	1	mem	16/24	2-4		x	x				
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY temp ← temp - 1	1	1	0	1	0	0	1	W	1	1	0	0	1	reg	7 + n	2		x	u				
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation: CY ← LSB of (mem), (mem) ← (mem) ÷ 2, MSB of (mem) ← CY temp ← temp - 1 n: number of shifts	1	1	0	1	0	0	1	W	mod	0	0	0	1	mem	19/27 + n	2-4		x	u				

Mnemonic	Operand	Operation	Operation Code	No. of Clocks	No. of Bytes	Flags				
			7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0			AC	CY	V	P	S
Rotation Instructions (cont)										
ROR	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← CY temp ← temp - 1	1 1 0 0 0 0 0 0 W 1 1 0 0 1	reg	7 + n	3		x	u	
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: CY ← LSB of (mem), (mem) ← (mem) ÷ 2 temp ← temp - 1	1 1 0 0 0 0 0 0 W mod 0 0 1	mem	19/27 + n	3-5		x	u	
Rotate Instruction										
ROLC	reg, 1	tmpcy ← CY, CY ← MSB of reg reg ← reg x 2 + tmpcy MSB of reg = CY: V ← 0 MSB of reg ≠ CY: V ← 1	1 1 0 1 0 0 0 0 W 1 1 0 1 0	reg	2	2		x	x	
	mem, 1	tmpcy ← CY, CY ← MSB of (mem) (mem) ← (mem) x 2 + tmpcy MSB of (mem) = CY: V ← 0 MSB of (mem) ≠ CY: V ← 1	1 1 0 1 0 0 0 0 W mod 0 1 0	mem	16/24	2-4		x	x	
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation: tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy temp ← temp - 1	1 1 0 1 0 0 1 W 1 1 0 1 0	reg	7 + n	2		x	u	
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation: tmpcy ← CY, CY ← MSB of (mem), (mem) ← (mem) x 2 + tmpcy temp ← temp - 1	1 1 0 1 0 0 1 W mod 0 1 0	mem	19/27 + n	2-4		x	u	
	reg, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: tmpcy ← CY, CY ← MSB of reg, reg ← reg x 2 + tmpcy temp ← temp - 1	1 1 0 0 0 0 0 0 W 1 1 0 1 0	reg	7 + n	3		x	u	
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: tmpcy ← CY, CY ← MSB of (mem) (mem) ← (mem) x 2 + tmpcy temp ← temp - 1	1 1 0 0 0 0 0 0 W mod 0 1 0	mem	19/27 + n	3-5		x	u	
n: number of shifts										

Mnemonic	Operand	Operation	Operation Code	No. of Clocks	No. of Bytes	Flags				
			7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0			AC	CY	V	P	S
Rotate Instructions (cont)										
RORC	reg, 1	tmpcy ← CY, CY ← LSB of reg reg ← reg ÷ 2, MSB of reg ← tmpcy MSB of reg ≠ bit following MSB of reg: V ← 1 MSB of reg = bit following MSB of reg: V ← 0	1 1 0 1 0 0 0 W 1 1 0 1 1 reg	2	2		x	x		
	mem, 1	tmpcy ← CY, CY ← LSB of (mem) (mem) ← (mem) ÷ 2, MSB of (mem) ← tmpcy MSB of (mem) ≠ bit following MSB of (mem): V ← 1 MSB of (mem) = bit following MSB of (mem): V ← 0	1 1 0 1 0 0 0 W mod 0 1 1 mem	16/24	2-4		x	x		
	reg, CL	temp ← CL, while temp ≠ 0, repeat this operation: tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2, MSB of reg ← tmpcy, temp ← temp - 1	1 1 0 1 0 0 1 W 1 1 0 1 1 reg	7 + n	2		x	u		
	mem, CL	temp ← CL, while temp ≠ 0, repeat this operation: tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← tmpcy, temp ← temp - 1	1 1 0 1 0 0 1 W mod 0 1 1 mem	19/27 + n	2-4		x	u		
	reg, imm8	temp ← imm8, while temp ≠ 0 repeat this operation: tmpcy ← CY, CY ← LSB of reg, reg ← reg ÷ 2 MSB of reg ← tmpcy, temp ← temp - 1	1 1 0 0 0 0 0 W 1 1 0 1 1 reg	7 + n	3		x	u		
	mem, imm8	temp ← imm8, while temp ≠ 0, repeat this operation: tmpcy ← CY, CY ← LSB of (mem), (mem) ← (mem) ÷ 2 MSB of (mem) ← tmpcy, temp ← temp - 1	1 1 0 0 0 0 0 W mod 0 1 1 mem	19/27 + n	3-5		x	u		
n: number of shifts										
Subroutine Control Instructions										
CALL	near-proc	(SP - 1, SP - 2) ← PC, SP ← SP - 2 PC ← PC + disp	1 1 1 0 1 0 0 0	16/20	3					
	regptr16	(SP - 1, SP - 2) ← PC, SP ← SP - 2 PC ← regptr16	1 1 1 1 1 1 1 1 1 1 0 1 0 reg	14/18	2					
	memptr16	(SP - 1, SP - 2) ← PC, SP ← SP - 2 PC ← (memptr16)	1 1 1 1 1 1 1 1 1 mod 0 1 0 mem	23/31	2-4					
	far-proc	(SP - 1, SP - 2) ← PS, (SP - 3, SP - 4) ← PC SP ← SP - 4, PS ← seg, PC ← offset	1 0 0 1 1 0 1 0	21/29	5					
	memptr32	(SP - 1, SP - 2) ← PS, (SP - 3, SP - 4) ← PC SP ← SP - 4, PS ← (memptr32 + 2), PC ← (memptr32)	1 1 1 1 1 1 1 1 1 mod 0 1 1 mem	31/47	2-4					

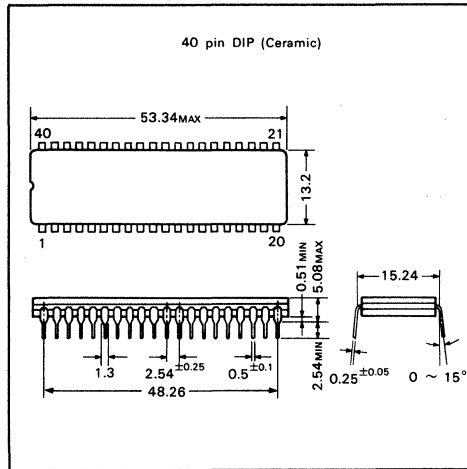
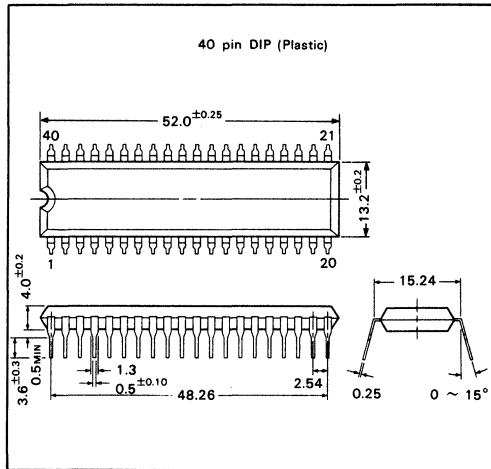
Mnemonic	Operand	Operation	Operation Code																No. of Clocks	No. of Bytes	Flags								
			7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0			AC	CY	V	P	S	Z			
Subroutine Control Instructions (cont)																													
RET		$PC \leftarrow (SP + 1, SP), SP \leftarrow SP + 2$	1	1	0	0	0	0	1	1													15/19	1					
	pop-value	$PC \leftarrow (SP + 1, SP)$ $SP \leftarrow SP + 2, SP \leftarrow SP + \text{pop-value}$	1	1	0	0	0	0	1	0													20/24	3					
		$PC \leftarrow (SP + 1, SP), PS \leftarrow (SP + 3, SP + 2)$ $SP \leftarrow SP + 4$	1	1	0	0	1	0	1	1													21/29	1					
	pop-value	$PC \leftarrow (SP + 1, SP), PS \leftarrow (SP + 3, SP + 2)$ $SP \leftarrow SP + 4, SP \leftarrow SP + \text{pop-value}$	1	1	0	0	1	0	1	0													24/32	3					
Stack Manipulation Instructions																													
PUSH	mem16	$(SP - 1, SP - 2) \leftarrow (\text{mem}16), SP \leftarrow SP - 2$	1	1	1	1	1	1	1	1	mod	1	1	0			mem						18/26	2-4					
	reg16	$(SP - 1, SP - 2) \leftarrow \text{reg}16, SP \leftarrow SP - 2$	0	1	0	1	0				reg												8/12	1					
	sreg	$(SP - 1, SP - 2) \leftarrow \text{sreg}, SP \leftarrow SP - 2$	0	0	0						sreg	1	1	0									8/12	1					
	PSW	$(SP - 1, SP - 2) \leftarrow \text{PSW}, SP \leftarrow SP - 2$	1	0	0	1	1	1	0	0													8/12	1					
	R	Push registers on the stack	0	1	1	0	0	0	0	0													35/67	1					
	imm	$(SP - 1, SP - 2) \leftarrow \text{imm}, SP \leftarrow SP - 2,$ When $S = 1$, sign extension	0	1	1	0	1	0	S	0													7/11 or 8/12	2-3					
POP	mem16	$(\text{mem}16) \leftarrow (SP + 1, SP), SP \leftarrow SP + 2$	1	0	0	0	1	1	1	1	mod	0	0	0			mem						17/25	2-4					
	reg16	$\text{reg}16 \leftarrow (SP + 1, SP), SP \leftarrow SP + 2$	0	1	0	1	1			reg													8/12	1					
	sreg	$\text{sreg} \leftarrow (SP + 1, SP)$ $\text{sreg} : \text{SS}, \text{DS}0, \text{DS}1$ $SP \leftarrow SP + 2$	0	0	0						sreg	1	1	1									8/12	1					
	PSW	$\text{PSW} \leftarrow (SP + 1, SP), SP \leftarrow SP + 2$	1	0	0	1	1	1	0	1													8/12	1	R	R	R	R	R
	R	Pop registers from the stack	0	1	1	0	0	0	0	1													43/75	1					
PREPARE	imm16, imm8	Prepare new stack frame	1	1	0	0	1	0	0	0												*	4						
			*: imm8 = 0: 12/16 imm8 ≥ 1: 22 + 20 × (imm8 - 1): Odd Address 18 + 12 × (imm8 - 1): Even Address																										
DISPOSE		Dispose of stack frame	1	1	0	0	1	0	0	1													6/10	1					
Branch Instruction																													
BR	near-label	$PC \leftarrow PC + \text{disp}$	1	1	1	0	1	0	0	1													12	3					
	short-label	$PC \leftarrow PC + \text{ext-disp}8$	1	1	1	0	1	0	1	1													12	2					
	regptr16	$PC \leftarrow \text{regptr}16$	1	1	1	1	1	1	1	1	1	1	1	0	0		reg						11	2					
	memptr16	$PC \leftarrow (\text{memptr}16)$	1	1	1	1	1	1	1	1	mod	1	0	0			mem						20/24	2-4					
	far-label	$PS \leftarrow \text{seg}, PC \leftarrow \text{offset}$	1	1	1	0	1	0	1	0													15	5					
	memptr32	$PS \leftarrow (\text{memptr}32 + 2), PC \leftarrow (\text{memptr}32)$	1	1	1	1	1	1	1	1	mod	1	0	1			mem						27/35	2-4					

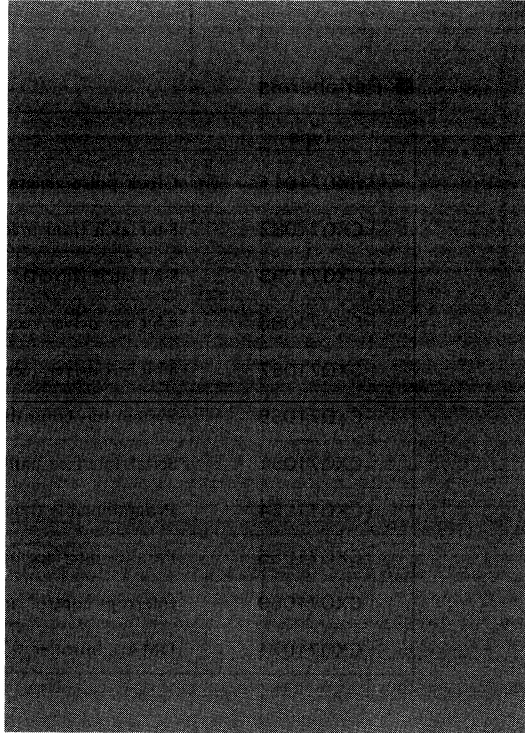
Mnemonic	Operand	Operation	Operation Code	No. of Clocks	No. of Bytes	Flags					
			7 6 5 4 3 2 1 0 7 6 5 4 3 2 1 0			AC	CY	V	P	S	Z
Conditional Branch Instructions											
BV	short-label	If V = 1, PC ← PC + ext-disp8	0 1 1 1 0 0 0 0	14/4	2						
BNV	short-label	If V = 0, PC ← PC + ext-disp8	0 1 1 1 0 0 0 1	14/4	2						
BC, BL	short-label	If CY = 1, PC ← PC + ext-disp8	0 1 1 1 0 0 1 0	14/4	2						
BNC, BNL	short-label	If CY = 0, PC ← PC + ext-disp8	0 1 1 1 0 0 1 1	14/4	2						
BE, BZ	short-label	If Z = 1, PC ← PC + ext-disp8	0 1 1 1 0 1 0 0	14/4	2						
BNE, BNZ	short-label	If Z = 0, PC ← PC + ext-disp8	0 1 1 1 0 1 0 1	14/4	2						
BNH	short-label	If CY OR Z = 1, PC ← PC + ext-disp8	0 1 1 1 0 1 1 0	14/4	2						
BH	short-label	If CY OR Z = 0, PC ← PC + ext-disp8	0 1 1 1 0 1 1 1	14/4	2						
BN	short-label	If S = 1, PC ← PC + ext-disp8	0 1 1 1 1 0 0 0	14/4	2						
BP	short-label	If S = 0, PC ← PC + ext-disp8	0 1 1 1 1 0 0 1	14/4	2						
BPE	short-label	If P = 1, PC ← PC + ext-disp8	0 1 1 1 1 0 1 0	14/4	2						
BPO	short-label	If P = 0, PC ← PC + ext-disp8	0 1 1 1 1 0 1 1	14/4	2						
BLT	short-label	If S XOR V = 1, PC ← PC + ext-disp8	0 1 1 1 1 1 0 0	14/4	2						
BGE	short-label	If S XOR V = 0, PC ← PC + ext-disp8	0 1 1 1 1 1 0 1	14/4	2						
BLE	short-label	If (S XOR V) OR Z = 1, PC ← PC + ext-disp8	0 1 1 1 1 1 1 0	14/4	2						
BGT	short-label	If (S XOR V) OR Z = 0, PC ← PC + ext-disp8	0 1 1 1 1 1 1 1	14/4	2						
DBNZNE	short-label	CW ← CW - 1 If Z = 0 and CW ≠ 0, PC ← PC + ext-disp8	1 1 1 0 0 0 0 0	14/5	2						
DBNZE	short-label	CW ← CW - 1 If Z = 1 and CW ≠ 0, PC ← PC + ext-disp8	1 1 1 0 0 0 0 1	14/5	2						
DBNZ	short-label	CW ← CW - 1 If CW ≠ 0, PC ← PC + ext-disp8	1 1 1 0 0 0 1 0	13/5	2						
BCWZ	short-label	If CW = 0, PC ← PC + ext-disp8	1 1 1 0 0 0 1 1	13/5	2						
Interrupt Instructions											
BRK	3	(SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS, (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0 PS ← (15, 14), PC ← (13, 12)	1 1 0 0 1 1 0 0	38/50	1						
	imm8 (≠ 3)	(SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS, (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0 PC ← (n x 4 + 1 n x 4) PS ← (n x 4 + 3, n x 4 + 2) n = imm8	1 1 0 0 1 1 0 1	38/50	2						

Mnemonic	Operand	Operation	Operation Code													No. of Clocks	No. of Bytes	Flags							
			7	6	5	4	3	2	1	0	7	6	5	4	3			2	1	0	AC	CY	V	P	S
Interrupt Instructions (cont)																									
BRKV		When V = 1 (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS, (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0 PS ← (19, 18), PC ← (17, 16)	1 1 0 0 1 1 1 0														40/52 3	1							
RETI		PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2), PSW ← (SP + 5, SP + 4), SP ← SP + 6	1 1 0 0 1 1 1 1														27/39	1	R	R	R	R	R	R	
CHKIND	reg16, mem32	When (mem32) > reg16 or (mem32 + 2) < reg16 (SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS, (SP - 5, SP - 6) ← PC, SP ← SP - 6 IE ← 0, BRK ← 0, PS ← (23, 22), PC ← (21, 20)	0 1 1 0 0 0 1 0	mod	reg	mem											53-56 /73-76 18/26	2-4							
BRKEM	imm8	(SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS, (SP - 5, SP - 6) ← PC, SP ← SP - 6 MD ← 0, PC ← (n x 4 + 1, n x 4) PS ← (n x 4 + 3, n x 4 + 2), n = imm8	0 0 0 0 1 1 1 1	1	1	1	1	1	1	1	1	1	1	1	1	1	38/50	3							
CPU Control Instructions																									
HALT		CPU Halt	1 1 1 1 0 1 0 0														2	1							
BUSLOCK		Bus Lock Prefix	1 1 1 1 0 0 0 0														2	1							
FPO1	fp-op	No Operation	1 1 0 1 1 X X X	1	1	Y	Y	Z	Z	Z								2	2						
	fp-op, mem	data bus ← (mem)	1 1 0 1 1 X X X	mod	Y	Y	Y	mem									11/15	2-4							
FPO2	fp-op	No Operation	0 1 1 0 0 1 1 X	1	1	Y	Y	Z	Z	Z								2	2						
	fp-op, mem	data bus ← (mem)	0 1 1 0 0 1 1 X	mod	Y	Y	Y	mem									11/15	2-4							
POLL		Poll and wait n: number of times POLL pin is sampled	1 0 0 1 1 0 1 1														2 + 5n	1							
NOP		No Operation	1 0 0 1 0 0 0 0														3	1							
DI		IE ← 0	1 1 1 1 1 0 1 0														2	1							
EI		IE ← 1	1 1 1 1 1 0 1 1														2	1							
8080 Mode Instructions																									
RETEM		PC ← (SP + 1, SP), PS ← (SP + 3, SP + 2), PSW ← (SP + 5, SP + 4), SP ← SP + 6	1 1 1 0 1 1 0 1	1	1	1	1	1	1	1	1	1	0	1											
CALLN	imm8	(SP - 1, SP - 2) ← PSW, (SP - 3, SP - 4) ← PS, (SP - 5, SP - 6) ← PC, SP ← SP - 6 MD ← 1, PC ← (n x 4 + 1, n x 4) PS ← (n x 4 + 3, n x 4 + 2), n = imm8	1 1 1 0 1 1 0 1	1	1	1	1	0	1	1	1	0	1	1	0	1	38/58	3							

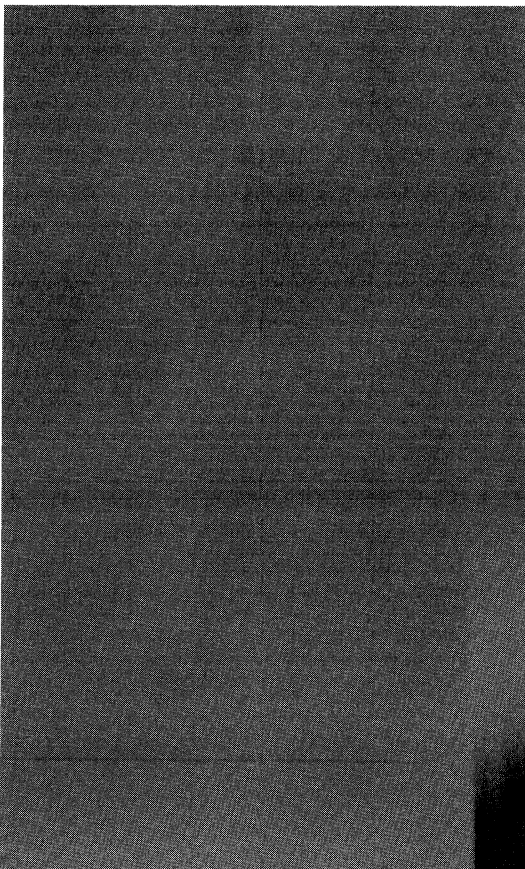
Package Outline

Unit: mm





Peripherals



■ Peripherals

Type	Function	Page
CXQ71011	Clock pulse generator/driver	123
CXQ71082	8-bit latch (non invert)	132
CXQ71083	8-bit latch (invert)	132
CXQ71086	8-bit bus driver/receiver (non invert)	137
CXQ71087	8-bit bus driver/receiver (invert)	137
CXQ71088	System bus controller	142
CXQ71051	Serial Interface unit	150
CXQ71054	Programmable timer counter	182
CXQ71055	Parallel interface unit	207
CXQ71059	Interrupt control unit	230
CXQ71071	DMA controller	264

Clock Pulse Generator/Driver

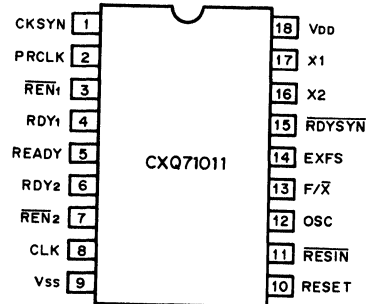
Description

The CXQ71011 is a clock pulse generator/driver for microprocessors and their peripherals with high speed CMOS technology.

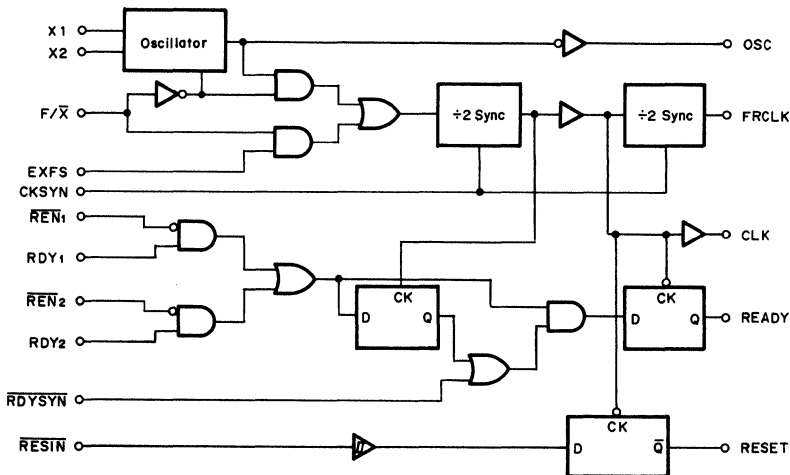
Features

- Clock pulse generator/driver for CXQ70108/70116 CPUs and their peripherals
- Frequency source can be a crystal or an external clock
- Internal frequency source power-down mode available when external clock is used ($F/\bar{X}='H'$)
- Reset signal with Schmitt-trigger circuit for CPU or peripherals
- Bus ready signal with two-bus system synchronization
- Clock synchronization with other CXQ71011s
- CMOS technology
- +5V single power supply
- 18-pin plastic DIP (300 mil)
- NEC μ PD71011 compatible

Pin Configuration (Top View)



Block Diagram



Pin Identification

No.	Symbol	Direction	Function
1	CKSYN	In	Clock synchronization input
2	PRCLK	Out	Peripheral clock output
3	$\overline{\text{REN}}_1$	In	Bus ready enable input 1
4	RDY ₁	In	Bus ready input 1
5	READY	Out	Ready output
6	RDY ₂	In	Bus ready input 2
7	$\overline{\text{REN}}_2$	In	Bus ready enable input 2
8	CLK	Out	Processor clock output
9	V _{SS}		Ground
10	RESET	Out	Reset output
11	$\overline{\text{RESIN}}$	In	Reset input
12	OSC	Out	Oscillator output
13	$\text{F}/\overline{\text{X}}$	In	External frequency source/crystal select
14	EXFS	In	External frequency source input
15	$\overline{\text{RDYSYN}}$	In	Ready synchronization select
16	X ₂	In	Crystal input
17	X ₁	In	Crystal input
18	V _{DD}		Power supply

Pin Functions

X₁, X₂ [Crystal Inputs]

A crystal is connected to these inputs to generate clocks for a CPU and its peripherals. The crystal frequency should be two times the frequency of CLK.

EXFS [External Frequency Source Input]

EXFS is external frequency input in the external frequency source mode ($\text{F}/\overline{\text{X}} = \text{'H'}$). A square TTL-level clock signal of two times the frequency of CLK output should be used for the source.

$\text{F}/\overline{\text{X}}$ [Frequency/Crystal Select]

$\text{F}/\overline{\text{X}}$ selects either the external frequency source or the crystal as the source of the CLK output. When $\text{F}/\overline{\text{X}}$ is low, CLK is generated from the crystal connected to X₁ and X₂. When $\text{F}/\overline{\text{X}}$ is high, CLK is generated from an external TTL-level frequency input on the EXFS pin and at the same time, the internal oscillation circuit will go into the power-down mode.

CLK [Processor Clock]

CLK supplies a 50% duty cycle clock to drive the CPU and its peripherals on the local bus. CLK has a half frequency of crystal or EXFS input. The CLK output is +0.4V higher than the other outputs.

PRCLK [Peripheral Clock]

PRCLK supplies a 50% duty cycle clock at one-half the frequency of CLK to drive peripheral devices.

OSC [Oscillator]

OSC outputs a TTL-level signal at the same frequency as the crystal input.

CKSYN [Clock Synchronization]

CKSYN synchronizes one CXQ71011 to other CXQ71011s. A high level at CKSYN resets the internal counter, and a low level enables it to count. CKSYN needs to be externally synchronized to EXFS. When using the crystal oscillator, CKSYN needs to be stopped to ground.

 $\overline{\text{RESIN}}$ [Reset Input]

This Schmitt trigger input generates the RESET output. An RC connection can be used to provide power-on-reset.

RESET [Reset]

This output is a reset signal for the CPU.

RDY1, RDY2 [Bus Ready]

A peripheral device sends RDY1 or RDY2 to signal that the data on the system bus has been received or is ready to be sent. $\overline{\text{REN1}}$ and $\overline{\text{REN2}}$ control the RDY1 and RDY2 signals.

 $\overline{\text{REN1}}$, $\overline{\text{REN2}}$ [Bus Ready Enable]

$\overline{\text{REN1}}$ and $\overline{\text{REN2}}$ qualify their respective RDY inputs.

 $\overline{\text{RDYSYN}}$ [Ready Synchronization Select]

$\overline{\text{RDYSYN}}$ selects the mode of READY signal synchronization. A low-level signal makes the synchronization a two-step process. This is used when RDY1 and RDY2 inputs are not synchronized to CLK. A high-level signal makes synchronization a one-step process. This is used when RDY1 and RDY2 are synchronized to CLK. See Block Diagram.

READY [Ready]

The READY signal to the processor is synchronized by the RDY inputs to the processor CLK. READY is cleared after the guaranteed hold time of the processor.

Absolute Maximum Ratings

($T_a=25^\circ\text{C}$, $V_{ss}=0\text{V}$)

Parameter	Symbol	Rating Value	Unit
Power supply voltage	V_{DD}	-0.5 to +7.0	V
Input voltage	V_I	-1.0 to $V_{DD}+1.0$	V
Output voltage	V_O	-0.5 to $V_{DD}+0.5$	V
Power dissipation	P_{DMAX}	500	mW
Operating temperature	T_{opr}	-40 to +85	$^\circ\text{C}$
Storage temperature	T_{stg}	-65 to +150	$^\circ\text{C}$

Comment: Exposing the device to stresses above those listed in the absolute maximum ratings could cause permanent damage. Exposure to absolute maximum ratings for extended periods may affect device reliability.

DC Characteristics

(Ta=-40 to +85°C, V_{DD}=5V±10%)

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Input voltage high	V _{IH}	2.2		V	
Input voltage high	V _{IH}	2.6		V	RESIN only
Input voltage low	V _{IL}		0.8	V	
Output voltage high	V _{OH}	V _{DD} -0.8		V	I _{OH} =-4 mA
Output voltage high	V _{OH}	V _{DD} -0.4		V	CLK, I _{OH} =-4 mA
Output voltage low	V _{OL}		0.45	V	I _{OL} =4 mA
Input current leakage	I _{IL}	-1.0	1.0	μA	
RDYSYN input current	I _i	-400	1.0	μA	
RESIN input hysteresis	V _H	0.25		V	
Power supply current (dynamic)	I _{DDdyn}		30	mA	f _{IN} =20 MHz
Power supply current (static)	I _{DD}		200	μA	

Capacitance

(Ta=25°C, V_{DD}=+5V)

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Input capacitance	C _{IN}		12	pF	f _c =1 MHz

AC Characteristics

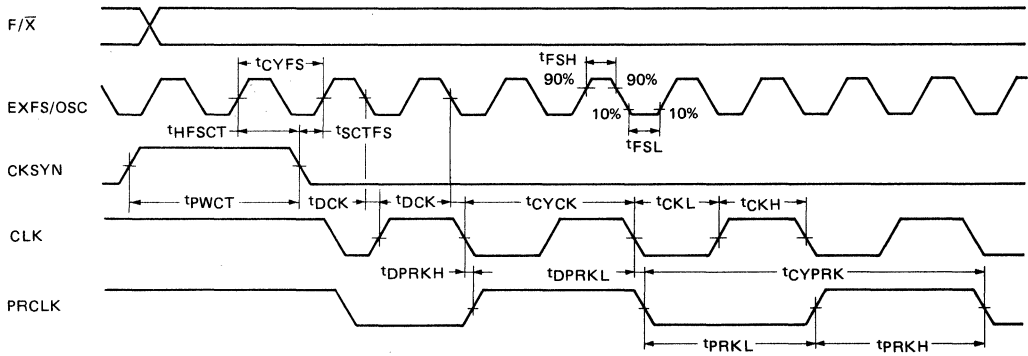
(f_{osc}=10 MHz: Ta=-40 to +85°C, V_{DD}=5V±10%)
 (f_{osc}=16 MHz: Ta=-10 to +70°C, V_{DD}=5V±5%)

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
EXFS cycle time	t _{CYFS}	50		ns	
EXFS high	t _{FSH}	20		ns	90%—90% V _{IN}
EXFS low	t _{FSL}	20		ns	10%—10% V _{IN}
OSC frequency	f _{osc}	8	20	MHz	
CKSYN width	t _{PWCT}	2t _{CYFS}		ns	
CKSYN hold for EXFS (active)	t _{HFST}	20		ns	
CKSYN setup (inactive)	t _{SCTFS}	20		ns	
CLK cycle time	t _{CYCK}	125		ns	
CLK high	t _{CKH}	50		ns	Test point 3.0V, f _{osc} =16 MHz
		80		ns	Test point 3.0V, f _{osc} =10 MHz

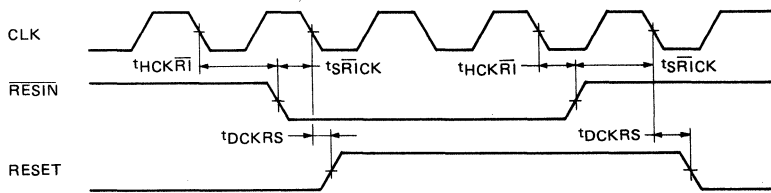
Parameter	Symbol	Min.	Max.	Unit	Test Conditions
CLK low	t _{CKL}	60		ns	Test point 1.5V, f _{osc} =16 MHz
		90		ns	Test point 1.5V, f _{osc} =10 MHz
CLK rise time	tr _{CK}		8	ns	Test point 1.5V to 3.0V, f _{osc} =16 MHz
			10	ns	Test point 1.5V to 3.0V, f _{osc} =10 MHz
CLK fall time	tf _{CK}		7	ns	Test point 3.0V to 1.5V, f _{osc} =16 MHz
			10	ns	Test point 3.0V to 1.5V, f _{osc} =10 MHz
OSC to CLK ↑ delay	td _{CK}	2	30	ns	
OSC to CLK ↓ delay	td _{CK}	-6	28	ns	
PRCLK cycle time	tc _{YPRK}	250		ns	
PRCLK high	tp _{RKH}	tc _{YCK} -20		ns	
PRCLK low	tp _{RKL}	tc _{YCK} -20		ns	
CLK ↓ to PRCLK ↑ delay	td _{PRKH}		22	ns	
CLK ↓ to PRCLK ↓ delay	td _{PRKL}		22	ns	
$\overline{\text{RESIN}}$ to CLK ↓ setup	ts _{RICK}	65		ns	
CLK ↓ to $\overline{\text{RESIN}}$ hold	th _{CKRI}	20		ns	
CLK ↓ to RESET delay	td _{CKRS}		40	ns	
$\overline{\text{REN}}_{1,2}$ to RDY _{1,2} setup	ts _{RERY}	15		ns	
CLK ↓ to $\overline{\text{REN}}_{1,2}$ hold	th _{CKRE}	0		ns	
RDY _{1,2} to CLK ↓ setup	ts _{RYCK}	35		ns	$\overline{\text{RDYSYN}}$ high
RDY _{1,2} to CLK ↑ setup	ts _{RYCK}	35		ns	$\overline{\text{RDYSYN}}$ low
CLK ↓ to RDY _{1,2} hold	th _{CKRY}	0		ns	
$\overline{\text{RDYSYN}}$ ↑ to CLK ↓ setup	ts _{RY$\overline{\text{S}}$CK}	50		ns	
CLK ↓ to $\overline{\text{RDYSYN}}$ ↓ hold	th _{CKRYS}	0		ns	
CLK ↓ to READY ↑ output delay	td _{CKRDY}		8	ns	
CLK ↓ to READY ↓ output delay	td _{CKRDY}		8	ns	
Input rise time	tr _I		20	ns	0.8V to 2.0V
Input fall time	tf _I		12	ns	2.0V to 0.8V
Output rise time	tr _O		20	ns	0.8V to 2.0V
Output fall time	tf _O		12	ns	2.0V to 0.8V

Timing Waveforms

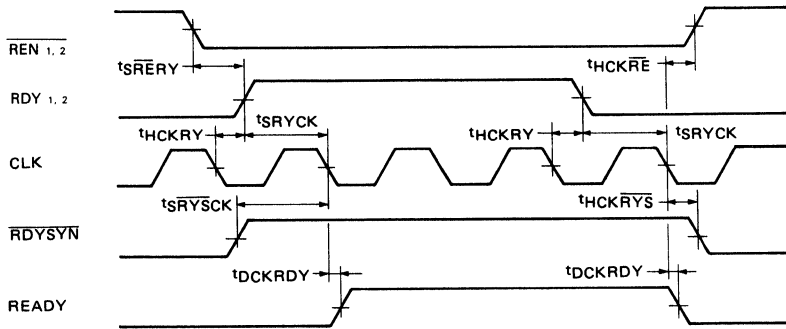
Clock Output



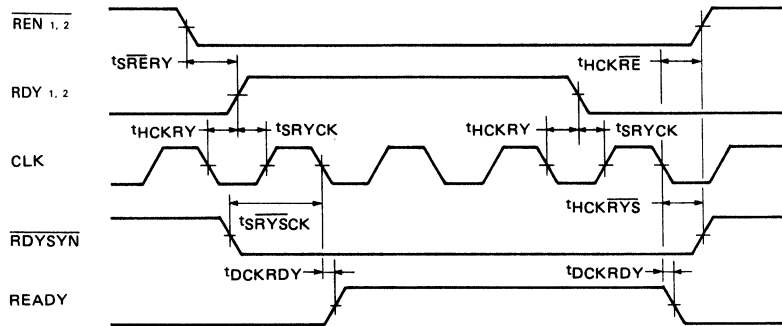
RESET Output



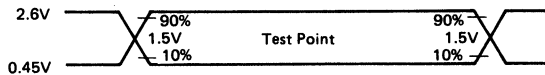
READY Output (RDYSYN High)



READY Output (RDYSYN Low)

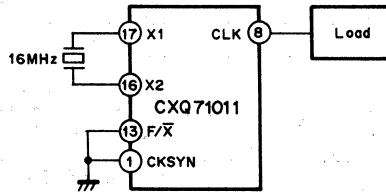


Test Circuit for CLK High or Low Time

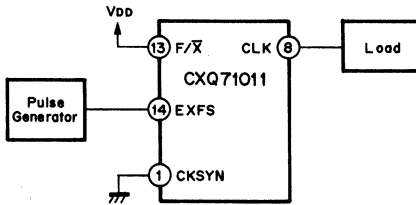


Test Circuits

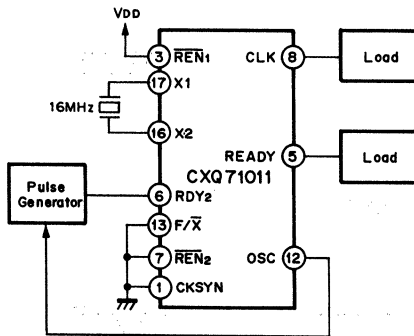
Test Circuit for CLK High or Low Time (in Crystal Oscillation Mode)



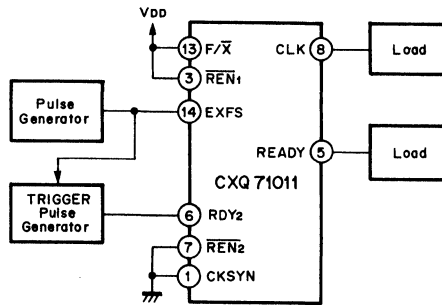
Test Circuit for CLK High or Low Time (in EXFS Oscillation Mode)



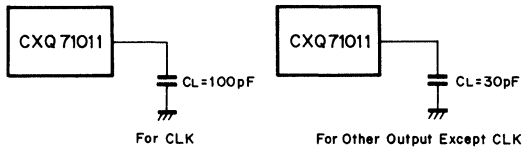
Test Circuit for CLK to READY (in Crystal Oscillation Mode)



Test Circuit for CLK to READY (in EXFS Oscillation Mode)

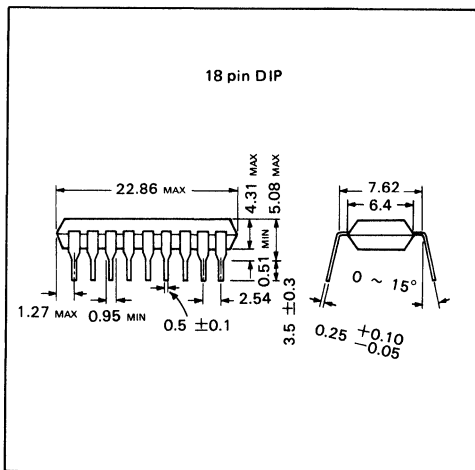


Loading Circuits



Package Outline

Unit: mm



8-Bit Latch

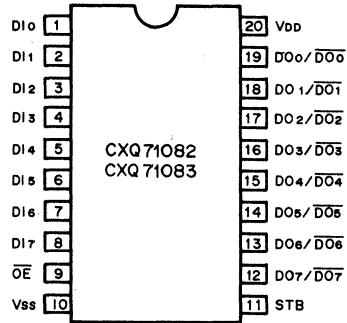
Description

CXQ71082 and CXQ71083 are CMOS 8-bit transparent latches with three-state output buffers. They are used as bus buffers or bus multiplexers in microprocessor systems. Their high-drive capability makes them suitable for data latch, buffer or I/O port applications.

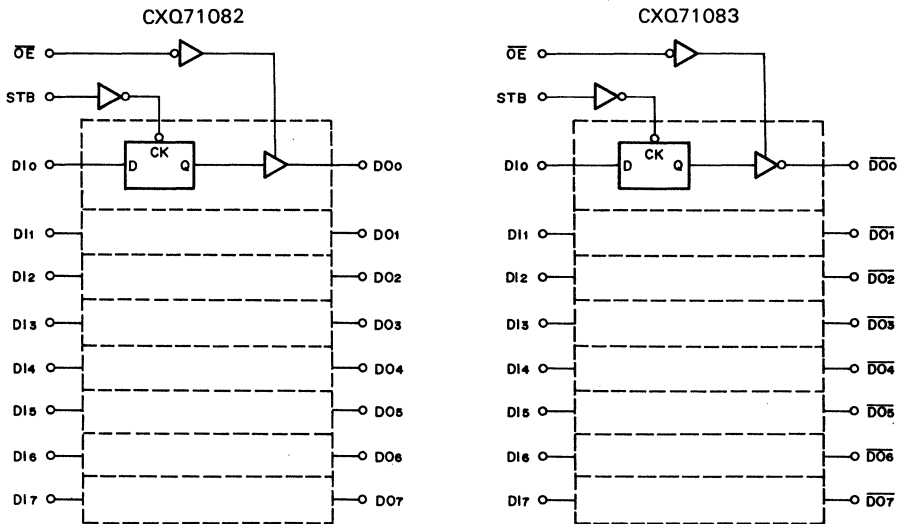
Features

- Transparent operation
- 8-bit parallel data register
- Three-state output buffer
- High drive capability output buffer ($I_{OL}=12\text{ mA}$)
- 8086, 8088, CXQ70108 and CXQ70116 CPU bus compatible
- CXQ71082: non-inverted output
CXQ71083: inverted output
- CMOS technology
- +5V single power supply
- 20-pin plastic DIP (300 mil)
- NEC μ PD71082, μ PD71083 compatible

Pin Configuration (Top View)



Block Diagram



Pin Identification

No.	Symbol	Direction	Function
1	DI ₀	In	Data input, bit 0
2	DI ₁	In	Data input, bit 1
3	DI ₂	In	Data input, bit 2
4	DI ₃	In	Data input, bit 3
5	DI ₄	In	Data input, bit 4
6	DI ₅	In	Data input, bit 5
7	DI ₆	In	Data input, bit 6
8	DI ₇	In	Data input, bit 7
9	\overline{OE}	In	Output enable input
10	V _{SS}		Ground
11	STB	In	Strobe input
12	DO ₇ / $\overline{DO_7}$	Out	Data output, bit 7
13	DO ₆ / $\overline{DO_6}$	Out	Data output, bit 6
14	DO ₅ / $\overline{DO_5}$	Out	Data output, bit 5
15	DO ₄ / $\overline{DO_4}$	Out	Data output, bit 4
16	DO ₃ / $\overline{DO_3}$	Out	Data output, bit 3
17	DO ₂ / $\overline{DO_2}$	Out	Data output, bit 2
18	DO ₁ / $\overline{DO_1}$	Out	Data output, bit 1
19	DO ₀ / $\overline{DO_0}$	Out	Data output, bit 0
20	V _{DD}		Power supply

Pin Functions

DI₇-DI₀ [Data Input]

DI₇-DI₀ are data input lines to the 8-bit data latch. Data on DI lines are latched with the trailing edge of STB (high to low). The data passes through the latch while STB is high.

DO₇-DO₀/ $\overline{DO_7}$ - $\overline{DO_0}$ [Data Output]

DO₇-DO₀/ $\overline{DO_7}$ - $\overline{DO_0}$ are data output lines from the 8-bit data latch. When \overline{OE} is high, these lines float to the high-impedance state. When \overline{OE} is low, data from the latch is output, either non-inverted (CXQ71082) or inverted (CXQ71083).

STB [Strobe]

STB is the strobe signal for the 8-bit latch. When STB is high, data on the DI lines passes through the 8-bit latch. Data is latched on the trailing edge of STB (high to low). When STB is low, latched data is stable.

\overline{OE} [Output Enable]

\overline{OE} is the output enable signal for the DO lines. When \overline{OE} is high, DO lines are high impedance. When \overline{OE} is low, data from the 8-bit latch is output to DO₇-DO₀.

STB	\overline{OE}	D07-D00	8-Bit Data Latch
Low	Low	Latched data from 8-bit data latch	DI line data has been latched with trailing edge of STB (high \rightarrow low)
	High	High impedance	
High	Low	Data on DI7-DI0	Pass through
	High	High impedance	

Absolute Maximum Ratings ($T_a=25^\circ\text{C}$, $V_{SS}=0\text{V}$)

Parameter	Symbol	Rating Value	Unit
Power supply voltage	V_{DD}	-0.5 to +7.0	V
Input voltage	V_I	-1.0 to $V_{DD}+1.0$	V
Output voltage	V_O	-0.5 to $V_{DD}+0.5$	V
Power dissipation	P_{DMAX}	500	mW
Operating temperature	T_{opr}	-40 to +85	$^\circ\text{C}$
Storage temperature	T_{stg}	-65 to +150	$^\circ\text{C}$

Comment: Exposing the device to stresses above those listed in the absolute maximum ratings could cause permanent damage. Exposure to absolute maximum ratings for extended periods may affect device reliability.

DC Characteristics ($T_a=-40$ to $+85^\circ\text{C}$, $V_{DD}=5\text{V}\pm 10\%$)

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Input voltage high	V_{IH}	2.2		V	
Input voltage low	V_{IL}		0.8	V	
Output voltage high	V_{OH}	$V_{DD}-0.8$		V	$I_{OH}=-4$ mA
Output voltage low	V_{OL}		0.45	V	$I_{OL}=12$ mA
Input current	I_I	-1.0	1.0	μA	$V_I=V_{DD}$, V_{SS}
Leakage current at high impedance	I_{OFF}	-10	10	μA	$\overline{OE}=V_{DD}$
Power supply current (static)	I_{DD}		80	μA	$V_I=V_{DD}$, V_{SS}
Power supply current (dynamic)	I_{DDdyn}		20	mA	$f_{IN}=1$ MHz $C=200$ pF

Capacitance ($T_a=25^\circ\text{C}$, $V_{DD}=+5\text{V}$)

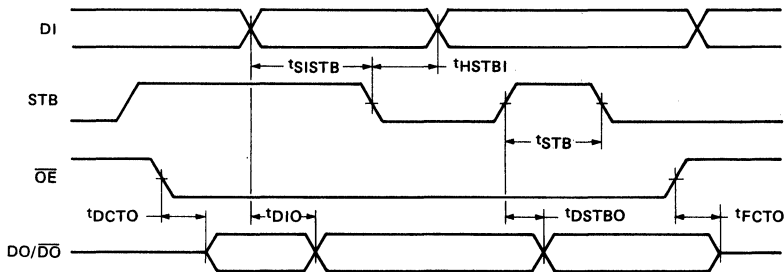
Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Input capacitance	C_{IN}		12	pF	$f_c=1$ MHz

AC Characteristics

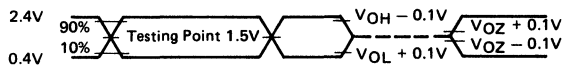
($T_a = -40$ to $+85^\circ\text{C}$, $V_{DD} = 5V \pm 10\%$)

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Input to output delay	t_{DIO}	5	40	ns	Load circuit [a]
STB to output delay	t_{DSTBO}	10	60	ns	Load circuit [a]
Data float time from \overline{OE} high	t_{FCTO}	5	30	ns	Load circuit [b]
Data output delay from \overline{OE} low	t_{DCTO}	10	40	ns	Load circuit [b]
Input to STB setup time	t_{SISTB}	0		ns	Load circuit [a]
Input to STB hold time	t_{HSTBI}	25		ns	Load circuit [a]
STB high time	t_{STB}	20		ns	Load circuit [a]
Signal rise time	t_{LH}		20	ns	0.8V to 2.0V
Signal fall time	t_{HL}		12	ns	2.0V to 0.8V

Timing Diagram

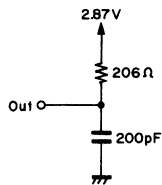


AC Testing Waveform

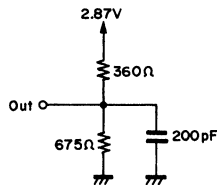


Loading Circuits for AC Testing

[a] V_{OL} , V_{OH} Outputs



[b] Three-State Outputs



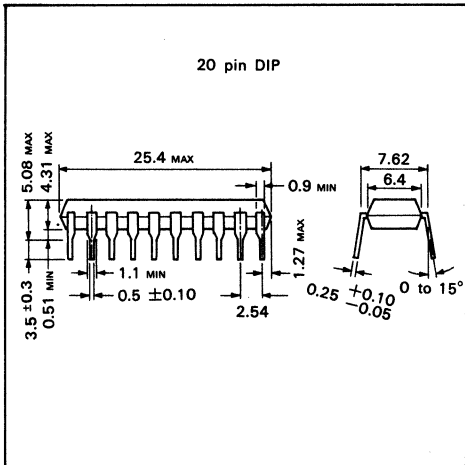
Loading Conditions: $I_{OL} = 12\text{ mA}$, $I_{OH} = -4\text{ mA}$, $C_L = 200\text{ pF}$

Functional Description

The CXQ71082 and CXQ71083 are 8-bit data latches strobed by the STB signal with high-drive capability output buffers controlled by the \overline{OE} signal. Data on the DI lines latched by the trailing edge of STB (high to low). When STB is high, data passes through the latch. When \overline{OE} is high, DO lines are high impedance. When \overline{OE} is low, the contents of the latches are output on DO7-DO0. The DO lines are isolated from \overline{OE} switching noise.

Package Outline

Unit: mm



8-Bit Bus Driver/Receiver

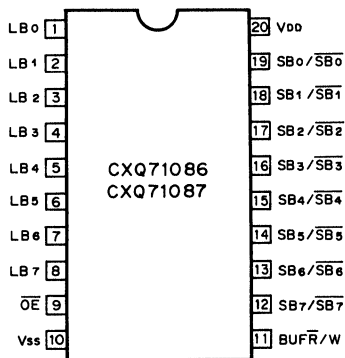
Description

The CXQ71086 and CXQ71087 are CMOS 8-bit, bidirectional bus driver/receivers with three-state output buffers.

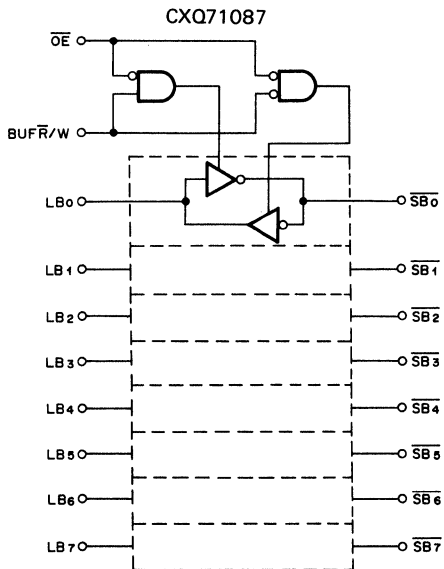
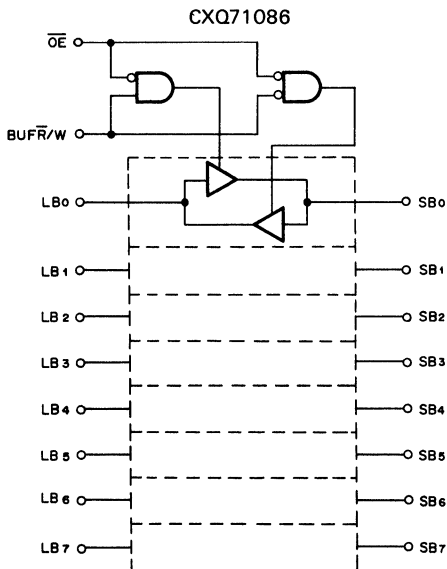
Features

- Bidirectional 8-bit parallel bus buffer
- Three-state output
- High drive capability system bus output ($I_{OL}=12\text{ mA}$)
- 8086, 8088, CXQ70108 and CXQ70116 CPU bus compatible
- CXQ71086: non-inverted system bus output
CXQ71087: inverted system bus output
- CMOS technology
- +5V single power supply
- 20-pin plastic DIP (300 mil)
- NEC μ PD71086, μ PD71087 compatible

Pin Configuration (Top View)



Block Diagram



Pin Identification

No.	Symbol	Direction	Function
1	LB ₀	I/O	CPU local data bus, bit 0
2	LB ₁	I/O	CPU local data bus, bit 1
3	LB ₂	I/O	CPU local data bus, bit 2
4	LB ₃	I/O	CPU local data bus, bit 3
5	LB ₄	I/O	CPU local data bus, bit 4
6	LB ₅	I/O	CPU local data bus, bit 5
7	LB ₆	I/O	CPU local data bus, bit 6
8	LB ₇	I/O	CPU local data bus, bit 7
9	\overline{OE}	In	Output enable input
10	V _{SS}		Ground
11	BUFR \overline{W}	In	Buffer read/write input
12	SB ₇ / \overline{SB} ₇	I/O	System data bus, bit 7
13	SB ₆ / \overline{SB} ₆	I/O	System data bus, bit 6
14	SB ₅ / \overline{SB} ₅	I/O	System data bus, bit 5
15	SB ₄ / \overline{SB} ₄	I/O	System data bus, bit 4
16	SB ₃ / \overline{SB} ₃	I/O	System data bus, bit 3
17	SB ₂ / \overline{SB} ₂	I/O	System data bus, bit 2
18	SB ₁ / \overline{SB} ₁	I/O	System data bus, bit 1
19	SB ₀ / \overline{SB} ₀	I/O	System data bus, bit 0
20	V _{DD}		Power supply

Pin Functions

LB₇-LB₀ [Local Data Bus]

LB₇-LB₀ are connected to the CPU local data bus. They input and output data between the CPU and memory, I/O or other peripherals. Data read/write mode is controlled by the BUFR \overline{W} signal input.

SB₇-SB₀/ \overline{SB} ₇- \overline{SB} ₀ [System Data Bus]

SB₇-SB₀/ \overline{SB} ₇- \overline{SB} ₀ are connected to the system bus, along with the memory, I/O or other peripherals. CXQ71086 outputs non-inverted signals, SB₇-SB₀. CXQ71087 outputs inverted signals, \overline{SB} ₇- \overline{SB} ₀.

\overline{OE} [Output Enable]

\overline{OE} controls the output buffers. When \overline{OE} is high, all output buffers float to the high-impedance state. When \overline{OE} is low, data is output from the buffers specified by the BUFR \overline{W} signal.

BUFR \overline{W} [Buffer Read/Write]

The data read/write mode is controlled by the BUFR \overline{W} signal input. When BUFR \overline{W} is high, LB lines are in input mode and SB lines are in output mode. When BUFR \overline{W} is low, SB lines are in input mode, and LB lines are output. See below.

\overline{OE}	BUFR/W	LB Pins	SB/ \overline{SB} Pins	Mode
Low	Low	Output	Input	System bus mode
	High	Input	Output	Local bus mode
High	Low	—	—	High impedance
	High	—	—	High impedance

Note: When \overline{OE} is high, all local and system bus pins float to high-impedance state.

Absolute Maximum Ratings

($T_a=25^\circ\text{C}$, $V_{SS}=0\text{V}$)

Parameter	Symbol	Rating Value	Unit
Power supply voltage	V_{DD}	-0.5 to +7.0	V
Input voltage	V_I	-1.0 to $V_{DD}+1.0$	V
Output voltage	V_O	-0.5 to $V_{DD}+0.5$	V
Power dissipation	$P_{D\text{MAX}}$	500	mW
Operating temperature	T_{opr}	-40 to +85	$^\circ\text{C}$
Storage temperature	T_{stg}	-65 to +150	$^\circ\text{C}$

Comment: Exposing the device to stresses above those listed in the absolute maximum ratings could cause permanent damage. Exposure to absolute maximum ratings for extended periods may affect device reliability.

DC Characteristics

($T_a=-40$ to $+85^\circ\text{C}$, $V_{DD}=5\text{V}\pm 10\%$)

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Input voltage high	V_{IH}	2.2		V	
Input voltage low	V_{IL}		0.8	V	
Output voltage high	V_{OH}	$V_{DD}-0.8$		V	$I_{OH}=-4$ mA
Output voltage low	V_{OL}		0.45	V	$I_{OL}(\text{LB})=4$ mA
Output voltage low	V_{OL}		0.45	V	$I_{OL}(\text{SB}/\overline{\text{SB}})=12$ mA
Input current	I_I	-1.0	1.0	μA	$V_I=V_{DD}$, V_{SS}
Leakage current at high impedance	I_{OFF}	-10	10	μA	$\overline{OE}=V_{DD}$
Power supply current (static)	I_{DD}		80	μA	$V_I=V_{DD}$, V_{SS}
Power supply current (dynamic)	$I_{DD\text{dyn}}$		40	mA	$f_{IN}=2$ MHz

Capacitance

($T_a=25^\circ\text{C}$, $V_{DD}=+5\text{V}$)

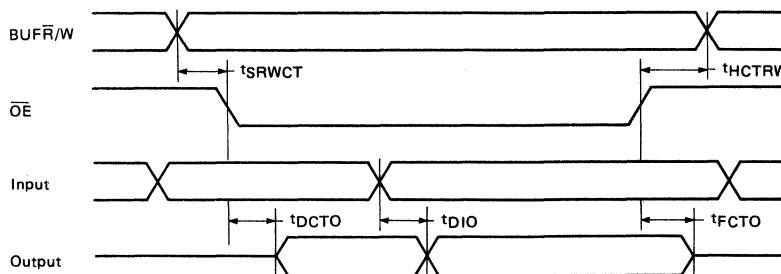
Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Input Capacitance	C_{IN}		24	pF	$f_c=1$ MHz LB lines

AC Characteristics

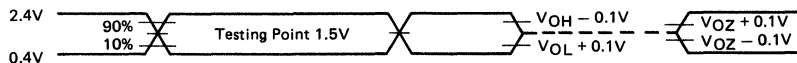
($T_a = -40$ to $+85^\circ\text{C}$, $V_{DD} = 5\text{V} \pm 10\%$)

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Input to Output Delay	tDIO	5	40	ns	Load [1], [1'] and [2], [2']
BUFR/W Setup Time to $\overline{\text{OE}}$	tSRWCT	10		ns	
BUFR/W Hold Time from $\overline{\text{OE}}$	tHCTRW	5		ns	
Data Float Time from $\overline{\text{OE}}$	tFCTO	5	30	ns	Load [3] and [3']
Data Output Delay from $\overline{\text{OE}}$	tDCTO	10	40	ns	Load [3] and [3']
Signal Rise Time	tR		20	ns	0.8V to 2.0V
Signal Fall Time	tF		12	ns	2.0V to 0.8V

Timing Waveforms

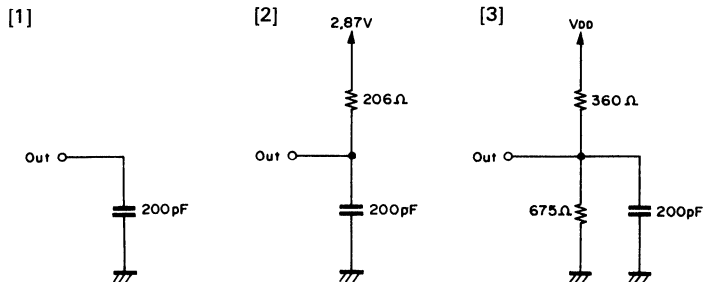


AC Testing Waveform

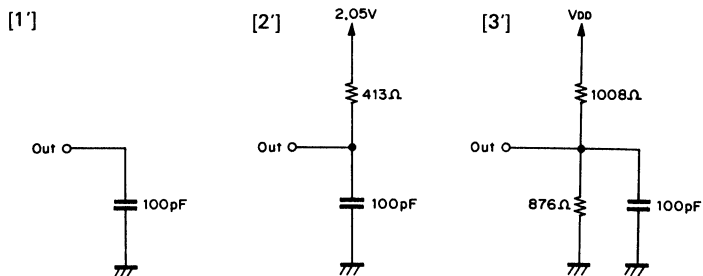


Loading Circuits for AC Test

LB to SB/ \overline{SB}



SB/ \overline{SB} to LB

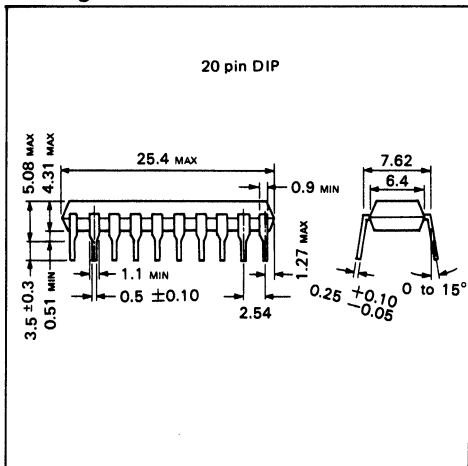


Functional Description

CXQ71086 and CXQ71087 are 8-bit, bidirectional bus driver/receivers with three-state outputs. The CXQ71086 provides a non-inverted system bus. The CXQ71087 provides an inverted system bus. These devices are used to expand CPU bus drive capability. The input/output lines are isolated from \overline{OE} and BUR/\overline{W} switching noise.

Package Outline

Unit: mm



System Bus Controller

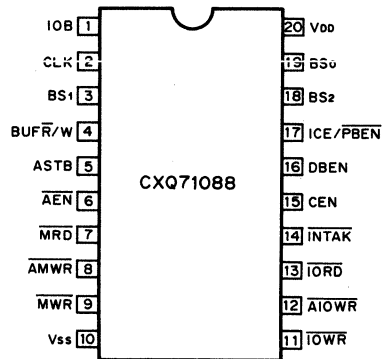
Description

The CXQ71088 is a CMOS system bus controller for a CXQ70108 or CXQ70116 CPU processor system. It controls the memory or I/O peripheral bus.

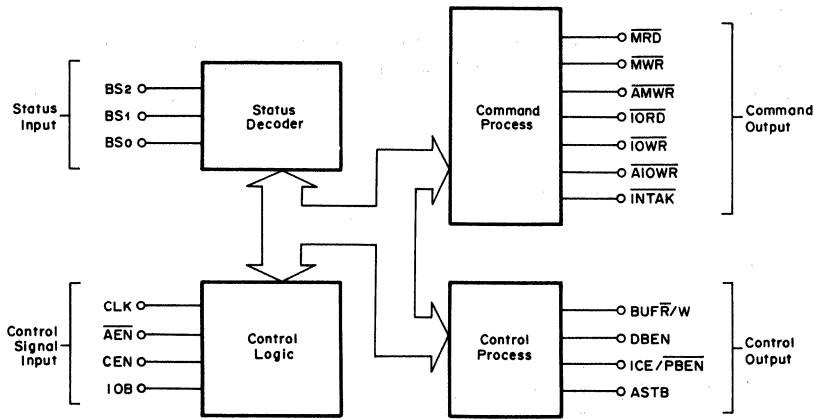
Features

- Bus controller for microcomputer system expansion
- Command outputs for system bus control
- Control outputs for I/O peripheral bus control
- High drive capability for command and control outputs (I_{OL}=12 mA)
- Three-state outputs for command outputs
- Advanced I/O and memory write command outputs
- CXQ70108/CXQ70116 CPU system compatible
- CMOS technology
- +5V single power supply
- 20-pin plastic DIP (300 mil)
- NEC μ PD71088 compatible

Pin Configuration (Top View)



Block Diagram



Pin Identification

No.	Symbol	Direction	Function
1	IOB	In	Input/output bus mode
2	CLK	In	Clock
3	BS ₁	In	Bus status input 1
4	BUFR/W	Out	Buffer read/write
5	ASTB	Out	Address strobe
6	$\overline{\text{AEN}}$	In	Address enable
7	$\overline{\text{MRD}}$	Out	Memory read
8	$\overline{\text{AMWR}}$	Out	Advanced memory write command
9	$\overline{\text{MWR}}$	Out	Memory write command
10	V _{SS}		Ground
11	$\overline{\text{IOWR}}$	Out	I/O write command
12	$\overline{\text{AIOWR}}$	Out	Advanced I/O write command
13	$\overline{\text{IORD}}$	Out	I/O read command
14	$\overline{\text{INTAK}}$	Out	Interrupt acknowledge
15	CEN	In	Command enable
16	DBEN	Out	Data buffer enable
17	$\overline{\text{ICE/PBEN}}$	Out	Interrupt cascade enable/Peripheral data bus enable
18	BS ₂	In	Bus status input 2
19	BS ₀	In	Bus status input 0
20	V _{DD}		Power supply

Pin Functions

BS₀-BS₂ [Bus Status Inputs 0-2]

BS₀-BS₂ are connected to the encoded CPU status outputs. The CXQ71088 decodes these status outputs into command and control outputs for timing control. See the command logic table in the functional description for an explanation of these inputs.

CLK [Clock]

CLK is connected to the same clock output that drives the CPU clock, usually the CLK output of a CXQ71011. It is the internal system clock of the CXQ71088.

$\overline{\text{AEN}}$ [Address Enable]

$\overline{\text{AEN}}$ controls the command output buffers. When IOB is low, a low-level $\overline{\text{AEN}}$ causes the command buffers to output command output signals. A high-level $\overline{\text{AEN}}$ makes all command lines go to high impedance. When IOB is high, the CXQ71088 is in I/O bus mode, and the command lines are not affected by $\overline{\text{AEN}}$.

CEN [Command Enable]

DBEN, PBEN and all command outputs are controlled by CEN. When CEN is high, all these outputs are active. When CEN is low, they are inactive.

IOB [I/O Bus Mode]

When IOB is high, the bus control mode is I/O mode. When IOB is low, the bus control mode is system bus mode.

 $\overline{\text{MRD}}$ [Memory Read Command]

$\overline{\text{MRD}}$ is the signal to read data from a memory device.

 $\overline{\text{MWR}}$ [Memory Write Command]

$\overline{\text{MWR}}$ is the signal to write data to a memory device.

 $\overline{\text{AMWR}}$ [Advanced Memory Write Command]

This command is the same as $\overline{\text{MWR}}$, except that it is generated one state (clock cycle) earlier than $\overline{\text{MWR}}$.

 $\overline{\text{IORD}}$ [I/O Read Command]

$\overline{\text{IORD}}$ is the signal to read data from an input device.

 $\overline{\text{IOWR}}$ [I/O Write Command]

$\overline{\text{IOWR}}$ is the signal to write data to an output device.

 $\overline{\text{AIOWR}}$ [Advanced I/O Write Command]

This command is the same as $\overline{\text{IOWR}}$, except that it is generated one state (clock cycle) earlier than $\overline{\text{IOWR}}$.

 $\overline{\text{INTAK}}$ [Interrupt Acknowledge]

$\overline{\text{INTAK}}$ acknowledges interrupt requests. Requesting devices output an interrupt vector address in response to $\overline{\text{INTAK}}$.

ASTB [Address Strobe]

ASTB is the control signal to latch the address outputs from the CPU in the external address latch of a CXQ71082 or CXQ71083. Address data should be strobed with the trailing edge (high to low) of ASTB.

DBEN [Data Buffer Enable]

DBEN is the control signal that activates the data bus driver/receiver of a CXQ71086 or CXQ71087 to input or output data between the CPU local bus and memory or I/O peripheral bus.

BUF $\overline{\text{R}}$ /W [buffer Read/Write]

BUF $\overline{\text{R}}$ /W is the signal that controls the data direction between CPU and memory or I/O peripherals. When BUF $\overline{\text{R}}$ /W is high, data is transferred from the CPU local bus to the memory or I/O system bus. When BUF $\overline{\text{R}}$ /W is low, data is transferred from the memory or I/O system bus to the CPU local bus.

ICE/ $\overline{\text{PBEN}}$ [Interrupt Cascade Enable/Peripheral Data Bus Enable]

The meaning of this multiplexed output signal depends on IOB. If IOB is low (system bus mode), it is the ICE output. ICE controls the cascade address transfer from master priority interrupt controller to slave priority interrupt controller. The slave reads the address from the master when ICE goes high.

When IOB is high, it becomes $\overline{\text{PBEN}}$. $\overline{\text{PBEN}}$ controls the I/O bus the same way that DBEN controls the system bus. In this case, however, the output is active low.

Absolute Maximum Ratings

(Ta=25°C, Vss=0V)

Parameter	Symbol	Rating Value	Unit
Power supply voltage	V _{DD}	-0.5 to +7.0	V
Input voltage	V _I	-1.0 to V _{DD} +1.0	V
Output voltage	V _O	-0.5 to V _{DD} +0.5	V
Power dissipation	P _D	500	mW
Operating temperature	T _{opr}	-40 to +85	°C
Storage temperature	T _{stg}	-65 to +150	°C

Comment: Exposing the device to stresses above those listed in the absolute maximum ratings could cause permanent damage. Exposure to absolute maximum ratings for extended periods may affect device reliability.

DC Characteristics(Ta=-40°C to +85°C, V_{DD}=5V±10%)

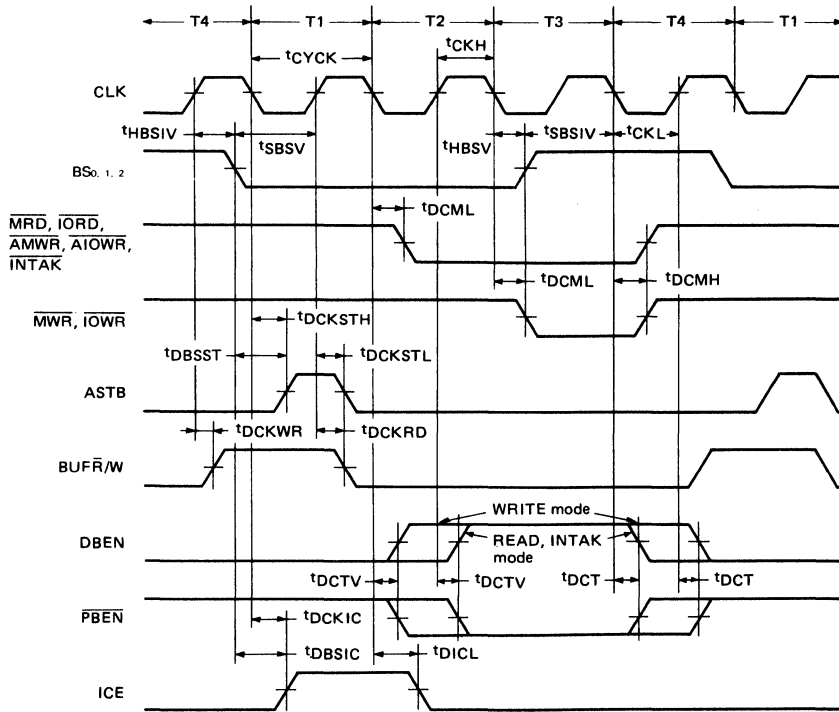
Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Input voltage high	V _{IH}	2.2		V	
Input voltage low	V _{IL}		0.8	V	
Output voltage high	V _{OH}	V _{DD} -0.8		V	I _{OH} =-4 mA
Output voltage low	V _{OL}		0.45	V	I _{OL} =12 mA, Command
Output voltage low	V _{OL}		0.45	V	I _{OL} =4 mA, Control
Input current	I _I	-1.0	1.0	μA	V _I =V _{DD} , V _{SS}
Leakage current at high impedance	I _{OFF}	-10	10	μA	
Power supply current (static)	I _{DD}		80	μA	V _I =V _{DD} , V _{SS}
Power supply current (dynamic)	I _{DDdyn}		20	mA	f _{IN} =10 MHz

AC Characteristics

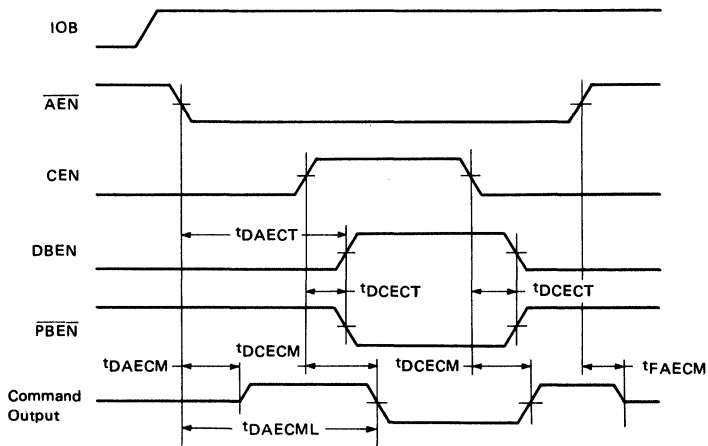
(Ta=-40 to +85°C, VDD=5±10%)

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
CLK cycle period	tCYCK	125		ns	
CLK pulse width low	tCKL	60		ns	
CLK pulse width high	tCKH	40		ns	
Setup time for bus status active to CLK↓	tsBSV	40		ns	
Hold time for bus status inactive from CLK↓	thBSV	10		ns	
Setup time for bus status inactive to CLK↓	tsBSIV	35		ns	
Hold time for bus status inactive from CLK↓	thBSIV	10		ns	
DBEN, $\overline{\text{PBEN}}$ active delay	tdCTV	10	50	ns	Load circuit [b]
DBEN, $\overline{\text{PBEN}}$ inactive delay	tdCT	10	50	ns	
ASTB active delay from CLK↓	tdCKSTH		30	ns	
ASTB active delay from bus status	tdBSST		25	ns	
ASTB inactive delay from CLK↑	tdCKSTL	7	25	ns	
ICE active delay from CLK↓	tdCKIC		30	ns	
ICE inactive delay from CLK↓	tdICL	10	50	ns	
ICE active delay from bus status	tdBSIC		25	ns	
$\overline{\text{BUFR}}/\overline{\text{W}}\downarrow$ output delay	tdCKRD		60	ns	
$\overline{\text{BUFR}}/\overline{\text{W}}\uparrow$ output delay	tdCKWR		40	ns	
$\overline{\text{AEN}}$ to DBEN, $\overline{\text{PBEN}}$ delay	tDAECT		30	ns	
CEN to DBEN, $\overline{\text{PBEN}}$ delay	tDCECT		30	ns	
CEN to command delay	tDCECM		tDCML	ns	
Command active delay from CLK↓	tDCML	10	40	ns	
Command inactive delay from CLK↓	tDCMH	10	40	ns	
Command enable delay from $\overline{\text{AEN}}$	tDAECM		40	ns	
Command output delay from $\overline{\text{AEN}}$	tDAECML	100	295	ns	
Command disable delay from $\overline{\text{AEN}}\uparrow$	tFAECM		50	ns	
Input/output rise time	tr		20	ns	0.8V to 2.0V
Input/output fall time	tf		12	ns	2.0V to 0.8V

Timing Waveforms

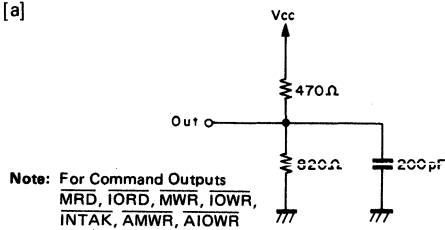


AEN, PBEN, DBEN Timing

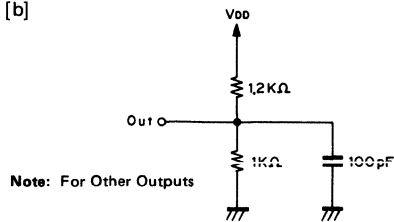


Loading Circuits for AC Testing

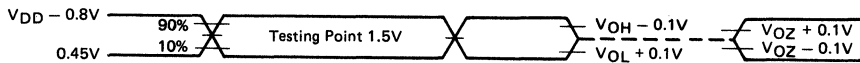
[a]



[b]



AC Testing Waveform



Functional Description

Command Logic

The CXQ71088 decodes the CPU bus status outputs into command outputs. The bus status outputs (BS₀-BS₂) and their decoded commands are shown below.

BS ₂	BS ₁	BS ₀	CPU Status	CXQ71088 Command Output
Low	Low	Low	Interrupt acknowledge	$\overline{\text{INTAK}}$
		High	I/O read mode	$\overline{\text{IORD}}$
	High	Low	I/O write mode	$\overline{\text{IOWR}}$, $\overline{\text{AIOWR}}$
		High	Halt mode	—
High	Low	Low	Instruction fetch mode	$\overline{\text{MRD}}$
		High	Memory read mode	$\overline{\text{MRD}}$
	High	Low	Memory write mode	$\overline{\text{MWR}}$, $\overline{\text{AMWR}}$
		High	No bus cycle mode	—

Bus Control Mode

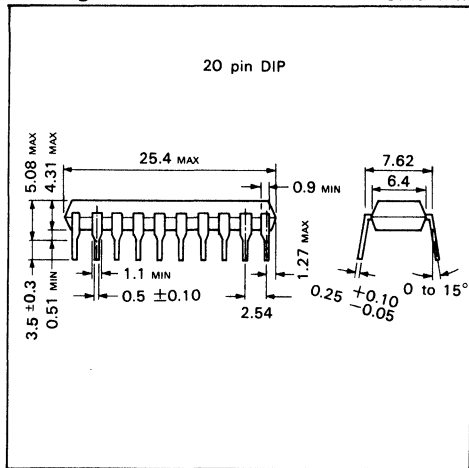
Bus control mode is controlled with IOB and $\overline{\text{AEN}}$ signals as shown below.

Control Input			Command Output		Control Output		Mode
CEN	IOB	AEN	Memory Command: MRD, MWR, AMWR	I/O Command: IOWR, AIOWR, IORD, INTAK	ICE/ $\overline{\text{PBEN}}$	BUFR/W, ASTB, DBEN	
H	H	H	High impedance	Output enable	$\overline{\text{PBEN}}$	Output enable	I/O bus mode
		L	Output enable				
	L	H	High impedance	High impedance	ICE	Output enable	System bus mode
		L	Output enable	Output enable			
L	X	X	H	H	H	Output enable (DBEN=L)	Command disable mode

Note: X=Don't care

Package Outline

Unit: mm



Serial Interface Unit

Description

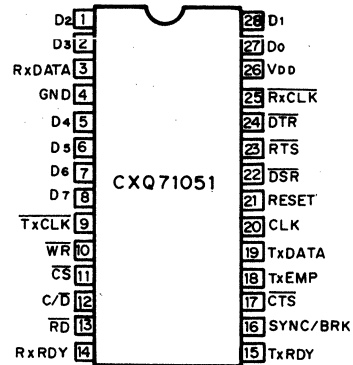
The CXQ71051 serial interface unit is a CMOS USART that provides serial data communications in microcomputer systems. It can be programmed to communicate in synchronous or asynchronous serial data transmission protocols, including IBM Bisync.

The USART receives serial data streams and converts them into parallel data characters for the CPU. While receiving serial data, the USART also accepts parallel data from the CPU, converts it to serial, and transmits the data. The USART signals the CPU when it has received or transmitted a character and requires service. The CPU may read complete USART status data at any time.

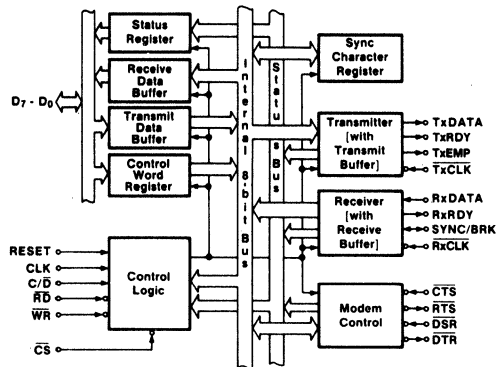
Features

- Synchronous operation
 - Single or double SYNC characters
 - Internal/external synchronization
 - Automatic SYNC character insertion
- Asynchronous operation
 - Clock rate: 1, 16, or 64×baud rate
 - Stop bits: 1, 1.5, or 2 bits
 - Break transmission
 - Automatic break detection
 - False start bit detection
- Baud rate ×1 clock: DC to 240 kilobaud
- Full duplex, double-buffered transmitter/receiver
- Error detection: parity, overrun, and framing
- Five- to eight-bit characters
- Low-power standby mode
- Compatible with standard microcomputer bus
- CMOS technology
- +5V single power supply
- 28-pin plastic DIP (600 mil)
- NEC μ PD71051 compatible.

Pin Configuration



Block Diagram



Pin Identification

No.	Symbol	Direction	Function
1, 2	D ₂ , D ₃	I/O	Data bus
3	RxDATA	In	Receive data
4	GND		Ground
5–8	D ₄ –D ₇	I/O	Data bus
9	$\overline{\text{TxCLK}}$	In	Transmitter clock
10	$\overline{\text{WR}}$	In	Write strobe
11	$\overline{\text{CS}}$	In	Chip select
12	C/D	In	Control or data select
13	$\overline{\text{RD}}$	In	Read strobe
14	RxRDY	Out	Receiver ready
15	TxRDY	Out	Transmitter ready
16	SYNC/BRK	In (SYNC) Out (BRK)	Synchronization/break
17	$\overline{\text{CTS}}$	In	Clear to send
18	TxEMP	Out	Transmitter empty
19	TxDATA	Out	Transmit data
20	CLK	In	Clock
21	RESET	In	Reset
22	$\overline{\text{DSR}}$	In	Data set ready
23	$\overline{\text{RTS}}$	Out	Return to send
24	$\overline{\text{DTR}}$	Out	Data terminal ready
25	RxCLK	In	Receiver clock
26	V _{DD}		Power supply
27, 28	D ₀ , D ₁	I/O	Data bus

Pin Functions

D7–D0 [Data Bus]

These pins are an 8-bit, 3-state, bidirectional data bus. The bus transfers data by connecting to the system data bus.

RESET [Reset]

A 'high' on this pin initializes the CXQ71051 and puts it into an idle state. It performs no operations in the idle state. The CXQ71051 enters standby mode when this signal falls from a high level to a low level. Standby mode is released when the CPU writes a mode byte to the CXQ71051. The reset pulse width must be at least 6 tcyk and the clock must be running.

CLK [Clock]

This clock input produces internal timing for the CXQ71051.

The clock frequency should be at least 30 times the transmitter or receiver clock input frequency ($\overline{\text{TxCLK}}$, $\overline{\text{RxCLK}}$) in sync or async mode with the $\times 1$ clock. This assures stable operation. The clock frequency must be more than 4.5 times the $\overline{\text{TxCLK}}$ or $\overline{\text{RxCLK}}$ in async mode using $\times 16$ or $\times 64$ clock mode.

$\overline{\text{CS}}$ [Chip Select]

A 'low' on this pin allows the CPU to read from or write to the CXQ71051.

When $\overline{\text{CS}}=1$, the CXQ71051 is not selected, the data bus D7–D0 is in the high impedance state, and the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals are ignored.

$\overline{\text{RD}}$ [Read Strobe]

A 'low' on this pin when $\overline{\text{CS}}=0$ allows the CPU to read data or status information from the CXQ71051.

$\overline{\text{WR}}$ [Write Strobe]

A 'low' on this pin when $\overline{\text{CS}}=0$ allows the CPU to write data or control byte to the CXQ71051.

C/ $\overline{\text{D}}$ [Control or Data]

This signal determines the data type transferred to and from the CXQ71051. When $C/\overline{\text{D}}=1$, the data is control byte or status. When $C/\overline{\text{D}}=0$, the data is character data. This pin is normally connected to the least significant bit (A_0) of the CPU address bus.

Table 1. Control Signals and Operations

$\overline{\text{CS}}$	$\overline{\text{RD}}$	$\overline{\text{WR}}$	C/ $\overline{\text{D}}$	CXQ71051 Operation	CPU Operation
0	0	1	0	Receive data buffer \rightleftarrows Data bus	Read receive data
0	0	1	1	Status register \rightleftarrows Data bus	Read status
0	1	0	0	Data bus \rightleftarrows Transmit data buffer	Write transmit data
0	1	0	1	Data bus \rightleftarrows Control byte register	Write control byte
0	1	1	X	Data bus: High impedance	None
1	X	X	X	Data bus: High impedance	None

$\overline{\text{DSR}}$ [Data Set Ready]

This is a general-purpose input pin that can be used for modem control. The status of this pin can be sensed by reading bit 7 of the status byte.

 $\overline{\text{DTR}}$ [Data Terminal Ready]

This is a general-purpose output pin that can be used for modem control. The state of this pin can be controlled by writing bit 1 of the command byte. If bit 1=0, then $\overline{\text{DTR}}=1$. If bit 1=1, then $\overline{\text{DTR}}=0$.

 $\overline{\text{RTS}}$ [Request to Send]

This is a general-purpose output pin that can be used for modem control. The status of this pin can be controlled by writing bit 5 of the command byte. If bit 5=1, then $\overline{\text{RTS}}=0$. If bit 5=0, then $\overline{\text{RTS}}=1$.

 $\overline{\text{CTS}}$ [Clear to Send]

This is an input pin that controls data transmission. The CXQ71051 is able to transmit serial data when $\overline{\text{CTS}}=0$ and the command byte sets TxEN=1. If $\overline{\text{CTS}}$ goes high during transmission, the sending operation will stop after sending all currently written data and the TxDATA pin will go high.

TxDATA [Transmit Data]

This pin transmits serial data.

TxRDY [Transmitter Ready]

This signal tells the CPU that the transmit data buffer in the CXQ71051 is empty; i.e., that new transmit data can be written. This signal is masked by the TxEN bit of the command byte and by the $\overline{\text{CTS}}$ input. It can be used as an interrupt signal to request data from the CPU.

The status of TxRDY can be sensed by reading bit 0 of the status byte. This allows the CXQ71051 to be polled. Note that TxRDY of the status byte is not masked by $\overline{\text{CTS}}$ or TxEN.

TxRDY is cleared to 0 by the falling edge of $\overline{\text{WR}}$ when the CPU writes transmit data to the CXQ71051. Data in the transmit data buffer that has not been sent is destroyed if transmit data is written while TxRDY=0.

TxEMP [Transmitter Empty]

The CXQ71051 reduces CPU overhead by using a double buffer; the transmit data buffer (second buffer) and the transmit buffer (first buffer) in the transmitter. When the CPU writes transmit data to the transmit data buffer (second buffer), the CXQ71051 transfers the contents of the second buffer to the first buffer, after transmitting the contents of the first buffer. This empties the second buffer and TxRDY is set to 1. TxEMP goes high when the second buffer is empty and the contents of the first buffer are sent. Thus, TxEMP=1 shows that both buffers are empty. In half-duplex operation, the CPU can know the timing needed to change from sending to receiving by testing TxEMP=1.

If TxEMP=1 occurs in async mode, the TxDATA pin goes high. When the CPU writes transmit data, TxEMP goes low and data transmission is resumed.

If TxEMP=1 occurs in sync mode, the CXQ71051 loads SYNC characters from the SYNC character register and sends them through the TxDATA pin. When the CPU writes a new transmit data to the CXQ71051, the data transmission will be resumed after the current (one or two) SYNC character(s) transmission and TxEMP will go low.

TxCLK [Transmitter Clock]

This pin is the reference clock input that determines the transmission rate. Data is transmitted at the same rate as TxCLK in sync mode. In async mode, TxCLK is set to 1, 16, or 64 times the transmission rate. Serial data from TxDATA is sent out on the falling edge of TxCLK.

For example, a rate of 19200 baud in sync mode means that TxCLK is 19.2 kHz. A rate of 2400 baud in async mode can represent:

$$\underline{\text{TxCLK}} = 2.4 \text{ kHz in } \times 1 \text{ clock mode.}$$

$$\underline{\text{TxCLK}} = 38.4 \text{ kHz in } \times 16 \text{ clock mode.}$$

$$\underline{\text{TxCLK}} = 153.6 \text{ kHz in } \times 64 \text{ clock mode.}$$

RxDATA [Receive Data]

This pin receives serial data.

RxRDY [Receiver Ready]

This signal goes high when the CXQ71051 receives one character of data and transfers that data to the receive data buffer; i.e., when the received data can be read. This signal can be used as an interrupt signal for data read request to the CPU. The CPU can know the status of RxRDY by reading bit 1 of the status byte in a polling operation. RxRDY goes low when the CPU reads the received data.

If the CPU fails to read a received data prior to the next single character being received and transferred to the receive buffer, overrun error occurs, and the OVE status bit is set. The previous data in the receive data buffer is overwritten by the newly transferred data and lost.

RxRDY remains low in the receive disable state. This state is realized by resetting the RxEN bit to 0 through the command byte. After RxEN is set to 1 (making receiving possible), RxRDY goes high whenever a new character is received and transferred to the receive data buffer.

SYNC/BRK [Synchronization/Break]

SYNC detects synchronization characters in sync mode. The SYNC mode byte selects internal or external SYNC detection. The SYNC pin becomes an output when internal synchronization is set, and an input when external synchronization is set.

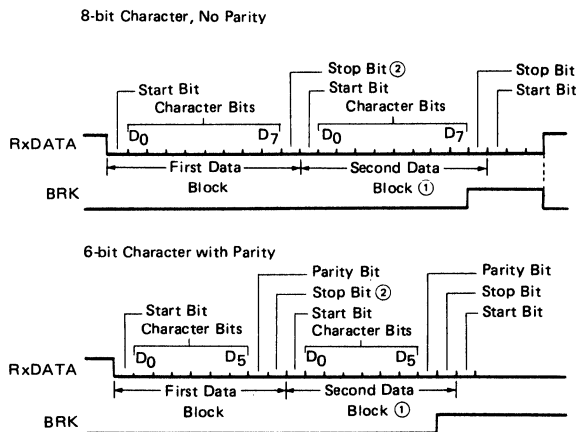
SYNC goes high when the CXQ71051 detects a SYNC character in internal synchronization. When two SYNC characters are used, SYNC goes high when the last bit of the two consecutive SYNC characters is detected. The CPU can read the status of the SYNC signal in bit 6 of the status byte. Both the SYNC pin and status are cleared to 0 by a read status operation.

In external synchronization, when the external circuit detects synchronization, a high level of at least one period of RxCLK is input to the SYNC pin. When the CXQ71051 detects the high level, it begins to receive data, starting at the rising edge of the next RxCLK. The high level input may be removed once synchronization is established.

BRK is used only in async mode and shows the detection of a break state. BRK goes high when the RxDATA input remains low through two successive character bit lengths (including the start, stop, and parity bits). As with SYNC, the CPU can read the status of BRK in bit 6 of the status byte. BRK is not cleared by the read operation. BRK signal is cleared when the RxDATA pin returns to high level, or when the CXQ71051 is reset by hardware or software. Figure 1 shows the break state and BRK signal.

Upon reset the SYNC/BRK pin becomes an output and goes low regardless of the previous mode.

Figure 1. Break Status and Break Signal



- Notes:**
1. When RxDATA goes high in the stop bit position of the second data block, the BRK signal level may [but does not always] become high for a maximum of one bit time.
 2. Only one bit of the stop bit is checked.

RxCLK [Receiver Clock]

This pin is a reference clock input that controls the receive data rate. In sync mode, the receiving rate is the same as $\overline{\text{RxCLK}}$. In async mode, $\overline{\text{RxCLK}}$ can be 1, 16, or 64 times the receive rate. Serial data from RxDATA is input on the rising edge of $\overline{\text{RxCLK}}$.

V_{DD} [Power]

+5V power supply.

GND [Ground]

Ground.

Absolute Maximum Ratings $T_a = +25^\circ\text{C}$

Parameter	Symbol	Rating Value	Unit
Power supply voltage	V_{DD}	-0.5 to +7.0	V
Input voltage	V_I	-0.5 to $V_{DD}+0.3$	V
Output voltage	V_O	-0.5 to $V_{DD}+0.3$	V
Power dissipation	P_{DMAX}	1.0	W
Operating temperature	T_{opr}	-40 to +85	$^\circ\text{C}$
Storage temperature	T_{stg}	-65 to +150	$^\circ\text{C}$

Comment: Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of this specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics $T_a = -40$ to $+85^\circ\text{C}$, $V_{DD} = +5\text{V} \pm 10\%$

Parameter	Symbol	Limits		Unit	Test Conditions
		Min.	Max.		
Input voltage high	V_{IH}	2.2	$V_{DD}+0.3$	V	
Input voltage low	V_{IL}	-0.5	0.8	V	
Output voltage high	V_{OH}	$0.7 \times V_{DD}$		V	$I_{OH} = -400 \mu\text{A}$
Output voltage low	V_{OL}		0.4	V	$I_{OL} = 2.5 \text{ mA}$
Input leakage current high	I_{LIH}		10	μA	$V_I = V_{DD}$
Input leakage current low	I_{LIL}		-10	μA	$V_I = 0\text{V}$
Output leakage current high	I_{LOH}		10	μA	$V_O = V_{DD}$
Output leakage current low	I_{LOL}		-10	μA	$V_O = 0\text{V}$
Supply current	I_{DD1}		10	mA	8 MHz operation
	I_{DD2}		50	μA	Stand-by mode

Capacitance $T_a = +25^\circ\text{C}$, $V_{DD} = 0\text{V}$

Parameter	Symbol	Limits		Unit	Test Conditions
		Min.	Max.		
Input capacitance	C_i		10	pF	$f_c = 1 \text{ MHz}$ Unmeasured pins returned to 0V
I/O capacitance	C_{IO}		20	pF	

AC Characteristics

Parameter	Symbol	Limits		Unit	Test Conditions
		Min.	Max.		
Read Cycle					
Address (\overline{CS} , C/\overline{D}) set-up to $\overline{RD} \downarrow$	tsAR	0		ns	
Address (\overline{CS} , C/\overline{D}) hold from $\overline{RD} \uparrow$	thRA	0		ns	
\overline{RD} width low	trRL	150		ns	
Data delay from $\overline{RD} \downarrow$	tDRD		120	ns	Cl=150 pF
Data float from $\overline{RD} \uparrow$	tFRD	10	80	ns	
Port (\overline{DSR} , \overline{CTS}) set-up to $\overline{RD} \downarrow$	tSPR	20		tCYK	
Write Cycle					
Address (\overline{CS} , C/\overline{D}) set-up to $\overline{WR} \downarrow$	tsAW	0		ns	
Address (\overline{CS} , C/\overline{D}) hold from $\overline{WR} \uparrow$	thWA	0		ns	
\overline{WR} width low	twWL	150		ns	
Data set-up to $\overline{WR} \uparrow$	tSDW	80		ns	
Data hold from $\overline{WR} \uparrow$	thWD	0		tCYK	
Port (\overline{DTR} , \overline{RTS}), TxEN delay from $\overline{WR} \uparrow$	tdWP		8	tCYK	
Write recovery time	trv	6		tCYK	Mode initialize
		8		tCYK	Async mode
		16		tCYK	Sync mode

Serial Transfer Timing						
CLK cycle time	t _{cyk}	125	DC	ns		
CLK pulse width high	t _{kkh}	50		ns		
CLK pulse width low	t _{kkL}	35		ns		
CLK rise time	t _{kR}	5	20	ns		
CLK fall time	t _{kF}	5	20	ns		
TxDATA delay from $\overline{\text{TxCLK}}$	t _{DTKTD}		0.5	μs		
Transmitter input clock pulse width low	$1 \times \text{BR}^1$	t _{TKTKL}	12		t _{cyk}	
	$16 \times, 64 \times \text{BR}$		1		t _{cyk}	
Transmitter input clock pulse width high	$1 \times \text{BR}$	t _{TKTKH}	15		t _{cyk}	
	$16 \times, 64 \times \text{BR}$		3		t _{cyk}	
Transmitter input clock frequency	$1 \times \text{BR}$	f _{rk} ²	DC	240	kHz	
	$16 \times \text{BR}$		DC	1536	kHz	
	$64 \times \text{BR}$		DC	1536	kHz	
Receiver input clock pulse width low	$1 \times \text{BR}$	t _{TRKRL}	12		t _{cyk}	
	$16 \times, 64 \times \text{BR}$		1		t _{cyk}	
Receiver input clock pulse width high	$1 \times \text{BR}$	t _{TRKRH}	15		t _{cyk}	
	$16 \times, 64 \times \text{BR}$		3		t _{cyk}	
Receiver input clock frequency	$1 \times \text{BR}$	f _{rk} ²	DC	240	kHz	
	$16 \times \text{BR}$		DC	1536	kHz	
	$64 \times \text{BR}$		DC	1536	kHz	
RxDATA set-up to sampling pulse	t _{SRDSP}	1		μs		
RxDATA hold from sampling pulse	t _{HSPRD}	1		μs		
TxEMP delay time	t _{DXEP}		20	t _{cyk}		
TxRDY delay time (TxRDY ↑)	t _{DTXR}		8	t _{cyk}		
TxRDY delay time (TxRDY ↓)	t _{DWTXR}		200	ns		
RxRDY delay time (RxRDY ↑)	t _{DRXR}		26	t _{cyk}		
RxRDY delay time (RxRDY ↓)	t _{DRRXR}		200	ns		
SYNC output delay time (for internal sync)	t _{DRKSY}		26	t _{cyk}		
SYNC input Set-up time (for external sync)	t _{SSYRK}	18		t _{cyk}		
RESET pulse width		6		t _{cyk}		

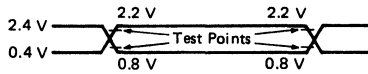
Notes: 1. BR=Baud Rate

2. $1 \times \text{BR}$: f_{rk} or f_{rk} ≤ 1/30 t_{cyk}; $16 \times, 64 \times \text{BR}$: f_{rk} or f_{rk} ≤ 1/4.5 t_{cyk}

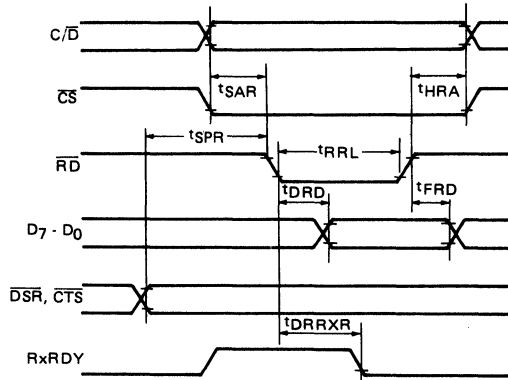
3. System CLK must be running during Reset operation

4. Status update can have a maximum delay of 28 t_{cyk} from the event affecting the status.

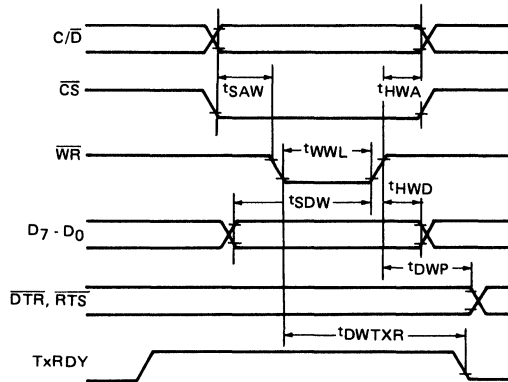
AC Test Input Waveforms:



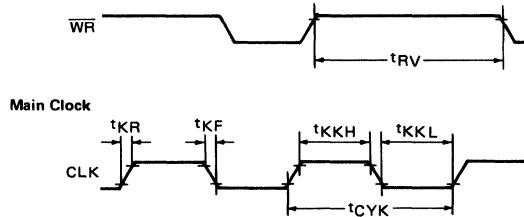
Read Data Cycle



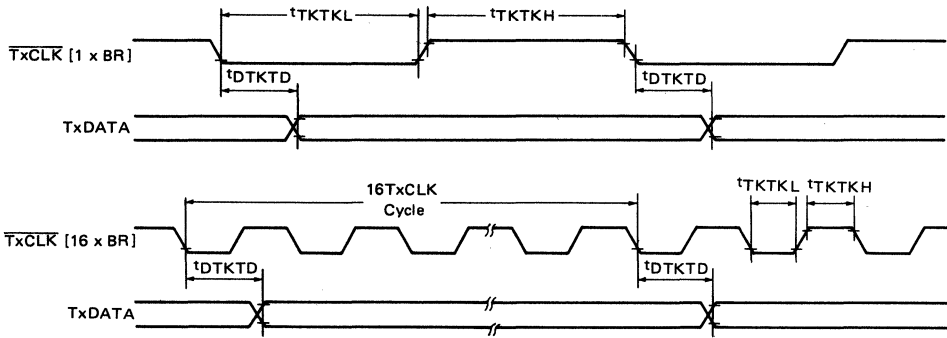
Write Data Cycle



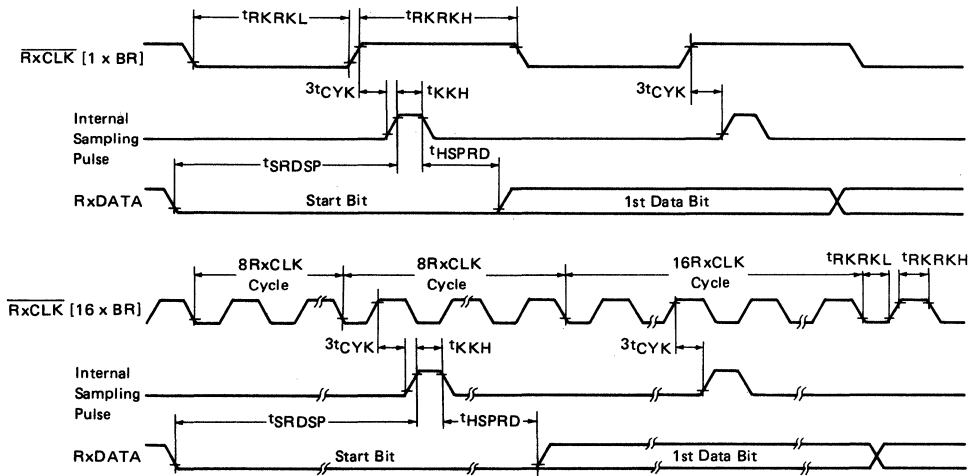
Write Recovery Time



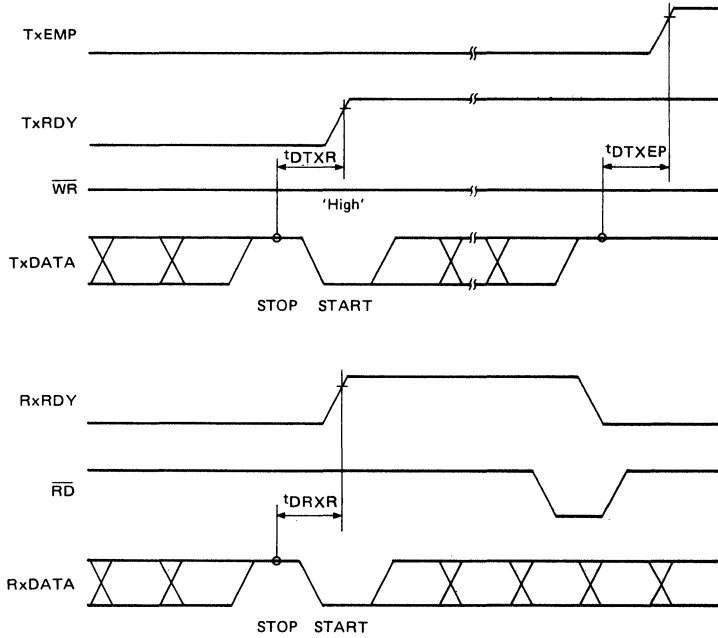
Transmitter Clock & TxDATA



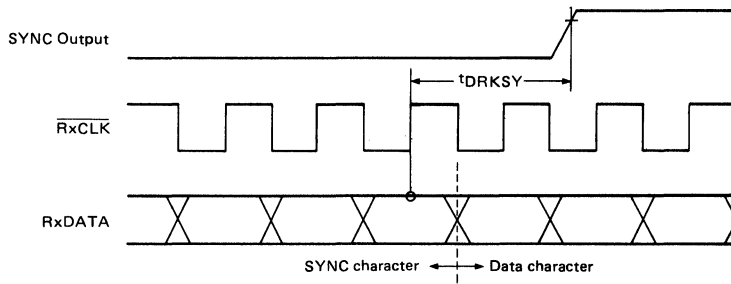
Receiver Clock & RxDATA Timing



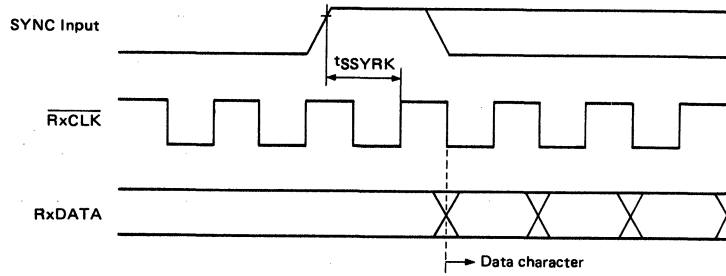
Flag Timing



Internal Sync



External Sync



Functions

The CXQ71051 is a CMOS serial interface unit that provides serial communications in microcomputer systems. The CPU handles the CXQ71051 as an ordinary I/O device.

The CXQ71051 can be operate in synchronous or asynchronous systems. In sync mode, the character bit length, number of sync characters, and sync detection mode must be defined. In async mode, the communication rate, character bit length, stop bit length, etc., must be defined. The parity bit may be designated in either mode.

The CXQ71051 converts parallel data received from the CPU into serial stream for transmission on the TxDATA pin, and converts serial input data into parallel data so that the CPU can read it (receiving operation).

The CPU can read the current status of the CXQ71051 and can process data after checking the status for transfer errors and CXQ71051 data buffer status.

The CXQ71051 is a low-power CMOS device and can be reset under hardware or software control to standby mode that consumes less power and removes the device from system operation. In this mode the previous operating mode is released and it waits for a mode byte to set the new mode. The CXQ71051 exits standby mode and shifts to designated operating mode when the CPU writes a mode byte to it.

Status Register

The status register allows the CPU to read the status of the CXQ71051 at any time except in standby mode. This register indicates the status of the errors and other conditions that require the CPU's attention.

Receive Data Buffer

When the receiver has converted the serial data input from the RxDATA pin into parallel data, the converted data is stored in the receive data buffer. The CPU can then read it. Data for one character entering the receive buffer is transferred to the receive data buffer and RxRDY becomes 1, requesting that the CPU read the data.

Transmit Data Buffer

The transmit data buffer holds the parallel data from the CPU that the transmitter will convert to serial data and output from TxDATA pin. When the CPU writes transmit data to the CXQ71051, the CXQ71051 stores data in the transmit data buffer. The transmit data buffer transfers the data to the transmitter, which sends the data from TxDATA pin.

Control Word Register

This register stores control word (mode byte, one or two SYNC characters, and command byte) which specifies the function of the CXQ71051. These control signals are then sent to the internal blocks.

Control Logic

The control logic sends control signals to the internal blocks and controls the operation of the CXQ71051 based on internal and external signals.

Synchronous Character Register

This register stores one or two SYNC characters used in sync mode. During transmission, the SYNC characters stored in this register will be output from the TxDATA pin when the CPU does not send a new character and TxEMP status is set. During receiving, synchronization will be achieved when the received characters and the SYNC characters stored in this register are equal.

Transmitter

The contents of the transmit data buffer are transferred to the transmitter, converted from parallel format to serial, and output from the TxDATA pin. The transmitter adds the appropriate characters or bits based on the mode.

Receiver

The receiver converts serial data input from the RxDATA pin into parallel data and transfers the parallel data to the receive data buffer, allowing the CPU to read it.

The receiver detects SYNC characters and checks parity bits in sync mode; and detects start and stop bits and checks parity bits in async mode.

In async mode, receiving does not begin (the start bit is not detected) until one effective stop bit (high level) is input to the RxDATA pin and Receive Enable is set (RxEN=1) after setting up the mode.

Modem Control

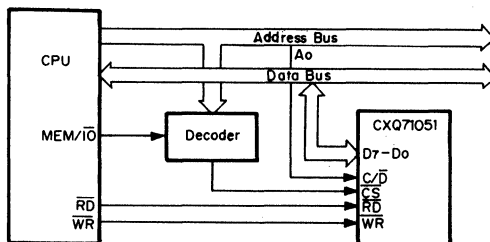
This block controls the $\overline{\text{CTS}}$, $\overline{\text{RTS}}$, $\overline{\text{DSR}}$, and $\overline{\text{DTR}}$ modem interface pins. The $\overline{\text{RTS}}$, $\overline{\text{DSR}}$, and $\overline{\text{DTR}}$ pins can also be used as general-purpose I/O pins.

Connecting the CXQ71051 to the System

The CPU uses the CXQ71051 as an I/O device by allocating two I/O addresses, assigned by C/\bar{D} . One I/O address is allocated when C/\bar{D} is low and becomes a port to transfer and receive data. The other I/O address is allocated when C/\bar{D} is high and becomes a port to write mode and command bytes and read status. Generally, the least significant bit (A_0) of the CPU address bus is connected to C/\bar{D} to get a continuous I/O address. This is shown in Figure 2.

Pins TxRDY and RxRDY are connected to the interrupt pins of the CPU or the interrupt controller when interrupts are used.

Figure 2. System Connection

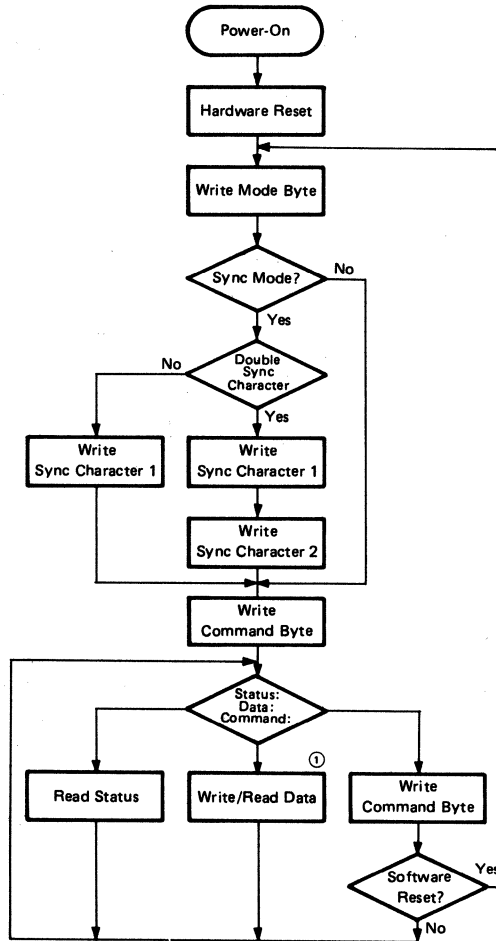


Operation

A hardware reset (a "high" pulse on the RESET pin) after power-on forces the CXQ71051 into the standby mode and it waits for a mode byte. The mode byte defines the communication protocol. In async mode, the CXQ71051 is ready for a command byte after the mode byte. In sync mode, the CXQ71051 waits for one or two SYNC characters to be written following the mode byte with $C/\bar{D}=1$. A command byte may be loaded after the SYNC characters are written. This operation sequence is shown in Figure 3.

In both modes, it is possible to write transmit data, read receive data, read status, and write another command byte once the first command byte has been written. When the command byte performs a software reset the CXQ71051 performs a reset operation, enters standby mode, and returns to a state where it waits for a mode byte.

Figure 3. CXQ71051 Operating Procedure

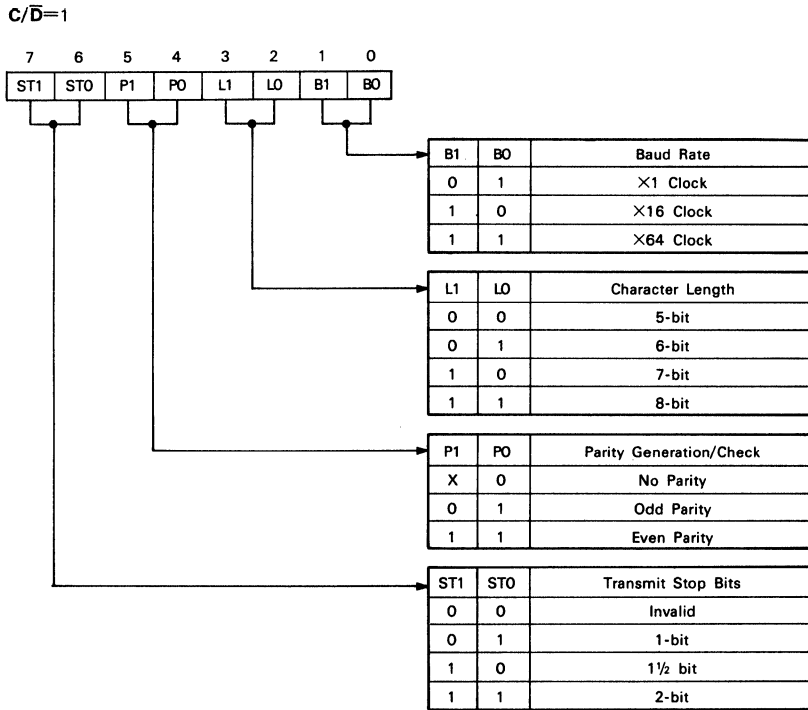


Note: 1 This is done with $C/\bar{D} = 0$. Others are operated with $C/\bar{D} = 1$.

Designating the Mode

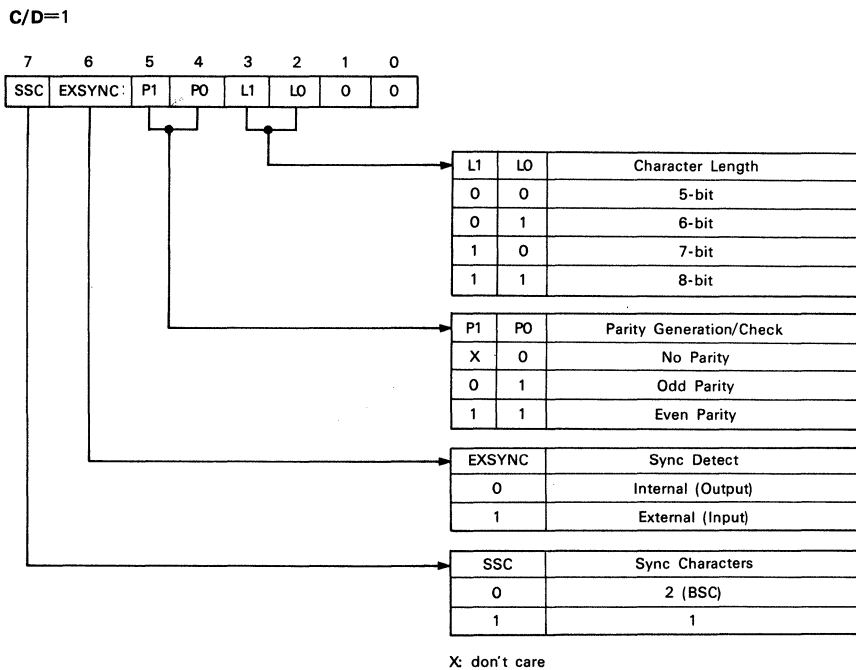
When the CXQ71051 is in standby mode following a reset operation, writing a mode byte exits the standby mode. Figure 4 shows the mode byte format for designating async mode. Figure 5 shows the mode byte format for designating sync mode. Bits 0 and 1 must be 00 to designate sync mode. Async mode is designated in all other cases.

Figure 4. Mode Byte Format for Asynchronous Mode



X: don't care

Figure 5. Mode byte Format for Synchronous Mode



The P1, P0 and L1, L0 bits are common to both modes. Bits P1 and P0 control the parity generation (during transmission) and checking (during receive) functions. These parity bit functions do not operate when P0=0. When P1P0=01, the CXQ71051 generates and checks odd parity. When P1P0=11, it generates and checks even parity.

Bits L1 and L0 define the number of bits per character: n. Additional bits such as parity bits are not included in this number. Given n bits, the CXQ71051 receives the lower n bits of the 8-bit data written by the CPU. The upper bits (8-n) of data that the CPU reads from the CXQ71051 are cleared to zero.

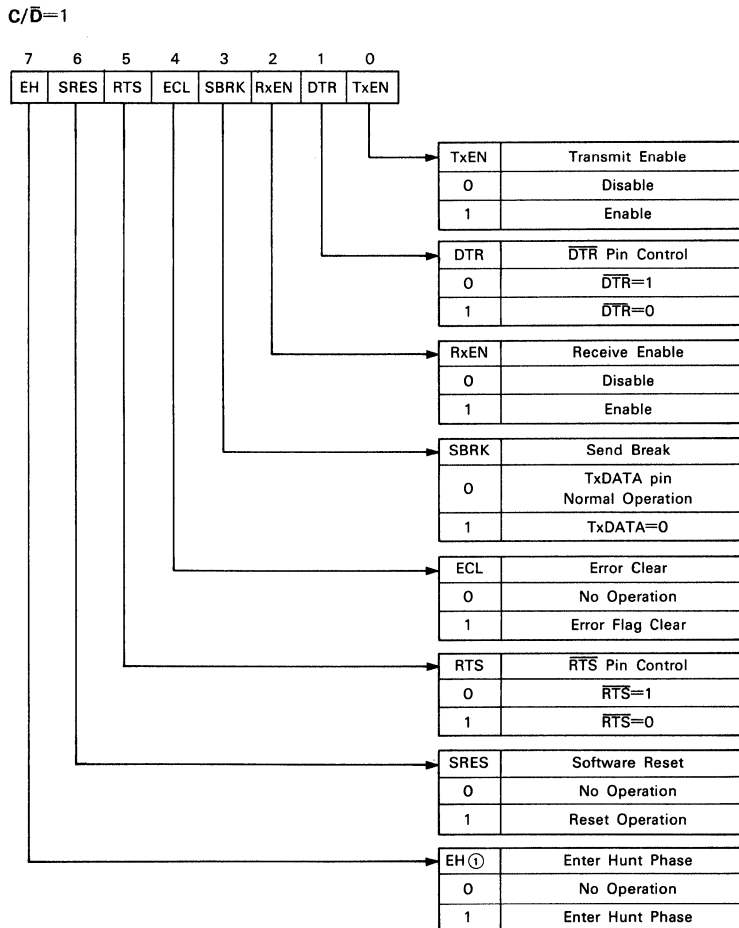
The ST1, ST0 and B1, B0 bits occur in async mode. The ST1 and ST0 bits determine the number of stop bits added by the CXQ71051 during transmission. The B1 and B0 bits determine the relationship between the baud rates for sending and receiving, and the clocks $\overline{\text{TxClock}}$ and $\overline{\text{RxClock}}$. B1 and B0 select the clock rates of 1, 16, or 64 times the required baud rate. The multiplier rate of 1 is not normally used in async mode. Note that the data and clock must be synchronized on the transmitter and receiver if $\times 1$ clock is used.

The SSC and EXSYNC bits are used in sync mode. The SSC bit determines the number of SYNC characters. SSC=1 designates one SYNC character. SSC=0 designates two SYNC characters. The number of SYNC characters determined by the SSC bit are written to the CXQ71051 right after writing the mode byte. The EXSYNC bit determines whether sync detection during receiving operations is internal or external. EXSYNC=1 selects external sync detection and EXSYNC=0 selects internal sync detection.

Commands and Operations

Commands are issued by command bytes that control the sending and receiving operations of the CXQ71051. A command byte is loaded following the mode byte in async mode; while in sync mode SYNC characters must be inserted prior to the command byte. The CPU must set $C/\bar{D}=1$. Figure 6 shows the command byte format.

Figure 6. Command Byte Format



Note: 1. The EH bit is used only in SYNC mode.

Bit EH is set to 1 when entering hunt phase to achieve synchronization in sync mode. Bit RxEN should also be set to 1 at the same time. Data receiving begins when the CXQ71051 has detected SYNC character(s) and achieved synchronization, thus ending hunt phase.

When Bit SRES is set to 1, a software reset is executed, and the CXQ71051 returns to the standby mode and waits for a mode byte.

Bits RTS controls the $\overline{\text{RTS}}$ output pin. $\overline{\text{RTS}}$ is low when the RTS bit=1, and is high when the RTS bit=0.

Setting bit ECL to 1 clears the error flags (PE, OVE, and FE) and the status. Set ECL to 1, when entering hunt phase (EH=1) or enabling the receiver (RxEN=1).

Bit SBRK transmits break characters. When SBRK=1, the data currently being transmitted is destroyed and the TxDATA pin goes low. Set SBRK=0 to release a break. Break also works when TxEN=0 (send disable).

Bit RxEN enables and disables the receiver. RxEN=1 enables the receiver and RxEN=0 disables the receiver. Synchronization is lost if RxEN=0 in sync mode.

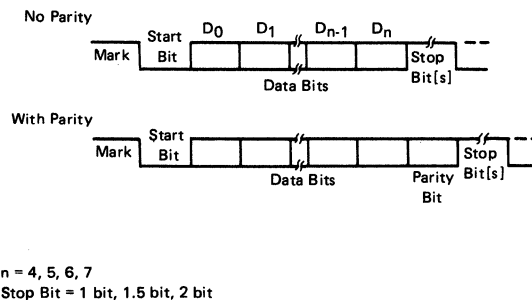
Bit DTR controls the $\overline{\text{DTR}}$ output pin. $\overline{\text{DTR}}$ is low when the DTR bit=1 and is high when the DTR bit=0.

The TxEN bit enables and disables the transmitter. TxEN=1 enables the transmitter and TxEN=0 disables the transmitter. When TxEN=0, the transmission will stop and the TxDATA pin go high (mark status) after all the currently written data is shifted out.

Transmission in Asynchronous Mode

The TxDATA pin is typically high (mark) when data is not being sent. When the CPU writes transmit data to the CXQ71051, the CXQ71051 transfers the transmit data from the transmit data buffer to the transmit buffer and sends the data from the TxDATA pin after adding one start bit (low level) and a programmed stop bit. If parity is used, a parity bit is inserted between the character and the stop bits. Figure 7 shows the data format for async mode. Serial data is shifted out on the falling edge of $\overline{\text{TxCLK}}$ at a rate equal to 1/1, 1/16, or 1/64 that of $\overline{\text{TxCLK}}$.

Figure 7. Asynchronous Mode Data Format



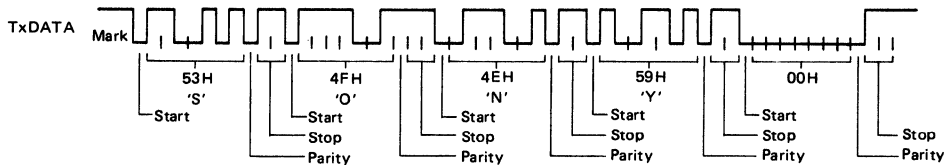
When bit SBRK is set to 1, the TxDATA pin will go low (break status), regardless of whether data is being transmitted. The following is an example of a program to transmit data in async mode. Figure 8 shows the output from the TxDATA pin.

```

ASYNTAX : CALL    ASYNMOD                ; set Async mode
           MOV    AL, 00010001B          ; Command: clear error flag, transmit enable
           OUT    PCTRL, AL
TXSTART : MOV    BW, OFFSET TXDADR        ; Transmit data area
           IN     AL, PCTRL
           TEST1  AL, 0                    ; Read status
           BZ     TXSTART                  ; Wait until TxRDY=1
           MOV    AL, [BW]                 ; Write transmit data
           OUT    PDATA, AL
           INC    BW                        ; Set next data address
           CMP    AL, 0
           BNE    TXSTART                  ; End if data=0
           RET
TXDADR   : DB     "SONY"                  ; Transmit data 53H, 4FH, 4EH, 59H
           DB     0                        ; Terminal data 00
ASYNMOD : MOV    AL, 0                    ; Write control bytes three times
           OUT    PCTRL, AL                ; with 00H to unconditionally
           OUT    PCTRL, AL                ; accept the new command byte
           OUT    PCTRL, AL
           MOV    AL, 01000000B           ; Software reset
           OUT    PCTRL, AL
           MOV    AL, 11111010B           ; Write mode byte
           OUT    PCTRL, AL                ; Stop bit=2 bits, even parity
           RET                               ; 7 bits/character, X16 clock

```

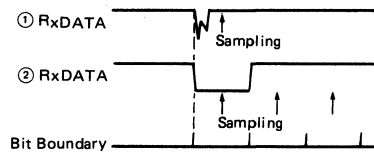
Figure 8. TxDATA Pin Output



Receiving in Asynchronous Mode

The RxDATA pin is normally high when data is not being received, as shown in Figure 9. When a low level signal enters it, the CXQ71051 detects the falling edge and presumes that it is a start bit.

Figure 9. Start Bit Detection



- Notes: 1. Start bit is not recognized because R x Data is high at the sampling time.
 2. Start bit is recognized because R x Data is low at the sampling time.

The CXQ71051 samples the level of the RxDATA input (only when $\times 16$ or $\times 64$ clock is selected) in a position 1/2 bit time behind the falling edge of the RxDATA input in order to ascertain a valid start bit. It is considered a valid start bit if a low level is detected at that time. If a low level is not detected, it is not regarded as a start bit and the CXQ71051 continues sampling for a new valid start bit.

When a start bit is detected, the sampling points of the data bits, parity bit (if used), and stop bits are decided by a bit counter. The sampling is performed on the rising edge of $\overline{\text{RxCLK}}$ at a rate equal to 1/1, 1/16, or 1/64 that of $\overline{\text{RxCLK}}$.

Data for one character entering the receive buffer is transferred to the receive data buffer and forces $\text{RxRDY}=1$, requesting that the CPU read the data. When the CPU reads the data, RxRDY returns to 0.

When a valid stop bit is detected, the CXQ71051 waits for the start bit of the next data. If a low level is detected in the stop bit, a framing error flag is set; however, the receive operation continues as if the correct high level had been detected. A parity error flag is set if a parity error is detected. An overrun error flag is set if the CPU has not read the data in time, and the next received data is transferred to the receive data buffer, overwriting the previous data. The CXQ71051's transmission and receive operations are not affected by these errors.

If a low level is input to the RxDATA pin for more than two data blocks during the receive operation, the CXQ71051 considers it a break state and the SYNC/BRK (pin and status) becomes 1.

Once async mode has been assigned following a reset and the receiver has been enabled, the start bit is not detected until a high level of more than one bit has been input to the RxDATA pin. The following is an example of a program to receive the data sent in the async transmission example in Figure 8. Note that the frequencies of $\overline{\text{TxCLK}}$ on the transmitter and $\overline{\text{RxCLK}}$ on the receiver are equal.

```

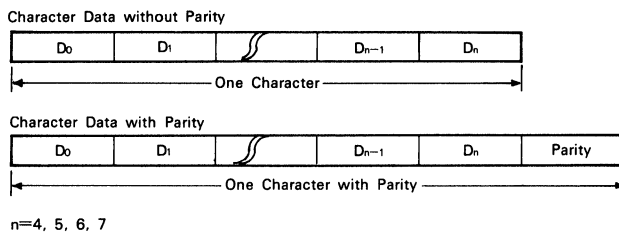
ASYNRX : CALL    ASYNMOD           ; Set Async mode
          MOV     AL, 00010100B     ; Command: clear error flag, receive enable
          OUT    PCTRL, AL
          MOV     BW, OFFSET RXDADR ; Receive data area
RXSTART : IN     AL, PCTRL          ; Read status
          TEST1  AL, 1
          BZ     RXSTART            ; Wait until RxRDY=1
          IN     AL, PDATA          ; Read and store the receive data
          MOV    [BW], AL           ; Store
          INC    BW                 ; Set next store address
          CMP    AL, 0              ; End if data=0
          BNE   RXSTART
          RET
RXDADR   DB     256 DUP (?)        ; Reserve receive data area

```

Transmission in Synchronous Mode

Following the establishment of sync mode and the enabling of the transmitter, the TxDATA pin stays high until the CPU writes the first character (normally, SYNC characters). When data is written, the TxDATA pin will send one bit for each falling edge of $\overline{\text{TxCLK}}$ if the $\overline{\text{CTS}}$ pin is low. Unlike async mode, start and stop bits are not used. However, a parity bit may be set. Figure 10 shows these data formats.

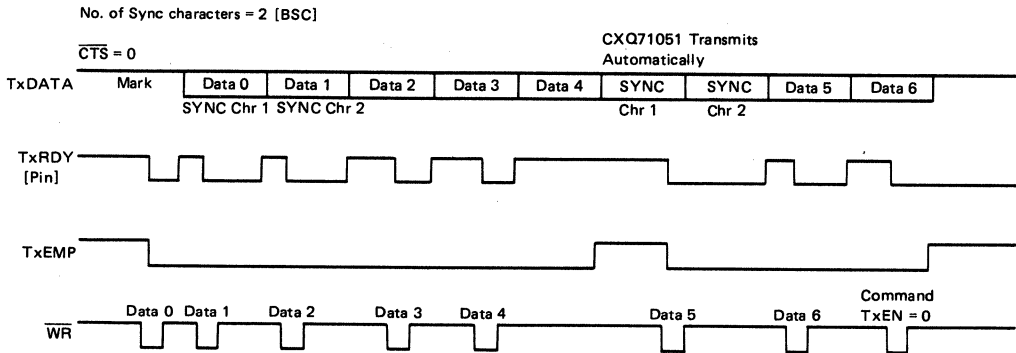
Figure 10. Synchronous Mode Data Format



Once transmission has begun, the CPU must write data to the CXQ71051 at the $\overline{\text{TxCLK}}$ rate. If TxEMP goes high because of a delay in writing by the CPU, the CXQ71051 inserts SYNC characters until the CPU writes a data. TxEMP goes low when a data is written, and the data is sent as soon as the transmission of the SYNC characters stops.

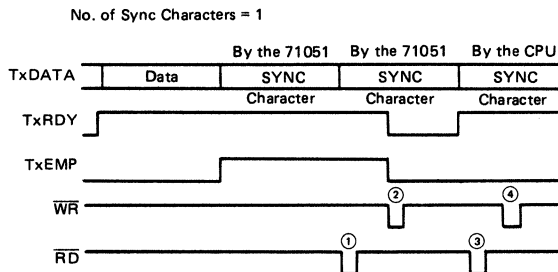
Automatic transmission of SYNC characters begins after the CPU writes a data. SYNC characters are not automatically sent merely by enabling the transmitter. Figure 11 shows these timing sequences.

Figure 11. Synchronous Mode Transmission Timing



If a command byte is written to the CXQ71051 while SYNC characters are being sent and TxEMP=1, the CXQ71051 may interpret the command as a data byte and transmit it as a data. If a command must be written under these conditions, the CPU should write a SYNC character to the CXQ71051 and send the command while the SYNC character is being transmitted. This is shown in Figure 12.

Figure 12. Issuing a Command During SYNC Character Transmission



- Notes:
1. Confirm the automatic transmission of the SYNC character by the TxEMP status.
 2. Write SYNC character data.
 3. Confirm the beginning of SYNC character transmission by the CPU by reading the status.
 4. Write command.

The following is an example of a program for transmitting in sync mode.

```

SYNTAX : CALL    SYNMOD                ; Set sync mode
          MOV     AL, 00010001B         ; Command: clear error flags, transmit enable
          OUT    PCTRL, AL
          MOV     BW, OFFSET TXDADR     ; Transmit data area
TXLEN :   IN     AL, PCTRL
          TEST1  AL, 0
          BZ     TXLEN
          MOV     AL, LDLEN             ; Transmit the length byte
          OUT    PDATA, AL
          MOV     CL, AL                ; Set number of transmit data to counter
          MOV     CH, 0
TXDATA : IN     AL, PCTRL
          TEST1  AL, 0
          BZ     TXDATA
          MOV     AL, [BW]             ; Transmit the number of
          OUT    PDATA, AL            ; bytes specified by LDLEN.
          INC    BW
          DBNZ   TXDATA
          MOV     AL, 00010000B        ; Command: clear error flags, transmit enable
          OUT    PCTRL, AL
          RET

SYNC1    DB     ?                    ; SYNC character 1
SYNC2    DB     ?                    ; SYNC character 2
LDLEN    DB     ?                    ; Set number of data to be
                                           ; transmitted (1 to 255)
TXDADR   DB     255 DUP (?)          ; Data to be transmitted
SYNMOD : MOV     AL, 0                ; Write control bytes
          OUT    PCTRL, AL            ; three times with 00H to
          OUT    PCTRL, AL            ; unconditionally accept
          OUT    PCTRL, AL            ; the new command byte
          MOV     AL, 01000000B        ; Software reset
          OUT    PCTRL, AL
          MOV     AL, 00111100B        ; Write mode byte: 2 SYNC characters,
          OUT    PCTRL, AL            ; internal sync detect,
                                           ; even parity, 8 bits/character
          MOV     AL, SYNC1            ; Write SYNC characters
          OUT    PCTRL, AL
          MOV     AL, SYNC2
          OUT    PCTRL, AL
          RET

```


When the receive buffer and the SYNC character coincide, the CXQ71051 ends hunt phase and SYNC becomes 1 in the center of the last SYNC bit to indicate that it has achieved synchronization. If parity exists, SYNC becomes 1 in the center of the parity bit. Data receiving starts from the following bit.

In external sync detection, synchronization is achieved by setting the SYNC pin high from an external circuit for at least one cycle of RxCLK. Hunt phase ends, and data receiving can start. At this time, the SYNC status bit becomes 1, and returns to 0 when the status is read. The SYNC status bit is set to 1 when the SYNC pin has a positive-going input followed by a high level of more than one cycle of RxCLK, even after synchronization is achieved.

If synchronization is lost, the CXQ71051 can regain it anytime by issuing an enter hunt phase command.

After synchronization, the SYNC character is compared with each data character regardless of whether internal or external synchronization is used. When the characters coincide, SYNC becomes 1, indicating that a SYNC character has been received. SYNC (SYNC status bit only in external detection) becomes 0 when the status is read.

Overrun and parity errors are checked the same as in async mode, affecting only the status flag. Parity checking is not performed in the hunt phase. The following is an example of a program that receives the data sent by the previous sync transmit program example. Note that the frequencies of TxCLK on the transmitter and RxCLK on the receiver are the same.

```

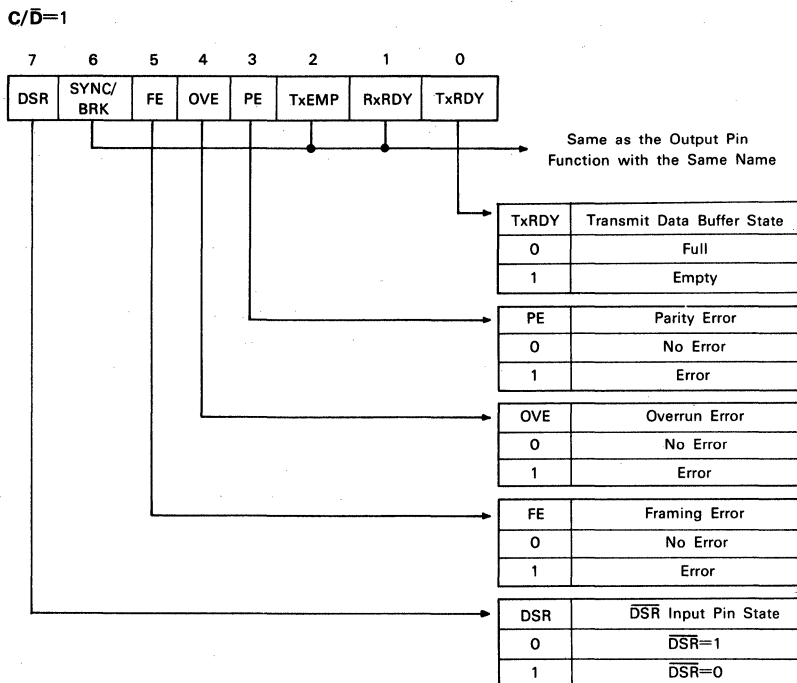
SYNRX :  CALL  SYNMOD           ; Set sync mode
         MOV   AL, 10010100B    ; Command: enter hunt phase,
         OUT  PCTRL, AL        ; clear error flags, receive enable
         MOV   BW, OFFSET RXDADR ; Set receive data store address
RXLEN  :  IN   AL, PCTRL
         TEST1 AL, 1
         BZ   RXLEN
         IN   AL, PDATA         ; Receive the number of receive data
         MOV  STLEN, AL        ; Set the number of
         MOV  CL, AL          ; receive data to both
         MOV  CH, 0           ; variable and counter
RXDATA :  IN   AL, PCTRL
         TEST1 AL, 1
         BZ   RXDATA
         IN   AL, PDATA         ; Receive and store the number of data bytes
         MOV  [BW], AL        ; by the counter
         INC  BW
         DBNZ RXDATA
         MOV  AL, 00000000B    ; Command: receive disable
         OUT  PCTRL, AL
         RET
STLEN  DB  ?                 ; Set number of receive data
RXDATR DB  256 DUP (0)      ; Reserve receive data area

```

Reading the Status Register

The CPU can read the status of the CXQ71051 at any time except when the CXQ71051 is in standby mode. Status can be read after setting $C/\bar{D}=1$ and $\bar{RD}=0$. Status is not updated while being read. Status updating is delayed at least 28 clock periods after an event that affects the status. Figure 14 shows the format of the status register.

Figure 14. Status Register Format

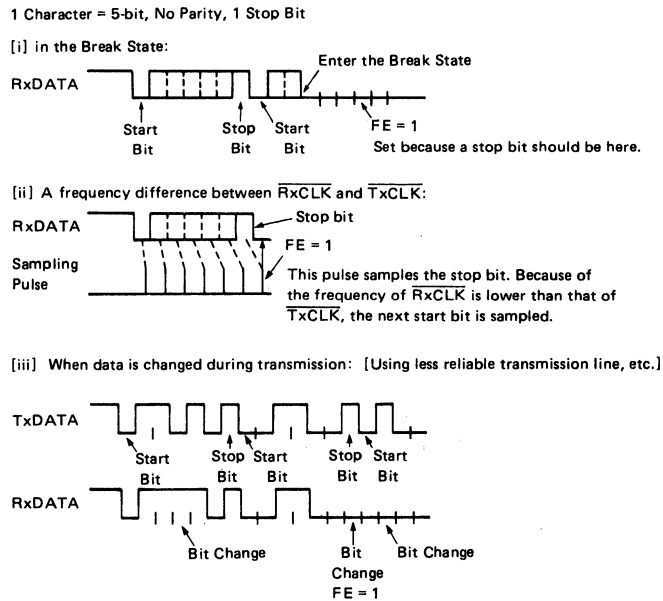


The TxEMP and RxRDY bits have the same meaning as the pins of the same name. The SYNC/BRK bit generally has the same meaning as the SYNC/BRK pin. In external synchronization mode, the status of the bit does not always coincide with the pin. In this case, the SYNC pin becomes an input and the status bit goes to 1 when a rising edge is detected at the input. The status bit remains at 1 until it is read, even after the input level at the SYNC pin goes low. The status bit becomes 1 when a SYNC character is input on the RxDATA input, even when the pin is low.

The DSR bit shows the status of the \bar{DSR} input pin. The status bit is 1 when the \bar{DSR} pin is low.

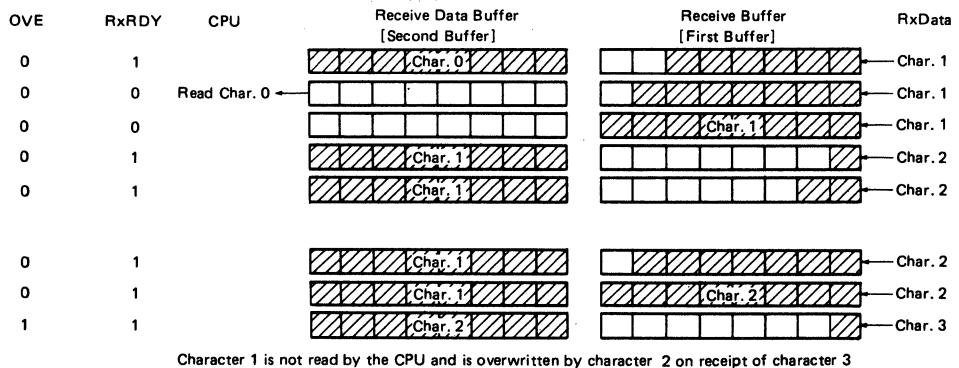
The FE bit (framing error) becomes 1 when more than one valid stop bit is not detected at the end of each data block during asynchronous receiving. Figure 15 shows how a framing error can happen.

Figure 15. Framing Error



The OVE bit (overrun error) becomes 1 when the CPU has not read the old data and the new data byte is received. In this case, the old data byte received is overwritten and lost in the receive data buffer. Figure 16 shows how an overrun error can happen.

Figure 16. Overrun Error



The PE bit (parity error) becomes 1 when a parity error occurs in a receive state.

Framing, Overrun, and parity errors do not disable the CXQ71051's operations. All three error flags are cleared to 0 by a command byte that sets the ECL bit to 1.

The TxRDY bit becomes 1 when the transmit data buffer is empty. The TxRDY output pin becomes 1 when the transmit data buffer is empty, the $\overline{\text{CTS}}$ pin is low, and TxEN=1. The TxRDY status bit and the TxRDY pin do not always have the same status.

$\text{TxRDY bit} = (\text{Transmit Data Buffer is empty})$

$\text{TxRDY pin} = (\text{Transmit Data Buffer is empty}) \cdot (\overline{\text{CTS}}=0) \cdot (\text{TxEN}=1)$

Standby Mode

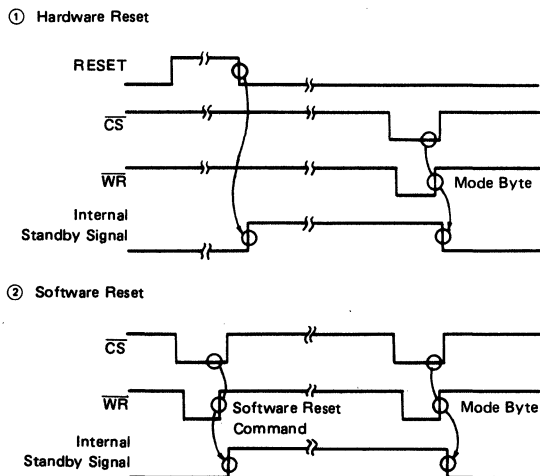
The CXQ71051 is a low-power CMOS device. In standby mode, it disables the external input clocks (CLK, TxCLK, and RxCLK) to the inside circuitry, thereby consuming less power.

A hardware reset is one way to enter standby mode. The input of a high level to the RESET pin causes the CXQ71051 to enter standby mode on the falling edge of the high level. A software reset command is the other way to enter standby mode. The only way to take the CXQ71051 out of standby mode is to write a mode byte.

In standby mode, the TxRDY, TxEMP, RxRDY, and SYNC/BRK pins are at low level and the TxDATA, $\overline{\text{DTS}}$, and $\overline{\text{RTS}}$ pins are at high level.

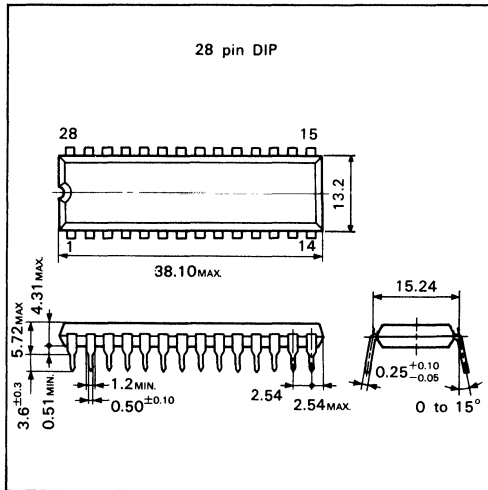
Figure 17 shows the timing for standby mode. While the internal standby signal is high, the external clocks to the CXQ71051 are ignored. If data ($\text{C}/\overline{\text{D}}=0$) is written to the CXQ71051 in standby mode, the reset operations are undefined and unpredictable operation may occur.

Figure 17. Standby Mode Timing



Package Outline

Unit: mm



Programmable Timer/Counter

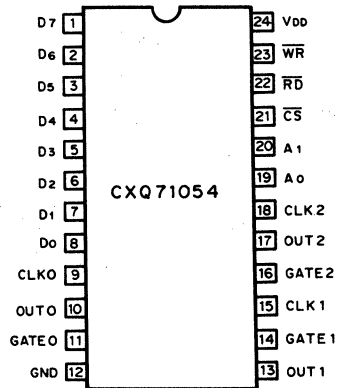
Description

The CXQ71054 is a CMOS high-performance programmable counter for microcomputer system timing control. It has three independent 16-bit counters, each capable of handling clock inputs up to 8 MHz. The programmer can use the CXQ71054 for timing applications to match his requirements, and can program the counters for the desired time delays. This eliminates the need for software timing loops.

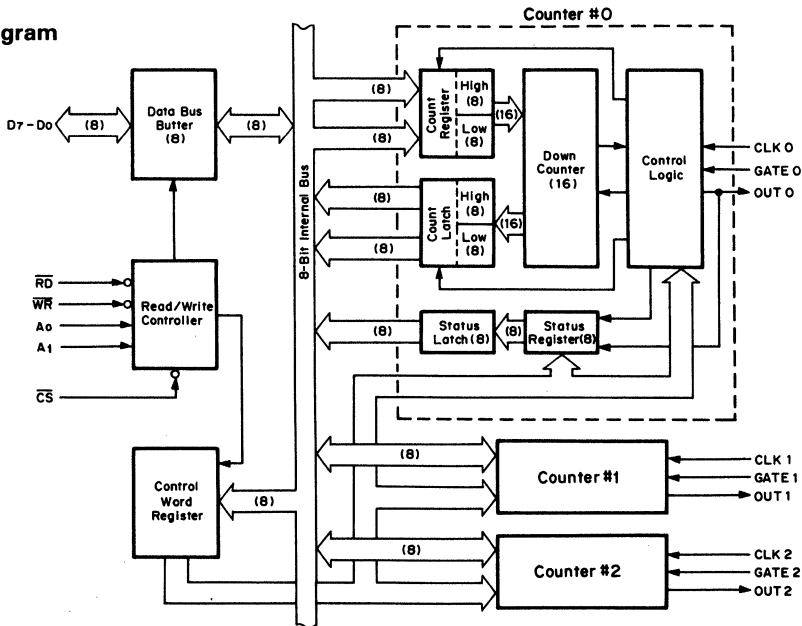
Features

- Three independent 16-bit counters
- Six programmable counter modes
- Binary or BCD count
- Multiple latch command
- Clock rate: DC (standby mode) to 8 MHz
- Low power standby mode
- CMOS technology
- +5V single power supply
- 24-pin plastic DIP (600 mil)
- NEC μ PD71054 compatible

Pin Configuration (Top View)



Block Diagram



Note: The internal architecture of counters #1 and #2 is the same as counter #0.

© 1985 NEC Electronics

Pin Identification

No.	Symbol	Direction	Function
1–8	D7–Do	In/Out	8-bit data bus
9	CLK0	In	Counter clock 0
10	OUT0	Out	Counter output 0
11	GATE0	In	Counter gate 0
12	GND		Ground
13	OUT1	Out	Counter output 1
14	GATE1	In	Counter gate 1
15	CLK1	In	Counter clock 1
16	GATE2	In	Counter gate 2
17	OUT2	Out	Counter output 2
18	CLK2	In	Counter clock 1
19, 20	A ₁ , A ₀	In	Address
21	\overline{CS}	In	Chip select
22	\overline{RD}	In	Read strobe
23	\overline{WR}	In	Write strobe
24	V _{DD}		Power supply

Pin Functions

D7–Do [Data Bus]

These pins are an 8-bit three-state bidirectional data bus.

This bus is used to program counter modes and to read status and count values. The data bus is active when $\overline{CS}=0$ and is in the high impedance state when $\overline{CS}=1$.

CLK_n [Counter Clock] n=0–2

These pins are the clock input that determines the count rate for counter n. The clock rate may be DC (standby mode) to 8 MHz.

OUT_n [Counter Output] n=0–2

These are the output pins for counter n. A variety of outputs is available depending on the count mode. When the CXQ71054 is used as an interrupt source, these pins can output an interrupt request signal.

GATE_n [Counter Gate] n=0–2

These input pins inhibit or trigger counter n according to the mode selected.

A₁, A₀ [Address]

These pins select one of the three counter or control word register. A₁, A₀ equal to 00, 01 or 10 selects counter 0, 1 or 2, respectively. The control word register is selected when A₁, A₀=11. These pins are normally connected to the system address bus.

\overline{CS} [Chip Select]

When $\overline{CS}=1$, all the bits of the data bus go to the high impedance state. \overline{CS} must be low to access the CXQ71054.

 \overline{RD} [Read Strobe]

\overline{RD} must be low to read data from the CXQ71054.

 \overline{WR} [Write Strobe]

\overline{WR} must be low to write data to the CXQ71054. The contents of the data bus are written to the CXQ71054 at the rising edge of \overline{WR} .

 V_{DD} [Power]

+5V power supply.

GND [Ground]

Ground.

Absolute Maximum Ratings*

(Ta=+25°C)

Parameter	Symbol	Rating Value	Unit
Power Supply Voltage	V _{DD}	-0.5 to +7.0	V
Input Voltage	V _I	-0.5 to V _{DD} +0.3	V
Output Voltage	V _O	-0.5 to V _{DD} +0.3	V
Power Dissipation	P _{DMAX}	1.0	W
Operating Temperature	T _{opr}	-10 to +70	°C
Storage Temperature	T _{stg}	-65 to +150	°C

* **Comment:** Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of this specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC CharacteristicsTa=-40°C to +85°C, V_{DD}=+5V±10%

Parameter	Symbol	Limits		Unit	Test Conditions
		Min.	Max.		
Input Voltage High	V _{IH}	2.2	V _{DD} +0.3	V	
Input Voltage Low	V _{IL}	-0.5	0.8	V	
Output Voltage High	V _{OH}	0.7×V _{DD}		V	I _{OH} =-400 μA
Output Voltage Low	V _{OL}		0.4	V	I _{OL} =2.5 mA
Input Leakage Current High	I _{LIH}		10	μA	V _I =V _{DD}
Input Leakage Current Low	I _{LIL}		-10	μA	V _I =0V
Output Leakage Current High	I _{LOH}		10	μA	V _O =V _{DD}
Output Leakage Current Low	I _{LOL}		-10	μA	V _O =0V
Supply Current	I _{DD1}		30	mA	8 MHz Operation
	I _{DD2}		50	μA	Stand-by Mode

CapacitanceTa=+25°C, V_{DD}=0V

Parameter	Symbol	Limits		Unit	Test Conditions
		Min.	Max.		
Input Capacitance	C _{IN}		10	pF	f _c =1 MHz Unmeasured pins returned to 0V
I/O Capacitance	C _{IO}		20	pF	

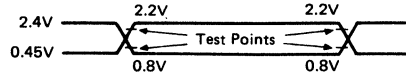
AC Characteristics

Ta=-40°C to +85°C, V_{DD}=5V±10%

Parameter	Symbol	Limits		Unit	Test Conditions
		Min.	Max.		
Read Cycle					
Address Set-up to \overline{RD} ↓	t _{SAR}	30		ns	
Address Hold from \overline{RD} ↑	t _{HRA}	10		ns	
\overline{CS} Set-up to \overline{RD} ↓	t _{SCR}	0		ns	
\overline{RD} Low Level Width	t _{RRL}	150		ns	
Data Delay from \overline{RD} ↓	t _{DRD}		120	ns	C _L =150 pF
Data Float from \overline{RD} ↑	t _{FRD}	10	85	ns	C _L =20 pF R _L =2 kΩ
Data Delay from Address	t _{DAD}		220	ns	C _L =150 pF
Read Recovery Time	t _{RV}	200		ns	
Write Cycle					
Address Set-up to \overline{WR} ↓	t _{SAW}	0		ns	
Address Hold from \overline{WR} ↑	t _{HWA}	0		ns	
\overline{CS} Set-up to \overline{WR} ↓	t _{SCW}	0		ns	
\overline{WR} Low Level Width	t _{WWL}	160		ns	
Data Set-up to \overline{WR} ↑	t _{SDW}	120		ns	
Data Hold from \overline{WR} ↑	t _{HWD}	0		ns	
Write Recovery Time	t _{RV}	200		ns	
CLK and Gate Timing					
CLK Cycle Time	t _{CYK}	125	DC	ns	
CLK High Level Width	t _{KKH}	60		ns	
CLK Low Level Width	t _{KKL}	60		ns	
CLK Rise Time	t _{KR}		25	ns	
CLK Fall Time	t _{KF}		25	ns	
GATE High Level Width	t _{GGH}	50		ns	
GATE Low Level Width	t _{GGL}	50		ns	
GATE Set-up to CLK ↑	t _{SGK}	50		ns	
GATE Hold from CLK ↑	t _{HKG}	50		ns	
Clock Delay from \overline{WR} ↑ (Count Transfer)	t _{DWK}	100		ns	t _{KKH} ≥ 125 ns
		225 - t _{KKH}		ns	t _{KKH} < 125 ns
Clock Set-up to \overline{WR} ↑ (Latch)	t _{SKW}	85		ns	
GATE Delay from \overline{WR} ↑	t _{DWG}	0		ns	
OUT Delay from GATE ↓	t _{DGO}		120	ns	C _L =150 pF
OUT Delay from CLK ↓	t _{DKO}		150	ns	C _L =150 pF
OUT Delay from \overline{WR} ↑ (Initial Out)	t _{DWO}		295	ns	C _L =150 pF

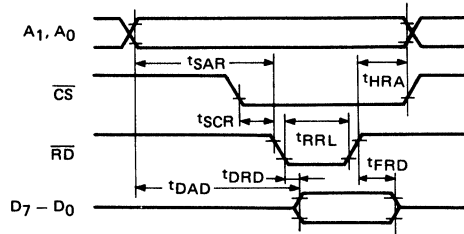
Note: AC timing test points for output V_{OH}=2.2V, V_{OL}=0.8V

AC Test Input Waveforms:

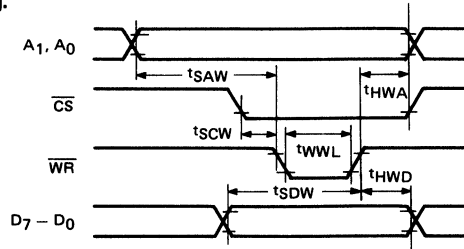


Testing Waveforms

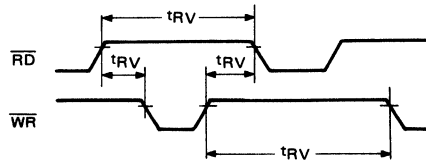
Read Cycle Timing:



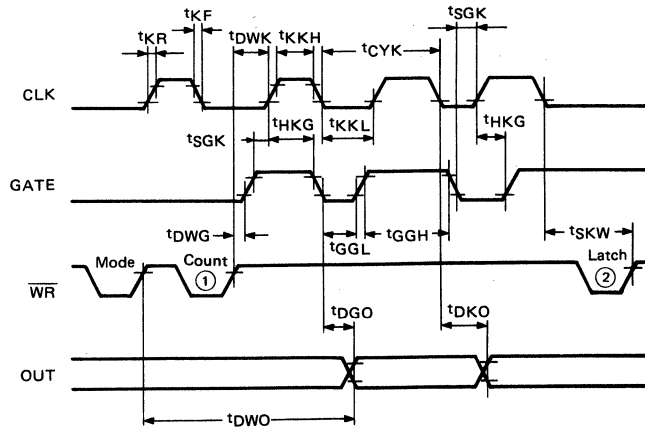
Write Cycle Timing:



Read/Write Recovery Time:



CLK and GATE Timing



- Notes:
- ① The last 1 byte of count number writing
 - ② Count latch command or multiple latch command

Block Functions

Data Bus Buffer

This is an 8-bit three-state bidirectional buffer that acts as an interface between the CXQ71054 and the system data bus. The data bus buffer handles control words, the count to be written to the count register, count data read from the count latch, and status data read from the status latch.

Read/Write Control

This circuit decodes signals from the system bus and sends control signals to other blocks of the CXQ71054. A_1 and A_0 select one of the counters or control word register. A low signal on \overline{RD} or \overline{WR} selects a read or write operation. \overline{CS} must be low to enable these operations.

Control Word Register

This is an 8-bit register into which the control word is written to determine the operating mode of the counter. Data is written to this register when the CPU executes an OUT command while $A_1, A_0=11$. The contents of this register cannot be read if the CPU executes an IN command while $A_1, A_0=11$. However, the multiple latch command allows you to read the mode and status of each counter.

Counter n (n=0-2)

The CXQ71054 contains three counters capable of binary or BCD operation. There are six programmable count modes. The counters operate independently and each can be set to a different mode. Address lines A₁, A₀ are used to select one of the three counter.

A 16-bit synchronous down counter performs the actual count operation within the counter. It is presettable and counts binary or BCD operation.

The count register is a 16-bit register that stores the count when it is newly written to the counter. The count is transferred to the down counter and a count operation for a specified number of counts begins.

The 8-bit width of the internal data bus permits the transfer of only eight bits at a time when the count is written to the count register. However, 16-bit data is transferred from the count register to the down counter at the same time. When the count is written to the count register while the counter is in read/write one byte mode, the remaining byte of the register will be zero.

The count latch normally holds the current value of the down counter. If the contents of the down counter change, the contents of the count latch also change so that the two values are the same. When the CXQ71054 receives a count latch command, the count latch latches the value of the down counter and holds it until the CPU reads it. When the data is read, the count latch returns to tracking the down counter.

When the control word is written to the counter, the lower six bits are copied to the lower six bits of the 8-bit status register. The remaining two bits show the status of the OUT pin and the NULL COUNT flag. When the Multiple Latch command is sent to latch counter, the current value of the status register is latched into the status latch. This data is held in the latch until the CPU reads it.

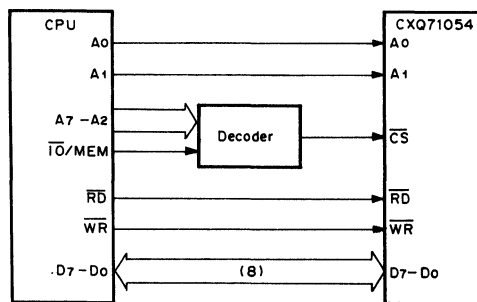
The control logic controls each internal block according to the mode and the state of the CLK and GATE pins. The result is output to and sets the state of the OUT pin.

CXQ71054 System Configuration Example

The CPU views the three counters and the control word register as four I/O ports. A₁, A₀ are connected to the A₁, A₀ pins of the system address bus. \overline{CS} is generated by decoding the address and \overline{IO}/MEM signals so that \overline{CS} goes low when the address bus is set to the target I/O address and I/O is selected. These connections are shown in Figure 1.

The CXQ71054 can be used with the memory-mapped I/O configurations. Then the decoding should be such that \overline{CS} goes low when memory is selected.

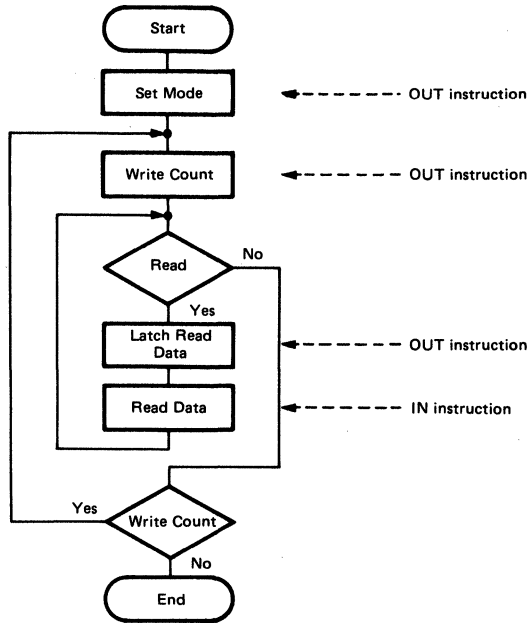
Figure 1. System Configuration Example



Programming and Reading the Counter

The counter must be programmed and the operating mode specified before the CXQ71054 can be used. Once a mode has been selected for counter, it operates in that mode until another mode is set. The count is written to the count register and when it is transferred to the down counter, a new count operation begins. The current count and status can be read while the counter operates. Figure 2 outlines the steps of operation.

Figure 2. Basic Operating Procedure



Programming the Counter

The CXQ71054 is controlled by a microcomputer program. The program must write a control word to set the counter mode and write a count data that determines the length of the count operation. Table 1 shows the values for A₁, A₀ that determine the target counter for write operations.

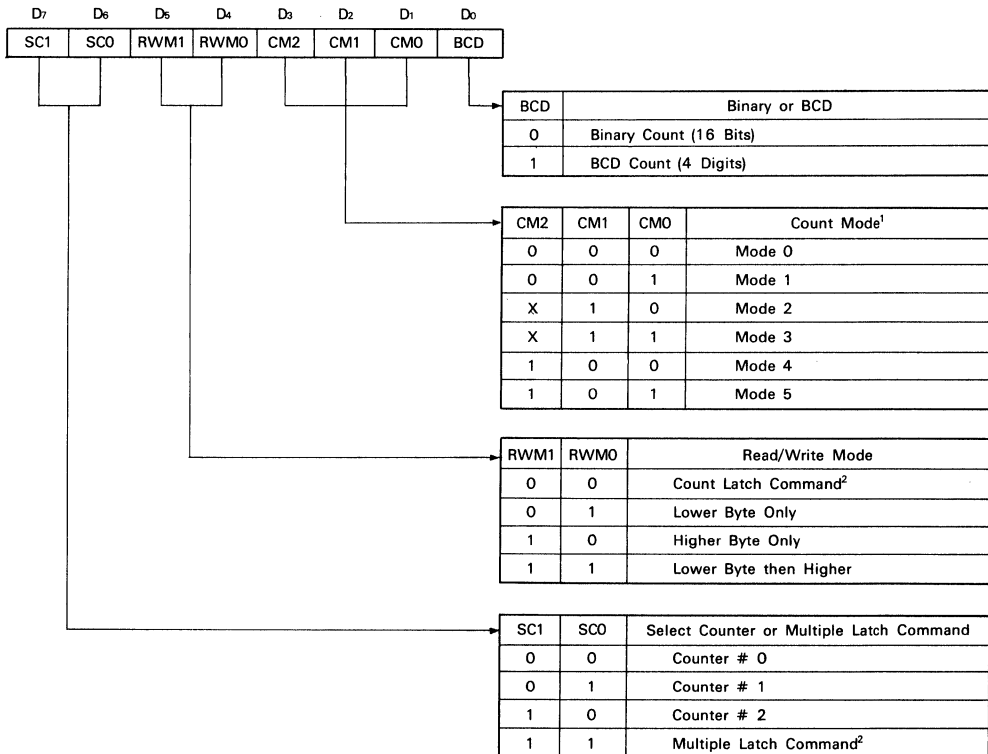
Table 1. Write Operations ($\overline{CS}=0$, $\overline{RD}=1$, $\overline{WR}=0$)

A ₁	A ₀	Write Target
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2
1	1	Control Word Register

Control Words and Mode Setting

The control word must be written to set the counter mode before operating the counter. A control word is written to the control word register if a write operation is performed when A₁, A₀=11. Figure 3 shows the format of the 8-bit control word.

Figure 3. Control Word Format (A₁, A₀=11, $\overline{CS}=0$, $\overline{RD}=1$, $\overline{WR}=0$)



1. $\overline{CS}=0$, $\overline{RD}=0$, $\overline{WR}=1$

2. See: Reading the Counter

X: Don't Care

SC1 and SC0 correspond to bits D7 and D6 and specify a counter or the Multiple Latch command. When a counter is chosen, the specifications described below apply to the counter.

RMW1 and RMW2 correspond to bits D5 and D4 and specify the read/write operation to the counter or select the Count Latch command.

CM2, CM1, and CM0 correspond to bits D3, D2, and D1 and set the counter mode (0 to 5).

BCD corresponds to bit D0 and selects binary or BCD operation. The count may be 0 to FFFFH in binary mode or 0 to 9999 in BCD.

If a control word specifies a mode, the lower six bits of the control word are copied to the lower six bits of the status register of the counter selected by SC1 and SC0. The mode selected remains in effect until a new mode is set. This is not true if the control word specifies the Count Latch or Multiple Latch command.

Writing the Count

The count is written to the counter after the mode is set. Set A1, A0 to specify the target counter as shown in Table 1. A new count can be written to a counter at any time, but the read/write mode selected (when the mode was written) must be used when writing the count.

In 1-byte mode (RWM1, RWM0=01, 10), the higher or lower byte of the count register is written by the first write. The write operation ends and 00H is written to the remaining byte. In the 2-byte mode (RWM1, RWM0=11), the lower byte is written by the first write and the higher byte by the second.

For example, if the 2-byte count 8801H is written to a counter set *in lower byte only mode mistakenly*, the lower byte (01H) is written first, followed by the higher byte (88H). Therefore, the data written to the count register is 0001H for the first write and 0088H for the second.

Table 2. Read/Write Mode and Count Write

Read/Write Mode	No. of Writes	Count Register	
		Higher Byte	Lower Byte
Low 1-byte	1	00H	nnH
High 1-byte	1	nnH	00H
Low/High 2-byte	2	nnH (2nd write)	nnH (1st write)

nn=two-digit hexadecimal value

Reading the Counter

There are three methods to read the contents of the down counter during operation. In particular, the multiple latch command reads the current count data and the counter mode or the state of the OUT pin. Table 3 shows the values of A1, A0 used to select the counter to be read.

Table 3. Read Operations ($\overline{CS}=0$, $\overline{RD}=0$, $\overline{WR}=1$)

A1	A0	Read Target
0	0	Counter 0
0	1	Counter 1
1	0	Counter 2

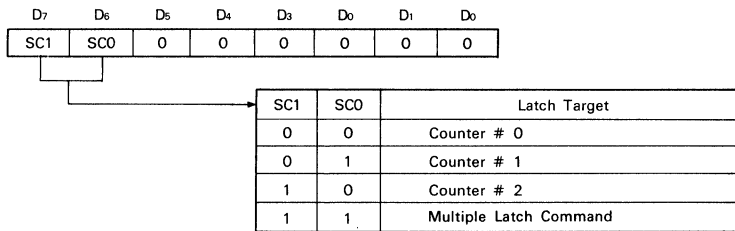
Directly Reading the Counter

It is possible to read the current value of the counter by reading the counter selected by A_1 , A_0 as shown in Table 3. This involves reading the count latch; since the value of the down counter may change while the count latch is read, this method may not provide an accurate result. The CLK or GATE input must be controlled to stop the counter and read it for a correct result.

Using the Count Latch Command

The Count Latch command is used to latch the current counter value into the count latch. This value is held by the latch until it is read or until a new mode is set. This provides an accurate reading of the counter value without affecting counter operation when the command is executed. Figure 4 shows the format for the Count Latch command.

Figure 4. Count Latch Command Format (A_1 , $A_0=11$, $\overline{CS}=0$, $\overline{RD}=1$, $\overline{WR}=0$)



Note that when bits D_7 and D_6 (SC1 and SC0) are 11, the command is not the Count Latch command; it is the Multiple Latch command.

If the counter value that was latched into the count latch is not read before a second latch command is issued, the second command is ignored. This is because the counter value latched by the first command is held until it is read or until a new mode is set. When the data in the count latch is read, the latch is then released and continues tracking the down counter.

Using the Multiple Latch Command

When the multiple latch command is received, the counter value and status register for any counter may be selectively latched into the count latch and status latch. Bits D_1 – D_5 of the Multiple Latch command specify the counter latching. The CPU can then read the status and counter value for the selected counter. Figure 5 shows the format for this command.

Bits CNT2, CNT1, and CNT0 correspond to counters 2, 1, and 0. The command is executed for all counters whose corresponding bit is 1. This allows the data for more than one counter to be latched by a signal Count Latch command.

When the **COUNT** bit is 0, the counter value of the selected counters is latched into the count latches.

When the **STATUS** bit is 0, the status of the selected counters is latched into the status latches. Bits D_5 – D_0 of the status register show the mode status of the counter. The **OUTPUT** bit (D_7) shows the state of the OUT pin of that counter. These bits are shown in Figure 6. The **NULL COUNT** bit (D_6) indicates whether the count data is valid. When the count is transferred from the count register to the down counter, this bit changes to 0 to show that the data is valid. Table 4 shows how the **NULL COUNT** flag operates.

Figure 5. Multiple Latch Command Format (A₁, A=11, CS=0, RD=1, WR=0)

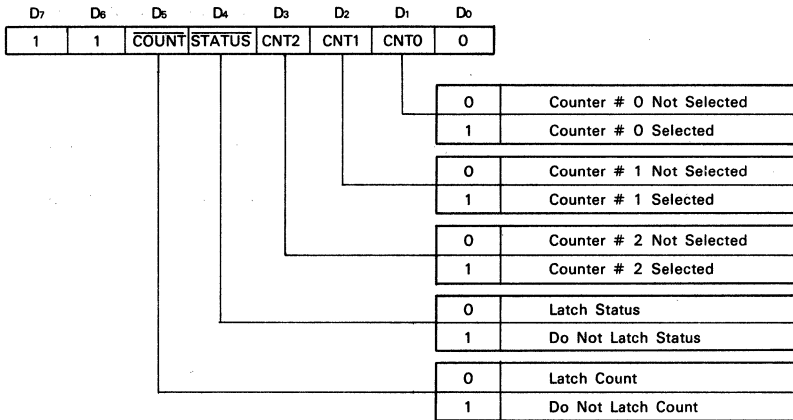


Figure 6. Status Data

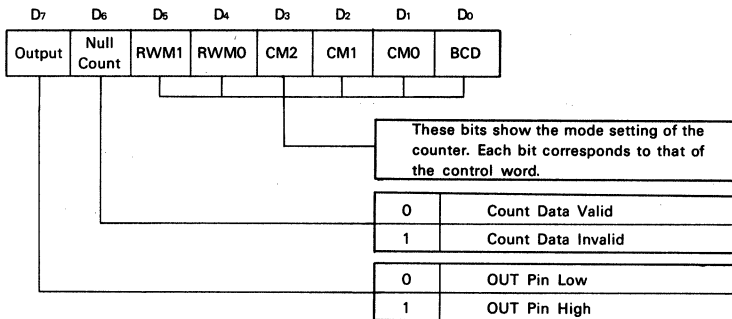


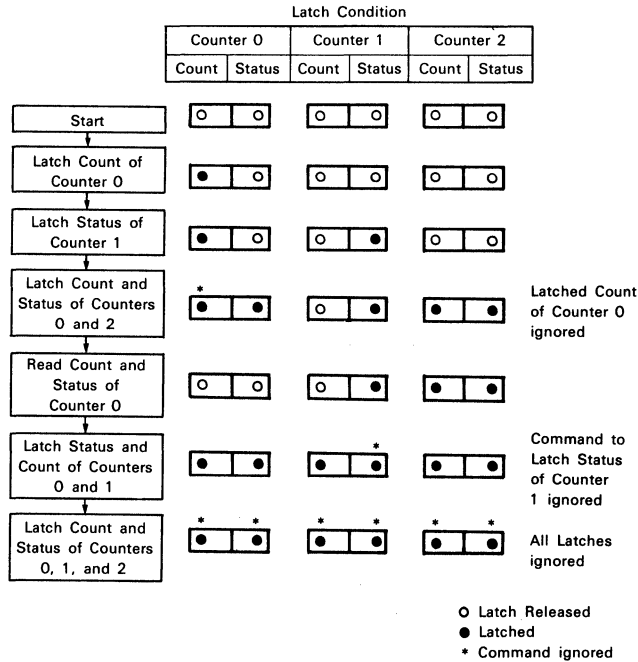
Table 4. Null Count Flag Operation

Operation	Null Count Flag
Write control word for mode set	1
Write count to count register*	1
Transfer count from count register to down counter	0

Note: * When 2-byte mode is selected, the flag becomes 1 when the second byte is written.

If the data that was latched is not read before a second multiple latch command is executed, the second command is ignored for those latches whose contents have not been read. This is because the data latched by the first command is held until it is read or until a new mode is set. When the data in the latch is read, the latch is then released.

Figure 7. Multiple Latch Command Execution Example



It is possible to latch both the count and status using the Multiple Latch command(s). The status data is always read first regardless of which data is latched first. The count data is read by the next read operation (1- or 2-step read as determined by read/write mode). If additional read commands are received, the unlatched count data (the contents of the down counter as reflected by the current counter value) is read.

Read operations must be performed in accordance with read/write mode. In 2-byte mode, two bytes of data must always be read. This does not imply that the second byte must be read right after the first; other counter operations may be inserted between the two reads. For example, you could read the lower byte, write a new lower byte, read the higher byte, and then write a new higher byte.

Definitions

CLK pulse refers to the time from the rising to the falling edge of the CLK_n input.

Trigger refers to the rising edge of the GATE_n input.

The **GATE_n** input is sampled at each rising edge of the CLK_n input. The GATE input can be level- or rising edge-sensitive. In the latter case, counter *n*'s internal flip-flop is set at the rising edge of the GATE signal, sensed at the rising edge of the next CLK pulse, and reset immediately. This allows edge-triggering to be sensed whenever it occurs.

Initial OUT refers to the state of the OUT pin immediately after the mode is set.

Count Transfer refers to the transfer from the count register to the down counter. The down counter is decremented at the falling edge of the CLK pulse.

Count Write refers to writing the count value to the count register.

Count Zero is state of the down counter when the counter has come to zero.

PCNT0, PCNT1, and PCNT2 are the I/O ports for counters 0, 1, and 2, respectively. **PCTRL** is the I/O port for the control word.

CW is control word.

HB is the higher byte of the count.

LB is the lower byte of the count.

In the timing charts for each counter mode, counter is in read/write 1-byte and binary count mode. If no GATE signal appears in the charts, a high level signal is assumed. The value shown below the OUT signal is the counter value. The maximum value that can be set for the count in each mode is 0. When this value is set, a maximum value of 10000H (binary count) or 10000 (BCD count) is enabled.

Counter Modes

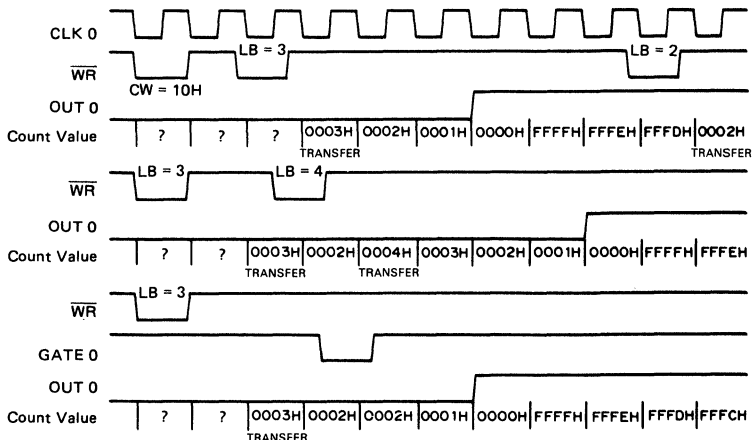
Mode 0: Interrupt on End of Count

In this mode, the OUT output changes from low to high level when the specified count has come to zero.

Table 5. Mode 0 Operation

Function	Result
Initial OUT	Low Level
GATE High	Count enable
GATE Low	Count disable
Count Write and Transfer	When the count is written, OUT will go low independent of the CLK pulse. In 2-byte mode counting is disabled while the first is written. OUT goes low immediately. Transfer is performed at the next CLK pulse after the count is written.
Count Operation	If the count is written while GATE is high; the decrement begins at the next CLK pulse after data transfer. If a count of n is set, OUT goes high after n+1 CLK pulse. If the count is written while GATE is low; after GATE goes high, the decrement begins at the next CLK pulse and OUT is low for n CLK pulse. One CLK pulse is not necessary for the transfer because it has already done while GATE is low.
Count Re-write	The new count is transferred at the next CLK pulse and counting is resumed from the new data.
Count Zero	The signal at the OUT pin goes high. The count operation does not stop and counts down to FFFFH (binary) or 9999 (BCD).
Minimum Count	1

Figure 8. Mode 0 Timing Chart



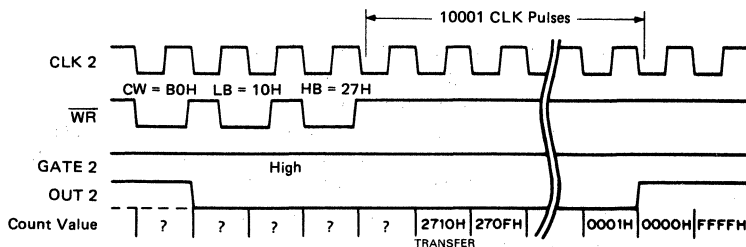
Mode 0 Program Example

This subroutine causes a delay of 10000 (2710H) CLK pulses. In this program, counter 2 is set to 2-byte mode and binary count.

```

SUBRO: MOV AL,10110000B ; set mode: counter 2.
        ; 2-byte mode.
        OUT PCTRL,AL ; Count mode 0, binary
        MOV AL,10H
        OUT PCNT2,AL
        MOV AL,27H ; write count 10000
        OUT PCNT2,AL
        RET
    
```

Figure 9. Mode 0 Program Example Timing Chart



Mode 1: GATE Retriggerable One-shot

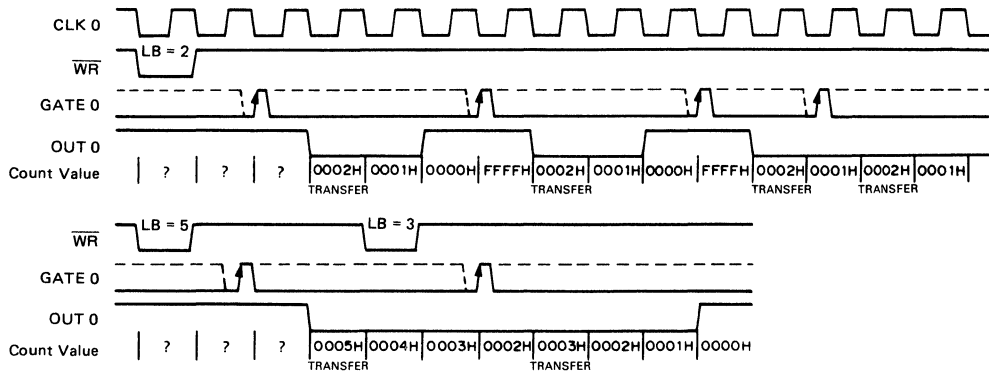
In Mode 1, a low-level one-shot pulse triggered by the GATE input is output from the OUT pin.

Table 6. Mode 1 Operation

Function	Result
Initial OUT	High Level
GATE Trigger*	OUT goes low at the next CLK pulse after the trigger.
Count Write and Transfer	Transfer is performed at the next CLK pulse after the GATE trigger.
Count Operation	OUT goes low at the next pulse following the trigger to start the one-shot pulse operation. The decrement then starts at the next CLK pulse. If a count of n is set, the one-shot output from OUT continues (remains low) for n CLK pulse. OUT then goes high when the count becomes zero. The one-shot operation is retriggerable, hence OUT remains low for n CLK pulse after any GATE trigger.
Count Re-write	If a new count is written during one-shot pulse operation, it does not affect the current operation unless the counter is retriggered. If retriggered, the new data is transferred.
Count Zero	The signal at the OUT goes high. The count operation does not stop and counts down to FFFFH (binary) or 9999 (BCD).
Mimumum Count	1

* The trigger is ignored when the count has not been written after the mode is set, or when only one byte of the count has been written in 2-byte.

Figure 10. Mode 1 Timing Chart



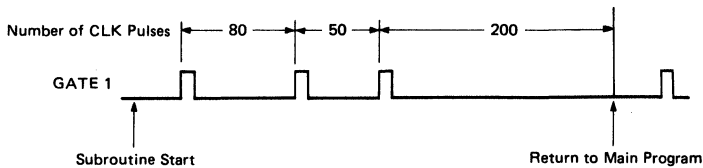
Mode 1 Program Example

This subroutine waits until no trigger is generated for an interval of 200 or more CLK pulses after the first gate trigger and returns to main program. Counter 1 is set to low-byte read/write mode and binary count.

```

SUBR1:  MOV  AL,01010010B ; set mode: counter 1, low-byte read/
        OUT  PCTRL,AL    ; write mode, count mode 1, binary
        MOV  AL,200
        OUT  PCNT1,AL    ; write low byte of count
;
FSTTRG: MOV  AL,11100100B ; multiple latch command: counter 1,
        OUT  PCTRL,AL    ; status
        IN   AL,PCNT1
        TEST AL,7        ; wait for first trigger
        BNZ  FSTTRG
;
WAIT:   MOV  AL,11100100B ; multiple latch command: counter 1,
        OUT  PCTRL,AL    ; status
        IN   AL,PCNT1
        TEST AL,7        ; wait until output goes high
        BZ   WAIT
        RET
    
```

Figure 11. Mode 1 Program Example Timing Chart



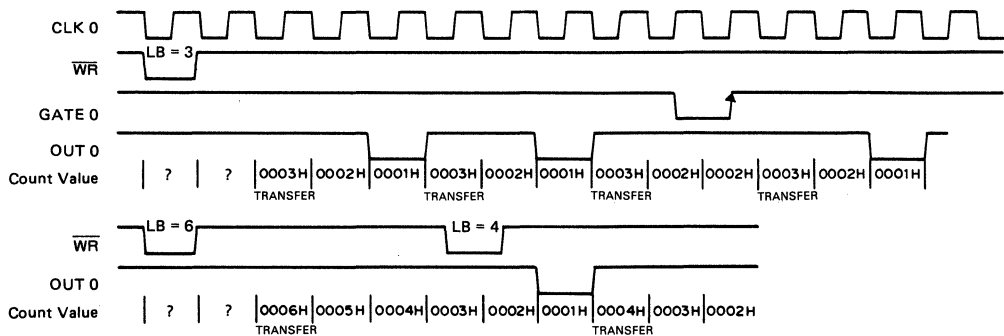
Mode 2: Rate Generator

In Mode 2, the signal from the OUT pin cyclically goes low for one clock period when the counter has reached 0001H. The counter operates as a frequency divider.

Table 7. Mode 2 Operation

Function	Result
Initial OUT	High Level
GATE High	Count Enable
GATE Low	Count disabled. If GATE goes low while OUT is low, OUT will go high (independent of the CLK pulse).
GATE Trigger*	Transfer is performed at the next CLK pulse after the trigger.
Count Write and Transfer	Transfer is performed at the next CLK pulse after the GATE trigger; after the count is written; after the count completion.
Write Operation	The decrement begins at the next CLK pulse after data transfer. When the count has decremented to one, OUT goes low for one CLK pulse. OUT then returns high and the transfer is performed again. The sequence is repeated. If a count of n is set, this sequence repeats every n CLK cycles.
Count Re-write	If a new count is written during counting, the current counting operation is not affected. The new count will be transferred at the next transfer: after the GATE trigger or after the count completion.
Count Zero	Not available in this mode.
Minimum Count	2

* The trigger is ignored when the count has not been written or when only one byte of the count has been written in 2-byte mode.

Figure 12. Mode 2 Operation Timing Chart

Mode 2 Program Example

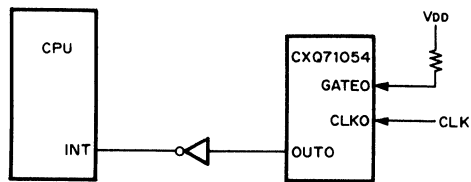
This subroutine generates an interrupt to the CPU every 10000 clock pulses. Counter 0 is in 2-byte mode and binary counting.

```

SUBR 2: MOV  AL,00110100B ; set mode: counter 0, 2-byte
        OUT  PCTRL,AL    ; mode, count mode 2, binary
        MOV  AL,10H
        OUT  PCNT0,AL
        MOV  AL,27H      ; write count 10000
        OUT  PCNT0,AL
        RET

```

Figure 13. Mode 2 Configuration



Mode 3: Square Wave Generator

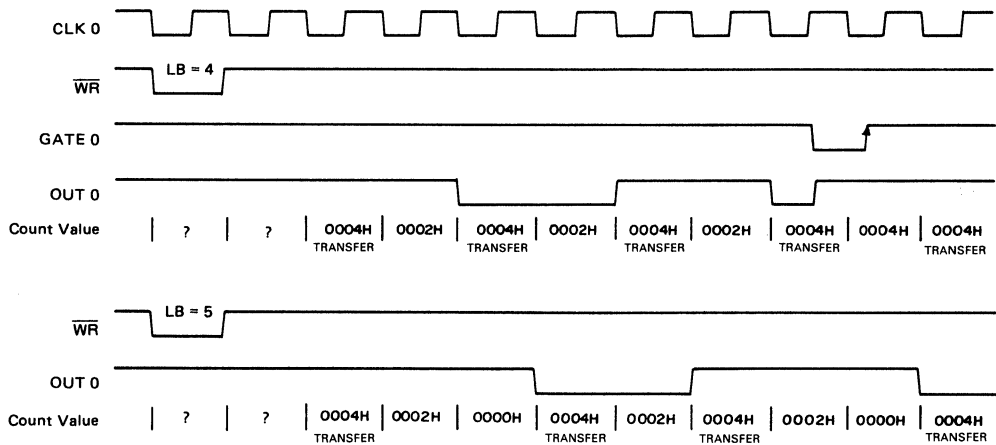
Mode 3 is a frequency divider similar to Mode 2, but with a different duty cycle of OUT.

Table 8. Mode 3 Operation

Function	Result
Initial OUT	High Level
GATE High	Count Enable
GATE Low	Count Disable. If GATE goes low while OUT is low. OUT will go high (Independent of the CLK pulse).
GATE Trigger*	Transfer is performed at the next CLK pulse after the trigger.
Count Write and Transfer	Transfer is performed at the next CLK pulse after the GATE trigger; after the count is written; after the end of the half-period of the count; after the count completion.
Count Operation	The operation depends on whether the programmed count n is even or odd. If n is even; the count is decremented by two. When the count reaches two (the end of the half-cycle), n will be transferred again and OUT will be inverted. The above sequence is taken as a half-cycle and repeated. If n is odd; $n-1$ is transferred and the count is decremented by two. The first half-cycle (while OUT is high) continues until the count reaches zero and $n-1$ is transferred again. The second half-cycle (while OUT is low) continues until the count reaches 2. Hence the first half-cycle is one CLK longer than the second half-cycle. (high OUT $\rightarrow (n+1)/2$ CLK; low OUT $\rightarrow (n-1)/2$ CLK) The above sequence is repeated.
Count Re-write	If a new count is written during counting, the current counting operation is not affected, The new count will be transferred at the next transfer: after the GATE trigger, after the current half-cycle.
Count Zero	Available only when the count is odd.
Minimum Count	2

* The trigger is ignored when the count has not been written after the mode is set or when only one byte of count has been written in 2-byte mode.

Figure 14. Mode 3 Timing Chart



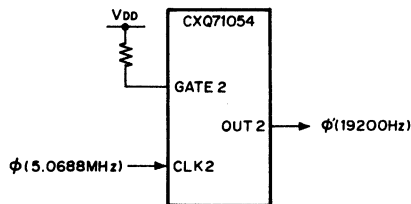
Mode 3 Program Example

This subroutine divides the input CLK frequency (5.0688 MHz) by 264 (108 H) to get a 19,200 Hz clock. Counter 2 is in 2-byte binary mode.

```

SUBR 3: MOV AL,10110110B ; set mode: counter 2, 2-byte
        OUT PCTRL,AL ; mode count mode 3, binary
        MOV AL,08H
        OUT PCNT2,AL
        MOV AL,01H ; 264 frequency division
        OUT PCNT2,AL
        RET
    
```

Figure 15. Frequency Division



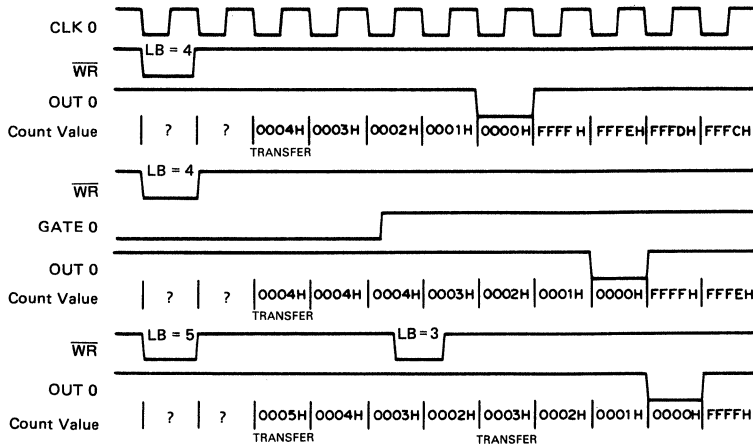
Mode 4: Software-triggered Strobe

In Mode 4, when the specified count comes to the end, OUT will go low for one CLK pulse.

Table 9. Mode 4 Operation

Function	Result
Initial OUT	High Level
GATE High	Count enable
GATE Low	Count disable
Count Write and Transfer	Transfer is performed at the next CLK pulse after the count is written. In 2-byte mode, data is transferred after the second byte is written.
Count Operation	If GATE is high, the decrements begins at the next CLK after data transfer. When the count becomes zero, OUT will go low for one CLK pulse and then go high again. For a count of n, OUT remains high for n+1 CLK pulses. If GATE is low, the decrement begins at the next CLK after GATE goes high.
Count Re-write	The new count is transferred at the next CLK pulse and counting is resumed from the new data.
Count Zero	OUT is low for one CLK pulse and returns to high. The down counter then counts to FFFFH (binary) or 9999 (BCD) without stopping counter operation.
Minimum Count	1

Figure 16. Mode 4 Timing Chart



Mode 5: Hardware-triggered Strobe (Retriggerable)

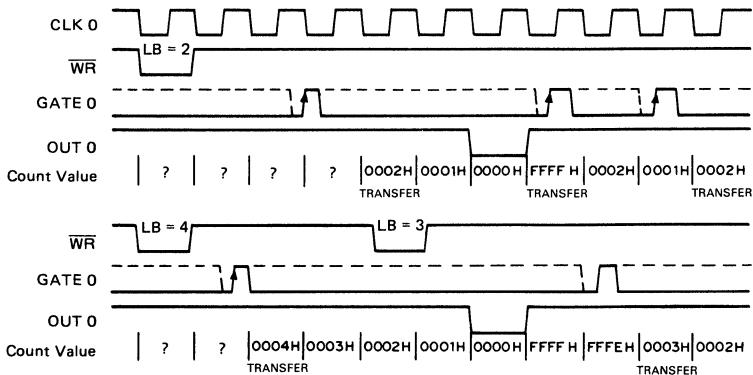
Mode 5 is similar to Mode 4 except that operation is triggered by the GATE input and can be retriggered.

Table 10. Mode 5 Operation

Function	Result
Initial OUT	High Level
GATE Trigger*	Transfer is performed at the next CLK pulse after the trigger.
Count Write and Transfer	Transfer is performed at the next CLK pulse after the GATE trigger.
Count Operation	The decrement begins at the next CLK pulses after the transfer. When the count becomes zero, OUT will go low for one CLK pulse and then go high again. For a count of n, OUT remains high for n+1 CLK pulses.
Count Re-write	The new count will be written without affecting the current counting operation.
Count Zero	OUT is low for one CLK and goes high again. The down counter then counts to FFFFH (binary) or 9999 (BCD) without stopping the counter operation.
Minimum Count	1

* The trigger is ignored when the count has not been written after the mode is set or when only one byte has been written in 2-byte mode.

Figure 17. Mode 5 Timing Chart



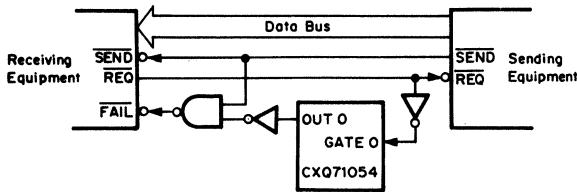
Mode 5: Program Example

Mode 5 can be used to add a fail-safe function to an interface. For example, the receiving equipment requests data by issuing a $\overline{\text{REQ}}$ signal to the sending equipment. The sending equipment responds by outputting data to the data bus and returning a $\overline{\text{SEND}}$ signal to the receiving equipment. In this type of system, if there is a malfunction in the sending equipment and no $\overline{\text{SEND}}$ signal is sent, the receiving equipment waits indefinitely for the $\overline{\text{SEND}}$ signal and system operation stops. The following subroutine is a remedy for this situation. If no $\overline{\text{SEND}}$ signal is output within a given period (50 CLK cycles in this example) after the $\overline{\text{REQ}}$ signal is output, the system assumes the sending equipment is malfunctioning and a $\overline{\text{FAIL}}$ signal is sent to the receiving equipment.

```

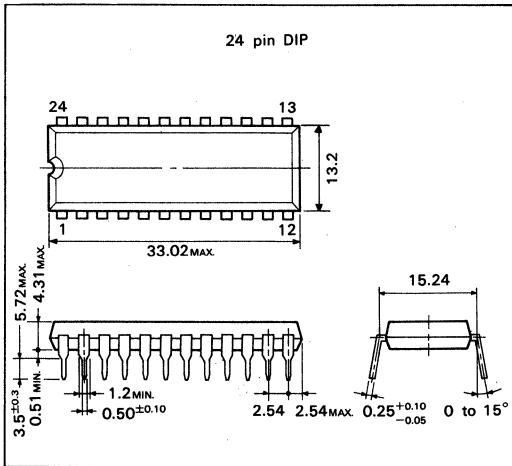
SUBR5:  MOV  AL,00011010B  ; set mode: counter 0, lower 1-byte
        OUT  PCTRL,AL      ; mode, count mode 5, binary
        MOV  AL,50         ; set interval: 50 CLK pulses
        OUT  PCNT0,AL
        RET
    
```

Figure 18. Interface Fail-safe Example



Package Outline

Unit: mm



Parallel Interface Unit

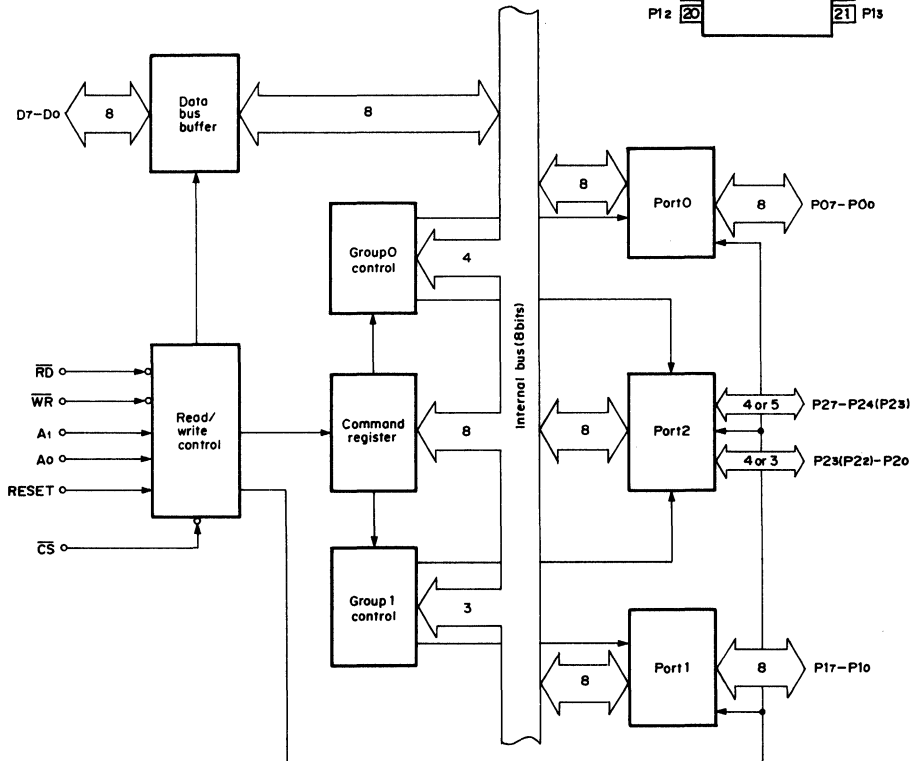
Description

The CXQ71055 is a low-power CMOS programmable parallel interface unit for use in microcomputer systems. The CXQ71055 has three I/O ports and is typically used to interface peripheral devices to a microcomputer system bus.

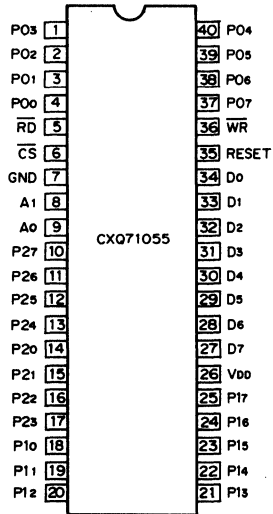
Features

- Three 8-bit I/O ports
- Three programmable operation modes
- Bit manipulation command
- Bus-compatible with most microcomputer
- 8 MHz operation
- CMOS technology
- +5V single power supply
- 40-pin plastic DIP (600 mil)
- NEC μ PD71055 compatible

Block Diagram



Pin Configuration (Top View)



Pin Identification

No.	Symbol	Direction	Function
1-4	P03-P00	In/Out	Port 0, bits 3-0
5	\overline{RD}	In	Read strobe
6	\overline{CS}	In	Chip select
7	GND		Ground
10-13	P27-P24	In/Out	Port 2, bits 7-4
14-17	P20-P23	In/Out	Port 2, bits 0-3
18-25	P10-P17	In/Out	Port 1
26	V _{DD}		Power supply
27-34	D7-Do	In/Out	Data bus
35	RESET	In	Reset
36	\overline{WR}	In	Write strobe
37-40	P07-P04	In/Out	Port 0, bits 7-4

Pin Functions

D7-Do [Data Bus]

D7-Do make up an 8-bit, three-state, bidirectional data bus. The bus is connected to the system data bus. It is used to send commands to the CXQ71055 and to send data to and from the CXQ71055.

\overline{CS} [Chip Select]

\overline{CS} is used to select the CXQ71055. When $\overline{CS}=0$, the CXQ71055 is selected. When $\overline{CS}=1$, the CXQ71055 is not selected and its data bus is high-impedance.

\overline{RD} [Read Strobe]

\overline{RD} is set low when data is being read from the CXQ71055 data bus.

\overline{WR} [Write Strobe]

\overline{WR} should be set low when data is to be written to the CXQ71055 data bus. The contents of the data bus are written to the CXQ71055 at the rising edge (low to high) of the \overline{WR} signal.

A1, A0 [Address]

The A1 and A0 inputs are used in combination with the \overline{RD} and \overline{WR} signals to select one of the three ports or the command register. A1 and A0 are usually connected to the lower two bits of the system address bus (table 1).

RESET [Reset]

When RESET is high, the CXQ71055 is reset. The group 0 and the group 1 ports are set to mode 0 (basic I/O port mode) and all ports are set for input.

P07-P00, P17-P10, P27-P20 [Ports 0,1,2]

Pins P07-P00, P17-P10, and P27-P20 are the port 0,1, and 2 I/O pins, respectively.

Table 1. Control Signals and Operation

CS	RD	WR	A ₁	A ₀	Operation	CPU Operation
0	0	1	0	0	Port 0 to Data bus	Input
0	0	1	0	1	Port 1 to Data bus	Input
0	0	1	1	0	Port 2 to Data bus	Input
0	0	1	1	1	Use Prohibited	
0	0	0	x	x		
0	1	0	0	0	Data bus to Port 0	Output
0	1	0	0	1	Data bus to Port 1	Output
0	1	0	1	0	Data bus to Port 2	Output
0	1	0	1	1	Data bus to Command register	Output
0	1	1	x	x	Data bus high impedance	
1	x	x	x	x		

Absolute Maximum Ratings

(T_a=25°C)

Parameter	Symbol	Rating Value	Units
Power supply voltage	V _{DD}	-0.5 to +7.0	V
Input voltage	V _I	-0.5 to V _{DD} +0.3	V
Output voltage	V _O	-0.5 to V _{DD} +0.3	V
Power dissipation	P _{DMAX}	500	mW
Operating temperature	T _{opr}	-40 to +85	°C
Storage temperature	T _{stg}	-65 to +150	°C

Comment: Exposing the device to stresses beyond those listed in Absolute Maximum Ratings could cause permanent damage. Exposure to absolute maximum ratings for extended periods may affect device reliability.

DC Characteristics

(T_a=-40 to +85°C, V_{DD}=5V±10%)

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Input voltage high	V _{IH}	2.2	V _{DD} +0.3	V	
Input voltage low	V _{IL}	-0.5	0.8	V	
Output voltage high	V _{OH}	0.7×V _{DD}		V	I _{OH} =-400 μA
Output voltage low	V _{OL}		0.4	V	I _{OL} =2.5 mA
Input leakage current high	I _{LIH}		10	μA	V _I =V _{DD}
Input leakage current low	I _{LIL}		-10	μA	V _I =0V
Output leakage current high	I _{LOH}		10	μA	V _O =V _{DD}
Output leakage current low	I _{LOL}		-10	μA	V _O =0V
Supply current	I _{DD1}		15	mA	Operation
	I _{DD2}		50	μA	Stand-by Mode

Capacitance

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Input capacitance	C _i		10	pF	f _c =1 MHz Unmeasured pins returned to 0V
I/O capacitance	C _{io}		20	pF	

AC Characteristics (T_a=-40 to +85°C, V_{DD}=5V±10%)**Read Timing**

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
A ₁ , A ₀ , \overline{CS} set-up to \overline{RD} ↓	tsAR	0		ns	
A ₁ , A ₀ , \overline{CS} hold from \overline{RD} ↑	thRA	0		ns	
\overline{RD} pulse width	trRL	160		ns	
Data delay from \overline{RD} ↓	tDRD		120	ns	C _L =150 pF
Data float from \overline{RD} ↑	tFRD	10	85	ns	C _L =20 pF R _L =2 kΩ
Read recovery time	trV	200		ns	

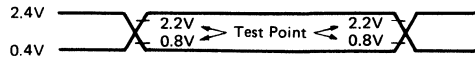
Write Timing

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
A ₁ , A ₀ , \overline{CS} set-up to \overline{WR} ↓	tsAW	0		ns	
A ₁ , A ₀ , \overline{CS} hold from \overline{WR} ↑	thWA	0		ns	
\overline{WR} pulse width	twWL	120		ns	
Data set-up to \overline{WR} ↑	tsDW	100		ns	
Data hold from \overline{WR} ↑	thWD	0		ns	
Write recovery time	trV	200		ns	

Other Timing

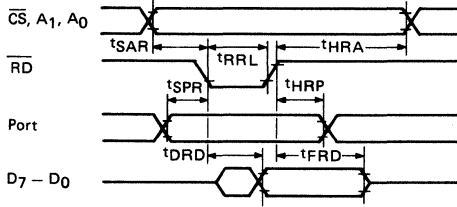
Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Port set-up time to \overline{RD} ↓	tSPR	0		ns	
Port hold time from \overline{RD} ↑	tHRP	0		ns	
Port set-up time to \overline{STB} ↓	tSPS	0		ns	
Port hold time from \overline{STB} ↑	tHSP	150		ns	
Port delay time from \overline{WR} ↑	tdWP		350	ns	$C_L=150$ pF
\overline{STB} pulse width	tSSL	350		ns	
\overline{DAK} pulse width	tdADAL	300		ns	
Port delay time from \overline{DAK} ↓ (mode 2)	tDDAP		300	ns	$C_L=150$ pF
Port float time from \overline{DAK} ↑ (mode 2)	tFDAP	20	250	ns	$C_L=20$ pF $R_L=2$ k Ω
\overline{OBF} set delay from \overline{WR} ↑	tdWOB		300	ns	$C_L=150$ pF
\overline{OBF} clear delay from \overline{DAK} ↓	tDDAOB		350	ns	
IBF set delay from \overline{STB} ↓	tDSIB		300	ns	
IBF clear delay from \overline{RD} ↑	tDRIB		300	ns	
INT set delay \overline{DAK} ↑	tDDAI		350	ns	
INT clear delay from \overline{WR} ↓	tdWI		450	ns	
INT set delay from \overline{STB} ↑	tDSI		300	ns	
INT clear delay from \overline{RD} ↓	tDRI		400	ns	
RESET pulse width	tRESET1	50		μ s	During or right after power-on
	tRESET2	500		ns	During operation

AC Testing Waveform

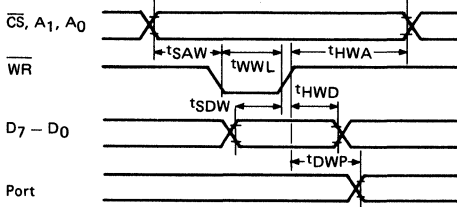


Timing Waveforms

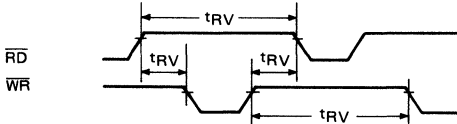
Timing Mode 0: Input



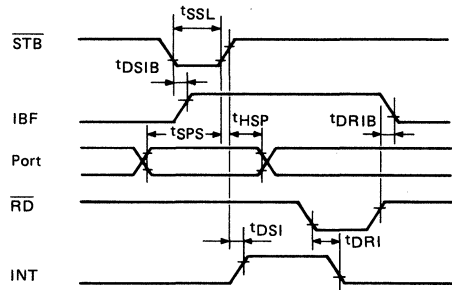
Mode 0: Output



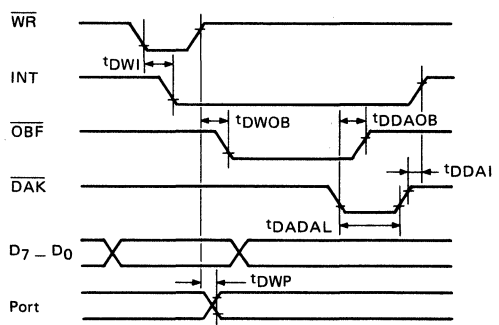
Recovery Time



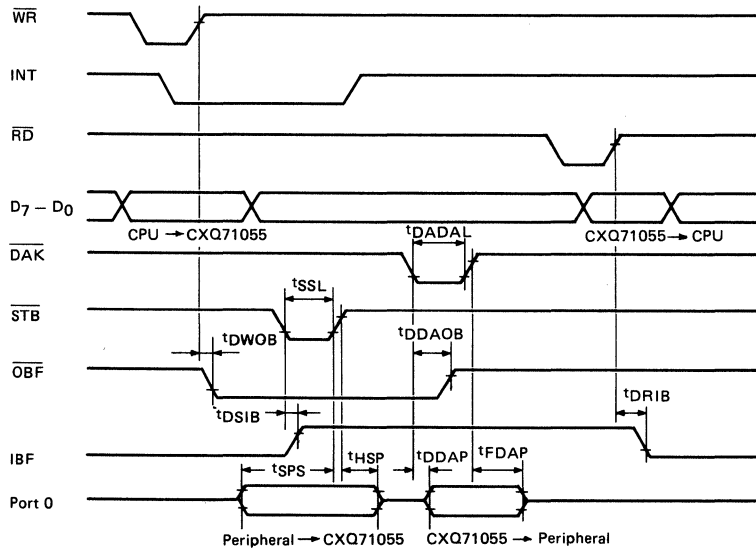
Mode 1: Input



Mode 1: Output



Mode 2



Functional Description

Ports 0, 1, 2

The CXQ71055 has three 8-bit I/O ports, referred to as port 0, port 1, and port 2. These ports are divided into two groups, group 0 and group 1. The groups can be programmed in one of three modes: mode 0, mode 1, and mode 2. Modes can be set independently for each group.

If group 0 is programmed in mode 0, port 0 and the four upper bits of port 2 belong to group 0; and port 1 and the four lower bits of port 2 belong to group 1. If group 0 is programmed in mode 1 or 2, port 0 and the 5 upper bits of port 2 belong to group 0; and port 1 and the three lower bits of port 2 belong to group 1.

Command Register

The CXQ71055 stores command words in this register. These commands control group 0 and group 1. Note that the contents of this register cannot be read.

Group 0 Control and Group 1 Control

These blocks control the operation of group 0 and group 1.

Read/Write Control

The read/write control controls the read/write operations for the ports and the data bus in response to the \overline{RD} , \overline{WR} , \overline{CS} , and address signals. It also handles RESET signals.

Data Bus Buffer

The data bus buffer latches information going to or from the system data bus.

CXQ71055 Commands

Two commands control CXQ71055 operation. The mode-select command determines the operation of group 0 and group 1 ports. The bit-manipulation command sets or resets the bits of port 2. These commands are executed by writing an 8-bit command word to the command register ($A_1A_0=11$).

Mode Select

The CXQ71055 port groups have three modes. Modes 0 and 1 can be specified for groups 0 and 1, but mode 2 can only be specified for group 0. The bits of all ports are cleared when a mode is selected.

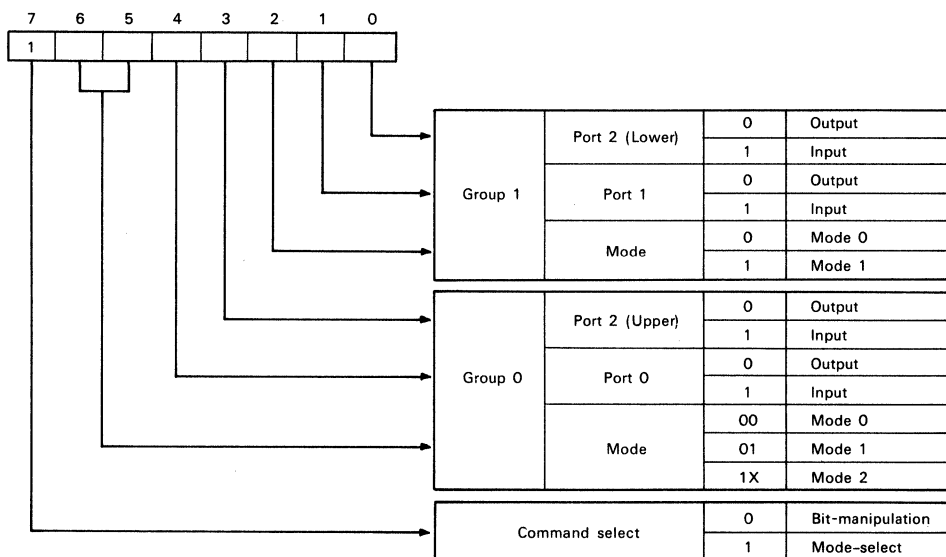
Mode 0. Basic input/output port operation.

Mode 1. Strobed input/output operation controlled by three or four bits of port 2 used as control/status signals.

Mode 2. (Only available for group 0). Port 0 is the bidirectional I/O port and the higher 5 bits of port 2 are used for status and control signals.

The mode is specified by writing the command (figure 1) to the command register.

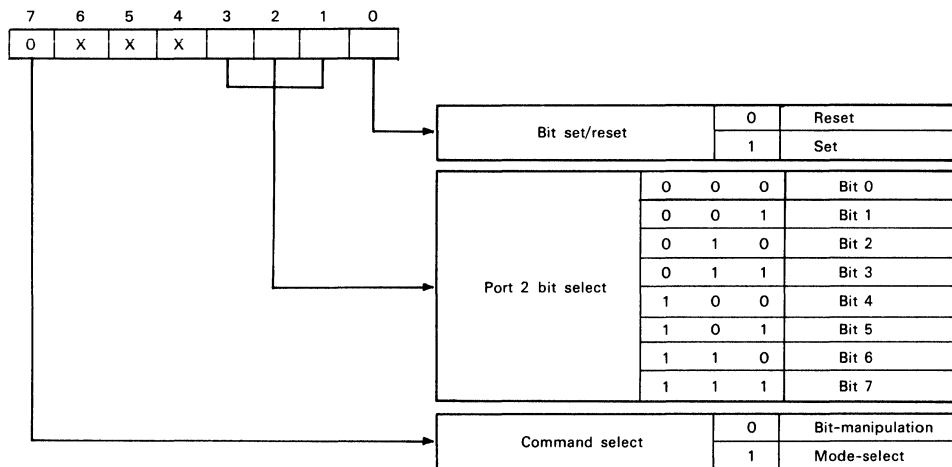
Figure 1. Mode-Select Command Format



Bit Manipulation Command

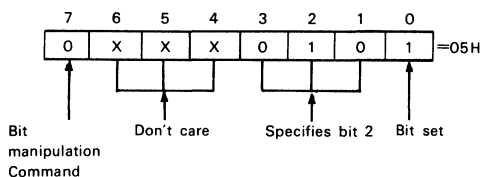
This command (figure 2) affects only port 2. It is mainly used in mode 1 and mode 2 to control the port 2 bits which are used as control/status signals. It is also used to enable and disable CXQ71055-generated interrupts and to set and reset Port 2 general input/output pins in mode 1.

Figure 2. Bit-Manipulation Command Format



For example, to set bit 2 of port 2 to 1 ($P2_2=1$), set the command word as shown in figure 3 (05H) in the command register.

Figure 3. Bit-Manipulation Command Example



Operation in Each Mode

The operation mode for each group in the CXQ71055 can be set according to the application. Group 0 can be programmed in modes 0,1, or 2, while group 1 is in mode 0 or 1.

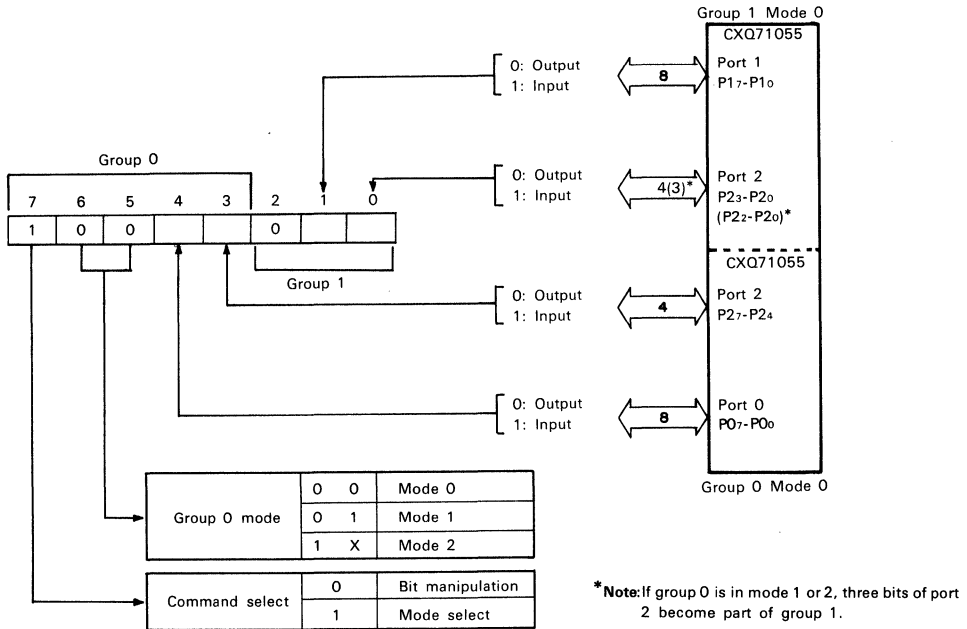
The \overline{RD} and \overline{WR} signals that appear in the descriptions of each mode refer to the port in question as addressed by A_1 and A_0 . These signals only affect the port addressed by A_1 and A_0 .

Where the group addressed may not be defined, 0 or 1 is appended to the signal name to indicate the group.

Mode 0

In this mode the ports of the CXQ71055 are used to perform basic I/O operations. Each port operates with a buffered input and a buffered, latched output. See figure 4.

Figure 4. Mode 0

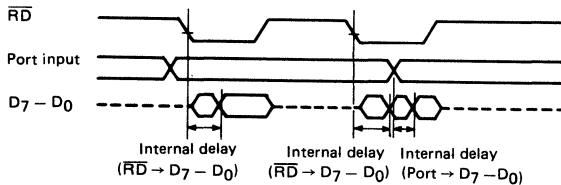


Depending on the command word sent to the CXQ71055 from the system bus, ports 0 and 1 and available bits of port 2 can be independently specified for input or output.

Input Port Operation

While the \overline{RD} signal is low, data from the port selected by the A_1A_0 signals is put on the data bus. See figure 5.

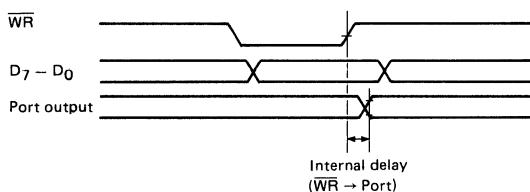
Figure 5. Mode 0 Input Timing



Output Port Operation

When data is written to the CXQ71055 ($\overline{WR}=0$), the data on the data bus will be latched in the port selected by the A_1A_0 signals at the rising edge of \overline{WR} and output to the port pins. See figure 6.

Figure 6. Mode 0 Output Timing



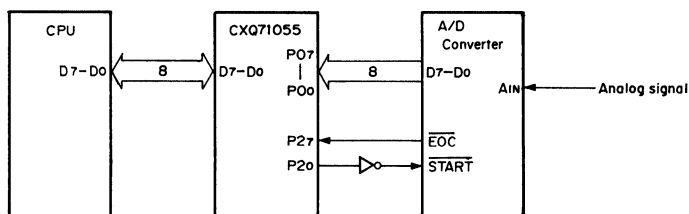
By reading a port which is programmed for output, the output value of the port can be obtained.

Note: When group 0 is in mode 1 or mode 2, only bits P22-P20 of port 2 can be used by group 1. Bits P23 belongs to group 0.

Mode 0 Example

This is an example of a CPU connected to an A/D converter via a CXQ71055. Here both group 0 and group 1 are set to mode 0 and port 2 is used to start conversion and detect the end of the conversion process.

Figure 7. A/D Converter Connection Example



This is a subroutine that reads the converted data from an A/D converter:

```

READ_A/D:  MOV     AL,10011000B    ; CXQ71055 Mode Setting:
           OUT     CTRLPORT, AL  ; Group 0, group 1 in mode 0
                                           ; Port 0 & port 2 (upper) are inputs
                                           ; Port 1 & port 2 (lower) are outputs

           MOV     AL, 00000001B
           OUT     CTRLPORT, AL  ; Conversion starts by setting P20 high
WAIT_EOC:  IN      AL, PORT2      ; End of conversion wait loop
           TEST1   AL, 7         ; Conversion ends when P27=0
           BNZ    WAIT_EOC
           IN      AL, PORT0     ; Read A/D converted values
           RET

```

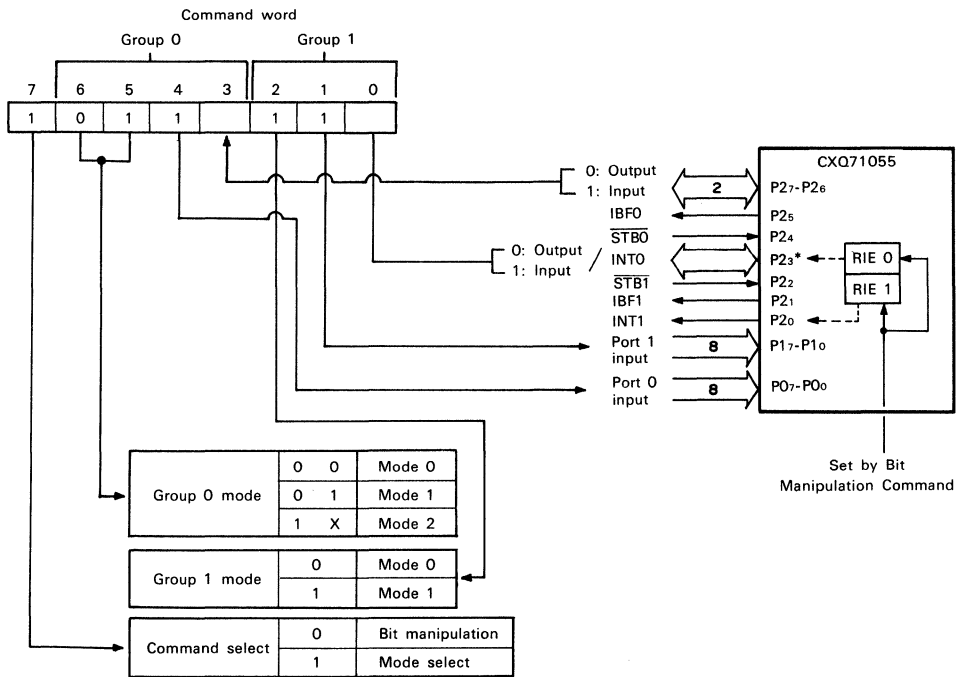
Mode 1

In this mode, the control and status signals control the I/O data. In group 0, port 0 functions as the data port and three of the upper five bits of port 2 function as control/status. In group 1, port 1 functions as the data port and the lower three bits of port 2 function as control/status. In mode 1, the bit-manipulation command is used to write the bits of port 2.

Group 0 Mode 1

When group 0 is used in mode 1, the upper five bits of port 2 become part of group 0. Of these five bits, three are used for control/status and the remaining two can be used for I/O (using the bit-manipulation command). See figure 8.

Figure 8. Mode 1 Input



* Note: Bit P23 is available in Group 1 only when Group 0 is Mode 0. For all other conditions P23 is part of Group 0. This diagram shows how bit P23 will be used if Group 1 is in Mode 1.

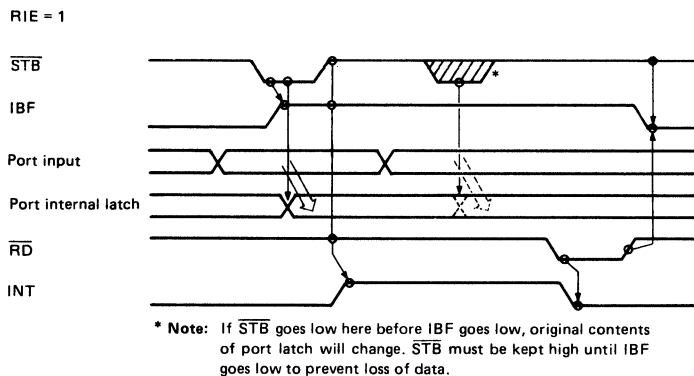
Group 1 Mode 1

When group 1 is used in mode 1, the lower three or four bits of port 2 become part of group 1. Of these four bits, three are used for control/status. The remaining bit, P23, can be used for I/O only if group 0 is in mode 0. Otherwise, P23 belongs to group 0 as a control/status bit. See figure 8.

Mode 1 Input Operation

In mode 1, port 0 is the data port for group 0, and port 1 for group 1. The control/status bits (port 2) are used as listed below. Figure 9 shows the signal timing.

Figure 9. Mode 1 Input Timing



\overline{STB} (Strobe)-Input ($\overline{STB0} \rightarrow P24$, $\overline{STB1} \rightarrow P22$). The data input at port 0 ($\overline{STB0}$) or port 1 ($\overline{STB1}$) is latched in port 0 or port 1 when \overline{STB} is brought low.

IBF (Input Buffer Full F/F)-Output ($IBF0 \rightarrow P25$, $IBF1 \rightarrow P21$). The IBF output goes high to indicate that the input buffer has become full. IBF goes high when the \overline{STB} signal goes low. IBF goes low at the rising edge of the \overline{RD} signal when $\overline{STB}=1$.

INT (Interrupt Request)-Output ($INT0 \rightarrow P23$, $INT1 \rightarrow P20$). INT goes high when the data is latched in the input port, when RIE is 1 and \overline{STB} , IBF and \overline{RD} are all high. INT goes low at the falling edge of the \overline{RD} signal. It can function as a data read request interrupt signal to a CPU.

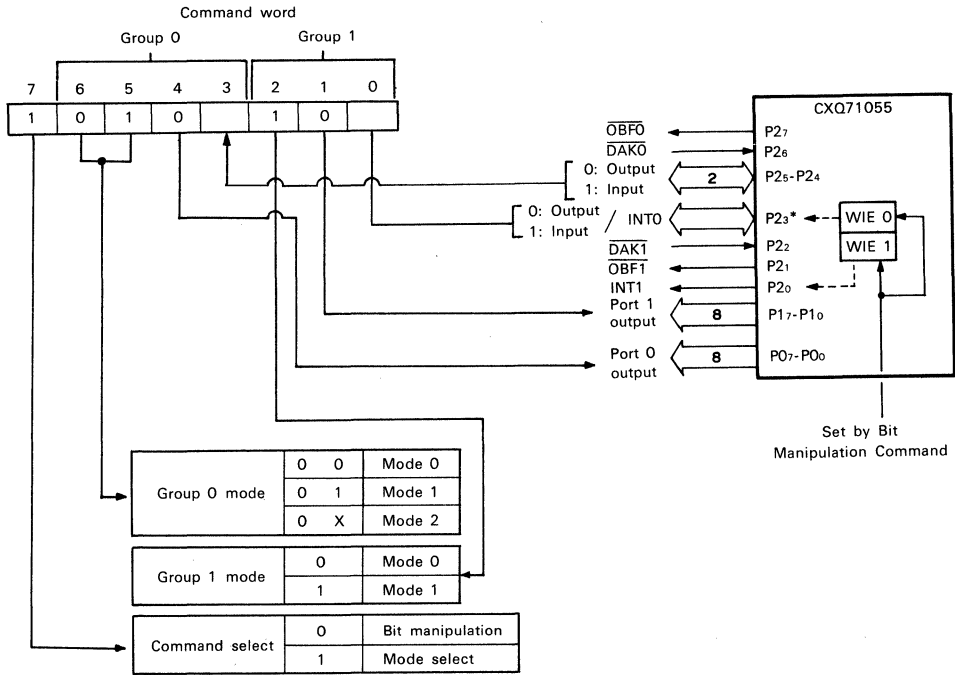
RIE (Read Interrupt Enable Flag)-(RIE0 \rightarrow P24, RIE1 \rightarrow P22). RIE controls the interrupt output. Interrupts can be enabled by using the bit-manipulation command to set this bit to 1, and disabled by resetting it to 0. This signal is internal to the CXQ71055 and is not an output. The state of RIE does not affect the function of \overline{STB} , which is addressed to the same bits of port 2.

When input is specified in mode 1, the status of IBF, INT and RIE can be read by reading the contents of port 2.

Mode 1 Output Operation

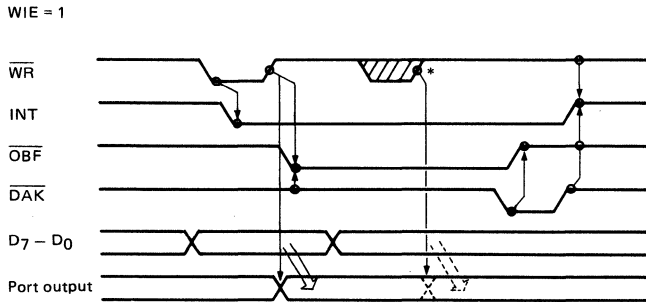
In mode 1 output operation (figure 10), the status/control bits (port 2) are used as listed below. Figure 11 shows the signal timing.

Figure 10. Mode 1 Output



*Note: Bit P23 is available in Group 1 only when Group 0 is Mode 0. For all other conditions P23 is part of Group 0. This diagram shows how bit P23 will be used if Group 1 is in Mode 1.

Figure 11. Mode 1 Output Timing



*Note: If data is written to the CXQ71055 before OBF goes high, the original contents of the port latch will change. Data must not be written while OBF is low to prevent loss of data.

$\overline{\text{OBF}}$ (Output Buffer Full F/F)-Output ($\overline{\text{OBF0}} \rightarrow \text{P27}$, $\overline{\text{OBF1}} \rightarrow \text{P21}$). $\overline{\text{OBF}}$ goes low when data is received from the CPU and is latched in the output port. It functions as a data receive flag. $\overline{\text{OBF}}$ goes low at the rising edge of $\overline{\text{WR}}$ when $\text{DAK}=1$ (write complete). It goes high when the $\overline{\text{DAK}}$ signal goes low.

$\overline{\text{DAK}}$ (Data Acknowledge)-Input ($\overline{\text{DAK0}} \rightarrow \text{P26}$, $\overline{\text{DAK1}} \rightarrow \text{P22}$). When this input is low, it signals the CXQ71055 that the peripheral device has received the output port data.

INT (Interrupt Request)-Output ($\text{INT0} \rightarrow \text{P23}$, $\text{INT1} \rightarrow \text{P20}$). INT goes high when the output data is taken when WIE is set to 1 and $\overline{\text{WR}}$, $\overline{\text{OBF}}$ and $\overline{\text{DAK}}$ are all high. It goes low at the falling edge of the $\overline{\text{WR}}$ signal. INT therefore functions as a write request signal to the CPU, indicating that it should send the next output data to the CXQ71055.

WIE (Write Interrupt Enable Flag)-(WIE0 \rightarrow P26, WIE1 \rightarrow P22). WIE controls the interrupt output. Interrupts can be enabled by using the bit-manipulation command to set this bit to 1 and disabled by resetting it to 0. This signal is internal to the CXQ71055 and is not an output. The state of WIE does not affect the function of $\overline{\text{DAK}}$ addressed to the same bits of port 2.

When output is specified in mode 1, the status of $\overline{\text{OBF}}$, INT and WIE can be obtained by reading the contents of port 2.

Table 2 shows a summary of these signals.

Table 2. Functions of Port 2 Bits in Mode 1

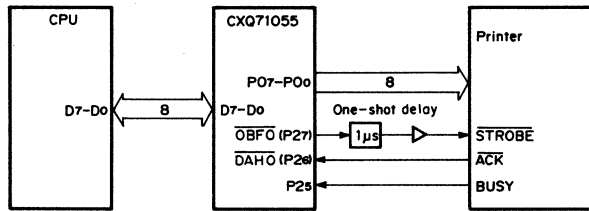
	Bit	Data Input	Data Output
Group 1	P20	INT1 (INTerrupt request)	INT1 (INTerrupt request)
	P21	IBF1 (Input Buffer Full f/f)	$\overline{\text{OBF1}}$ (Output Buffer Full f/f)
	P22	$\overline{\text{STB1}}$ (STroBe input) RIE1 (Read Interrupt Enable flag)	$\overline{\text{DAK1}}$ (Data AcKnowledge input) WIE1 (Write Interrupt Enable flag)
	P23	I/O*	I/O*
Group 0	P23	INT0 (INTerrupt request)	INT0 (INTerrupt request)
	P24	$\overline{\text{STB0}}$ (STroBe input) RIE0 (Read Interrupt Enable flag)	I/O
	P25	IBF0 (Input Buffer Full f/f)	I/O
	P26	I/O	$\overline{\text{DAK0}}$ (Data AcKnowledge input) WIE0 (Write Interrupt Enable flag)
	P27	I/O	$\overline{\text{OBF0}}$ (Output Buffer Full f/f)

*Note: Can be used only when group 0 is set to mode 0. In other modes, P23 belongs to group 0.

Mode 1 Example

This example (figure 12) demonstrates connecting a printer with the CXQ71055. Group 0 is used in Mode 1. Group 1 can operate in mode 0 or 1; in this example it is set to mode 0.

Figure 12. Connection to Printer



INIT:	MOV	AL,10101000B	; CXQ71055 Mode Setting:
			; Group 0: mode 1 output, Port 2 input
			; Group 1: mode 0
	OUT	CTRLPORT, AL	
	RET		
SENDPRN:	MOV	BW, DATA	; Output data address
PRNLOOP:	MOV	AL, [BW]	
	CMP	AL, 0FFH	; End if data=0FFH
	BNZ	WAIT	
	RET		
WAIT:	IN	AL, PORT2	
	TEST1	AL, 7	; Wait until output buffer is empty
	BZ	WAIT	
	TEST1	AL, 5	; Wait until printer can accept data
	BNZ	WAIT	
	MOV	AL, [BW]	; Send data to printer
	OUT	PORT0, AL	
	INC	BW	
	BR	PRNLOOP	

Mode 2

Mode 2 can only be used by group 0. In this mode, port 0 functions as a bidirectional 8-bit data port operating under the control of the upper five bits of port 2 as control/status signals. In this mode, port 0 combines the input and output operations of mode 1. See figures 13 and 14.

In mode 2, the status of the following signals can be determined by reading port2: $\overline{\text{OBFO}}$, IBFO, INTO, WIEO, and RIEO.

Figure 13. Mode 2

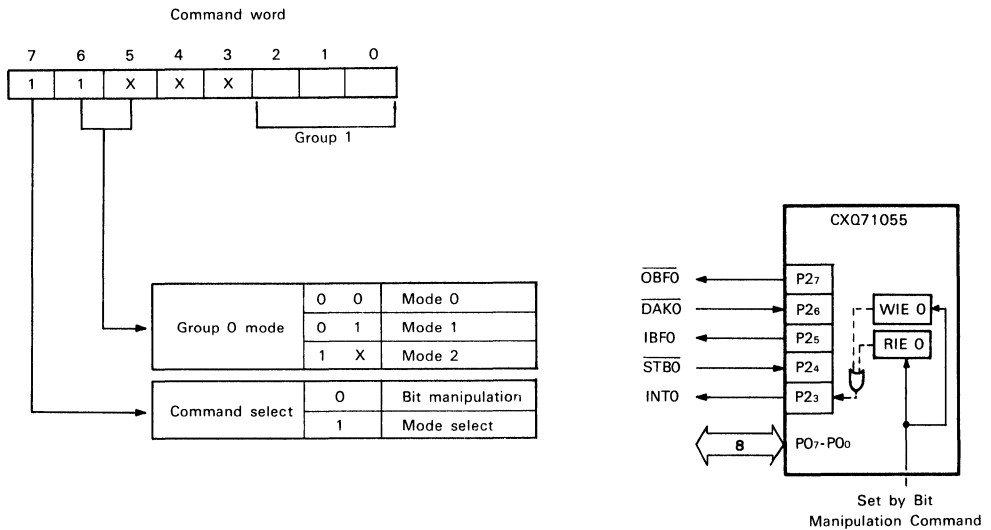
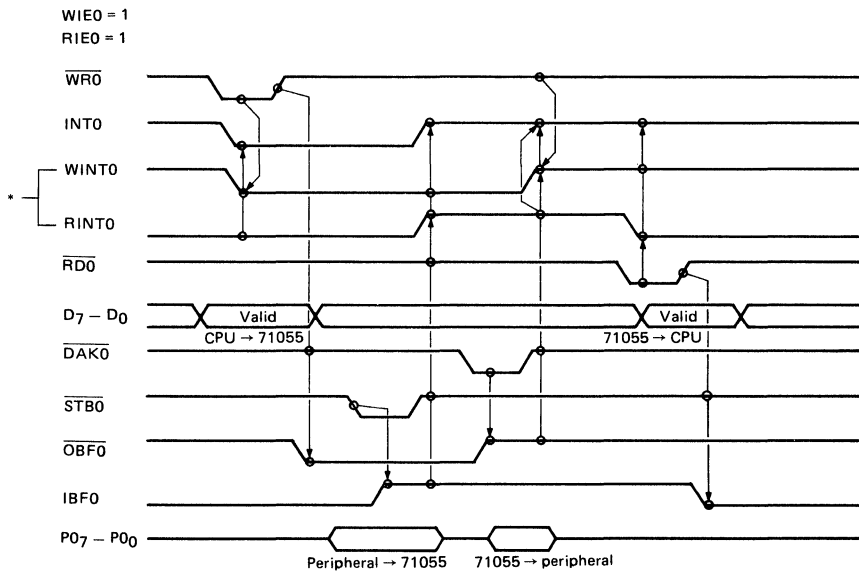


Figure 14. Mode 2 Timing



***Note:** WINT0 and RINT0 are internal signals and are write and read interrupt request signals to the CPU, respectively.
 $WINT0 = OBF0 \cdot WIE0 \cdot DAK0 \cdot WR0$
 $RINT0 = IBF0 \cdot RIE0 \cdot STB0 \cdot RD0$
 Also note that
 $INT0 = WINT0 + RINT0$

The $\overline{DAK0}$ and $\overline{STB0}$ signals are used to select input or output for port 0. By using these signals, bidirectional operation between the CXQ71055 and peripheral can be realized.

In mode 2, the bit-manipulation command is used to write to port 2.

Control/Status Port Operation

The following control/status signals are used for output operation of port 0:

$\overline{OBF0}$ (Output Buffer Full)-Output (P27). $\overline{OBF0}$ goes low when data is received from the D0-D7 data bus and is latched in the port 0 output buffer. It therefore functions as a receive request signal to the peripheral. $\overline{OBF0}$ goes low at the rising edge of the $\overline{WR0}$ signal (end of data write). It goes high when $\overline{DAK0}$ is low (output data from port 0 received).

$\overline{DAK0}$ (Data Acknowledge)-Input (P26). $\overline{DAK0}$ is sent to the CXQ71055 in response to the $\overline{OBF0}$ signal. It should be set low when peripheral reads data from port 0 of the CXQ71055, which causes the three-state output buffer to be in output state.

WIE0 (Write Interrupt Enable Flag)-Output (P26). WIE0 controls the write interrupt request output. Interrupts are enabled by using the bit-manipulation command to set this bit to 1 and disabled by resetting it to 0. The state of WIE does not affect the \overline{DAK} function of this pin.

The following control/status signals are used for input operations of port 0:

$\overline{STB0}$ (Strobe Input)-Input (P24). When $\overline{STB0}$ goes low, the data sent to the CXQ71055 is latched in port 0.

IBF0 (Input Buffer Full F/F)-Output (P25). When IBF0 goes high, it indicates that the input buffer is full. It functions as a signal which can be used to prohibit further data transfer. IBF0 goes high when $\overline{STB0}$ goes low. It goes low at the rising edge of $\overline{RD0}$ when $\overline{STB0}=1$ (read complete).

RIE0 (Read Interrupt Enable Flag)-Output (P24). RIE0 controls the read interrupt request output. Interrupts are enabled by using the bit-manipulation command to set this bit to 1 and disabled by resetting it to 0. The state of RIE0 does not affect the $\overline{STB0}$ function of this pin.

This control/status signal is used for both input and output operations:

INT0 (Interrupt Request)-Output (P23). During input operations, INT0 functions as a read request interrupt signal to the CPU. During output, it functions as a write request interrupt signal to the CPU. This signal is the logical OR of the INT signal for data read (RINT0) and the INT signal for write (WINT0) in mode 1 (RINT0 OR WINT0).

Table 3 is a summary of these signals.

Table 3. Functions of Port 2 in Mode 2

Bit	Function
P23	INT0 (INTerrupt request)
P24	$\overline{STB0}$ (STroBe input) RIE0 (Read Interrupt Enable flag)
P25	IBF0 (input Buffer Full f/f)
P26	$\overline{DAK0}$ (Data AcKnowledge input) WIE0 (Write Interrupt Enable flag)
P27	$\overline{OBF0}$ (Output Buffer Full f/f)

Mode 2 Example

Figures 15, 16, and 17 show data transfer between two CPUs.

Figure 15. Connecting Two CPUs

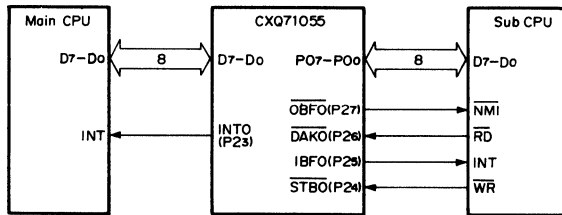


Figure 16. Main CPU Flowchart

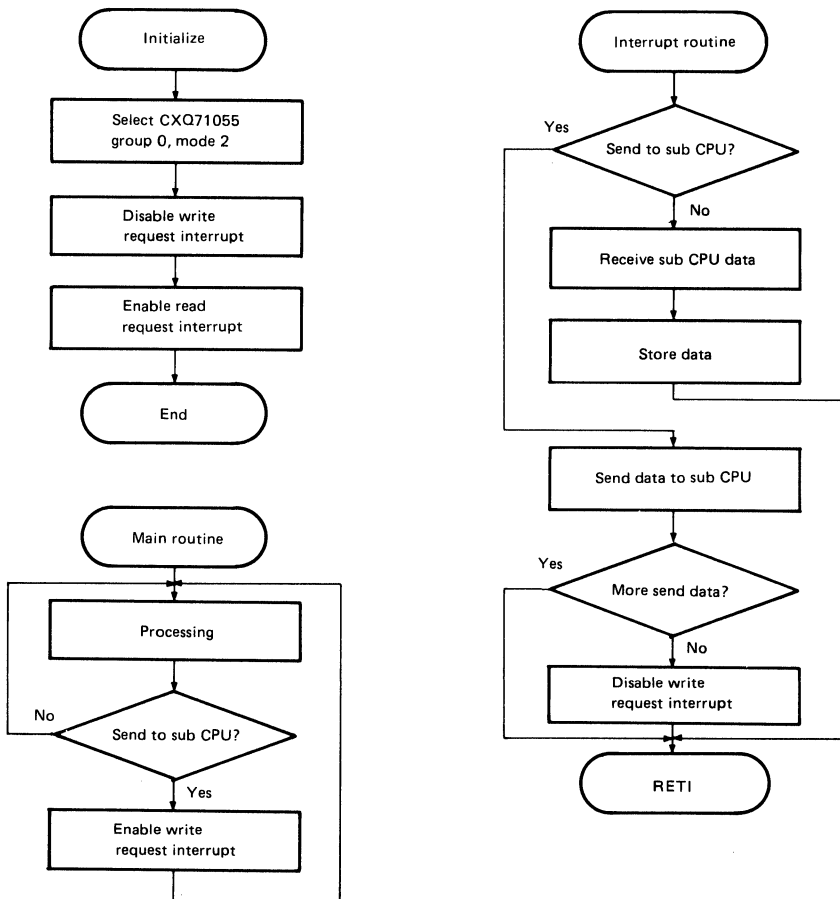
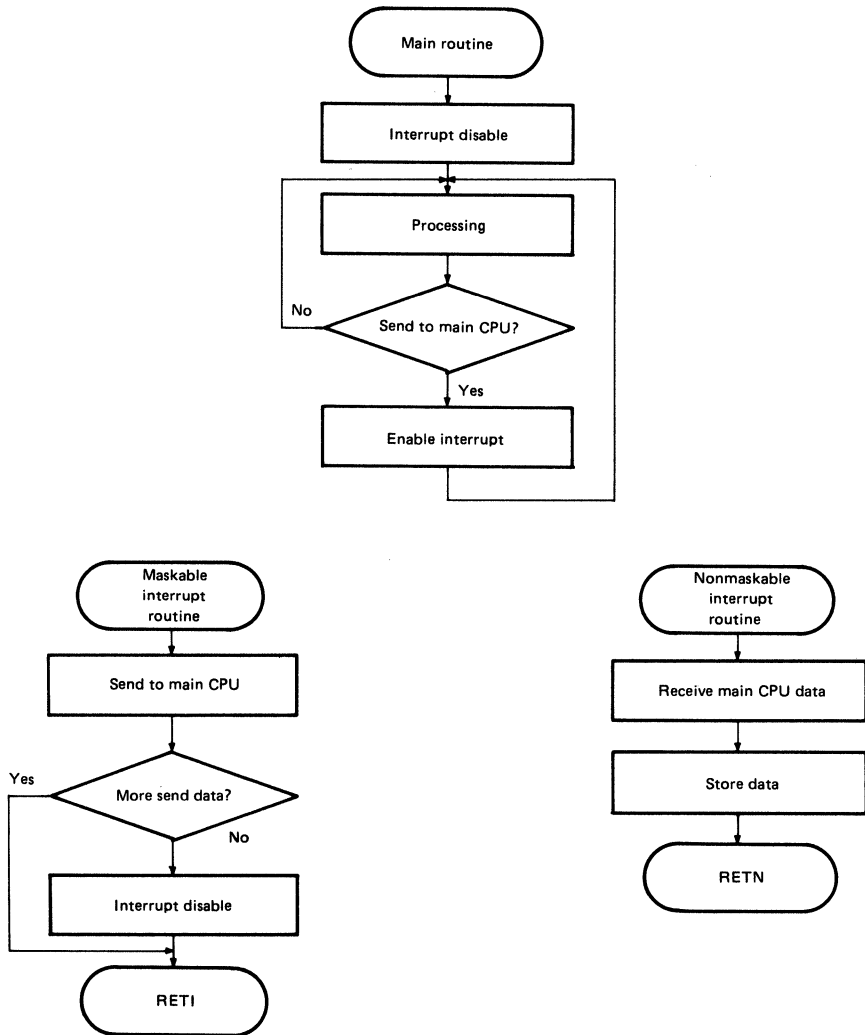


Figure 17. Sub CPU Flowchart



Mode Combinations

Table 4 is a complete list of all the combinations of modes and groups, and the function of the port 2 bits in each mode.

Table 4. Mode Combinations and Port 2 Bit Functions

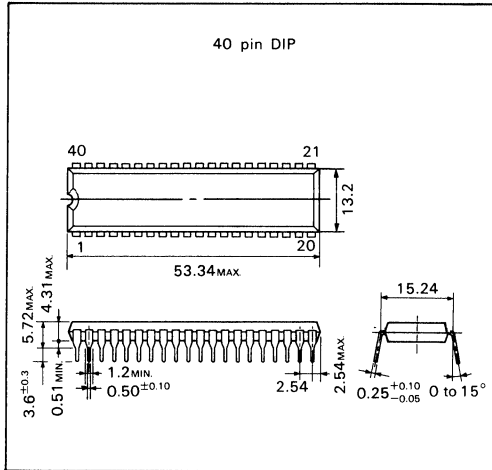
Group 0		Group 1		Port 2							
Mode	Port 0	Mode	Port 1	P27	P26	P25	P24	P23	P22	P21	P20
0	Input	0	Input	D	D	D	D	D	D	D	D
				B							
0	Input	0	Output	D	D	D	D	D	D	D	D
				B							
0	Input	1	Input	D	D	D	D	D	RIE1 STB1	IBF1	INT1
				B							
0	Input	1	Output	D	D	D	D	D	WIE1 DAK1	ÖBF1	INT1
				B							
0	Output	0	Input	D	D	D	D	D	D	D	D
				B							
0	Output	0	Output	D	D	D	D	D	D	D	D
				B							
0	Output	1	Input	D	D	D	D	D	RIE1 STB1	IBF1	INT1
				B							
0	Output	1	Output	D	D	D	D	D	WIE1 DAK1	ÖBF1	INT1
				B							
1	Input	0	Input	D	D	IBF0	RIE0 STB0	INT0	D	D	D
				B							
1	Input	0	Output	D	D	IBF0	RIE0 STB0	INT0	D	D	D
				B							
1	Input	1	Input	D	D	IBF0	RIE0 STB0	INT0	RIE1 STB1	IBF1	INT1
				B							
1	Input	1	Output	D	D	IBF0	RIE0 STB0	INT0	WIE1 DAK1	ÖBF1	INT1
				B							

1	Output	0	Input	OBFO	WIEO DAKO	D	D	INTO	D	D	D
				B							
1	Output	0	Output	OBFO	WIEO DAKO	D	D	INTO	D	D	D
				B							
1	Output	1	Input	OBFO	WIEO DAKO	D	D	INTO	RIE1 STB1	IBF1	INT1
				B					B		
1	Output	1	Output	OBFO	WIEO DAKO	D	D	INTO	WIE1 DAK1	OB F 1	INT1
				B					B		
2	I/O	0	Input	OBFO	WIEO DAKO	IBFO	RIEO STBO	INTO	D	D	D
				B							
2	I/O	0	Output	OBFO	WIEO DAKO	IBFO	RIEO STBO	INTO	D	D	D
				B							
2	I/O	1	Input	OBFO	WIEO DAKO	IBFO	RIEO STBO	INTO	RIE1 STB1	IBF1	INT1
				B					B		
2	I/O	1	Output	OBFO	WIEO DAKO	IBFO	RIEO STBO	INTO	WIE1 DAK1	OB F 1	INT1
				B					B		

- Notes:**
1. In this chart, "D" indicates data that is used by the user.
 2. The symbol "B" indicates bits that can only be rewritten by the bit-manipulation command.
 3. Shaded area belongs to group 1.

Package Outline

Unit: mm



Interrupt Control Unit

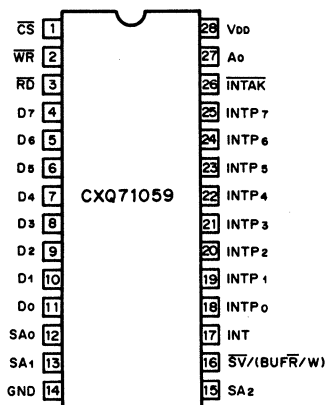
Description

The CXQ71059 is a low-power CMOS programmable interrupt control unit for microcomputer systems. It can process eight interrupt requests and can be expanded to 64 interrupt requests by adding other CXQ71059s. It transfers the interrupt request with the highest priority to the CPU, along with the interrupt address information.

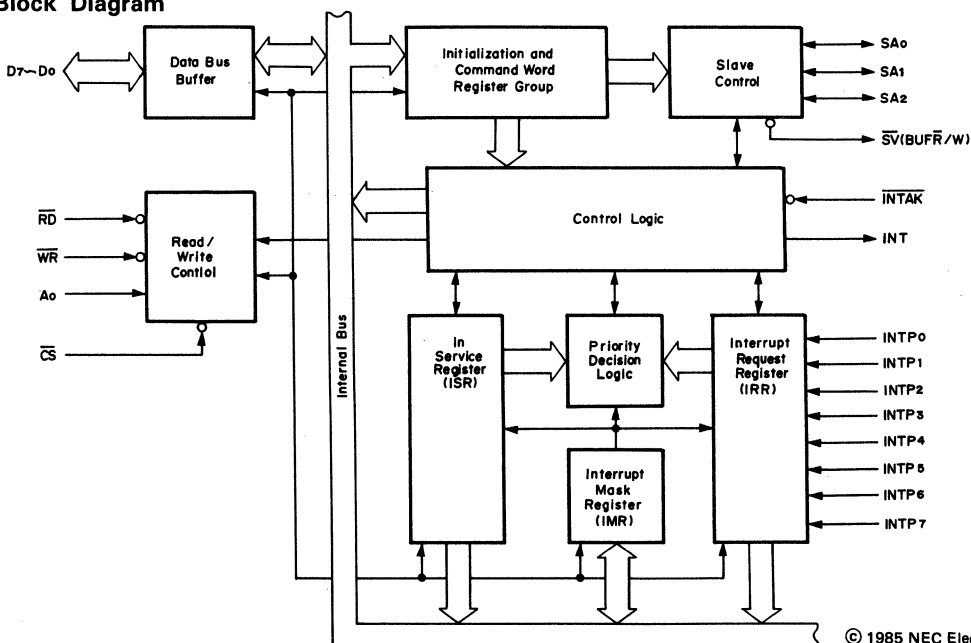
Features

- 8085A compatible (CALL mode)
- CXQ70108/70116 compatible (vector mode)
- Eight interrupt request inputs per chip
- Up to 64 interrupt requests inputs per system (extended mode)
- Edge- or level-triggered interrupt request inputs
- Each interrupt maskable
- Programmable priority level
- Polling operation
- CMOS technology
- +5V single power supply
- 28-pin plastic DIP (600 mil)
- NEC μ PD71059 compatible

Pin Configuration (Top View)



Block Diagram



© 1985 NEC Electronics

Pin Identification

No.	Symbol	Direction	Function
1	\overline{CS}	In	Chip select input
2	\overline{WR}	In	Write strobe input
3	\overline{RD}	In	Read strobe input
4-11	D7–D0	In/Out	Data bus
12-13	SA0–SA1	In/Out	Slave address, bits 0, 1
14	GND		Ground potential
15	SA2	In/Out	Slave address, bit 2
16	$\overline{SV}/(\overline{BUFR}/W)$	In/Out	Slave (Buffer read write)
17	INT	Out	Interrupt output
18-25	INTP0–INTP7	In	Interrupt inputs
26	\overline{INTAK}	In	Interrupt acknowledge input
27	A0	In	Address input
28	V _{DD}		Power supply

Pin Functions

D7–D0 [Data Bus]

The 8-bit, three-state bidirectional bus is used to interface the 71059 to the system bus. The control words, status information, and interrupt-vector data are transferred.

\overline{CS} [Chip Select]

The CPU sets \overline{CS} low when selecting CXQ71059 to read from (IN instructions) or write to (OUT instructions). The \overline{RD} and \overline{WR} signals to the CXQ71059 are enabled when \overline{CS} is low. \overline{CS} is not used for the INTAK sequence.

\overline{RD} [Read Strobe]

The CPU sets \overline{RD} low when reading the internal registers IMR, IRR and ISR, and the polling data. during polling operations.

\overline{WR} [Write Strobe]

The CPU sets \overline{WR} low when writing initializing words IW1–IW4 and command words IMW, PFCW and MCW.

A0 [Address]

A0 is used with \overline{CS} , \overline{RD} , and \overline{WR} to read or write to the CXQ71059. Normally, A0 is connected to A0 of the system address bus. Table 1 shows the relationship between read/write operations and the control signals (\overline{CS} , \overline{WR} , \overline{RD} and A0).

Table 1. Read/Write Operations

\overline{CS}	\overline{RD}	\overline{WR}	A_0	Other Conditions	CXQ71059 Operation	CPU Operation
0	0	1	0	IRR set by MCW	IRR to Data bus	IRR read
				ISR set by MCW	ISR to Data bus	ISR read
				Polling phase ¹	Polling data to Data Bus	Polling
0	0	1	1		IMR to Data bus	IMR read
0	1	0	0	D ₄ =1	Data bus to IW1 register	IW1 write
				D ₄ , D ₃ =0	Data bus to PFCW register	PFCW write
				D ₄ =0, D ₃ =1	Data bus to MCW register	MCW write
0	1	0	1	Note 2	Data bus to IW2 register	IW2 write
					Data bus to IW3 register	IW3 write
					Data bus to IW4 register	IW4 write
				After initializing	Data bus to IMR	IMW write
0 1	1 X	1 X	X X		Data bus: high impedance	
0	0	0	X		Illegal	

- Notes:** 1. In the polling phase, polling data is read instead of IRR and ISR.
2. Refer to Control Words section for IW2–IW4 writing procedure.

INTP₇–INTP₀ [Interrupt Request from Peripheral]

INTP₇–INTP₀ are eight asynchronous interrupt request inputs. They can be set to be either edge- or level-triggered. These pins are pulled up by an internal resistance. Their power consumption is lower at high-level input than at low-level input.

INT [Interrupt]

INT is the interrupt request output from a CXQ71059 to the CPU or master CXQ71059. When an interrupt from a peripheral is input to an INTP pin and acknowledged, the CXQ71059 asserts INT high to generate an interrupt request to the CPU or master CXQ71059.

INTAK [Interrupt Acknowledge]

INTAK informs the CXQ71059 that its interrupt request is being acknowledged by the CPU. During this acknowledgement, the CPU returns three low-level pulses (CALL mode) or two low-level pulses (vector mode).

$\overline{SV}/(\overline{BUF\overline{R}/W})$ [Slave, Buffer Read/Write]

This pin has two functions. When in non-buffer mode, it is the \overline{SV} input to designate a slave ($\overline{SV}=0$) or master ($\overline{SV}=1$). \overline{SV} has no meaning when the CXQ71059 is set to signal mode.

This signal allows a bus transceiver to be controlled by the CXQ71059, if buffer mode is used. This pin becomes a $\overline{BUF\overline{R}/W}$ output. When the CXQ71059 changes its data bus to output, $\overline{BUF\overline{R}/W}$ goes low. $\overline{BUF\overline{R}/W}$ goes high when the data bus changes to input.

SA₂–SA₀ [Slave Address]

These pins are only used in systems with cascaded CXQ71059s. The master CXQ71059 uses these pins to address up to eight slave CXQ71059s. These pins are output pins for masters, and input pins for slaves.

Note: In the single mode, SA₂–SA₀ are output pins, but the output data has no meaning.

V_{DD} [Power Supply]

This is the positive power supply.

GND [Ground]

This is the ground potential.

Absolute Maximum Ratings

T_a=25°C

Parameter	Symbol	Rating Value	Unit
Power supply voltage	V _{DD}	−0.5 to +7.0	V
Input voltage	V _I	−0.5 to V _{DD} +0.3	V
Output voltage	V _O	−0.5 to V _{DD} +0.3	V
Power dissipation	P _{DMAX}	500	mW
Operating temperature	T _{opr}	−40 to +85	°C
Storage temperature	T _{stg}	−65 to +150	°C

Comment: Exposing the device to stresses above those listed in the absolute maximum ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational section of this specification. Exposure to absolute maximum ratings for extended periods may affect device reliability.

DC Characteristics

T_a=−40 to +85°C; V_{DD}=5V±10%

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Input voltage high	V _{IH}	2.2	V _{DD} +0.3	V	
Input voltage low	V _{IL}	−0.5	0.8	V	
Output voltage high	V _{OH}	0.7×V _{DD}		V	I _{OH} =−400 μA
Output voltage low	V _{OL}		0.4	V	I _{OL} =2.5 mA
Input leakage current high	I _{LIH}		10	μA	V _I =V _{DD}
Input leakage current low	I _{LIL}		−10	μA	V _I =0V
Output leakage current high	I _{LOH}		10	μA	V _O =V _{DD}
Output leakage current low	I _{LOL}		−10	μA	V _O =0V
INTP input leakage current high	I _{LIPH}		10	μA	
INTP input leakage current low	I _{L IPL}		−300	μA	
Supply current	I _{DD1}		9	mA	Operation
	I _{DD2}		50	μA	Stand-by Mode

Capacitance

Ta=25°C; VDD=0V

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Input capacitance	C _I		10	pF	f _c =1 MHz Unmeasured pins returned to 0V
I/O capacitance	C _{IO}		20	pF	

AC Characteristics (Ta=-40 to +85°C; VDD=5V±10%)**Read Timing**

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Ao, \overline{CS} set-up to \overline{RD} ↓	t _{SAR}	0		ns	
Ao, \overline{CS} hold from \overline{RD} ↑	t _{HRA}	0		ns	
\overline{RD} pulse width low	t _{RRL}	160		ns	
\overline{RD} pulse width high	t _{RRH}	120		ns	
Data delay from \overline{RD} ↓	t _{DRD}		120	ns	C _L =150 pF
Data float from \overline{RD} ↑	t _{FRD}	10	85	ns	C _L =100 pF
Data delay from Ao, \overline{CS}	t _{DAD}		200	ns	C _L =150 pF
BUFR/W delay from \overline{RD} ↓	t _{DRBL}		100	ns	
BUFR/W delay from \overline{RD} ↑	t _{DRBH}		150	ns	

Write Timing

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Ao, \overline{CS} set-up to \overline{WR} ↓	t _{SAW}	0		ns	
Ao, \overline{CS} hold from \overline{WR} ↑	t _{HWA}	0		ns	
\overline{WR} pulse width low	t _{WWL}	120		ns	
\overline{WR} pulse width high	t _{WWH}	120		ns	
Data set-up to \overline{WR} ↑	t _{SDW}	120		ns	
Data hold from \overline{WR} ↑	t _{HWD}	0		ns	

Interrupt Timing

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
INTP pulse width	tIPIPL	100		ns	See note
SA set-up to second, third $\overline{\text{INTAK}} \downarrow$	tsSIA	40		ns	Slave
$\overline{\text{INTAK}}$ pulse width low	tIAIAL	160		ns	
$\overline{\text{INTAK}}$ pulse width high	tIAIAH	120		ns	$\overline{\text{INTAK}}$ Sequence
INT delay from INTP \uparrow	tDIPI		300	ns	$C_L=150$ pF
SA delay from first $\overline{\text{INTAK}} \downarrow$	tDIAS		360	ns	Master, $C_L=150$ pF
Data delay from $\overline{\text{INTAK}} \downarrow$	tDIAD		120	ns	$C_L=150$ pF
Data float from $\overline{\text{INTAK}} \uparrow$	tFIAD	10	85	ns	$C_L=100$ pF
Data delay from SA	tDSD		200	ns	Slave, $C_L=150$ pF
BUFR/W delay from $\overline{\text{INTAK}} \downarrow$	tDIABL		100	ns	$C_L=150$ pF
BUFR/W delay from $\overline{\text{INTAK}} \uparrow$	tDIABH		150	ns	

Note: The time to clear the input latch in edge-trigger mode.

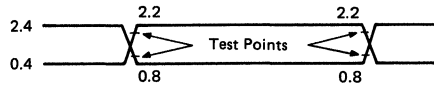
Other Timing

Parameter	Symbol	Min.	Max.	Unit	Test Conditions
Command recovery time	trV1	120		ns	Note 1
$\overline{\text{INTAK}}$ recovery time	trV2	250		ns	Note 2
$\overline{\text{INTAK}}$ /command recovery time	trV3	250		ns	Note 3

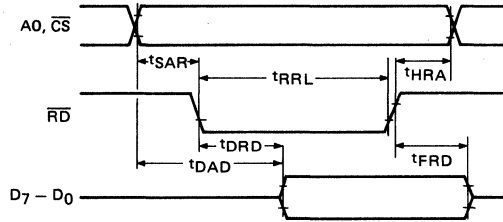
- Notes:**
1. The time to move from read to write operation.
 2. The time to move $\overline{\text{INTAK}}$ to the next $\overline{\text{INTAK}}$ operation.
 3. The time to move $\overline{\text{INTAK}}$ to/from command (read/write).

Timing Waveforms

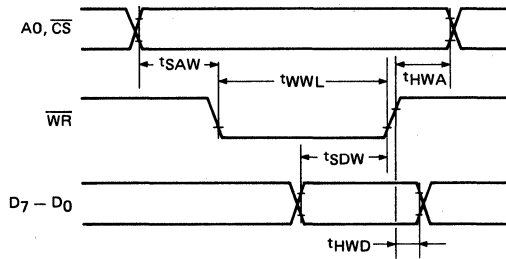
AC Test Input/Output Waveform



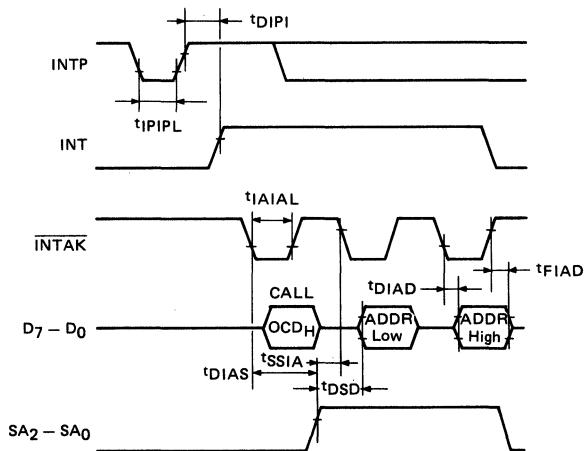
Read Cycle



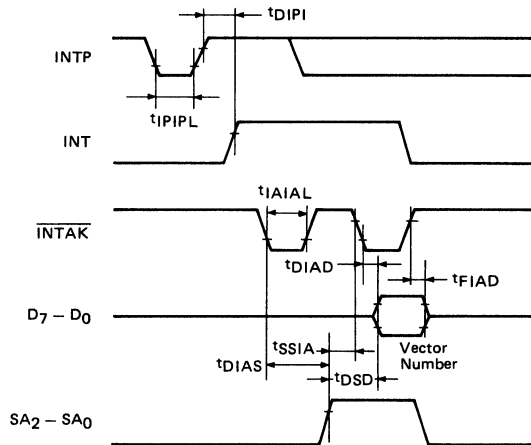
Write Cycle



INTAK Sequence (CALL Mode) 8085A

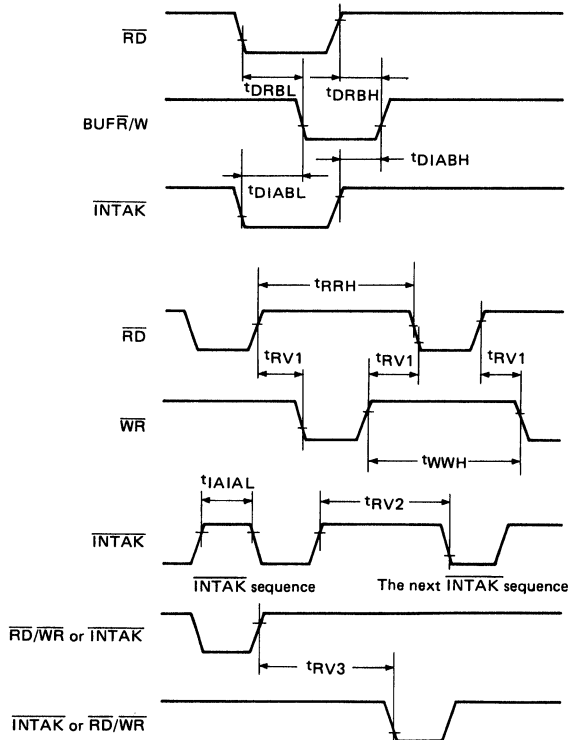


INTAK Sequence (Vector Mode) CXQ70108/70116



INTP Input should be maintained at high level until the leading edge of the 1st INTAK pulse.

Other Timing



Block Diagram Functions

Data Bus Buffer

The data bus buffer is a buffer between D7–D0 and the CXQ71059's internal bus.

Read/Write Control

The read/write control controls the reading from and writing to the CXQ71059 registers.

Initialization and Command Word Registers

These registers store initializing words IW1–IW4 and command words PFCW (priority and finish control word) and MCW (mode control word). The CPU cannot read these registers.

Interrupt Mask Register (IMR)

The interrupt mask register stores the interrupt mask word (IMW). Each bit masks an interrupt. If bit n of this register is 1, the interrupt request $INTP_n$ is masked and cannot be accepted by the CXQ71059. The CPU can read this register by performing an IN instruction with $A_0=1$.

Interrupt Request Register (IRR)

The interrupt request register shows which interrupt levels are currently being requested. If bit n of the IRR is 1, $INTP_n$ is requesting service. The CPU can read this register.

In-Service Register (ISR)

The in-service register shows all interrupt levels currently in service. If bit n of this register is 1, the interrupt routine corresponding to $INTP_n$ is currently being executed. The CPU can read this register.

Priority Decision Logic

The priority decision logic decides the highest priority interrupt request in IRR. The decision is based on the state of IMR, ISR, and the mode setting.

Control Logic

The control logic receives and generates the signals that control the sequence of events in an interrupt.

Slave Control

Slave control is used in systems with cascaded CXQ71059s. A master CXQ71059 uses it to control slave CXQ71059s, and a slave uses it to interface with the master CXQ71059.

Interrupt Operation

Almost all microcomputer systems use interrupts to reduce the overhead when controlling peripherals. However, the number of interrupt pins on a CPU is limited. When the number of interrupt lines increases beyond that limit, external circuits like the CXQ71059 become necessary.

The CXQ71059 can process eight interrupt request according to an allocated priority order and transmit the signal with the highest priority to the CPU. It also supplies the CPU with information to ascertain the interrupt routine start address. Cascading CXQ71059s by connecting up to eight "slave" CXQ71059s to a single "master" CXQ71059 permits expansion up to 64 interrupt request signals.

Interrupt system scale, interrupt routine addresses, interrupt request priority, and interrupt request masking are all programmable, and can be defined by the CPU.

Normal interrupt operation for a single CXQ71059 is as follows. First, the initialization registers are set-up by a sequence of initialization words. When the CXQ71059 detects an interrupt request from a peripheral to an INTP pin, it sets the corresponding bit of the interrupt request register (IRR). The interrupt is checked with the interrupt mask register (IMR) and the interrupt service register (ISR). If the interrupt is not masked and there is no other interrupt with a higher priority in service or requesting service, it generates an INT signal to the CPU.

The CPU acknowledges the interrupt by bringing the $\overline{\text{INTAK}}$ line low. The CXQ71059 sets the corresponding bit in its ISR to indicate that this interrupt is in service and to disable interrupts with lower priority. It resets the bit in the IRR at this point. The CXQ71059 then peaces interrupt CALL or vector data onto the data bus in response to $\overline{\text{INTAK}}$ pulses. When the CPU has finished processing the interrupt, it will inform the CXQ71059 by sending a finish interrupt (FI) command. This resets the bit in the ISR and allows the CXQ71059 to accept interrupts with lower priorities. If the CQ71059 is programmed in the self-FI mode, the ISR bit is reset automatically and this step is not necessary.

Software Features

The CXQ71059 has the following software features:

- Interrupt types: CALL/vector
- Interrupt masking: Normal/extended nesting
- End of interrupt: Self-FI/normal FI/specific FI
- Priority rotation: Normal nested/extended nested/exceptional nested
Automatic priority rotation
Rotate to specific priority
- Polled mode
- CPU-readable registers

Hardware Configurations

The CXQ71059 has the following hardware configurations:

- Interrupt input: Edge/level sensitive
- Cascading CXQ71059s: Single/extended (master/slave)
- Output driver control: Buffered/non-buffered

Mode Control

These features and configurations are selected and controlled by the four initialization words (IW1—IW4) and the three command words (IMW, PFCW, and MCW). The format of these words are shown in figures 2 and 3, respectively.

Control Words

These are two types of CXQ71059 control words: initialization words and command words.

There are four initialization words: IW1–IW4. These words must be written to the CXQ71059 to initialize it prior to normal operation. They must be written in a sequence of two to four bytes.

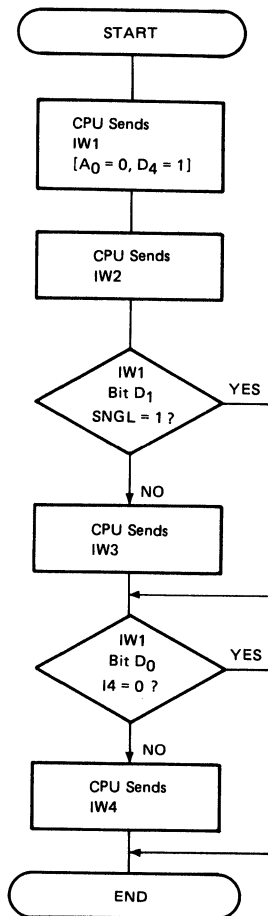
There are three types of command words: interrupt mask word (IMW), priority and finish control word (PFCW), and the mode control word (MCW). These words can be written freely after initialization.

Initialization Words

Initialization sequence. When a data is written to a CXQ71059 with $A_0=0$ and $D_4=1$, the data is always accepted as IW1. This results in a default initialization as shown below. See figure 1.

- (1) The edge-trigger circuit of the INTP input is reset. IRR is cleared in the edge-trigger mode.
- (2) ISR and IMR are cleared.
- (3) INTP₇ receives the lowest priority; INTP₀ receives the highest.
- (4) The exceptional nesting mode is cleared. IRR is set as the register to be read.
- (5) Register IW4 is cleared. The normal nesting mode, non-buffer mode, FI command mode, and CALL mode are entered.

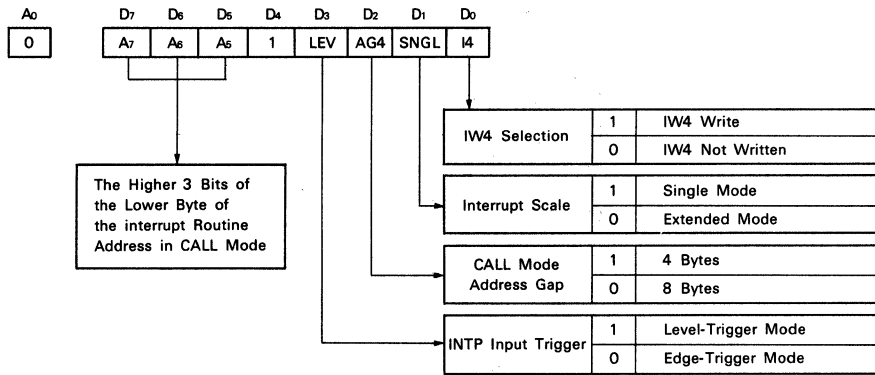
Figure 1. Initialization Sequence



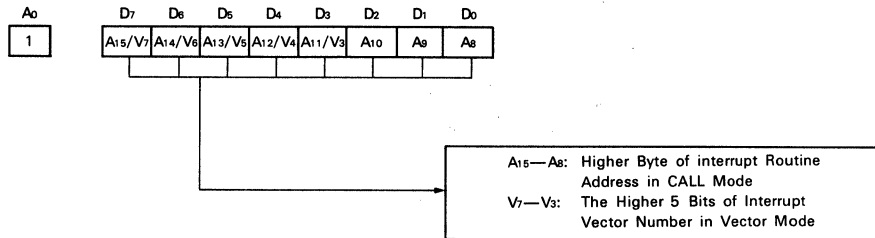
The initialization words are written in a consecutive format. The first two, IW1 and IW2 must be written for any mode of the operation, and designate the interrupt address or vector. IW3 specifies which interrupts have slaves if the CXQ71059 is used for a master, and defines the slave number for a slave. Therefore, IW3 is only required in extended systems. The CXQ71059 will only accept it if bit D1 of IW1, SNGL=0. IW4 is only accepted if bit D0 of IW1, I4=1. See figure 2 for the format of the initialization words.

Figure 2. Initialization Word Format

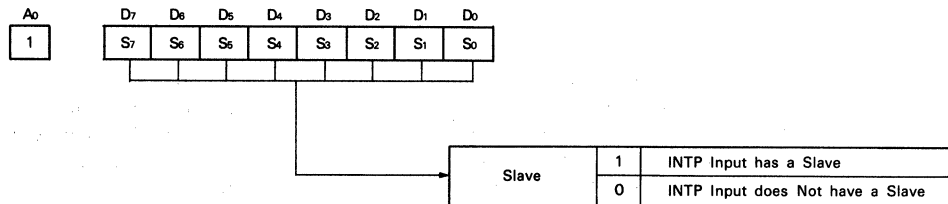
IW1 [Initialization Word 1]



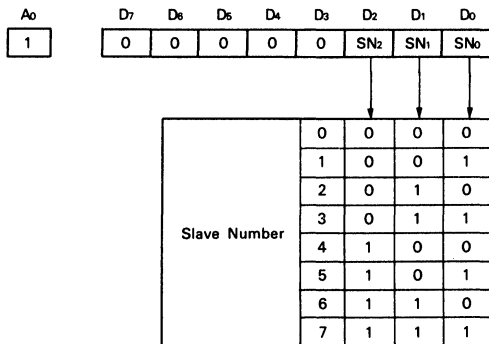
IW2 [Initialization Word 2]



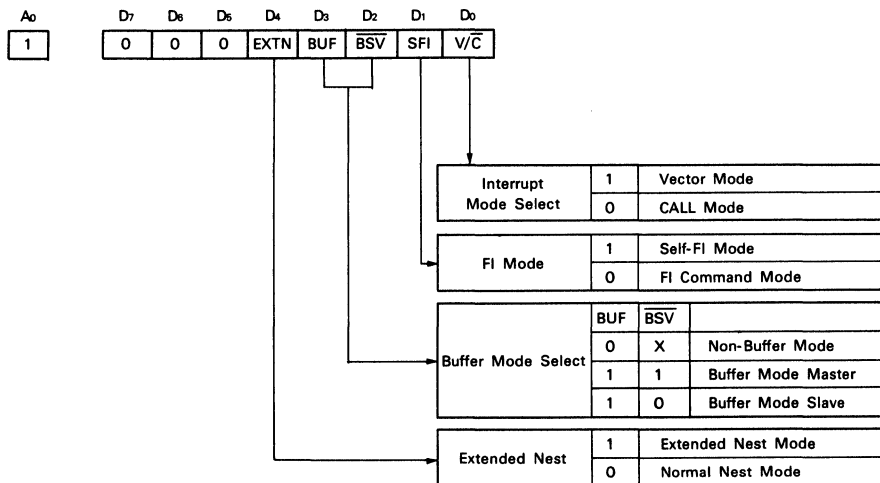
IW3 [Initialization Word 3] Master Mode



IW3 [Initialization Word 3] Slave Mode



IW4 [Initialization Word 4]



IW1:

A7—A5 are the higher three bits of the lower byte of an interrupt routine address given to the CPU in CALL mode.

LEV is used to select the trigger mode of the INTP inputs. If LEV=0, the edge-trigger mode is selected. If LEV=1, the level-trigger mode is selected.

AG4 (address gap 4 bytes) sets the spacing of interrupt routine addresses in the CALL mode to either 4 or 8. For example, when the address of INTP₀'s interrupt routine is 1000H, INTP₁'s address is 1004H when AG4=1 and 1008H when AG4=0.

SINGL is used to designate the scale of the interrupt system. If SINGL=1, only one CXQ71059 is used in a system, and IW3 will not be accepted.

IW4 will be written when I4=1. It will not be written when I4=0.

IW2:

A15—A8 is a higher byte of the interrupt routine address given to the CPU in the CALL mode.

V7—V3 are the five higher bits of the interrupt vector number given to the CPU in the vector mode.

IW3:

IW3 has meaning only in the extended mode.

When the CXQ71059 is the master in an extended mode system, S₇—S₀ define whether INTP₇—INTP₀ have slaves or peripherals. For example, if S₂=1, that indicates that interrupts to pin INTP₂ are from slave CXQ71059. The master CXQ71059 will output the slave number 2 on SA₂—SA₀ during the INTAK cycle and will not place the interrupt address or vector number on the data bus. The slave then outputs the interrupt address or vector number. When S₂=0, the master will place the interrupt address or vector numbers on the data bus for INTP₂ interrupts.

In the slave mode, the lower three bits of IW3 set the slave number of the CXQ71059. This number will be compared with the number put on pins SA₂—SA₀ output by the master CXQ71059 during an INTAK cycle. If there is a match, the slave knows that its interrupt is being honored and issues the interrupt address or vector number onto the data bus.

IW4:

EXTN (extended mode) sets the nesting mode. When EXTN=0, the normal nesting mode is set. When EXTN=1, the extended nesting mode is set.

SFI (self-finish interrupt mode) set the FI mode. When SFI=0, the FI (finish interrupt) command mode is set. In FI command mode, an FI command must be sent to the CXQ71059 to terminate the interrupt. When SFI=1, the self-FI mode is set and the CXQ71059 will automatically perform an FI command at the end of every INTAK cycle.

BUF (buffer) is used to designate the buffer mode. If BUF=1, the buffer mode is set.

BSV (buffered slave) is used with BUF. If BUF=1, BSV defines whether the CXQ71059 is a master or a slave. When BSV=0, the CXQ71059 is master; when BSV=1, a slave.

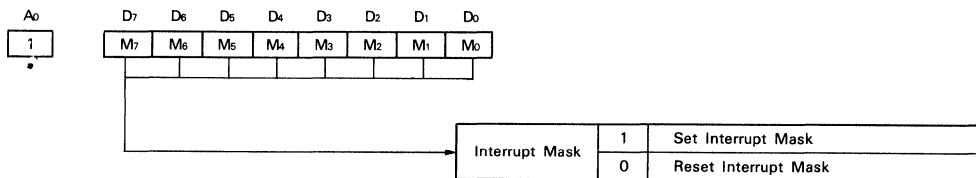
V/C sets the vector or CALL mode. The vector mode is set when V/C=1 and the CALL mode is set when V/C=0. These modes should be programmed to match the system's CPU.

Command Words

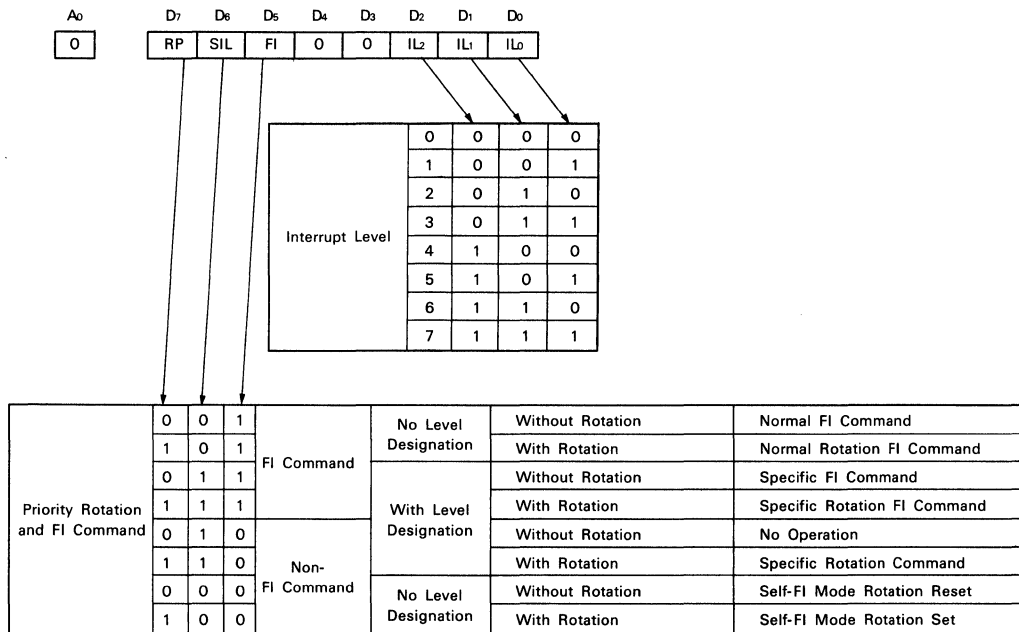
The command words give various commands to a CXQ71059 during its operation to change interrupt masks and priorities, to end interrupt processing, etc. See figure 3.

Figure 3. Command Word Format

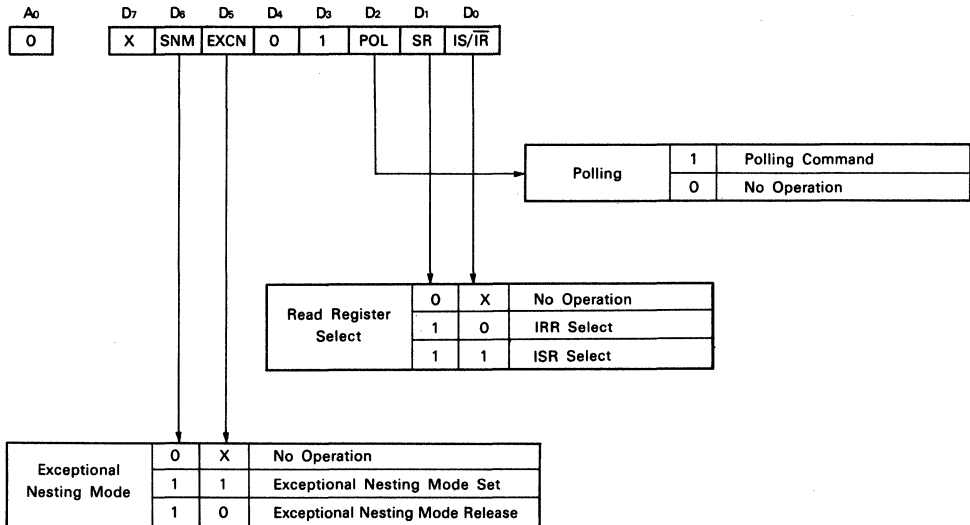
IMW [Interrupt Mask Word]



PFCW [Priority Finish and Control Word]



MCW [Mode Control Word]



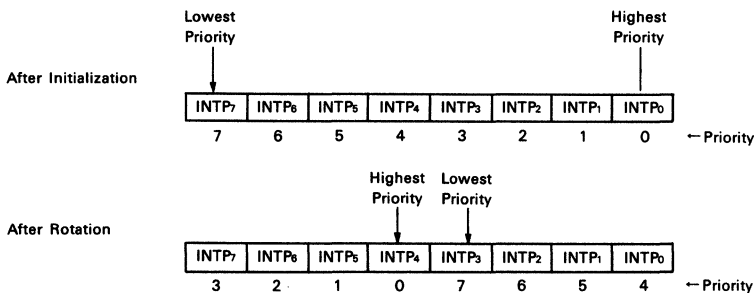
IMW (Interrupt Mask Word). This word masks the IRR and disables the corresponding INTP interrupt requests. It also masks the ISR in the exceptional nesting mode. Bits M7–M0 correspond to the interrupt levels of INTP7–INTP0, respectively. IMW is stored in the IMR.

Interrupts corresponding to the bits of IRR and ISR in the exceptional nesting mode are masked if the Mn bit is set to 1.

PFCW (Priority and Finish Control Word). This word designates the FI command that defines the way how interrupts are ended, and the commands that change interrupt request priorities.

When RP (rotate priority) is set to 1, the priorities of the interrupt request change (rotate). The priority order of the 8 INTP pins is circular, as shown in Figure 4. If a level is assigned to the lowest priority, the priorities for all the other levels will be defined correspondingly. For example, if INTP3 becomes the lowest priority, INTP4 will be the highest. (INTP7 has the lowest priority after initialization).

Figure 4. INTP Priority Order



SIL (specify interrupt level) is set to 1 to change the priority order or designate an interrupt level. It is used with the RP and FI bits (bits D7 and D5). When SIL =1 and RP=1, the level identified by IL2—IL0 is designated as the lowest priority level. The other priorities will be assigned correspondingly. When SIL=1 used with FI=1, it resets the ISR bit corresponding to the interrupt level IL2—IL0. If SIL=0, the IL2—IL0 bits have no meaning.

MCW (Mode Control Word). This word is used to enter the exceptional nesting mode, to poll the CXQ71059, and to read the ISR and IRR registers.

Bits SR (set register) and IS/ $\bar{I}R$ (ISR/IRR) are used to read the contents of the IRR and ISR registers. When SR=0, no operation is performed. To read IRR or ISR, set A0=0 and select the IRR or ISR register by writing to MCW. To select the IRR register, write MCW with SR=1 and IS/ $\bar{I}R$ =0. To select the ISR, write MCW with SR=1 and IS/ $\bar{I}R$ =1. The selection is retained, and MCW does not have to be rewritten to read the same register again. IRR and ISR are not masked by the IMR.

Bits SNW (set nesting mode) and EXCN (exceptional nesting mode) are used to set or clear the exceptional nesting mode. If SNW=0, EXCN executes no operation. If SNW=1 with EXCN=1, the exceptional nesting mode is set. If with EXCN=0, it is cleared.

POL (polling) is used to enable the polling operation. If POL=1, the polling command is issued. If POL=0, it is not issued.

CALL OR VECTOR MODES

The CXQ71059 passes interrupt routine address data to the CPU in two modes, depending on the CPU type. This mode is selected by bit V/\bar{C} in initialization word IW4. V/\bar{C} is set to one to select the vector mode for CXQ70108/70116 CPUs, and reset to zero to select the CALL mode for 8085A CPUs.

CALL MODE (8085A CPUs)

In this mode, when an interrupt is acknowledged by the CPU, the CXQ71059 outputs three bytes of interrupt data to the data bus in its \overline{INTAK} sequence. During the first \overline{INTAK} pulse from the CPU, the CXQ71059 outputs the CALL opcode OCDH. During the next \overline{INTAK} pulse, it outputs the lower byte of a two-byte interrupt routine address. During the third \overline{INTAK} pulse, it outputs the upper byte of the address. The CPU interprets these three bytes as a CALL instruction and executes the CALL interrupt routine. See figure 5.

Interrupt routine addresses are defined by using words IW1 and IW2 through the initialization. However, only the higher 10 or 11 bits of the interrupt addresses are designated: A15—A6 or A15—A5. The remaining lower bits (D5—D0 or D4—D0) are set to zero to get the address of INTPO's interrupt routine. The addresses for INTPI—INTP7 are set at equally spaced gaps based on it. The gap between interrupt addresses is determined by the AG4 bit (address gap 4 bytes) of IW1. When AG4=1, the interrupt routine starting addresses are 4 bytes apart. Therefore, the starting address for INTPI is the starting address for INTPO plus four times i. When AG4=0, starting addresses are eight bytes apart, so the starting address for INTPI is the starting address for INTPO plus eight times i. See figure 6.

Figure 5. CALL Mode Interrupt Sequence

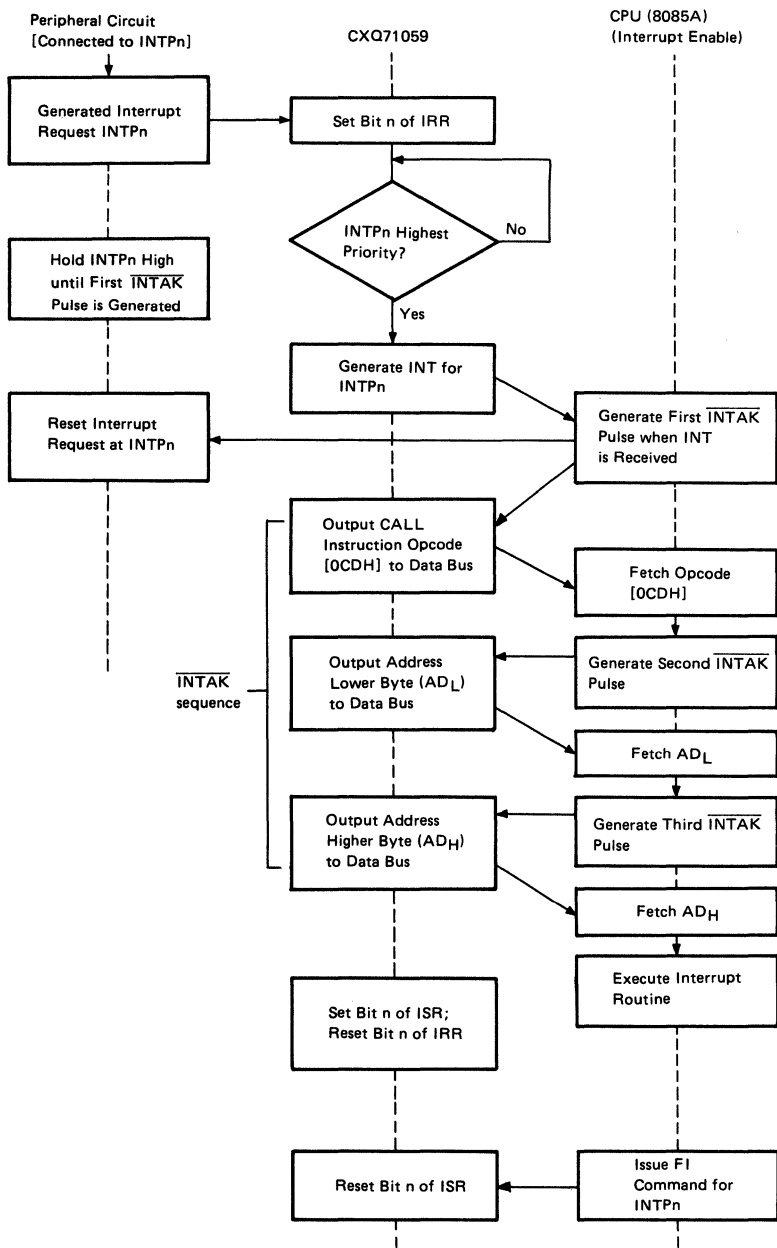


Figure 6. CALL Mode Interrupt Address Sequence**Address Lower Byte [AD_L] During Second INTAK**

AG4=1 (4-Byte Spacing Address)

Interrupt Level	D7	D6	D5	D4	D3	D2	D1	D0
INTP ₀	A ₇	A ₆	A ₅	0	0	0	0	0
INTP ₁	A ₇	A ₆	A ₅	0	0	1	0	0
INTP ₂	A ₇	A ₆	A ₅	0	1	0	0	0
INTP ₃	A ₇	A ₆	A ₅	0	1	1	0	0
INTP ₄	A ₇	A ₆	A ₅	1	0	0	0	0
INTP ₅	A ₇	A ₆	A ₅	1	0	1	0	0
INTP ₆	A ₇	A ₆	A ₅	1	1	0	0	0
INTP ₇	A ₇	A ₆	A ₅	1	1	1	0	0

AG4=0 (8-Byte Spacing Address)

Interrupt Level	D7	D6	D5	D4	D3	D2	D1	D0
INTP ₀	A ₇	A ₆	0	0	0	0	0	0
INTP ₁	A ₇	A ₆	0	0	1	0	0	0
INTP ₂	A ₇	A ₆	0	1	0	0	0	0
INTP ₃	A ₇	A ₆	0	1	1	0	0	0
INTP ₄	A ₇	A ₆	1	0	0	0	0	0
INTP ₅	A ₇	A ₆	1	0	1	0	0	0
INTP ₆	A ₇	A ₆	1	1	0	0	0	0
INTP ₇	A ₇	A ₆	1	1	1	0	0	0

Note: When AG4=0, bit A₅ is ignored.**Address Higher Byte [AD_H] During Third INTAK**

D7	D6	D5	D4	D3	D2	D1	D0
A ₁₅	A ₁₄	A ₁₃	A ₁₂	A ₁₁	A ₁₀	A ₉	A ₈

Vector Mode (CXQ70108/70116 CPUs)

In the vector mode, the CXQ71059 outputs a one-byte interrupt vector number to the data bus in the INTAK sequence. The CPU uses that vector number to generate an interrupt routine address. See figure 7.

The higher five bits of the vector number, V7-V3, are defined by IW2 during initialization. The CXQ71059 assigns the remaining three bits to successive interrupt vector numbers for the eight interrupts. See figure 8.

Figure 7. Vector Mode Interrupt Sequence

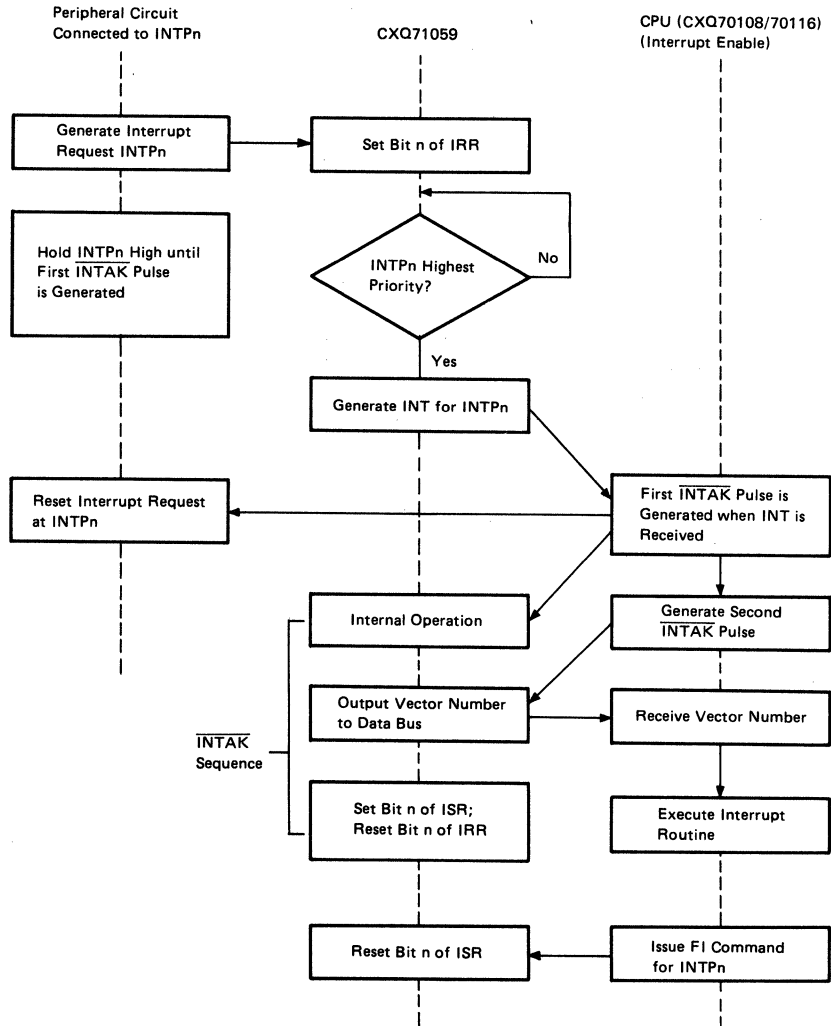


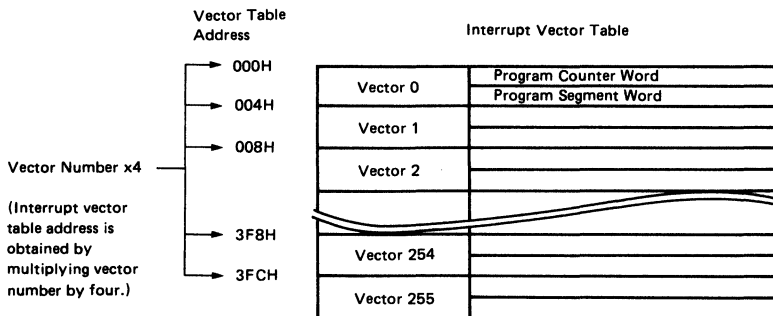
Figure 8. Vector Numbers Output in Vector Mode

Output During the Second $\overline{\text{INTAK}}$

Interrupt Level	D7	D6	D5	D4	D3	D2	D1	D0
INTP ₀	V ₇	V ₆	V ₅	V ₄	V ₃	0	0	0
INTP ₁	V ₇	V ₆	V ₅	V ₄	V ₃	0	0	1
INTP ₂	V ₇	V ₆	V ₅	V ₄	V ₃	0	1	0
INTP ₃	V ₇	V ₆	V ₅	V ₄	V ₃	0	1	1
INTP ₄	V ₇	V ₆	V ₅	V ₄	V ₃	1	0	0
INTP ₅	V ₇	V ₆	V ₅	V ₄	V ₃	1	0	1
INTP ₆	V ₇	V ₆	V ₅	V ₄	V ₃	1	1	0
INTP ₇	V ₇	V ₆	V ₅	V ₄	V ₃	1	1	1

The CPU generates an interrupt vector by multiplying the vector number by four, and using the result as the address of a location in an interrupt vector table located at addresses 000H–3FFH. See figure 9.

Figure 9. Interrupt Vectors for the CXQ70108/70116



System Scale Modes

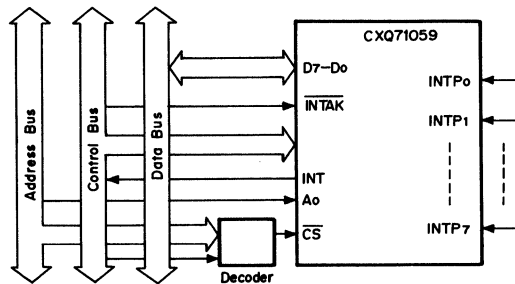
The CXQ71059 can operate in either single mode, with up to eight interrupt lines or extended mode, with more than one CXQ71059 and more than eight interrupt lines. In extended mode a CXQ71059 is in either master or slave mode.

Bit D₁ of the first initialization word, IW₁, SNGL (single mode) designates the scale of the interrupt system. SNGL=1 designates that only one CXQ71059 is being used (single mode system). SNGL=0 designates an extended mode system with a master and slave CXQ71059s. In the single mode (SNGL=1), the SV input and IW₄ buffer mode bits D₃ and D₂ are ignored by the CXQ71059.

Single Mode

This mode is the normal mode of CXQ71059 operation. It has been described in the Interrupt Operation section. See figure 10 for a system example.

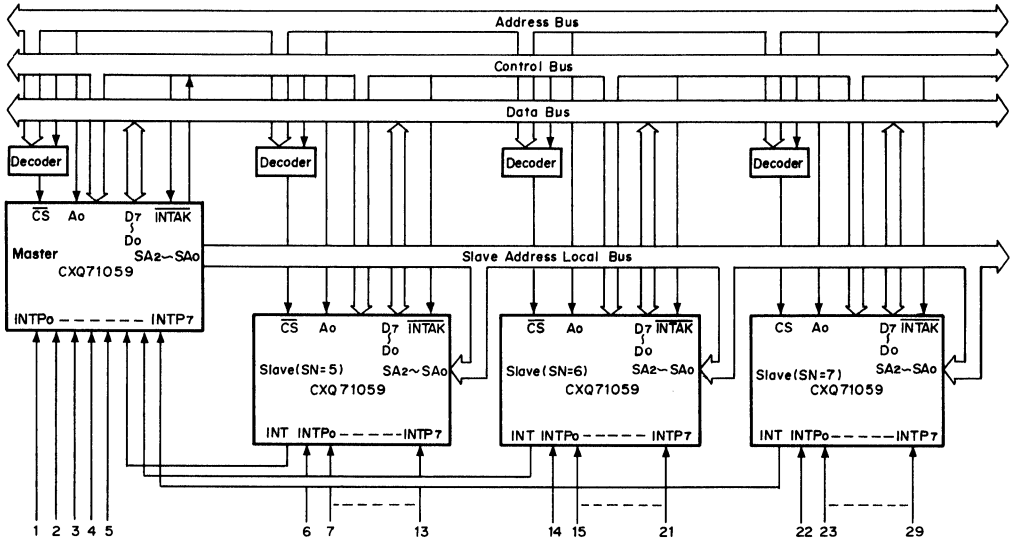
Figure 10. Single Mode System



Extended Mode

In this mode, up to 64 interrupt requests can be processed using a master connected to a maximum of eight slaves. See figure 11 for a system example.

Figure 11. Extended System Example with Three Slaves



Master Mode

In the master mode, S7–S0 are defined by IW3, identifying which INTP inputs have slaves and which do not. Let us consider an interrupt request from INTP_n.

If S_n=0, the interrupt is not from a slave (for example, INTP₀ of the master in Figure 11), and the CXQ71059 treats it the same way it would if it were in the single mode. SA2–SA0 outputs are low.

If S_n=1, the interrupt is from a slave (for example, INTP₇ of the master). The master sends an interrupt to the CPU if the slave requesting the interrupt has priority. The master then outputs slave address n to pins SA2–SA0 on the first INTAK pulse by the CPU. It lets the slave n perform the rest of the INTAK sequence.

Slave Mode

When a slave receives an interrupt request from a peripheral (assuming this request is higher than other requests and in-service levels on the slave), it sends an interrupt request to the master through its INT output. When the interrupt is accepted by the CPU through the master, the master outputs the slave's address on pins SA2–SA0. Each slave compares the address on SA2–SA0 to its own address. The slave that has sent interrupt will find a match. It completes the INTAK sequence the same way as a single CXQ71059 would.

The master outputs slave address 0 when it is processing a non-slave interrupt. Therefore, do not use 0 as a slave address if there are less than eight slaves connected to the master.

Figures 12 and 13 show the interrupt operating sequences for slaves in the extended mode.

Figure 12. Interrupt from Slave (CALL Mode)

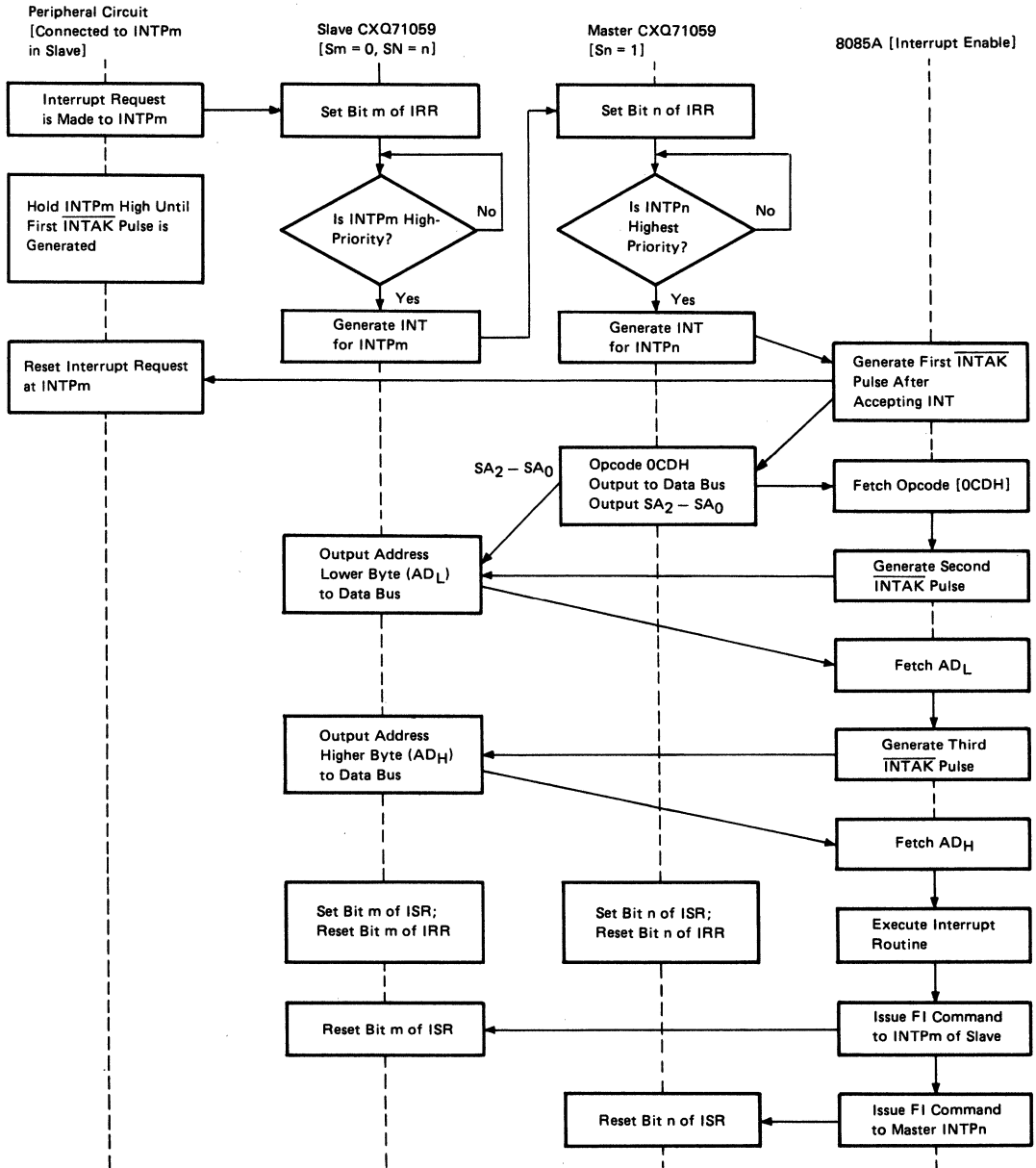
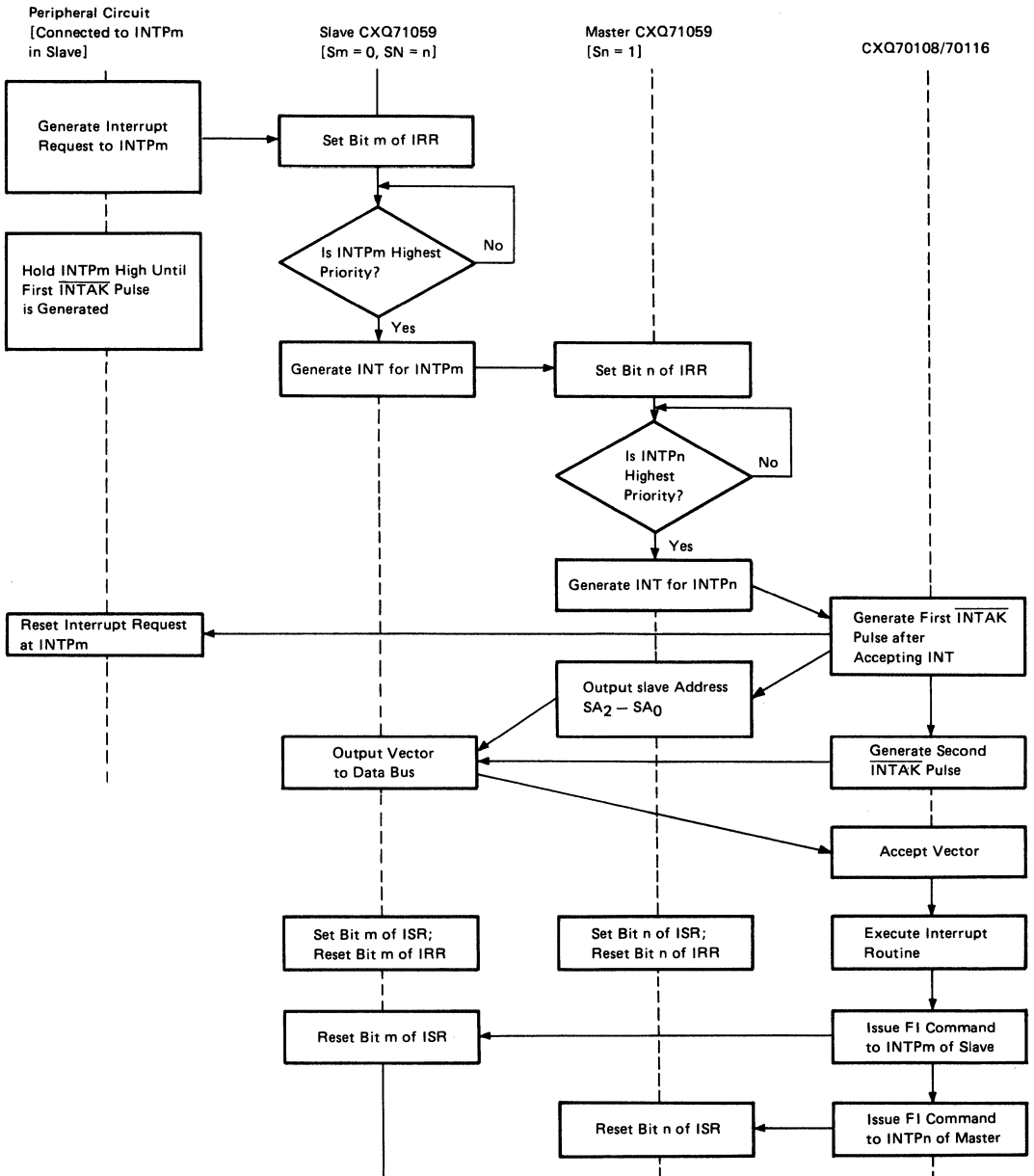


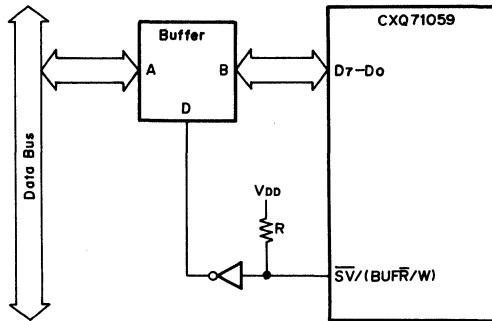
Figure 13. Interrupt from Slave (Vector Mode)



BUFFER AND NON-BUFFER MODES

When the system scale is large, a buffer may be needed for the CXQ71059 to drive the data bus. A buffer mode is supplied, with a signal to specify the buffer direction. In the buffer mode, $\overline{SV}/(BUF\overline{R}/W)$ is used to select the buffer direction and \overline{SV} cannot be used to specify the master/slave mode. The master/slave selection must be done by IW4. IW4 bit D3, BUF (buffer) and D2, \overline{BSV} (buffered slave) are used together to define the buffer mode and master/slave relation. When BUF=0, the non-buffer mode is designated and \overline{BSV} has no meaning. When BUF=1, the buffer mode is designated. In buffer mode, the CXQ71059 is a master when $\overline{BSV}=1$, a slave when $\overline{BSV}=0$. See figure 14.

Figure 14. Buffer Mode



- Note 1:** D determines data direction
 Low Level: A → B
 High Level: A ← B
- 2:** The CXQ71059 is set to input \overline{SV} in its initial state and is pulled up by R to set D to the low level during initialization.

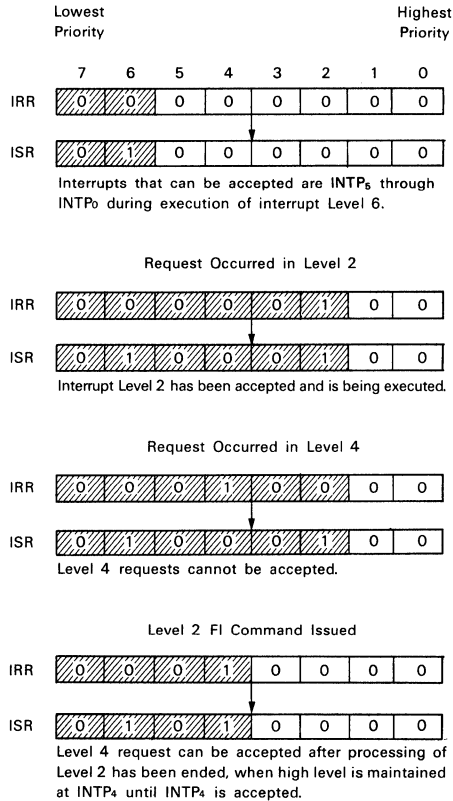
NESTING MODES

A variety of nesting modes are available for supporting a multilevel-interrupt structure.

Normal Nesting Mode

This mode is entered when IW4 is not written or when IW4 has EXTN=0. It is the most common nesting mode. See figure 15.

While an interrupt is being executed in this mode (corresponding bit of ISR=1), only interrupt requests of higher priority can be accepted and the requests of the same or lower priority cannot.

Figure 15. Normal Nesting Mode**Extended Nesting Mode**

This mode is only applicable to a master in the extended mode. Eight interrupt priority levels of a slave become only one priority level when viewed by the master.

Therefore, a request made by a slave with a higher priority than one in service from the same slave will not be accepted. This is because the master's ISR bit is set, ignoring all requests of equal or lower priority in normal nesting mode. This cannot be called complete nesting since priority ranking within slave loses its significance.

When the extended nesting mode is entered (EXTN=1 in IW4 of the master), interrupt requests of the same priority level can be accepted only in interrupt requests by slaves which permit complete nesting operations.

Care should be taken when issuing an FI (finish interrupt) command in the extended nesting mode. Upon an interrupt by a slave, the CPU first issued as FI command to the slave. Then, the CPU reads the slave's in-service register (ISR) to see if that slave still has interrupts in service. If there are no interrupts in service (ISR=00H), an FI command can be issued to the master, as in the single mode when an interrupt is made by a peripheral.

Exceptional Nesting Mode

A CXQ71059 in the normal or extended nesting mode cannot accept interrupts of a lower priority than the interrupt in service. Sometimes, however, it is desirable that requests with lower priority be accepted while higher-priority interrupt is being serviced. It is enabled by using the exceptional nesting mode. Once the exceptional nesting mode is entered, it is retained until reset. When it is reset, the previous nesting mode is resumed.

The exceptional nesting mode is programmed by SNM (set nesting mode) and EXCN (exceptional nesting mode) (bit D6 and D5 of MCW), used in pairs. They set and clear the exceptional nesting mode. The mode will not changed when SNW=0. Exceptional nesting mode is entered if EXCN= and cleared if EXCN=0 when SNM=1.

In the exceptional nesting mode IMR masks the corresponding bits of IRR and ISR (though in other mode only IRR is masked). Therefore, if a bit in the IMR is set to a 1 in the exceptional nesting mode, it inhibits interrupts to that level and any other masked bit, but it allows interrupts to all other levels, higher or lower priority.

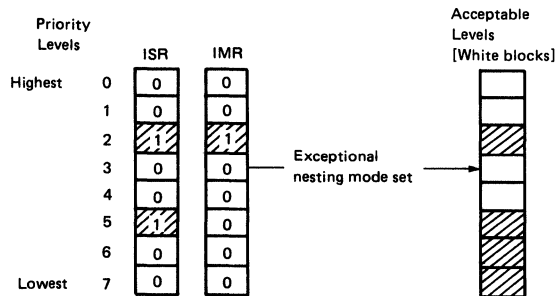
Cautions must be taken when issuing an FI command to a level masked in the exceptional nesting mode. Since the ISR bit is masked, the normal FI command will not work. For this reason, a specific FI command specifying the ISR bits must be issued. After the exceptional mode is released, the normal FI command may be used.

The procedure for the exceptional nesting mode is as follows:

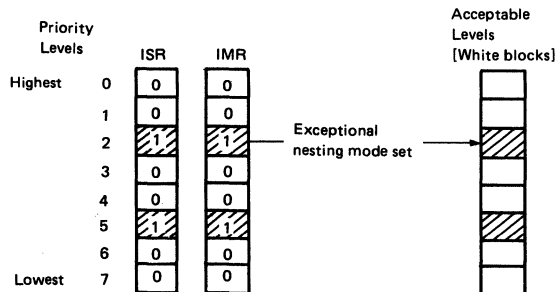
- (1) Read the ISR.
- (2) Write the read data to the IMR.
- (3) Set the exceptional nesting mode.

In this way, all interrupt requests not currently in service will be enabled. See figure 16.

Figure 16. Exceptional Nesting Mode



Requests from INTP₆ and INTP₇ cannot be accepted when only bit 2 of the IMR is set to 1.



All requests other than those being executed can be accepted when the IMR is set the same as the ISR, in exceptional nesting mode.

Finishing Interrupts and Changing the Priority Levels

The priority and finish control word (PFCW) issues FI commands and changes interrupt priorities.

Normal FI Command

	D7	D6	D5	D4	D3	D2	D1	D0
PFCW=	0	0	1	0	0	X	X	X

When a normal FI command is issued, the CXQ71059 resets the ISR bit corresponding to the highest priority level selected from the interrupts in service. This operation confirms to the CXQ71059 that the highest priority routine (assuming it is the last) of the routines in service is finished. If an interrupt routine changes the priority level or the exceptional nesting mode is used, this command will not operate correctly because the highest priority interrupt is not necessarily the last interrupt in service.

Specific FI Command

	D7	D6	D5	D4	D3	D2	D1	D0
PFCW=	0	1	1	0	0	IL ₂	IL ₁	IL ₀

When the specific FI command is issued, the CXQ71059 resets the ISR bit designated by bits IL₂—IL₀ of the PFCW. This command is used when the CXQ71059 cannot automatically determine the level of a completed service routine.

Self-FI Mode

When SFI (of IW4)=1, the CXQ71059 is set to the self-FI mode. In this mode, the ISR bit corresponding to the interrupt is set at the leading edge of the last $\overline{\text{INTAK}}$ pulse in the $\overline{\text{INTAK}}$ sequence. The same bit is reset on the trailing edge of the last $\overline{\text{INTAK}}$ pulse. Therefore, the CPU does not have to issue an FI command when the interrupt routine ends. In this mode, however, the ISR stores no information to specify the routine in service. Unless interrupts are disable by the interrupt routine, newly generated interrupt requests are granted without priority limitation by the ISR. This can cause a stack overflow when frequent interrupt requests occur, or when the interrupt is level-trigger

Self-FI Rotation

With Rotation:

	D7	D6	D5	D4	D3	D2	D1	D0
PFCW=	1	0	0	0	0	X	X	X

Without Rotation:

	D7	D6	D5	D4	D3	D2	D1	D0
PFCW=	0	0	0	0	0	X	X	X

Rotation of interrupt priorities can be added to the self-FI mode. In this case, the corresponding interrupt is assigned to the lowest priority level when the bit is reset in the ISR at the end of the $\overline{\text{INTAK}}$ sequence.

Normal Rotation FI Command

	D7	D6	D5	D4	D3	D2	D1	D0
PFCW=	1	0	1	0	0	X	X	X

When the normal rotation FI command is issued, the CXQ71059 resets the ISR bit corresponding to the highest priority level selected from the interrupts in service, then rotates the priority levels so that the interrupt just completed becomes the lowest priority.

Specific Rotation FI Command

	D7	D6	D5	D4	D3	D2	D1	D0
PFCW=	1	1	1	0	0	IL2	IL1	IL0

When the specific rotation FI command is issued, the CXQ71059 resets the ISR bit designated by bits IL2—IL0 of the PFCW and rotates the interrupt priorities so that the interrupt just reset becomes the lowest priority. As this change in priority levels is different from the normal nesting mode, it is the user's responsibility to manage nesting.

Specific Rotation Command

	D7	D6	D5	D4	D3	D2	D1	D0
PFCW=	1	1	0	0	0	IL2	IL1	IL0

When the specific rotation command is issued, the CXQ71059 specifies the interrupt designated by bits IL2—IL0 of the PFCW to the lowest priority and rotates the priority levels. In this case also, the user must manage nesting.

Triggering Mode

Bit D3 of the first initialization word, IW1, is LEV (level-trigger mode bit). LEV defines the trigger mode of the INTP input. The level-trigger mode is set when LEV=1. The edge-trigger mode is set when LEV=0.

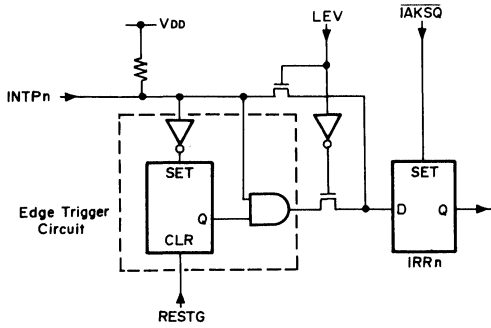
Edge-Trigger Mode

In the edge-trigger mode, an interrupt is detected by the rising edge of the signal on an INTP input. Although the corresponding IRR bit becomes '1' when INTP is high, the IRR bit is not latched until after the CPU returns an $\overline{\text{INTAK}}$ pulse. The INTP input must be maintained high until after $\overline{\text{INTAK}}$ is received. The IRR bit will be unlatched only after the INTP input goes low. To send the next interrupt request, temporarily lower the INTP input, then raise it.

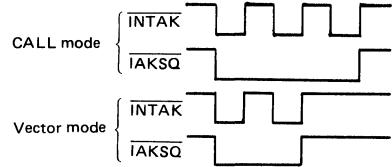
Level-Trigger Mode

In the level-trigger mode, an IRR bit is set by the INTP input being at high level. As in the edge-trigger mode, the INTP must be maintained high until after the INTAK is received. Interrupts are requested as long as the INTP input remains high. Care should be taken so as not to cause a stack overflow in the CPU. See figure 17.

Figure 17. INTP Input



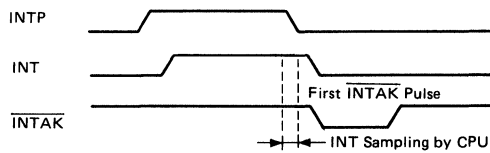
o IAKSQ:



o RESTG goes high when IWI is written or on the low-to-high transition of IAKSQ.

Note: In both the edge- and level-trigger modes the INTP requests must be maintained high until after the INTAK is received. If on any INTP inputs the CXQ71059 INT output goes low before the first INTAK pulse, the CXQ71059 operates as if the INTP₇ interrupt occurred. This incomplete interrupt request will not set the bit 7 of ISR. In case this will occur, the interrupt routine for the INTP₇ must see whether the current request is a complete one or not by reading the ISR. The FI command should not be issued for incomplete interrupts. See figure 18.

Figure 18. Incomplete Interrupt Request



Reading Registers

The contents of the internal registers, IRR, ISR, and IMR, can be read.

The IMR can be read when a direct read with $A_0=1$ is done.

To read the IRR or ISR, MCW must be issued to select either the IRR or ISR. Then a read operation with $A_0=0$ will issue the data of the selected register. Once the MCW is issued, the CXQ71059 will retain the selection. Therefore, no MCW is required when the same register with the previous one is to be read. The data of the IRR or ISR to be read is not masked by the IMR.

Polling Operation

When polling, the CPU should first disable its INT input. Next, it issues a polling command to the CXQ71059 using MCW with $POL=1$. This command sets the CXQ71059 in the polling mode until the CPU reads one of the CXQ71059's registers. When the CPU performs a read operation with $A_0=0$ in the polling mode, polling data as shown in figure 19 is read instead of ISR or IRR. The CXQ71059 then ends the polling mode.

If INT in the polling data is 1, the CXQ71059 sets the ISR bit corresponding to the interrupt level shown by bits PL_2 - PL_0 of the polling data and considers that interrupt as being executed. The CPU then processes the interrupt routine accordingly, based on the polling data read. An FI command should be issued when this processing ends.

Note: When a read is performed with $A_0=1$ after the polling command is sent to the CXQ71059, the IMR will be read instead of polling data. However, after the polling command is issued, the CXQ71059 operates in the same manner when $A_0=1$ as it would when $A_0=0$. This means that although A_0 is set to 1, the CXQ71059 will send the contents of the IMR, but it will also set an ISR bit just as it would if A_0 had been set to zero. This may disturb the nesting. Therefore, performing a read operation with $A_0=1$ immediately after sending the polling command should be avoided.

Figure 19. Polling Data

	D7	D6	D5	D4	D3	D2	D1	D0
MCW =	INT	0	0	0	0	PL_2	PL_1	PL_0

INT (Interrupt)

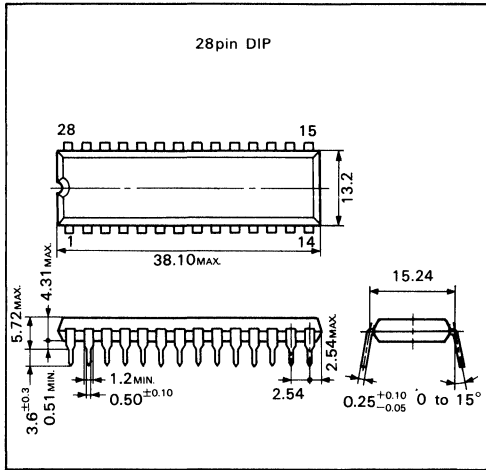
This bit has the same meaning as the INT pin. When it is set to 1, it means that the CXQ71059 has accepted an INTP input.

PL_2 - PL_0 (Permitted Level)

These bits show which INTP input requested an interrupt when $INT=1$.

Package Outline

Unit: mm



DMA Controller

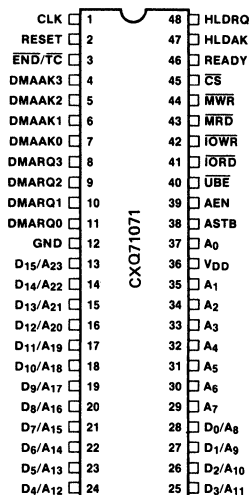
Description

The CXQ71071 is a high-speed, high-performance Direct Memory Access (DMA) controller that provides high speed data transfers between peripheral devices and memories. A programmable bus width allows bidirectional data transfer in both 8- and 16-bit systems. In addition, CMOS technology reduces power consumption.

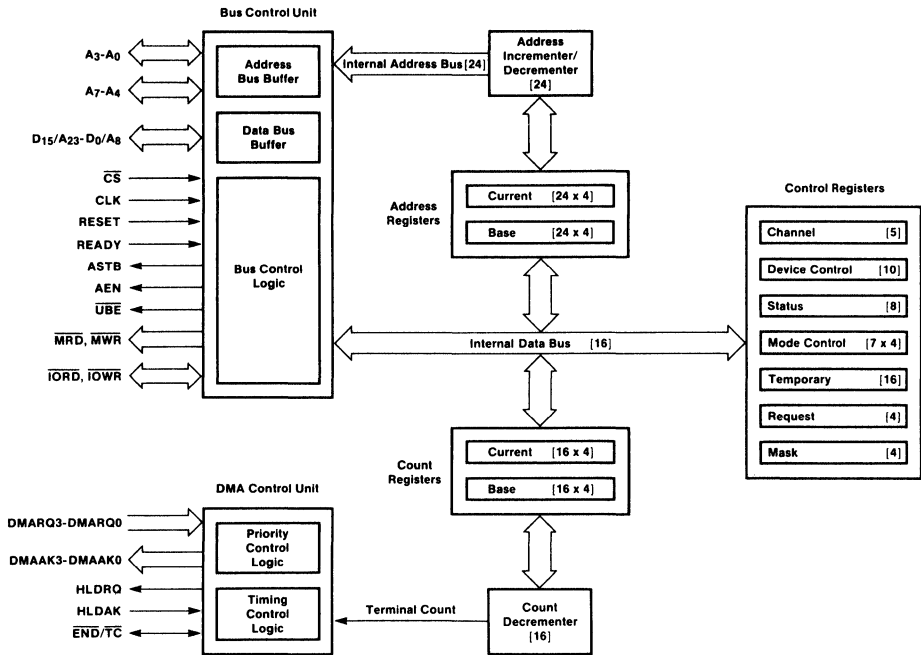
Features

- Four independent DMA channels
- 24 address lines to access 16M byte memory
- 64K byte/word transfer count
- 8- or 16-bit programmable data bus width
- Abundant transfer functions
 - Byte/word transfer
 - Three transfer modes: single, demand, and block
 - Two bus modes: bus release and bus hold
 - Two transfer timings: normal and compressed
 - Memory-to-memory, memory-to-I/O, or I/O-to-memory transfer
- Enable/disable of individual DMA requests
- Software DMA requests
- Autoinitialize enable/disable of individual DMA channels
- Address increment/decrement
- Fixed/rotational DMA channel priority
- Terminal count output
- Forced transfer termination input
- Cascade capability
- Programmable DMA request and acknowledge signal polarities
- High performance: transfers up to 5.33M bytes/second
- 8 MHz operation
- CXQ70108/70116 CPU system compatible
- CMOS technology
- Single power supply
- 48-pin plastic/ceramic DIP
- NEC μ PD71071 compatible

Pin Configuration



Block Diagram



Pin Functions

No.	Symbol	Name	Type	Function
1	CLK	Clock	Input	Controls internal operation and data transfer rate.
2	RESET	Reset	Input	Initializes the controller's internal registers and enters the controller in the Idle cycle (CPU controls the bus). Active high.
3	$\overline{\text{END}}/\overline{\text{TC}}$	End DMA Transfer/ Terminal Count	Input/Output	The DMA service can be terminated either internally or externally. The CXQ71071 allows an external signal to end a DMA service by pulling the $\overline{\text{END}}$ low. The CXQ71071 also generates a pulse ($\overline{\text{TC}}$) when the designated terminal count for any channel has been reached. Open drain, it requires external pull-up resistor. Active low.
4-7	DMAAK3- DMAAK0	DMA Acknowledge	Output	Indicate to peripheral devices that DMA service has been acknowledged. DMAAK3-DMAAK0 respond respectively to DMA Channels 3-0. The sense of these line is programmable.
8-11	DMARQ3- DMARQ0	DMA Request	Input	Accept DMA service requests from peripheral devices. DMARQ3-DMARQ0 respond respectively to DMA Channels 3-0. The sense of these lines is programmable.
12	GND	Ground		Ground
13-20	A23-A16/ D15-D8	Address/ Data	Input/Output 3-State	In 8-bit systems, these pins output the upper 8 bits of the address. In 16-bit systems, the address is multiplexed with the upper byte of the data bus. In the Idle mode, these pins become data bus to communicate with the CPU. In the DMA cycles, these lines become address/data bus.
21-28	A15-A8/ D7-D0	Address/ Data	Input/Output 3-state	These pins output the middle 8 bits of the address multiplexed with the lower byte of the data bus. In the Idle mode, these lines function as data bus. In the DMA cycle, they become address/data bus.

No.	Symbol	Name	Type	Function
29–35, 37	A7–A0	Address	Input/Output 3-state	Function as the lower 8 bits of the address bus. A7–A4 output memory address during the DMA cycle and become high impedance in the Idle cycle. A3–A0 function as the lower 4 bits of the address bus. In the Idle cycle, A3–A0 become address inputs to select internal registers for the CPU to read or write. In the DMA cycle, A3–A0 output memory address.
36	VDD	Power Supply		+5V power supply.
38	ASTB	Address Strobe	Output	Latches address information into an external address latch on the falling edge of ASTB during a DMA cycle. Latches A23–A8 in 16-bit mode and A15–A8 in 8-bit mode. Active high.
39	AEN	Address Enable	Output	Enables the output of an external latch that holds DMA addresses. AEN becomes high during the DMA cycle. Active high.
40	\overline{UBE}	Upper Byte Enable	Input/Output 3-state	Indicates the upper byte of the data bus is valid during 16-bit mode. In the Idle cycle during data transfer, it receives \overline{UBE} from the CPU when the upper data byte is valid. During a DMA cycle, \overline{UBE} goes low to signify the presence of valid data on D15–D8. \overline{UBE} has no meaning in 8-bit mode and becomes high impedance in the Idle cycle and high level in the DMA cycle. Active low.
41	$\overline{IOR\overline{D}}$	I/O Read	Input/Output	In the Idle cycle, it inputs a read signal from the CPU. In the DMA cycle, it outputs a read signal to I/O devices. Active low.
42	$\overline{IOW\overline{R}}$	I/O Write	Input/Output	In the Idle cycle, it inputs a write signal from the CPU. In the DMA cycle, it outputs a write signal to I/O devices. Active low.
43	$\overline{MR\overline{D}}$	Memory Read	Output 3-state	During the DMA cycle, it outputs a read signal to memory. $\overline{MR\overline{D}}$ is high impedance during the Idle cycle. Active low.
44	\overline{MWR}	Memory Write	Output 3-state	During the DMA cycle, it outputs a write signal to memory. \overline{MWR} is high impedance during the Idle cycle. Active low.

No.	Symbol	Name	Type	Function
45	$\overline{\text{CS}}$	Chip Select	Input	During the Idle cycle, it selects the CXQ71071 as an I/O device. Active low.
46	READY	Ready	Input	During a DMA operation, it indicates that a data transfer for one cycle has been completed and may be terminated. To meet the requirements of low speed I/O devices and memory, the CXQ71071 inserts a wait state (if READY is low), to extend the bus cycle until READY becomes high. Active high.
47	HLDK	Hold Acknowledge	Input	Accepts the bus hold acknowledge signal from the CPU. Active high.
48	HLDK	Hold Request	Output	Outputs a bus hold request to the CPU. Active high.

Absolute Maximum Ratings

(Ta=25°C)

Parameter	Symbol	Rating Value	Unit
Voltage on any pin with respect to Ground	V	-0.5 to +7.0	V
Operating temperature	Topr	-40 to +85	°C
Storage temperature	Tstg	-65 to +150	°C

Comment: Exposing the device to stresses above those listed in Absolute Maximum Ratings could cause permanent damage. The device is not meant to be operated under conditions outside the limits described in the operational sections of this specification. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

DC Characteristics

(Ta=-40 to +85°C, VDD=5V±10%)

Parameter	Symbol	Limits		Unit	Test Conditions
		Min.	Max.		
Input high voltage	VIH	3.3	VDD+0.5	V	CLK input pin
		2.2	VDD+0.5	V	Other inputs
Input low voltage	VIL	-0.5	0.8	V	
Output high voltage	VOH	0.7VDD		V	IOH=-400µA
Output low voltage	VOL		0.4	V	IOL=2.5mA
Input leakage current	II1		±10	µA	0V≤VI≤VDD
Output leakage current	ILO		±10	µA	0V≤VO≤VDD
Supply current (dynamic)	IDD1		30	mA	
Supply current (static)	IDD2		100	µA	

Capacitance

(Ta=25°C)

Parameter	Symbol	Min.	Typ.	Max.	Unit	Test Conditions
Input capacitance	CI		8	15	pF	fc=1.0MHz Unmeasured pins returned to 0V
Output capacitance	CO		4	8	pF	
I/O capacitance	CIO		10	18	pF	

AC Characteristics (DMA Mode)

(Ta=-40 to -85°C, VDD=5V±10%)

Parameter	Symbol	Limits		Unit	Test Conditions
		Min.	Max.		
Clock cycle	t _{CYK}	125		ns	
Clock pulse width high	t _{KKH}	44		ns	
Clock pulse width low	t _{KKL}	55		ns	
Clock rise time	t _{KR}		10	ns	1.5V→3.0V
Clock fall time	t _{KF}		10	ns	3.0V→1.5V
Input rise time	t _{IR}		20	ns	
Input fall time	t _{IF}		12	ns	
Output rise time	t _{OR}		20	ns	
Output fall time	t _{OF}		12	ns	
DMARQ setup time to CLK high	t _{SDQ}	35		ns	S1, S0, S3, SW, S4w
HLDRO high delay time from CLK low	t _{DHQH}		100	ns	S1, S4w
HLDRO low delay time from CLK low	t _{DHQL}		100	ns	S1, S0, S4w
HLDRO low level period	t _{HQHQL}	2t _{CYK} -50		ns	S4w
HLDAR high setup time to CLK low	t _{SHA}	35		ns	S0, S4, S4w
AEN high delay time from CLK low	t _{DAEH}		90	ns	S1, S2
AEN low delay time from CLK low	t _{DAEL}		90	ns	S1, S4w
ASTB high delay time from CLK low	t _{DSTH}		70	ns	S1
ASTB low delay time from CLK high	t _{DSTL}		70	ns	S1
ASTB high level period	t _{STSTH}	t _{KKL} -15		ns	
ADR/ $\overline{\text{UBE}}$ / $\overline{\text{RD}}$ / $\overline{\text{WR}}$ active delay time from CLK low	t _{DA}		100	ns	S1, S2
ADR/ $\overline{\text{UBE}}$ / $\overline{\text{RD}}$ / $\overline{\text{WR}}$ float time from CLK low	t _{FA1}		70	ns	S1, S4w
ADR setup time to ASTB low	t _{SAST}	t _{KKL} -50		ns	
ADR hold time from ASTB low	t _{HSTA}	t _{KKH} -20		ns	
ADR/ $\overline{\text{UBE}}$ hold time from CLK high	t _{HA}	10		ns	S2, S4
ADR float time from CLK low	t _{FA2}	0	70	ns	S1, S2
$\overline{\text{RD}}$ low delay time from ADR float	t _{DAR}	-10		ns	

Parameter	Symbol	Limits		Unit	Test Conditions
		Min.	Max.		
Input data setup time to CLK low	tsID	35		ns	S14
Input data hold time from CLK low	thID	10		ns	S14
Output data delay time from CLK low	tdOD	10	100	ns	S22
Output data hold time from CLK high	thOD	10		ns	S24
Output data hold time from MWR high	thMWOD	tkKL-50		ns	
\overline{RD} low delay time from CLK low	tdKLR		70	ns	S2 Normal Timing
\overline{RD} low delay time from CLK high	tdKHR		70	ns	S2 Compressed Timing
\overline{RD} low level period	trRL1	2tcYK-50		ns	Normal Timing
	trRL2	tcYK+tkKH-50			Compressed Timing
\overline{RD} high delay time from CLK low	tDRH	15	100	ns	S4
ADR delay time from \overline{RD} high	tdRA	tcYK-40		ns	
\overline{WR} low delay time from CLK low	tdWL1	10	70	ns	S3 Normal Write
	tdWL2	10	70	ns	S2 Extended Write, Normal Timing
\overline{WR} low delay time from CLK high	tdWL3	10	70	ns	S2 Compressed Timing
\overline{WR} low level period	twWL1	tcYK-50		ns	Normal Write
	twWL2	2tcYK-50		ns	Extended Write, Normal Timing
	twWL3	tcYK+tkKH-50		ns	Extended Write, Compressed Timing
\overline{WR} high delay time from CLK low	tdWH	10	80	ns	S4
DMAAK setup time to \overline{RD} , \overline{WR} low	tsDARW	0		ns	S1, S2
\overline{RD} high delay time from \overline{WR} high	tdWHRH	5		ns	
DMAAK delay time from CLK high	tdKHDA	10	70	ns	S1

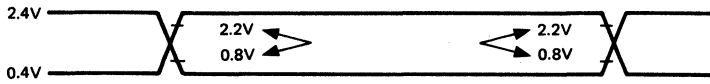
Parameter	Symbol	Limits		Unit	Test Conditions
		Min.	Max.		
DMAAK delay time from CLK low	tDKLDA	10	115	ns	S1 Cascade Mode
DMAAK inactive delay time from CLK high	tDDAI1	10	70	ns	S4
DMAAK inactive delay time from HLDAAK low	tDDAI2	5	-80	ns	S4 Cascade Mode
\overline{TC} low delay time from CLK high	tDTCL		100	ns	S3
\overline{TC} off delay time from CLK high	tDTCF		40	ns	S4
\overline{TC} high delay time from CLK high	tDTCH		t _{KKH} + t _{CYK} -10	ns	0V→2.2V*1
\overline{TC} low level period	tTCTCL	t _{CYK} -15		ns	
\overline{END} low setup time to CLK high	tSED	35		ns	S2
\overline{END} low level period	tEDEL	100		ns	
READY setup time to CLK high	tSRY	35		ns	S3, SW
READY hold time from CLK high	tHRY	20		ns	S3, SW

Notes: *1 $\overline{END}/\overline{TC}$ has a 75 pF maximum input capacitance. To meet t_{DTCH}, use a 2.2K ohm or greater pull-up resistor with a load capacitance of 75 pF. The maximum output load capacitance for pins other than $\overline{END}/\overline{TC}$ is 100 pF.

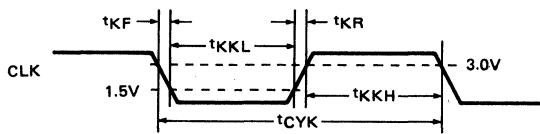
AC Characteristics (Programming Mode and RESET)

Parameter	Symbol	Limits		Unit	Test Conditions
		Min.	Max.		
$\overline{\text{IOWR}}$ low level period	tIWIWL	100		ns	
$\overline{\text{CS}}$ low setup time to $\overline{\text{IOWR}}$ high	tSCSIW	100		ns	
$\overline{\text{CS}}$ hold time from $\overline{\text{IOWR}}$ high	tHIWCS	0		ns	
ADR/ $\overline{\text{UBE}}$ setup time to $\overline{\text{IOWR}}$ high	tSAIW	100		ns	
ADR/ $\overline{\text{UBE}}$ hold time from $\overline{\text{IOWR}}$ high	tHIWA	0		ns	
Input data setup time to $\overline{\text{IOWR}}$ high	tSIDIW	100		ns	
Input data hold time from $\overline{\text{IOWR}}$ high	tHIWID	0		ns	
$\overline{\text{IORD}}$ low level period	tIRIRL	150		ns	
ADR/ $\overline{\text{CS}}$ setup time to $\overline{\text{IORD}}$ low	tSAIR	35		ns	
ADR/ $\overline{\text{CS}}$ hold time from $\overline{\text{IORD}}$ high	tHIRA	0		ns	
Output data delay time from $\overline{\text{IORD}}$ low	tDIROD		120	ns	
Output data float time from $\overline{\text{IORD}}$ high	tFIROD		100	ns	
RESET high level period	tRESET	2tcyk		ns	
VDD setup time to RESET low	tSVDD	500		ns	
$\overline{\text{IOWR}}$ / $\overline{\text{IORD}}$ wait time from RESET low	tSYIWR	2tcyk		ns	RESET Low to first Read/Write
$\overline{\text{IOWR}}$ / $\overline{\text{IORD}}$ recovery time	tRVIWR	200		ns	

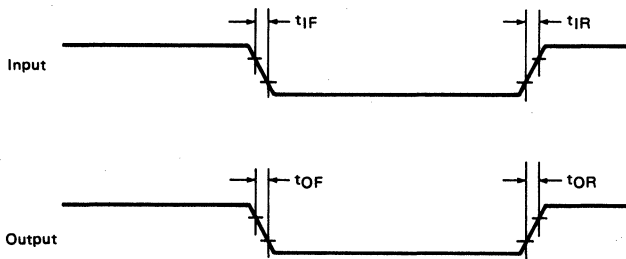
AC Test Waveforms



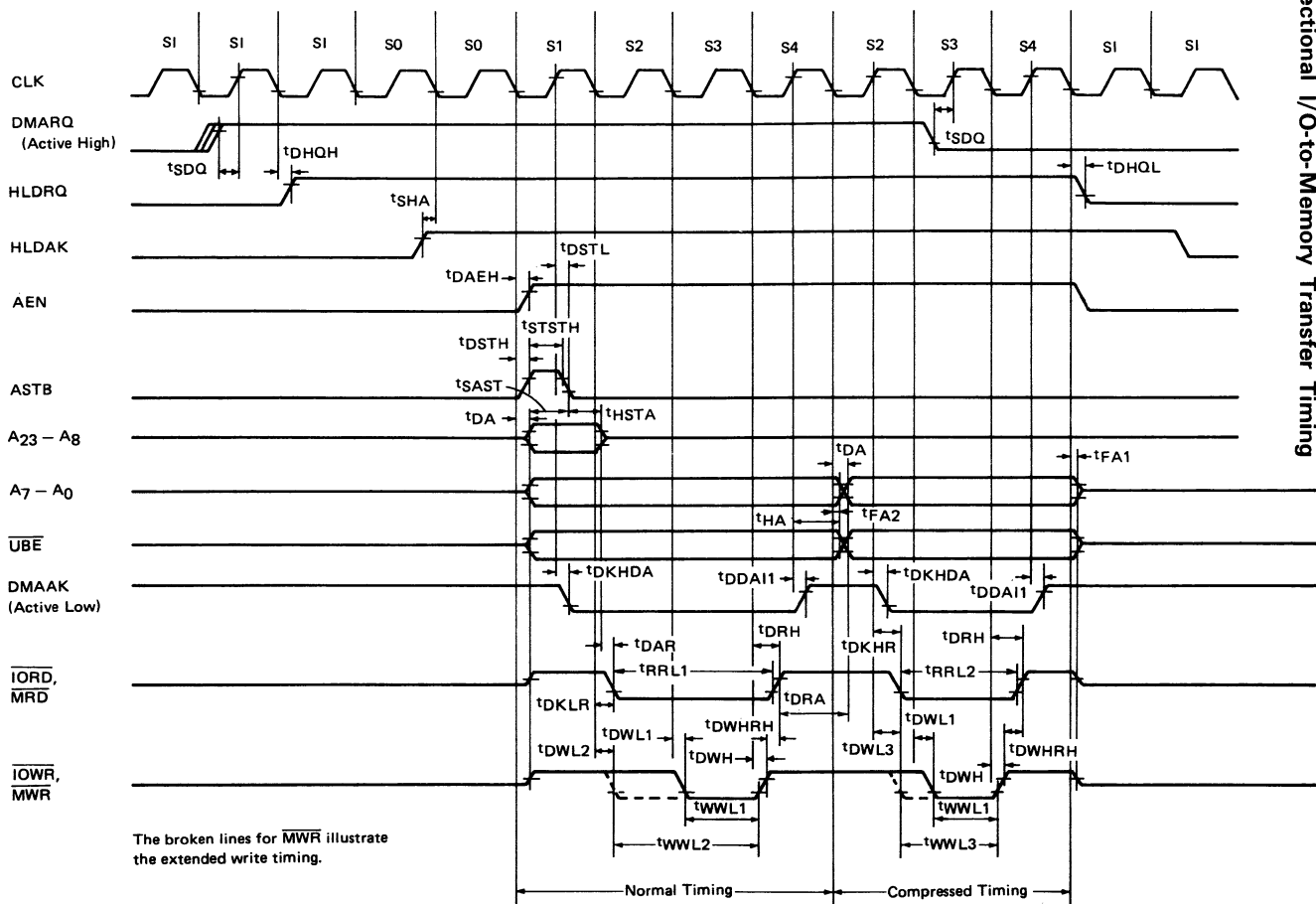
Clock Timing



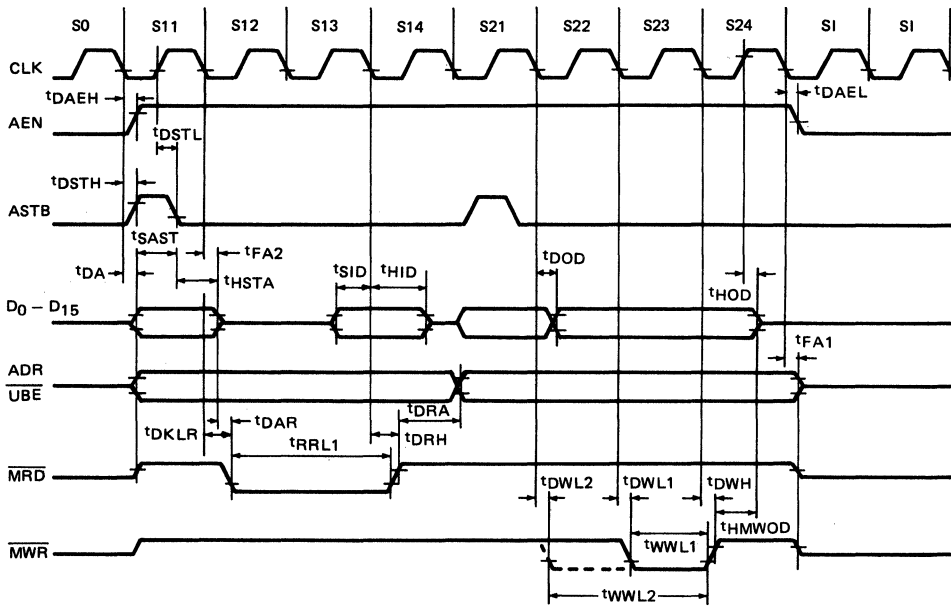
Input/Output Edge Timing



Directional I/O-to-Memory Transfer Timing

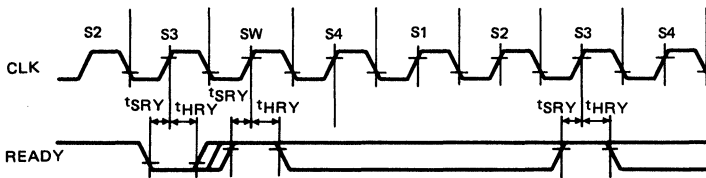


Memory-to-Memory Transfer Timing

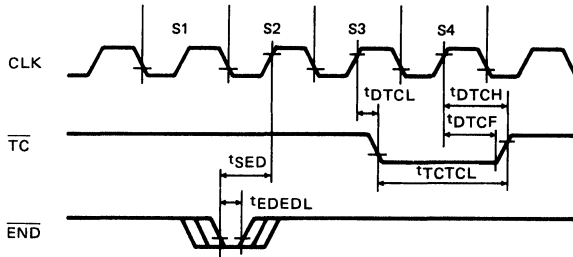


The broken line for \overline{MWR} illustrates the extended write timing.

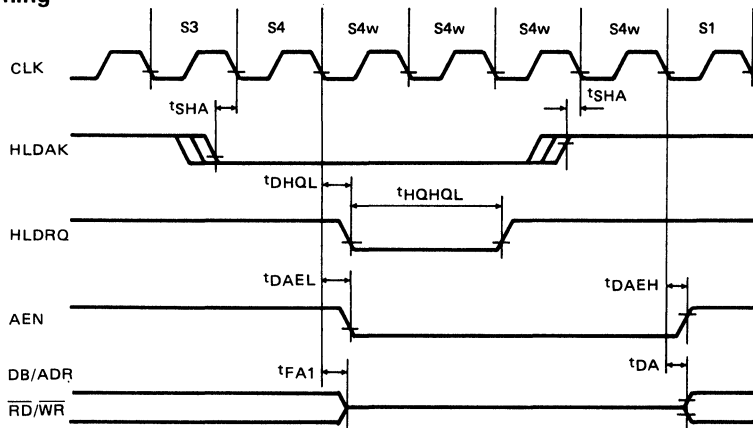
Ready Timing



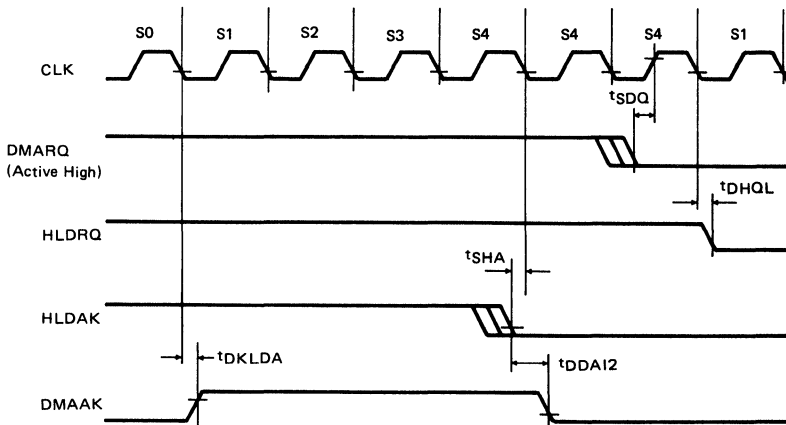
END/TC Timing



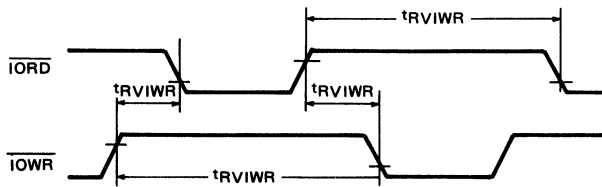
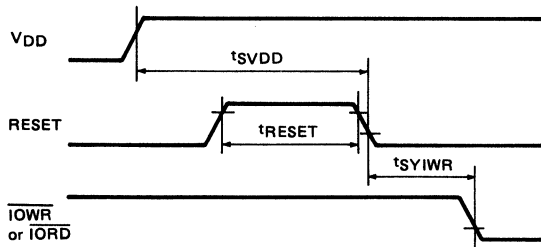
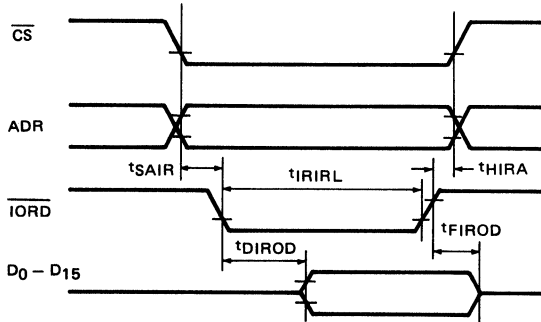
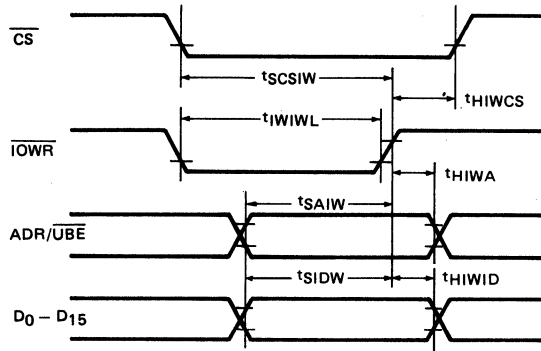
Bus Wait Timing



Cascade Timing



Programming Mode and RESET Timing



Functional Description

Bus Control Unit

The Bus Control Unit consists of the address and data buffers, and bus control logic. The Bus Control Unit generates and receives signals that control address and data on the internal address and data buses.

DMA Control Unit

The DMA Control Unit contains the priority and timing control logic. The priority control logic determines the priority level of DMA requests and arbitrates the use of the system bus according to the priority level. The timing control logic provides internal timing and controls DMA operations.

Address Registers

Each DMA channel has one 24-bit Base Address Register and one 24-bit Current Address Register. The Base Address Registers hold the value written by the CPU and transfer the value to the corresponding Current Address Register during autoinitialization (address and count are automatically initialized). The Current Address Register is automatically incremented/decremented for each transfer and always contains the address to be transferred next.

Address Incrementer/Decrementer

The Address Incrementer/Decrementer updates the contents of the Current Address Register whenever a DMA transfer for one bus cycle has finished.

Count Registers

Each DMA channel has one 16-bit Base Count Register and one 16-bit Current Count Register. The Base Count Register holds the value written by the CPU and transfers the value to the corresponding Current Count Register during autoinitialization. The Current Count Register is automatically decremented for each transfer and generates a terminal count when it reaches zero.

Note: The number of DMA transfer cycles is actually the value of the Current Count Register + 1.

Therefore, when programming the Count Register, specify the number of DMA transfers minus one.

Count Decrementer

The Count Decrementer decrements the contents of the Current Count Register by one when each DMA transfer cycle has finished.

Control Registers

The CXQ71071 contains the following control registers:

- Channel
- Device
- Status
- Mode
- Temporary
- Request
- Mask

These registers control bus mode, pin active level, DMA operation mode, mask bits, and other CXQ71071 operating functions.

DMA Operation

The CXQ71071 operates in two cycles: Idle and DMA. In an Idle cycle, the CPU uses the system bus, while in a DMA cycle, the CXQ71071 uses it.

Idle Cycle

In an Idle cycle, there are no DMA requests active or there are one or more active DMA requests, but the CPU has not released the bus. The CXQ71071 will sample the four DMARQ inputs every clock cycle to determine if any channel is requesting a DMA service. If one or more inputs are active, the corresponding DMA request bits (RQ) are set and the CXQ71071 will output a bus hold request (HLDRQ) to the CPU. The CXQ71071 continues to sample DMA requests until it obtains the bus.

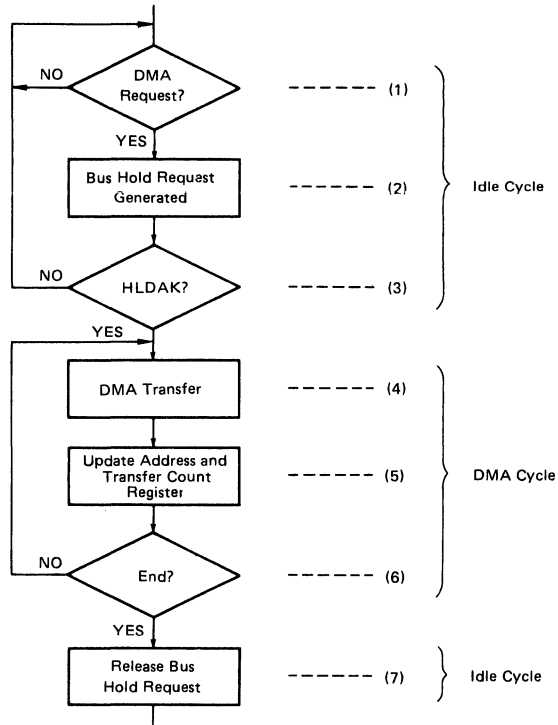
After the CPU returns a HLDACK signal and the CXQ71071 obtains the bus, the CXQ71071 stops DMA sampling and selects the highest priority channel from the valid DMA request signals.

The CXQ71071 will also sample \overline{CS} pin, checking an attempt by the CPU to read or write the internal registers of the device. Address lines A0-A3 select which registers will be read or written and \overline{IOR} and \overline{IOW} are used to select reading or writing.

DMA Cycle

In a DMA cycle, the CXQ71071 controls the bus in order to perform DMA transfer operations based on the programmed information. Figure 1 outlines the sequential flow of a DMA operation.

Figure 1. DMA Operation Flow



Data Bus Width

In order to allow an easy interface with an 8- or 16-bit CPU, the data bus width of the CXQ71071 is user programmable for 8 or 16 bits. A 16-bit data bus allows access to the 16-bit internal registers in one I/O bus cycle.

The initial bus width after reset is set to 8 bits.

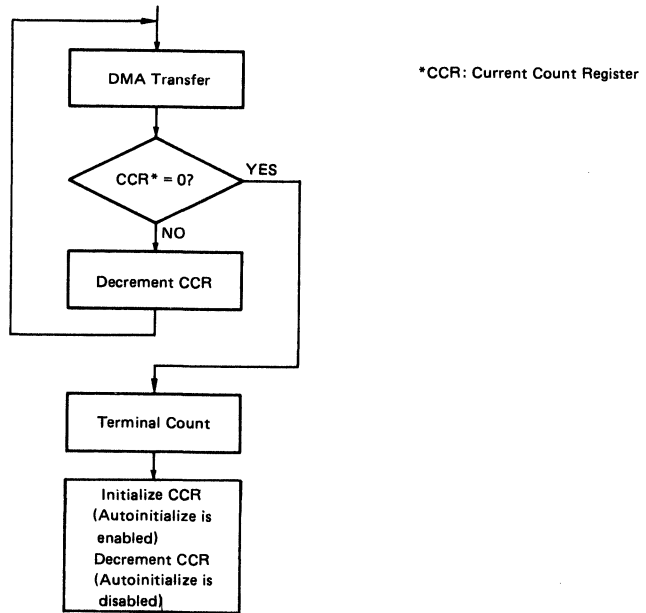
The following table shows the relationship of the data bus width, A0, \overline{UBE} , and the internal registers.

Bus Width	A0	\overline{UBE}	Internal Read/Write Registers
8 bits	X	X	D7 — D0 \longleftrightarrow 8-bit internal register
16 bits	0	1	D7 — D0 \longleftrightarrow 8-bit internal register
	1	0	D15 — D8 \longleftrightarrow 8-bit internal register
	0	0	D15 — D0 \longleftrightarrow 16-bit internal register

Terminal Count

The CXQ71071 ends DMA service when it internally generates a terminal count (\overline{TC}) or when \overline{END} externally becomes active. A terminal count is produced when the contents of the Current Count Register become 0 and output a low level pulse to the \overline{TC} pin. Figure 2 shows the relationship between the generation of the Terminal Count and the Current Count Register. The Current Count Register is tested after each DMA transfer and prior to decrementing it so that the DMA transfer is actually performed one more than the programmed value of the Current Count Register.

Figure 2. Generation of Terminal Count (\overline{TC})



Unless a channel is programmed for autoinitialize when DMA service ends, the corresponding bit of the Mask Register is set, and the DMARQ input of the channel is masked.

DMA Transfer Type

The type of transfer the CXQ71071 performs depends on the following conditions:

- Memory-to-memory transfer enable
- Direction of memory-to-I/O transfer (each channel)
- Transfer mode (each channel)
- Bus mode

Memory-to-Memory Transfer Enable.

The CXQ71071 can perform memory-to-I/O transfer (one transfer in one bus cycle) and memory-to-memory transfer (one transfer in two bus cycles).

Memory-to-memory transfer is enabled, when bit 0 of the Device Control Register is set to 1. The DMA channel used for memory-to-memory transfer is fixed, with Channel 0 as the source channel and Channel 1 as the destination channel. For this reason, the contents of the Count Registers and word/byte transfer modes of Channels 0 and 1 should be the same when performing memory-to-memory transfer. When DMARQ0 (Channel 0) becomes active by software, the transfer is initiated. The CXQ71071 performs the following operations until a Channel 1 terminal count or $\overline{\text{END}}$ input is present:

- The memory data pointed to by the Current Address Register of Channel 0 is read into the temporary register and the address and count of Channel 0 are updated.
- The temporary register data is written to the memory location shown by the Current Address Register of Channel 1, and the address and count of Channel 1 are updated.

Note: If DMARQ1 (Channel 1) becomes active while memory-to-memory transfer is enabled, the CXQ71071 will perform memory-to-I/O transfer. Since this may cause erroneous memory-to-memory transfers, bit 1 of the Mask Register should be set to 1.

During the memory-to-memory transfers, the source side (Channel 0) can be programmed to retain the same address by setting bit 1 of the Device Control Register to 1. In this manner, a range of memory can be initialized with the same value.

During memory-to-memory transfer, the DMAAK signals and Channel 0's terminal count (TC) pulse are not output.

Direction of Memory-to-I/O Transfers

All DMA transfers use memory as a reference point. A DMA Read reads data from memory and writes to an I/O port. A DMA Write reads data from an I/O port and writes to memory. In memory-to-I/O transfer, use the Mode Control Register to set one of the following transfer directions for each channel and activate the appropriate control signals.

Type	Transfer Direction	Activated Signals
DMA Read	Memory \longrightarrow I/O	$\overline{\text{IOWR}}$, $\overline{\text{MRD}}$
DMA Write	I/O \longrightarrow Memory	$\overline{\text{IORD}}$, $\overline{\text{MWR}}$
Verify	Verify transfer outputs address only and does not perform actual transfer	—

Transfer Modes

In a memory-to-I/O transfer, the Mode Control Register selects the single, demand, or block mode of DMA transfer for each channel. The conditions for the termination of each transfer characterize each transfer mode. The following table shows the various transfer modes and termination conditions:

Transfer Mode	Transfer End Conditions
Single	After each byte/word
Demand	<ul style="list-style-type: none"> • $\overline{\text{END}}$ input • Generation of terminal count • When DMA request of the channel in service becomes inactive • When DMA request of a channel in higher priority becomes active (Bus Hold mode)
Block	<ul style="list-style-type: none"> • $\overline{\text{END}}$ input • Generation of terminal count

Note: DMA transfer operations using memory-to-memory transfers are identical to the block transfer mode.

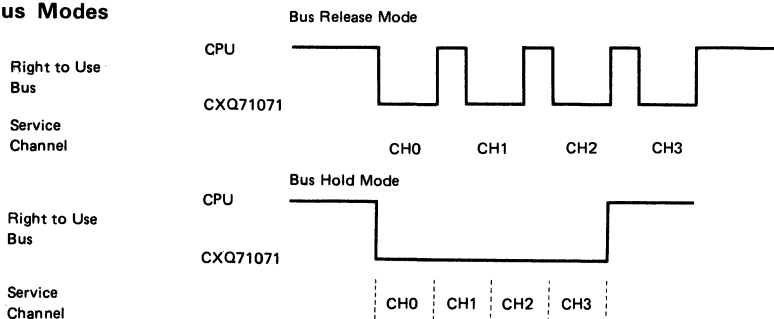
Bus Mode

The Device Control Register selects either the Bus Release or Bus Hold mode. The bus mode determines how the CXQ71071 returns the bus to the CPU.

Figure 3 shows that in Bus Release mode, only one channel can receive service for each DMA operation. Whenever DMA service terminates (transfer end conditions depend on the transfer mode), the channel returns the bus to the CPU (regardless of the state of other DMA requests) and the CXQ71071 enters the Idle cycle. When the CXQ71071 regains the bus, another DMA operation will begin.

In Bus Hold mode, several channels can receive service without releasing the bus after obtaining it. If there is another valid DMA request when a channel's DMA service is finished, the new DMA service can begin after the previous service without returning the bus to the CPU. Transfer end conditions depend on the transfer mode.

Figure 3. Bus Modes



The operation of Single, Demand, and Block Mode transfers depends on whether the CXQ71071 is in Bus Release or Bus Hold mode.

Single Mode Transfer

In Bus Release mode, when a channel completes the transfer of a single byte or word, the CXQ71071 enters the Idle cycle regardless of the state of the DMA request inputs. In this manner, other devices will be able to access the bus on alternate bus cycles.

In Bus Hold mode, when a channel completes the transfer of a single byte or word, the CXQ71071 terminates the channel's service even if it is still asserting a DMA request signal. The CXQ71071 will then service the highest priority requesting channel. If there are no requests from any other channel, the CXQ71071 releases the bus and enters the Idle cycle.

Demand Mode Transfer

In Bus Release mode, the current active channel continues its data transfer as long as the DMA request of that channel is active, even though other DMA channels are issuing higher priority requests. When the DMA request of the channel in service goes inactive, the CXQ71071 releases the bus to the CPU and enters the Idle state, even if the DMA requests from other channels are active.

In Bus Hold mode, when the active channel completes a single transfer, the CXQ71071 checks the other DMA request lines without ending the state of the current service. If there is a higher priority request, the CXQ71071 suspends servicing the current channel and starts servicing the highest priority channel requesting service, without releasing the bus. If there is no higher request than the current one, the CXQ71071 continues to service the currently active channel. Lower priority DMA requests are honored without releasing the bus after the current channel service is completed.

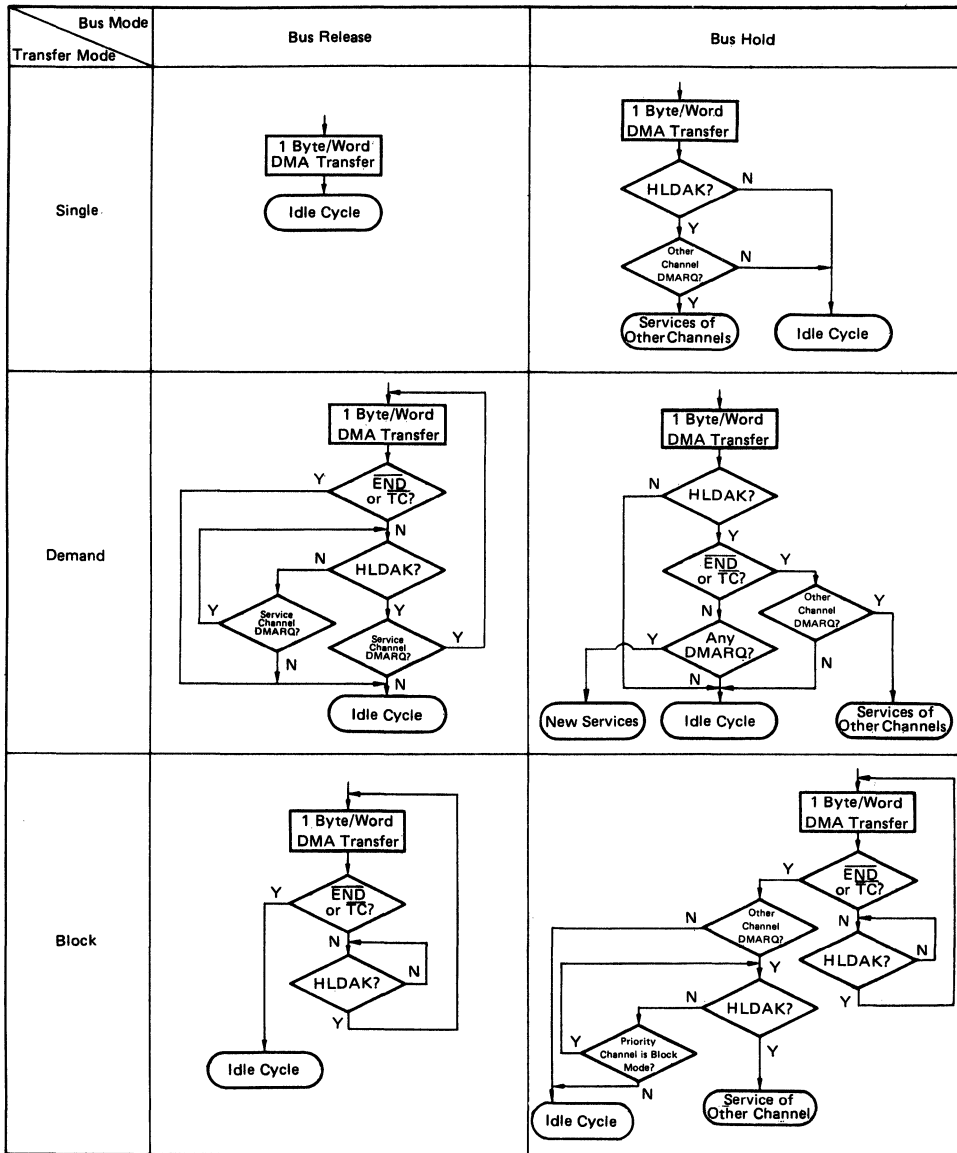
Block Mode Transfer

In Bus Release mode, the current channel continues data transfer until a terminal count or the external $\overline{\text{END}}$ signal becomes active. During this time, the CXQ71071 ignores all other DMA requests. After completion of the block transfer, the CXQ71071 releases the bus and enters the Idle cycle even if the DMA requests from other channels are active.

In Bus Hold mode, the current channel transfers data until an internal or external $\overline{\text{END}}$ signal becomes active. When the service is completed, the CXQ71071 checks all DMA requests without releasing the bus. If there is an active request, the CXQ71071 immediately begins servicing the request. The CXQ71071 releases the bus after it honors all DMA requests.

Figure 4 shows the operation flow for the six possible transfer and bus mode operations in DMA transfer.

Figure 4. Transfer and Bus Modes Operations



Byte/Word Transfer

If the Initialize Command selects a 16-bit data bus width, the Mode Control Register can specify DMA transfer in byte or word units for each channel. The following table shows the byte count by which the Address and Count Registers are incremented or decremented during byte/word transfer.

Register	Byte Transfer	Word Transfer
Address	±1	±2
Count	-1	-1

During word transfers, two bytes starting at an even address are handled as one word. If the initial value of the programmed address is odd, transfer is started after decrementing the address by 1. For this reason, always select even addresses as the initial value to avoid destroying data. Byte and word transfers are controlled by the A_0 and \overline{UBE} signals.

The following table shows the relationship between the data bus width, A_0 and \overline{UBE} signals, and data bus status.

Data Bus Width	A_0	\overline{UBE}	Data Bus Status
8 bits	X	1*	D7—D0 valid byte
16 bits	0	1	D7—D0 valid byte
	1	0	D15—D8 valid byte
	0	0	D15—D0 valid word

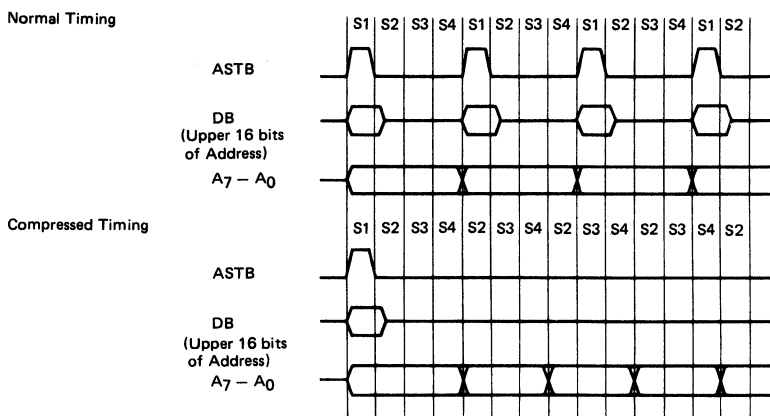
*Note: Always 1

Compressed Timing

A DMA transfer cycle is normally executed in four clocks. However, when the Device Control Register selects compressed timing, one DMA cycle can be executed in three clock cycles. Compressed timing is only available in block mode (except memory-to-memory) and in demand mode during Bus Release mode, for 33% more efficiency.

The CXQ71071 is able to omit one clock during compressed timing by not updating the upper 16 bits of the latched address. In Block mode and Demand mode during bus release, addresses are output sequentially and the upper 16 bits of addresses latched in external latches need not be updated except after a carry or borrow from A_7 to A_8 . For this reason, during compressed timing, the S1 state (output of upper 16 bits of an address for external latching) can be omitted in the bus cycles except during the first bus cycle and when the upper 16 bits of an address is changed. Figure 5 shows wave forms for normal and compressed timing.

Figure 5. Normal and Compressed Timing Waveforms



Software DMA Requests

The CXQ71071 can accept software DMA requests in addition to DMA requests from the four DMARQ pins. Software DMA requests are generated by setting the appropriate bit in the Request Register. Software DMA requests are not masked by the Mask Register and operate differently depending on which bus or transfer mode is used.

Bus Mode

When Bus Release mode is set, the highest priority channel among software DMA requests and DMARQ pins will be serviced, and all bits of the Request Register will be cleared when the service is over. There may be a chance that other software DMA requests will be cancelled.

When Bus Hold mode is set, only the corresponding bit of the Request Register will be cleared after a DMA service is over. All software DMA requests will be serviced in the sequence of their priority level.

Precaution must be taken for software DMA requests for cascade channels (See the Cascade Connection) in Bus Hold mode. While a cascade channel is serviced, the master CXQ71071 operational mode is changed to Bus Release mode temporarily and all bits of the request register are cleared when the cascade channel service is over. To avoid this, it is necessary to mask any cascade channels before issuing a software DMA request. After confirming that all DMA software services are completed and all bits of the Request Register are cleared, the cascade channel masks can be cleared.

Transfer Mode

When Single or Demand mode is programmed, the corresponding request bits are cleared and software DMA service ends with the transfer of one byte/word. When Block mode (memory-to-memory) is programmed, service continues until END is input or a terminal count is generated. The corresponding request bits are cleared when service ends.

Autoinitialize

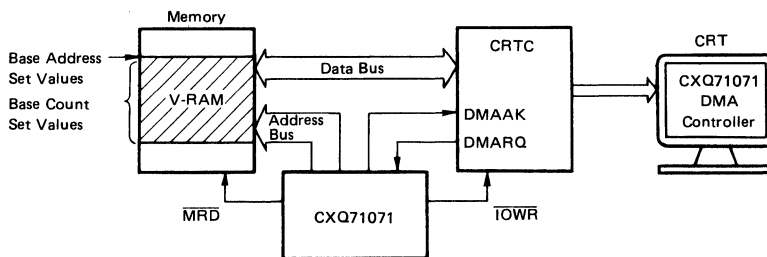
When the Mode Control Register selects autoinitialize for a channel, the CXQ71071 automatically initializes the address and count when $\overline{\text{END}}$ is input or a terminal count is generated. Then the contents of the Base Address and Base Count Registers are transferred to the Current Address and Current Count Registers, respectively. The corresponding bit of the Mask Register is cleared. However, the bit of the Mask Register is set for channels not programmed for autoinitialize.

Use the autoinitialize function for the following types of transfers:

Repetitive Input/Output of Memory Area

Figure 6A shows an example of DMA transfer between a CRT controller and memory. After setting the same value in the Base and Current Registers, autoinitialize allows repetitive DMA transfer without CPU involvement.

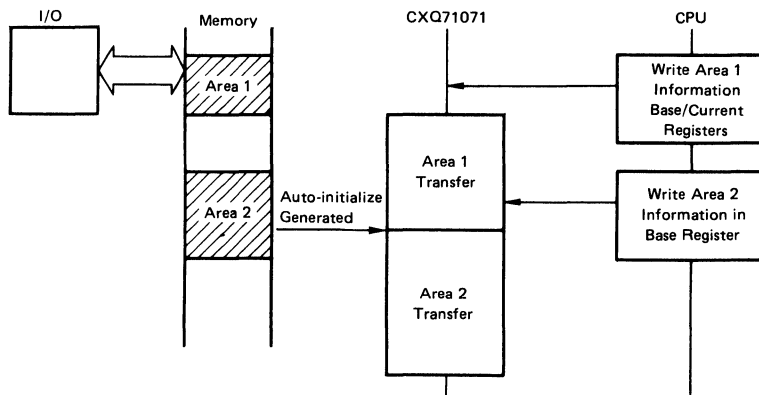
Figure 6A. Autoinitialize, Application 1



Continuous Transfer of Several Memory Areas

The CPU can write the CXQ71071 only to the Address/Count Base Registers for programming the address/count information. The autoinitialize function can be used to perform continuous transfer of several contiguous or non-contiguous memory areas during Single or Demand mode in the Bus Release mode. If the CPU sets information for Area 2 in Base Registers during the previous transfer of data for Area 1, the Current Registers will be automatically restored from the Base Registers following the generation of a terminal count. Figure 6B illustrates this procedure.

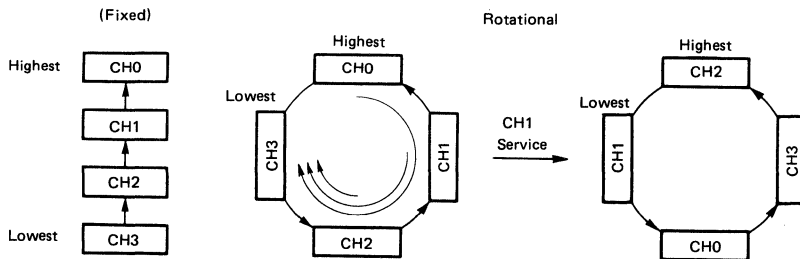
Figure 6B. Autoinitialize, Application 2



Channel Priority

Each of the CXQ71071's four channels has its own priority. When there are DMA requests from several channels simultaneously, the channel with the highest priority will be serviced. The Device Control Register selects two types of priority encoding: fixed and rotational priority. In fixed priority, channel 0 is assigned the highest priority through channel 3 the lowest. In rotational priority, priority order is rotated so that the last channel to get service is assigned the lowest priority with the others rotating accordingly. This method prevents exclusive servicing of some channel (s). Figure 7 shows the two priority order methods.

Figure 7. Priority Order

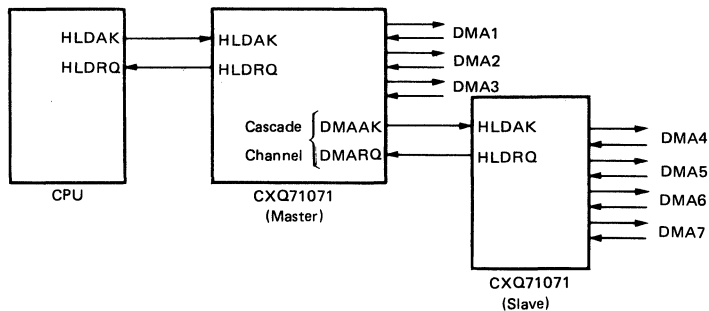


Cascade Connection

The CXQ71071 can be cascaded to expand the system DMA channel capacity. To connect a CXQ71071 for cascading (Figure 8),

1. Connect pins HLDK and HLDK of the slave CXQ71071 to pins DMARQ and DMAAK of any channels of the master CXQ71071.
2. Set bits 7 and 6 of the Mode Control Register to 11 in order to select Cascade mode for the Master's channel.

Figure 8. Cascade Connection Example



In the master CXQ71071, DMARQ, DMAAK, HLDRQ, and HLDK only become valid during the DMA service of the channel set to the Cascade mode. The other signals are disabled so as not to conflict with the outputs of the active channel in the slave CXQ71071. The master cascade channel only propagates hold request/hold acknowledge between the slave and CPU.

The master CXQ71071 always operates in the Bus Release mode while a cascade channel is in service even when the Bus Hold mode is set. Other DMA requests are ignored while a cascade channel is in service. When the slave CXQ71071 ends DMA service and moves into an Idle cycle, the master also moves to an Idle cycle and releases the bus. At this time, all bits of the master's Request Register are cleared. The master operates the other non-cascaded channels normally.

Bus Waiting Operation

In systems using a V40™/V50™ as the CPU, even during a DMA cycle, the on-chip refresh control unit in the CPU may lower the level of the HLDK signal to inactive and use the bus. Therefore, the CXQ71071 automatically performs a bus waiting operation in a system that has a bus master whose priority level is higher than that of the CXQ71071.

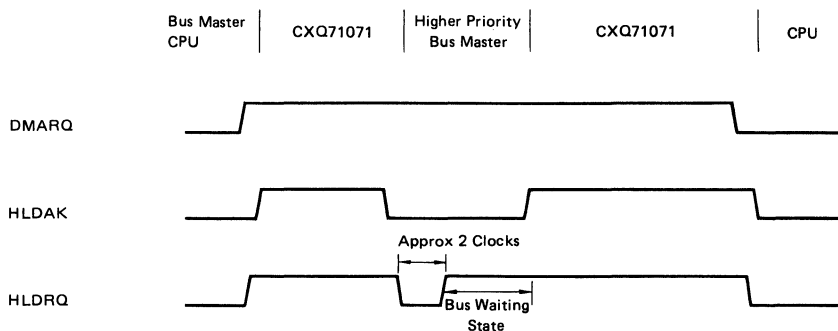
The CXQ71071 executes the bus waiting operation if the HLDK signal becomes inactive during service in an operating mode where transfer is executed continuously during block transfer mode; during demand transfer mode in bus release mode; and during memory-to-memory transfer.

When HLDK becomes inactive during service in other operating modes, the device returns to the Idle cycle and passes the control of the bus to the higher bus master.

Figure 9 shows that when the HLDK signal becomes inactive during service for continuous transfer, the CXQ71071 is set up in an S4w state (bus waiting). Operation moves to an Idle cycle if DMARQ is inactive in the demand transfer mode.

The HLDRQ signal becomes inactive for a period of about two clocks and the bus is released. The S4w state is repeated until the HLDK signal again becomes active and the interrupted service is immediately restarted.

Figure 9. Bus Waiting Operation



Programming the CXQ71071

To prepare a channel for DMA transfer, the following information must be defined:

- starting address for the transfer
- number of transfer cycles
- DMA operating modes
- data bus widths
- active levels of the DMARQ and DMAAK signals

The CXQ71071 contain 395 bits of internal memory in the form of registers. The address lines A₃—A₀ are used to address the register to be read or written.

The following tables show the register and command configurations.

Register Configuration

Register	Bit size	Number
Channel	5	1
Base Address	24	4
Current Address	24	4
Base Count	16	4
Current Count	16	4
Mode Control	7	4
Device Control	10	1
Status	8	1
Request	4	1
Mask	4	1
Temporary	16	1

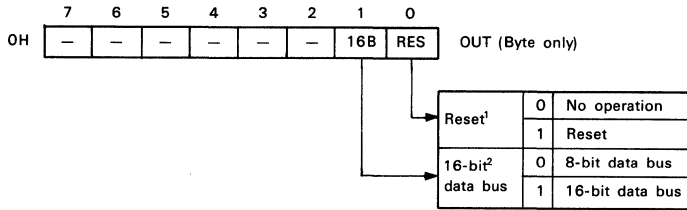
Note: When using a 16-bit CPU and selecting a 16-bit data bus, use the word IN/OUT instruction to read/write information two bytes at a time. However, the commands suffixed with "(B)" in the following table, must be issued with the byte IN/OUT instruction.

Command Configuration

Address	R/W	Command Name	MSB	Format				LSB		
0H	W(B)	Initialize	—	—	—	—	—	16B	RES	
1H	R(B)	Channel Register Read	—	—	—	BASE	SEL3	SEL2	SEL1	SEL0
	W(B)	Channel Register Write	—	—	—	—	—	—	BASE	SELCH
2H	R/W	Count Register Read/Write	C7	C6	C5	C4	C3	C2	C1	C0
3H	R/W		C15	C14	C13	C12	C11	C10	C9	C8
4H	R/W	Address Register Read/Write	A7	A6	A5	A4	A3	A2	A1	A0
5H	R/W		A15	A14	A13	A12	A11	A10	A9	A8
6H	R/W(B)		A23	A22	A21	A20	A19	A18	A17	A16
8H	R/W	Device Control Reg. Read/Write	AKL	RQL	EXW	ROT	CMP	DDMA	AHLD	MTM
9H	R/W		—	—	—	—	—	—	WEV	BHLD
0AH	R(B)	Status Register Read	RQ3	RQ2	RQ1	RQ0	TC3	TC2	TC1	TC0
0BH	R/W(B)	Mode Control Reg. Read/Write	TMODE		ADIR	AUTI	TDIR		—	W/B
0CH	R	Temporary Reg. (lower) Read	T7	T6	T5	T4	T3	T2	T1	T0
0DH	R	Temporary Reg. (higher) Read	T15	T14	T13	T12	T11	T10	T9	T8
0EH	R/W(B)	Request Reg. Read/Write	—	—	—	—	SRQ3	SRQ2	SRQ1	SRQ0
0FH	R/W(B)	Mask Reg. Read/Write	—	—	—	—	M3	M2	M1	M0

Initialize

The following figure shows the CXQ71071 initialize process.



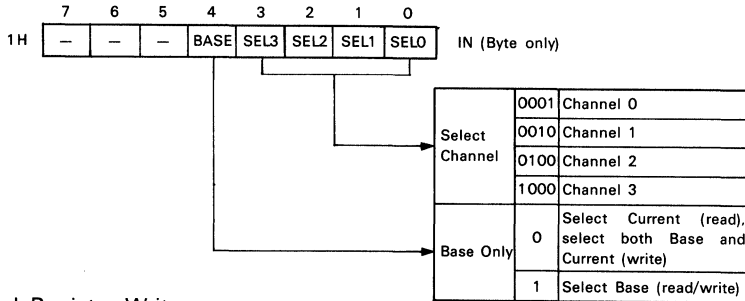
- Notes:**
- The CXQ71071 initializes as follows:

Register	Initialization Operation
Initialize	Clears all bits
Address	No change
Count	No change
Channel	Selects Channel 0
Mode Control	Clears all bits
Device Control	Clears all bits
Status	Clears all bits
Request	Clears all bits
Mask	Sets all bits (masks all channels)
Temporary	Clears all bits
 - When using the CXQ71071 in a 16-bit system, set this bit immediately after a hardware reset since the CXQ71071 always initializes in the 8-bit data bus mode.

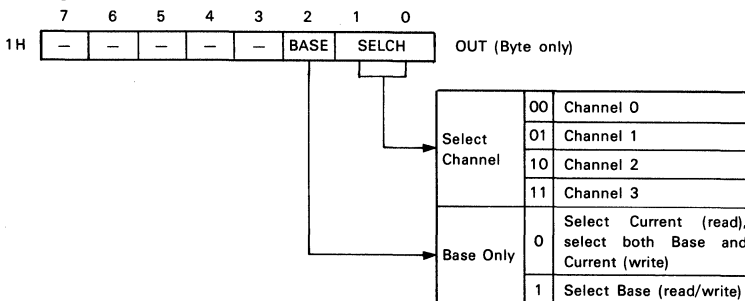
Channel Register

This command reads and writes the Channel Register that selects one of four DMA channels for programming by the Address, Count and Mode Control Registers. This command must be issued by the byte IN/OUT instruction.

Channel Register Read



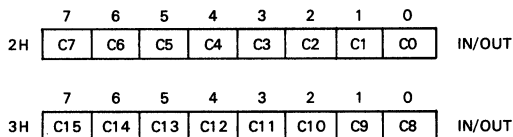
Channel Register Write



Count Register Read/Write

If bit 2 of the Channel Register is cleared, a write to the Count Register updates both the Base and Current Count Registers with the new data. If bit 2 of the Channel Register is set, a write to the Count Register only affects the Base Count Register.

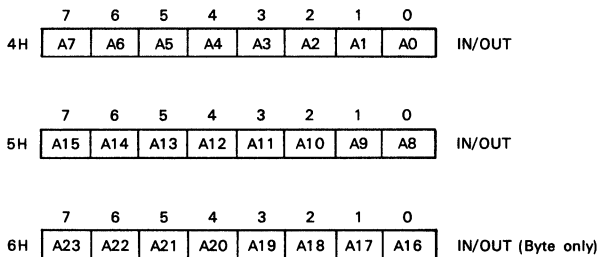
The Base Count Registers hold the initial count value until a new count is specified. If autoinitialization is enabled, this value is transferred to the Current Count Register when an END or TC is generated. For each DMA transfer, the Current Count Register is decremented by one.



Address Register Read/Write

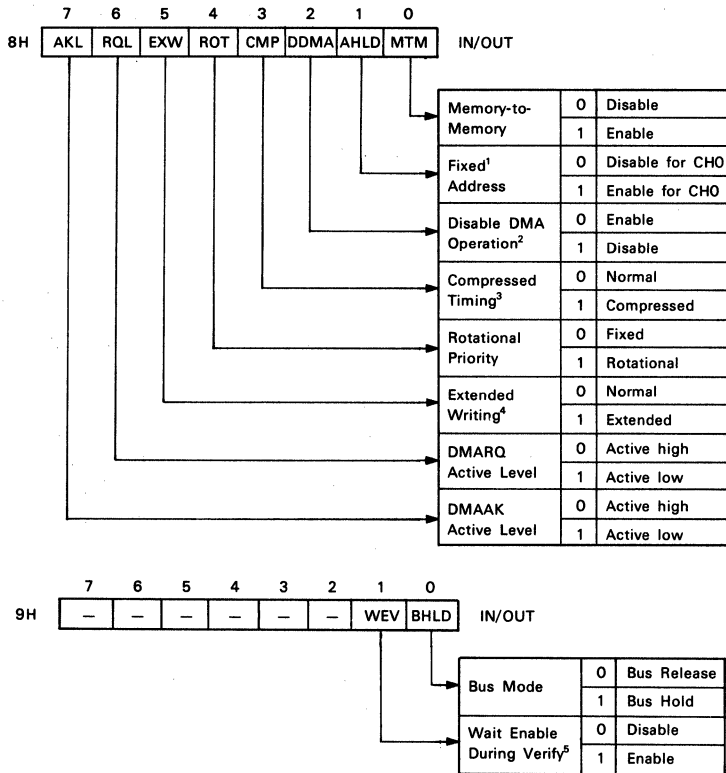
The word IN/OUT instruction is used to specify the lower two bytes (4H and 5H) of the register if a 16-bit data bus width is selected. The byte IN/OUT instruction must be used to specify the upper byte (6H) of the register. When bit 2 of the Channel Register is cleared, a write to the Address Register updates both the Base and Current Address Registers with the new data. If bit 2 of the Channel Register is set, a write to the Address Register only affects the Base Address Register.

The Base Register holds the starting address value until a new value is specified and this value is transferred to the Current Address Register during autoinitialization. For each DMA transfer, the Current Address Register is incremented or decremented by two during word transfer and by one during byte transfer.



Device Control Register Read/Write

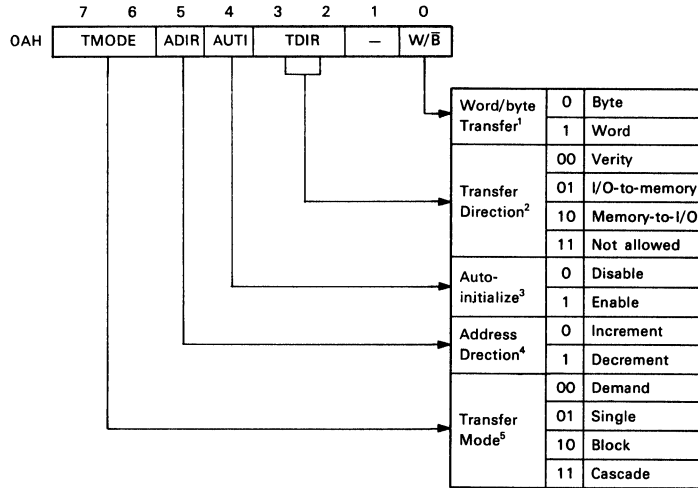
The Device Control Register Read/Write command reads from and writes to the Device Control Register. If using a 16-bit data bus, the word IN/OUT instruction is used to read and write 16-bit data.



- Notes:**
1. This bit is meaningless when MTM = 0.
 2. Disables HLDRQ to the CPU to prevent incorrect DMA operation while the CXQ71071's registers are being initialized or modified.
 3. Performs compressed timing DMA transfer in block or demand mode during bus release.
 4. When EXW is 0, the write signal becomes active (normal write) during S3 and SW (see the timing waveforms). When 1, the write signal becomes active during S2, S3, and SW (like the read signal).
 5. Wait states are generated by the READY signal during a verify transfer.

Mode Control Register Read/Write

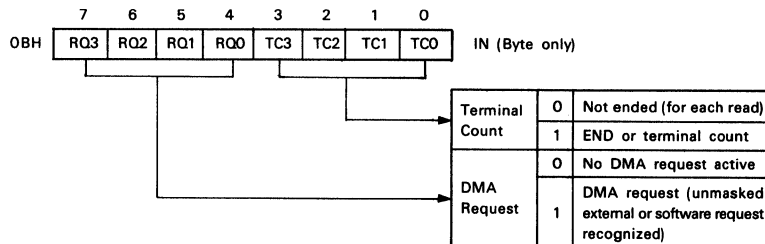
This command reads from and writes to the Mode Control Register to specify the operating mode for each channel. The Channel Register selects the Mode Control Register. This command must be issued by the byte IN/OUT instruction.



- Notes:**
1. When a 16-bit data bus is selected, this bit selects DMA transfer by word or byte.
 2. These bits select the DMA transfer direction between memory and I/O. These bits are meaningless during memory-to-memory transfer.
 3. Channel 0 and 1 must have the same AUTI bit value when performing memory-to-memory transfer.
 4. This bit decides the update direction of the Current Address Register. When ADIR is 0, the register increments by 1 for a byte transfer and by 2 for a word transfer. When ADIR is 1, the register decrements by 1 for a byte transfer and by 2 for a word transfer.
 5. These bits select the transfer mode during DMA transfer between memory and I/O, and are meaningless during memory-to-memory transfer.

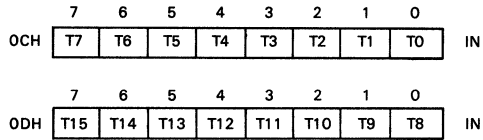
Status Register Read

This command reads the Status Register for the individual DMA channel that has DMA request states and terminal count or END information. This command must be issued by the byte IN instruction.



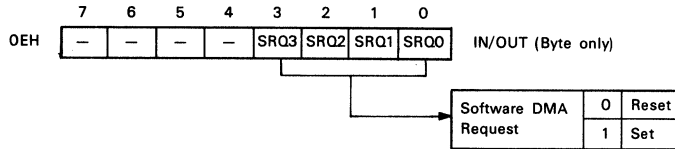
Temporary Register Read

If a 16-bit data bus is selected, the word IN instruction is used to read 16-bit data with this command. The last data transferred in memory-to-memory transfer is stored in the temporary register. If an 8-bit data bus is selected, the value of the upper byte becomes undefined.



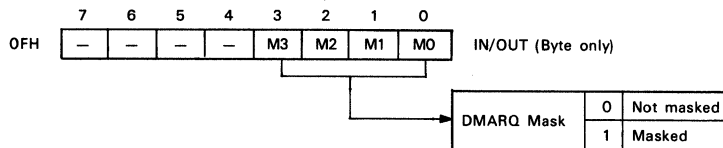
Request Register Read/Write

This command reads from and writes to the Request Register to generate DMA requests by software for the four corresponding DMA channels. This command must be issued by the byte IN/OUT instruction.



Mask Register Read/Write

This command reads from and writes to the Mask Register to control DMA request for the corresponding four DMA channels by DMARQ3—DMARQ0. This command must be issued by the byte IN/OUT instruction.



DMA Transfer Modes

Figures 10 through 15 show state transition diagrams for the different modes of DMA transfer.

Figure 10. Idle Cycle

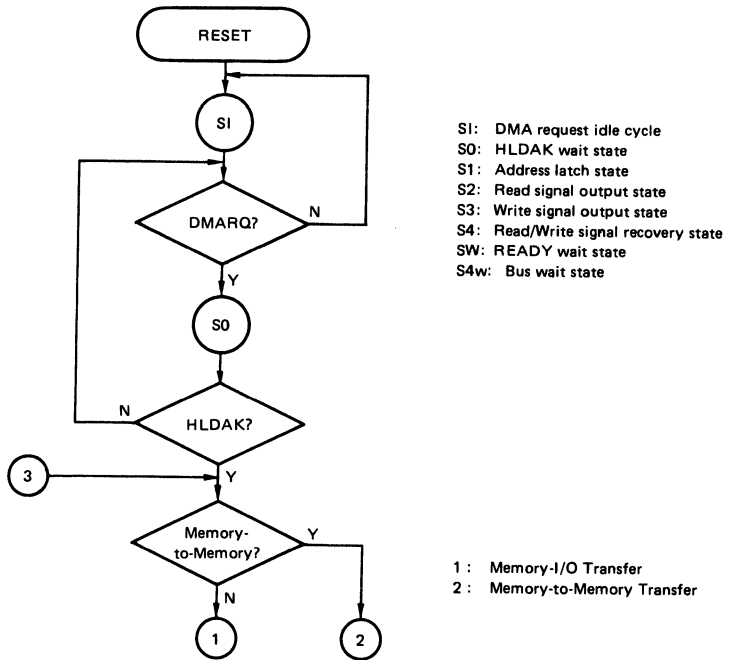


Figure 11. DMA Cycle, Cascade Mode

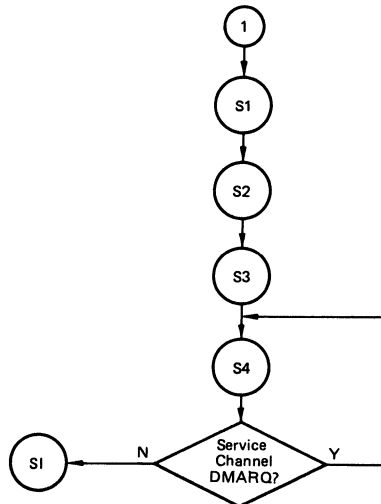


Figure 12. DMA Cycle, Single Mode

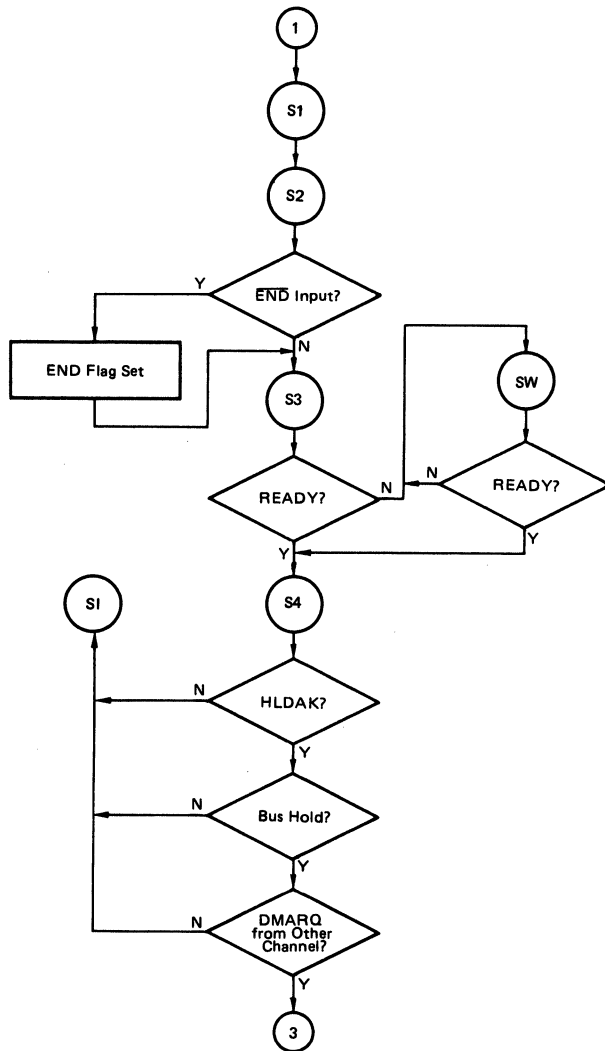
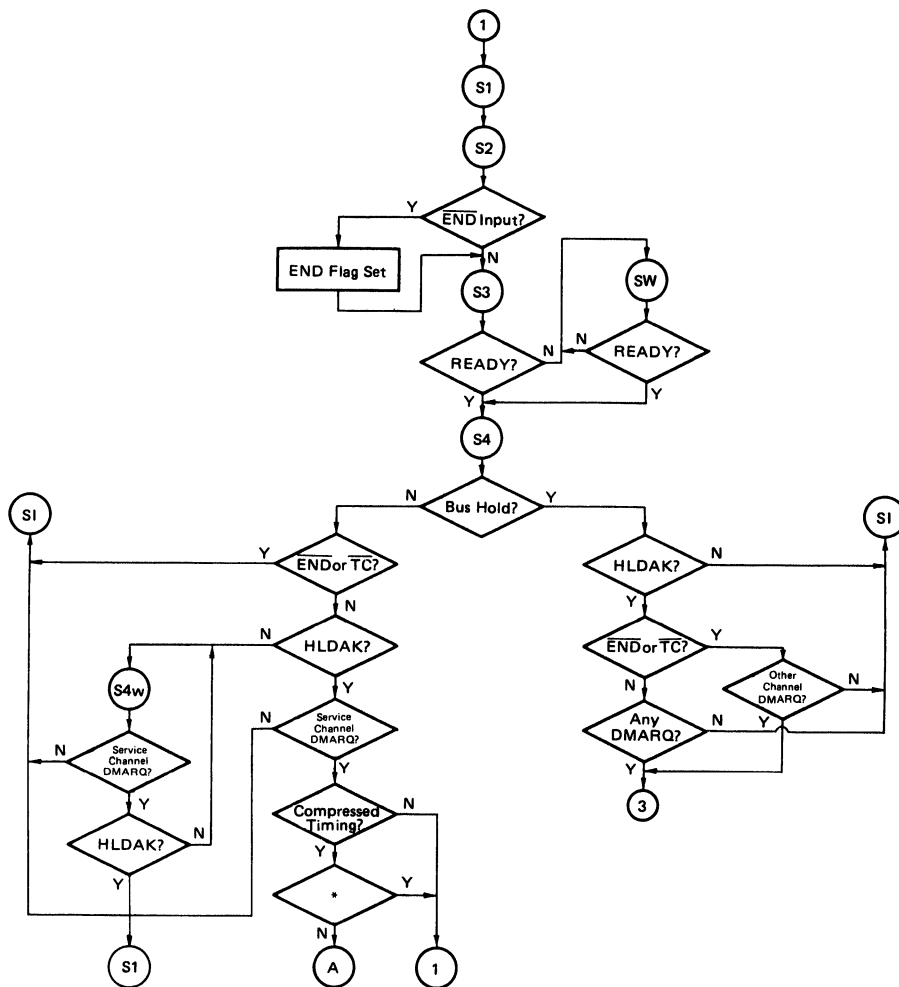


Figure 13. DMA Cycle, Demand Mode



* Carry or Borrow to Upper Two-Bytes of Address?

Figure 14. DMA Cycle, Block Mode

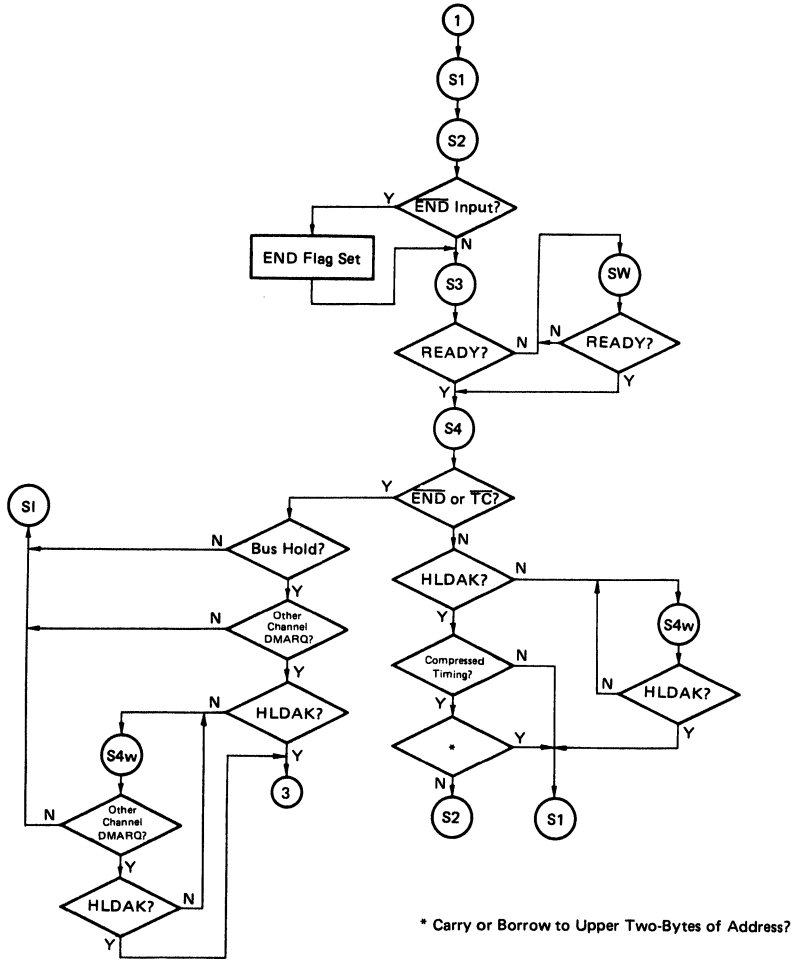
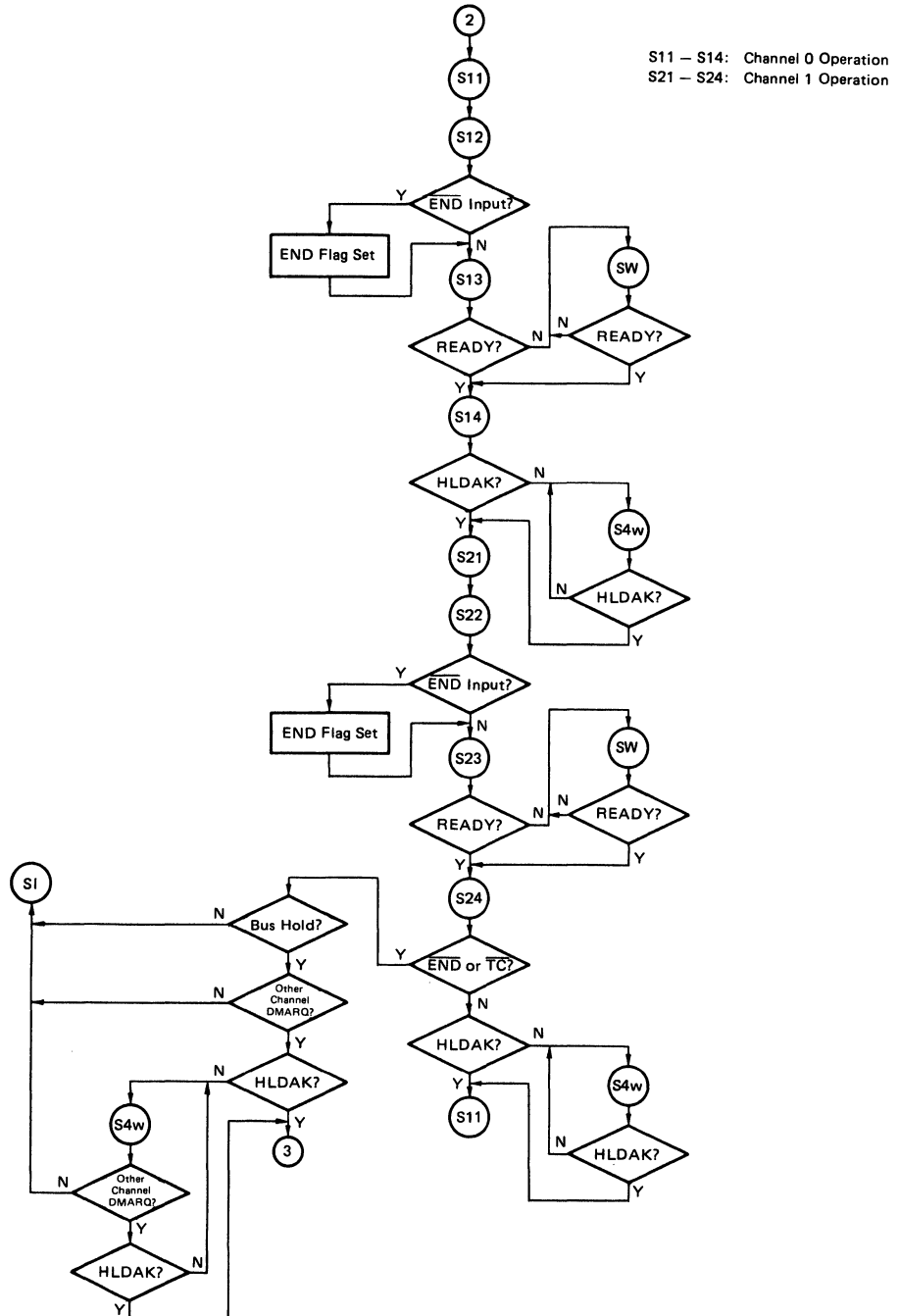


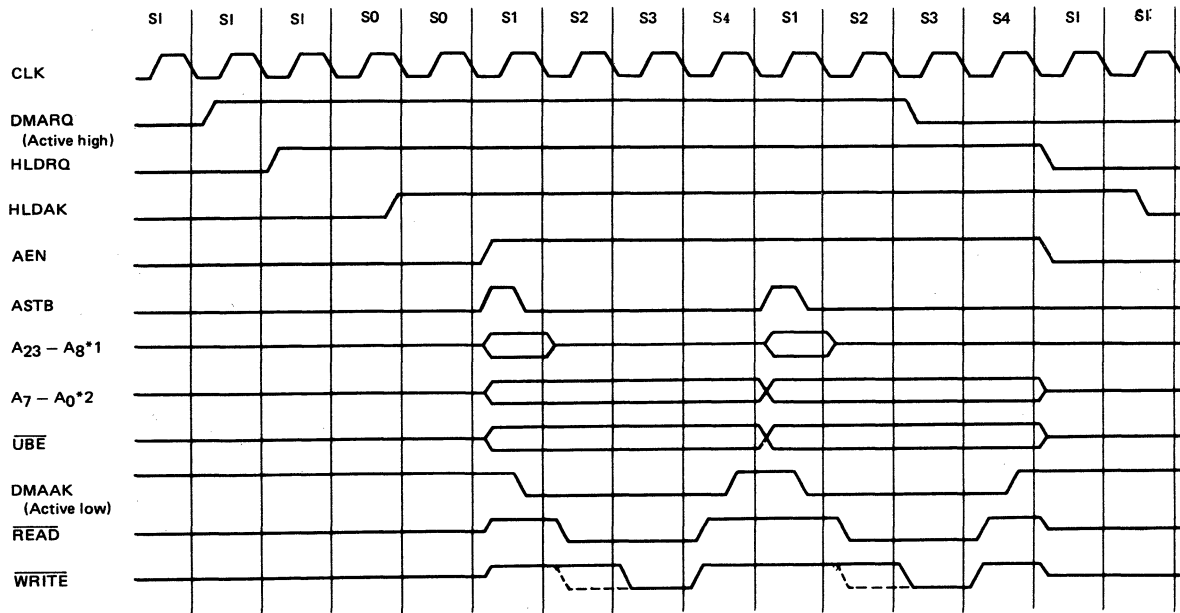
Figure 15. DMA Cycle, Memory-to-Memory Transfer



Transfer Timing

Figures 16 through 18 show CX071071 timing waveforms.

Figures 16. Memory-I/O Transfer, Normal Timing



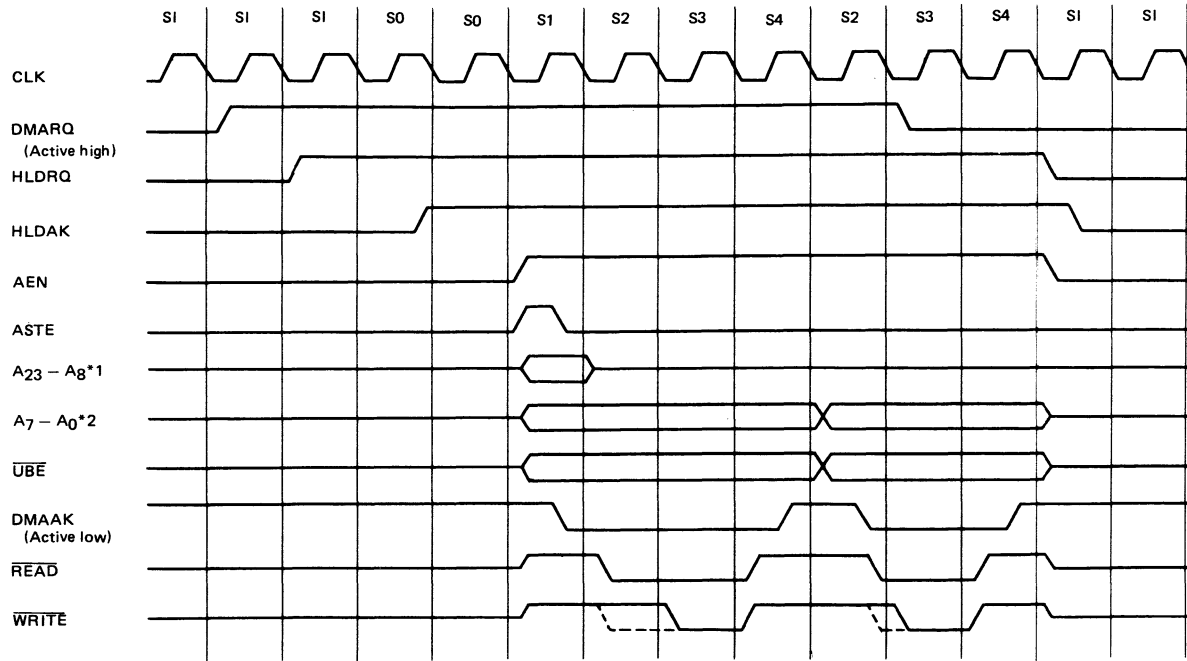
Notes: If an 8-bit data bus is selected,

1: A15—A8

2: A23—A16, A7—A0

The broken lines of the $\overline{\text{WRITE}}$ signal are for extended write timing.

Figure 17. Memory-I/O Transfer, Compressed Timing



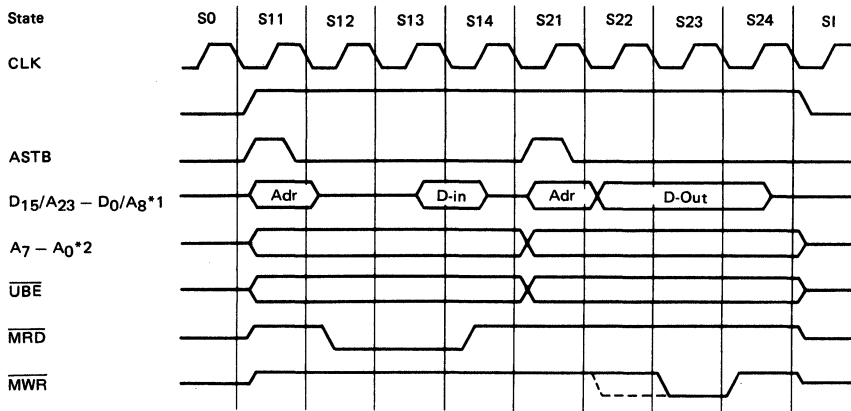
Notes: If an 8-bit data bus is selected,

1: $A_{15}-A_8$

2: $A_{23}-A_{16}$, A_7-A_0

The broken lines of the WRITE signal are for extended write timing.

Figure 18. Memory-to-memory Transfer

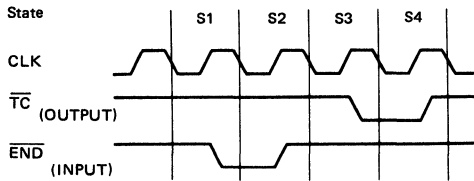


Notes: If an 8-bit data bus is selected,

- 1: D7/A15—D0/A8
- 2: A23—A16, A7—A0

The broken lines of the $\overline{\text{MWR}}$ signal are for extended write timing.

Figure 19. $\overline{\text{END}}/\overline{\text{TC}}$ Timing



Examples of System Configuration

Figures 20 through 22 show system configuration examples using the 8-bit CXQ70108 CPU and the 16-bit CXQ70116 CPU. The CXQ71082 externally latches addresses and data.

Figure 20. System Configuration with CXQ70108

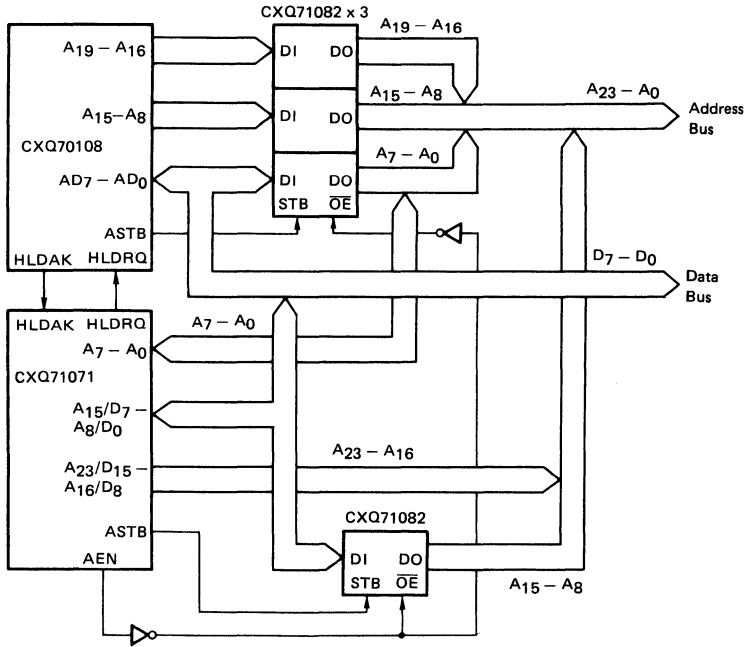


Figure 21. System Configuration with CXQ70116, Byte Transfer

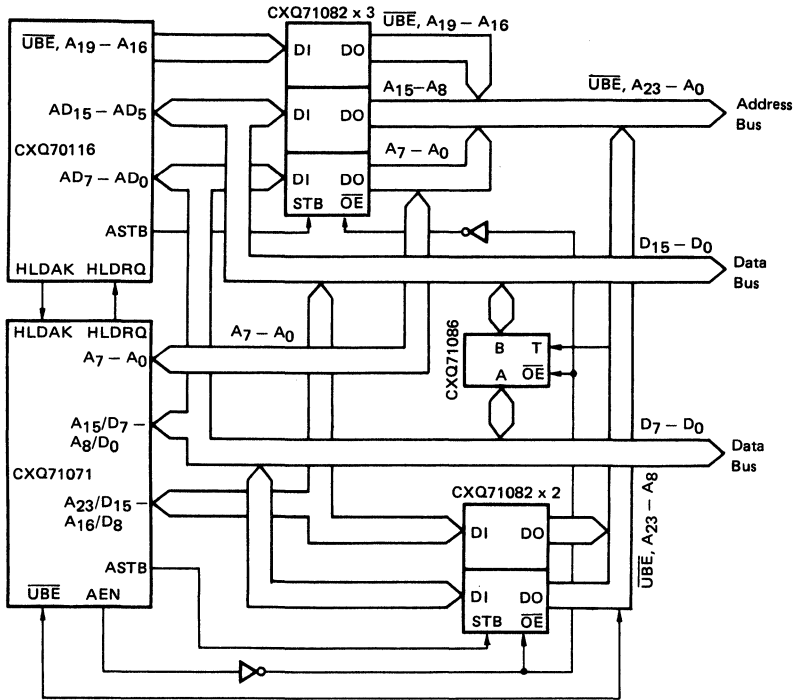
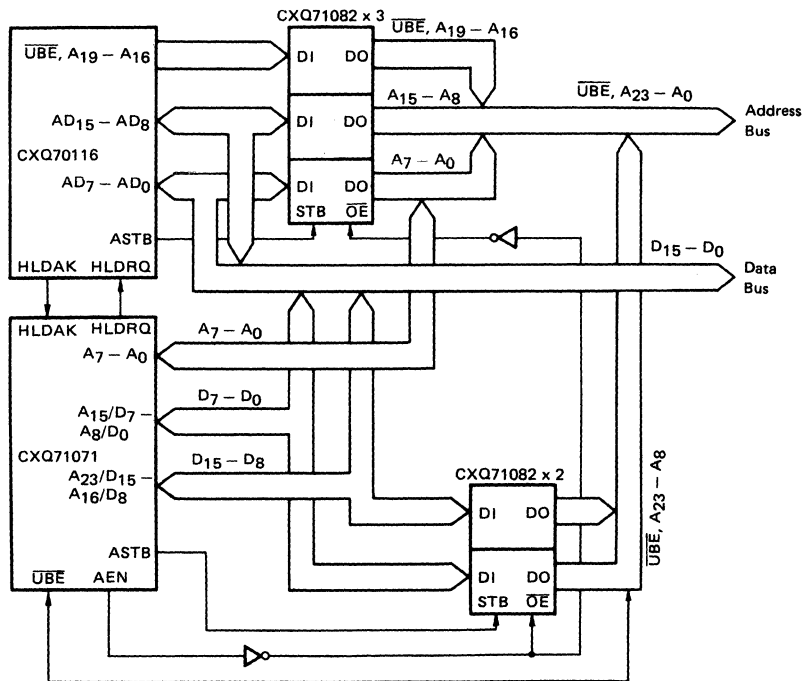
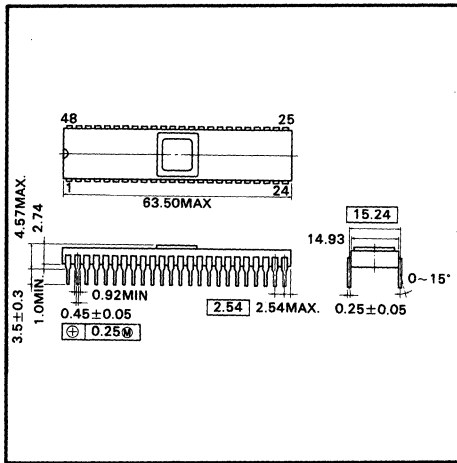


Figure 22. System Configuration with CXQ70116, Word Transfer



Package Outline

Unit: mm



Sony SALES OFFICES:

Sites	Address	Phone	Fax
Corporate & Administration Western Office	23430 Hawthorne Blvd. Suite No. 330 Torrance, CA 90505	213-373-9425	213-375-8358
South East Office	3175 A. Northwoods Pkwy Norcross, GA 30071	404-263-9888	404-446-3615
North East Office	57 Wells Avenue Newton, MA 02159	617-527-4560	617-244-2518
Central Office	3201 Premier Drive Suite 100 Irving, TX 75063	214-550-5200	214-550-5300

AREA SALES OFFICES:

NORTHEAST AREA: Sony Corporation of America (617) 527-4540

SOUTHEAST AREA: Sony Corporation of America (404) 263-9888

CENTRAL AREA: Sony Corporation of America (214) 550-5200

WESTERN AREA: Sony Corporation of America (213) 373-9425

REPRESENTATIVE OFFICES:

ALABAMA: Interep Assoc., Inc., (205) 478-1036, (205) 533-1730

ARIZONA: Shefler-Kahn, (602) 257-9015

ARKANSAS: B.P. Sales, (918) 744-9964

CALIFORNIA: (San Diego and Imperial) Addem Sales, (619) 729-9216; (Southern Ca., excluding San Diego) Varigon, Inc., (714) 855-0233, (Northern Ca. Counties: Monterey/kings/Tulare/Inyo) Brooks Technical Group, (415) 960-3880

COLORADO: Electrodyne, (303) 695-8903

CONNECTICUT: New England Technical Sales, (203) 237-8827

DELAWARE: Beacon North, Inc., (800) 336-3747

DISTRICT OF COLUMBIA: Beacon North, Inc., (703) 478-2480

FLORIDA: Donato & Assoc., (305) 352-0727

GEORGIA: Interep Assoc., Inc., (404) 449-8680

HAWAII: Brooks Technical Group, (415) 960-3880

IDAHO: Electrodyne, (801) 486-3801

ILLINOIS: (Northern) Micro-Tex, Inc., (312) 382-3001; (Southern) Kebco, (314) 576-4111

INDIANA: Giesting & Assoc., (317) 844-5222

IOWA: J.R. Sales, (319) 393-2232

KANSAS: Kebco, (913) 541-8431 or (316) 733-1301

KENTUCKY: Giesting & Assoc., (513) 385-1105

LOUISIANA: B.P. Sales, (214) 234-8438

MAINE: New England Technical Sales, (617) 272-0434

MARYLAND: Beacon North, Inc., (800) 336-3747

MASSACHUSETTS: New England Technical Sales, (617) 272-0434

MICHIGAN: Giesting & Assoc., (313) 478-8106

MINNESOTA: High-Tech Sales, (612) 944-7274

MISSISSIPPI: Interep Assoc., Inc., (205) 533-1730

MISSOURI: Kebco, (314) 576-4111

MONTANA: Electrodyne, (801) 486-3801

NEBRASKA: J.R. Sales, (319) 393-2232

NEVADA: (Clark County) Shefler-Kahn, (602) 257-9015, Brooks Technical Group, (916) 477-8096

NEW HAMPSHIRE: New England Technical Sales, (617) 272-0434

NEW JERSEY: S.J. Assoc., Inc. (718) 291-3232, (609) 866-1234

NEW MEXICO: Shefler-Kahn, (505) 345-3591

NEW YORK: (Manhattan) S.J. Assoc., Inc., (718) 291-3232 (Upstate) Advance Comp. Corp., (315) 699-2671

NORTH CAROLINA: Naylor Assoc., Inc., (919) 544-6630

NORTH DAKOTA: High-Tech Sales, (612) 944-7274

OHIO: Giesting & Assoc., (513) 385-1105, (216) 261-9705, or (513) 433-5832
OKLAHOMA: B. P. Sales, (918) 744-9964
OREGON: Vantage Corp., (503) 620-3280
PENNSYLVANIA: (East) S.J. Assoc., Inc., (609) 866-1234; (West) Giesting & Assoc., (513) 385-1105
RHODE ISLAND: New England Technical Sales, (617) 272-0434
SOUTH CAROLINA: Naylor Assoc., Inc., (919) 544-6630
SOUTH DAKOTA: High-Hech Sales, (612) 944-7274
TENNESSEE: Interep Assoc., Inc., (615) 639-3491
TEXAS: B.P. Sales, (214) 234-8438, (713) 531-4144, or (512) 346-9186; (El Paso County) Shefler-Kahn (505) 345-3591
UTAH: Electrodyne, (801) 486-3801
VERMONT: New England Technical Sales, (617) 272-0434
VIRGINIA: Beacon North, Inc., (703) 478-2480, (804) 239-8486
WASHINGTON: Vantage Corp., (206) 455-3460
WEST VIRGINIA: Giesting & Assoc., (513) 385-1105
WISCONSIN: (Lower) Micro-Tex, Inc., (414) 542-5352; (West Wisconsin Counties: Polk/Barron/Dunn/St. Croix/Pierce/Pepin/Chippewa, Eau Claire) High-Tech Sales, (612) 944-7274
WYOMING: Electrodyne, (801) 486-3801

DISTRIBUTOR OFFICES:

ALABAMA: HuntsvilleHammond Electronics, (205) 830-4764
 HuntsvilleMarshall Industries, (205) 881-9235
ARIZONA: PhoenixMarshall Industries, (602) 968-6181
 ScottsdaleWestern Micro, (602) 948-4240
 TempeInsight, (602) 829-1800
CALIFORNIA: ChatsworthDiplomat Electronics, (818) 700-8700
 IrvineMarshall Industries, (714) 458-5399
 CupertinoWestern Micro, (408) 725-1660
 Los Angeles ...Marshall Industries, (818) 407-0101
 Northridge.....Western Micro, (818) 700-9922, (800) 538-3401
 OrangeRyno Electronics, (714) 637-0200
 SacramentoMarshall Industries, (916) 635-9700
 San DiegoInsight, (619) 587-0471
 San DiegoMarshall Industries, (619) 578-9600
 San DiegoRyno Electronics, (619) 453-8430
 San Francisco...Marshall Industries, (408) 943-4600
 Sunnyvale.....Diplomat Electronics, (408) 737-0204
COLORADO: DenverMarshall Industries, (303) 427-1818
CONNECTICUT: Danbury.....Diplomat Electronics, (203) 797-9674
 WallingfordMarshall Industries, (203) 265-3822
FLORIDA: Clearwater.....Diplomat Electronics, (813) 443-4514
 Ft. Lauderdale ...Diplomat Electronics, (305) 974-8700
 Ft. Lauderdale ...Hammond Electronics, (305) 973-7103
 Ft. Lauderdale ...Marshall Industries, (305) 928-0661
 Ft. Lauderdale ...Reptron, (305) 979-8227
 OrlandoHammond Electronics, (305) 841-1010
 OrlandoMarshall Industries, (305) 841-1878
 TampaMarshall Industries, (813) 576-1399
 TampaReptron, (813) 855-4656
GEORGIA: AtlantaMarshall Industries, (404) 923-5750
 Norcross ...Hammond Electronics, (404) 449-1996
 Norcross ...Reptron, (404) 446-1300
ILLINOIS: ChicagoMarshall Industries, (312) 490-0155
 Schaumburg...Reptron, (312) 882-1700
INDIANA: Indianapolis...Marshall Industries, (317) 297-0483
KANSAS: Kansas City...Marshall Industries, (913) 492-3121

MARYLAND: BaltimoreHammond Electronics, (301) 583-2525
 ColumbiaDiplomat Electronics, (301) 995-1226
 ColumbiaVantage Components, (301) 720-5100
 Gaithersburg.....Marshall Industries, (301) 840-9450
MASSACHUSETTS: BostonMarshall Industries, (617) 272-8200
 Burlington.....Western Micro, (617) 229-5819
 BillericaDiplomat Electronics, (617) 667-4670
MICHIGAN: Livonia...Marshall Industries, (313) 525-5850
 Livonia...Reptron, (313) 525-2700
MINNESOTA: Minneapolis ...Marshall Industries, (612) 559-2211
 Minnetonka ...Reptron, (612) 938-0000
 PlymouthDiplomat Electronics, (612) 559-2500
NEW JERSEY: CliftonVantage Components, (201) 777-4100
 FairfieldMarshall Industries, (201) 882-0320
 TotowaDiplomat Electronics, (201) 785-1830
NEW YORK: BinghamtonMarshall Industries, (607) 798-1611
 CommackVantage Components, (516) 543-2000
 HenriettaDiplomat Electronics, (716) 359-4400
 LiverpoolDiplomat Electronics, (315) 652-5000
 Long Island.....Marshall Industries, (516) 273-2424
 MelvilleDiplomat Electronics, (516) 454-6400
 RochesterMarshall Industries, (716) 235-7620
NORTH CAROLINA: Greensboro ...Hammond Electronics, (919) 275-6391
 RaleighMarshall Industries, (919) 878-9882
OHIO: ClevelandMarshall Industries, (216) 248-1788
 DaytonMarshall Industries, (513) 236-8088
 Worthington...Reptron, (614) 436-6675
OREGON: Beaverton...Western Micro, (503) 629-2082
 Portland.....Marshall Industries, (503) 644-5050
PENNSYLVANIA: PhiladelphiaMarshall Industries, (609) 234-9100
 Pittsburgh.....Marshall Industries, (412) 963-0441
TEXAS: Austin.....Diplomat Electronics, (512) 836-8707
 Austin.....Marshall Industries, (512) 837-1991
 Dallas.....Diplomat Electronics, (214) 980-1888
 Dallas.....Marshall Industries, (214) 233-5200
 Houston.....Marshall Industries, (713) 895-9200
UTAH: Salt Lake City...Diplomat Electronics, (801) 486-4134
 Salt Lake City...Marshall Industries, (801) 261-0901
WASHINGTON: Redmond ...Western Micro (206) 881-6737
 SeattleMarshall Industries, (206) 747-9100
WISCONSIN: Milwaukee...Marshall Industries, (414) 797-8400

Sony Semiconductor Integrated Circuit Data Book

1986. Sept. 1st Edition

Printed in USA © Copyright 1986, Sony Corporation of America

SONY®

Sony Semiconductor

