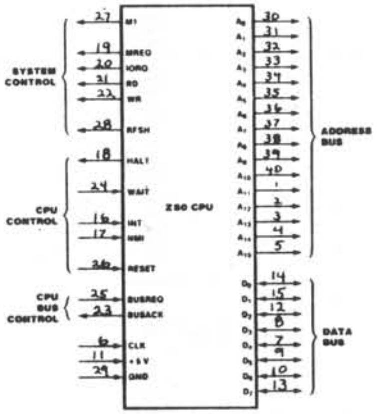
Zilog

Z8400
Z80® CPU Central Processing Unit

Features



Pin Descriptions

A_0-A_{15}. *Address Bus* (output, active High, 3-state). A_0-A_{15} form a 16-bit address bus. The Address Bus provides the address for memory data bus exchanges (up to 64K bytes) and for I/O device exchanges.

BUSACK. *Bus Acknowledge* (output, active Low). Bus Acknowledge indicates to the requesting device that the CPU address bus, data bus, and control signals MREQ, IORQ, RD, and WR have entered their high-impedance states. The external circuitry can now control these lines.

BUSREQ. *Bus Request* (input, active Low). Bus Request has a higher priority than NMI and is always recognized at the end of the current machine cycle. BUSREQ forces the CPU address bus, data bus, and control signals MREQ, IORQ, RD, and WR to go to a high-impedance state so that other devices can control these lines. BUSREQ is normally wire-ORed and requires an external pullup for these applications. Extended BUSREQ periods due to extensive DMA operations can prevent the CPU from properly refreshing dynamic RAMs.

D_0-D_7. *Data Bus* (input/output, active High, 3-state). D_0-D_7 constitute an 8-bit bidirectional data bus, used for data exchanges with memory and I/O.

HALT. *Halt State* (output, active Low). HALT indicates that the CPU has executed a Halt instruction and is awaiting either a non-maskable or a maskable interrupt (with the mask enabled) before operation can resume. While halted, the CPU executes NOPs to maintain memory refresh.

INT. *Interrupt Request* (input, active Low). Interrupt Request is generated by I/O devices. The CPU honors a request at the end of the current instruction if the internal software-controlled interrupt enable flip-flop (IFF) is enabled. INT is normally wire-ORed and requires an external pullup for these applications.

IORQ. *Input/Output Request* (output, active Low, 3-state). IORQ indicates that the lower half of the address bus holds a valid I/O address for an I/O read or write operation. IORQ is also generated concurrently with M1 during an interrupt acknowledge cycle to indicate that an interrupt response vector can be placed on the data bus.

M1. *Machine Cycle One* (output, active Low). M1, together with MREQ, indicates that the current machine cycle is the opcode fetch cycle of an instruction execution. M1, together with IORQ, indicates an interrupt acknowledge cycle.

MREQ. *Memory Request* (output, active Low, 3-state). MREQ indicates that the address bus holds a valid address for a memory read or memory write operation.

NMI. *Non-Maskable Interrupt* (input, active Low). NMI has a higher priority than INT. NMI is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop, and automatically forces the CPU to restart at location 0066H.

RD. *Memory Read* (output, active Low, 3-state). RD indicates that the CPU wants to read data from memory or an I/O device. The addressed I/O device or memory should use this signal to gate data onto the CPU data bus.

RESET. *Reset* (input, active Low). RESET initializes the CPU as follows: it resets the interrupt enable flip-flop, clears the PC and Registers I and R, and sets the interrupt status to Mode 0. During reset time, the address and data bus go to a high-impedance state, and all control output signals go to the inactive state. Note that RESET must be active for a minimum of three full clock cycles before the reset operation is complete.

RFSH. *Refresh* (output, active Low). RFSH, together with MREQ, indicates that the lower seven bits of the system's address bus can be used as a refresh address to the system's dynamic memories.

WAIT. *Wait* (input, active Low). WAIT indicates to the CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter a Wait state as long as this signal is active. Extended WAIT periods can prevent the CPU from refreshing dynamic memory properly.

WR. *Memory Write* (output, active Low, 3-state). WR indicates that the CPU data bus holds valid data to be stored at the addressed memory or I/O location.

Instruction Set

The Z80 microprocessor has one of the most powerful and versatile instruction sets available in any 8-bit microprocessor. It includes such unique operations as a block move for fast, efficient data transfers within memory or between memory and I/O. It also allows operations on any bit in any location in memory.

The following is a summary of the Z80 instruction set and shows the assembly language mnemonic, the operation, the flag status, and gives comments on each instruction. The *Z80 CPU Technical Manual* (03-0029-01) and *Assembly Language Programming Manual* (03-0002-01) contain significantly more details for programming use.

The instructions are divided into the following categories:

- □ 8-bit loads
- □ 16-bit loads
- □ Exchanges, block transfers, and searches
- □ 8-bit arithmetic and logic operations
- □ General-purpose arithmetic and CPU control
- □ 16-bit arithmetic operations
- □ Rotates and shifts
- □ Bit set, reset, and test operations
- □ Jumps
- □ Calls, returns, and restarts
- □ Input and output operations

A variety of addressing modes are implemented to permit efficient and fast data transfer between various registers, memory locations, and input/output devices. These addressing modes include:

- □ Immediate
- □ Immediate extended
- □ Modified page zero
- □ Relative
- □ Extended
- □ Indexed
- □ Register
- □ Register indirect
- □ Implied
- □ Bit

8-Bit Load Group

Mnemonic	Symbolic Operation	S	Z		H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
LD r, r'	r ← r'	•	•	X	•	X	•	•	•	01 r r'		1	1	4	r, r' Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
LD r, n	r ← n	•	•	X	•	X	•	•	•	00 r 110 ← n →		2	2	7	
LD r, (HL)	r ← (HL)	•	•	X	•	X	•	•	•	01 r 110		1	2	7	
LD r, (IX+d)	r ← (IX+d)	•	•	X	•	X	•	•	•	11 011 101 01 r 101 ← d →	DD	3	5	19	
LD r, (IY+d)	r ← (IY+d)	•	•	X	•	X	•	•	•	11 111 101 01 r 110 ← d →	FD	3	5	19	
LD (HL), r	(HL) ← r	•	•	X	•	X	•	•	•	01 110 r		1	2	7	
LD (IX+d), r	(IX+d) ← r	•	•	X	•	X	•	•	•	11 011 101 01 110 r ← d →	DD	3	5	19	
LD (IY+d), r	(IY+d) ← r	•	•	X	•	X	•	•	•	11 111 101 01 110 r ← d →	FD	3	5	19	
LD (HL), n	(HL) ← n	•	•	X	•	X	•	•	•	00 110 110 ← n →	36	2	3	10	
LD (IX+d), n	(IX+d) ← n	•	•	X	•	X	•	•	•	11 011 101 00 110 110 ← d → ← n →	DD 36	4	5	19	
LD (IY+d), n	(IY+d) ← n	•	•	X	•	X	•	•	•	11 111 101 00 110 110 ← d → ← n →	FD 36	4	5	19	
LD A, (BC)	A ← (BC)	•	•	X	•	X	•	•	•	00 001 010	0A	1	2	7	
LD A, (DE)	A ← (DE)	•	•	X	•	X	•	•	•	00 011 010	1A	1	2	7	
LD A, (nn)	A ← (nn)	•	•	X	•	X	•	•	•	00 111 010 ← n → ← n →	3A	3	4	13	
LD (BC), A	(BC) ← A	•	•	X	•	X	•	•	•	00 000 010	02	1	2	7	
LD (DE), A	(DE) ← A	•	•	X	•	X	•	•	•	00 010 010	12	1	2	7	
LD (nn), A	(nn) ← A	•	•	X	•	X	•	•	•	00 110 010 ← n → ← n →	32	3	4	13	
LD A, I	A ← I	↕	↕	X	0	X	IFF	0	•	11 101 101 01 010 111	ED 57	2	2	9	
LD A, R	A ← R	↕	↕	X	0	X	IFF	0	•	11 101 101 01 011 111	ED 5F	2	2	9	
LD I, A	I ← A	•	•	X	•	X	•	•	•	11 101 101 01 000 111	ED 47	2	2	9	
LD R, A	R ← A	•	•	X	•	X	•	•	•	11 101 101 01 001 111	ED 4F	2	2	9	

NOTES: r, r' means any of the registers A, B, C, D, E, H, L.
IFF the content of the interrupt enable flip-flop (IFF) is copied into the P/V flag.

For an explanation of flag notation and symbols for mnemonic tables, see Symbolic Notation section following tables.

"Z80 is a trademark of Zilog, Inc.
with whom the publisher is not associated."

16-Bit Load Group

	Symbolic				Flags					Opcode		No. of	No. of M	No. of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
LD dd, nn	dd ← nn	•	•	X	•	X	•	•	•	00 dd0 001 ← n → ← n →		3	3	10	dd Pair 00 BC 01 DE 10 HL 11 SP
LD IX, nn	IX ← nn	•	•	X	•	X	•	•	•	11 011 101 00 100 001 ← n → ← n →	DD 21	4	4	14	
LD IY, nn	IY ← nn	•	•	X	•	X	•	•	•	11 111 101 00 100 001 ← n → ← n →	FD 21	4	4	14	
LD HL, (nn)	H ← (nn + 1) L ← (nn)	•	•	X	•	X	•	•	•	00 101 010 ← n → ← n →	2A	3	5	16	
LD dd, (nn)	dd_H ← (nn + 1) dd_L ← (nn)	•	•	X	•	X	•	•	•	11 101 101 01 dd1 011 ← n → ← n →	ED	4	6	20	
LD IX, (nn)	IX_H ← (nn + 1) IX_L ← (nn)	•	•	X	•	X	•	•	•	11 011 101 00 101 010 ← n → ← n →	DD 2A	4	6	20	
LD IY, (nn)	IY_H ← (nn + 1) IY_L ← (nn)	•	•	X	•	X	•	•	•	11 111 101 00 101 010 ← n → ← n →	FD 2A	4	6	20	
LD (nn), HL	(nn + 1) ← H (nn) ← L	•	•	X	•	X	•	•	•	00 100 010 ← n → ← n →	22	3	5	16	
LD (nn), dd	(nn + 1) ← dd_H (nn) ← dd_L	•	•	X	•	X	•	•	•	11 101 101 01 dd0 011 ← n → ← n →	ED	4	6	20	
LD (nn), IX	(nn + 1) ← IX_H (nn) ← IX_L	•	•	X	•	X	•	•	•	11 011 101 00 100 010 ← n → ← n →	DD 22	4	6	20	
LD (nn), IY	(nn + 1) ← IY_H (nn) ← IY_L	•	•	X	•	X	•	•	•	11 111 101 00 100 010 ← n → ← n →	FD 22	4	6	20	
LD SP, HL	SP ← HL	•	•	X	•	X	•	•	•	11 111 001	F9	1	1	6	
LD SP, IX	SP ← IX	•	•	X	•	X	•	•	•	11 011 101 11 111 001	DD F9	2	2	10	
LD SP, IY	SP ← IY	•	•	X	•	X	•	•	•	11 111 101 11 111 001	FD F9	2	2	10	
PUSH qq	(SP - 2) ← qq_L (SP - 1) ← qq_H SP ← SP - 2	•	•	X	•	X	•	•	•	11 qq0 101		1	3	11	qq Pair 00 BC 01 DE 10 HL 11 AF
PUSH IX	(SP - 2) ← IX_L (SP - 1) ← IX_H SP ← SP - 2	•	•	X	•	X	•	•	•	11 011 101 11 100 101	DD E5	2	4	15	
PUSH IY	(SP - 2) ← IY_L (SP - 1) ← IY_H SP ← SP - 2	•	•	X	•	X	•	•	•	11 111 101 11 100 101	FD E5	2	4	15	
POP qq	qq_H ← (SP + 1) qq_L ← (SP) SP ← SP + 2	•	•	X	•	X	•	•	•	11 qq0 001		1	3	10	
POP IX	IX_H ← (SP + 1) IX_L ← (SP) SP ← SP + 2	•	•	X	•	X	•	•	•	11 011 101 11 100 001	DD E1	2	4	14	
POP IY	IY_H ← (SP + 1) IY_L ← (SP) SP ← SP + 2	•	•	X	•	X	•	•	•	11 111 101 11 100 001	FD E1	2	4	14	

NOTES dd is any of the register pairs BC, DE, HL, SP
qq is any of the register pairs AF, BC, DE, HL
$(PAIR)_H$, $(PAIR)_L$ refer to high order and low order eight bits of the register pair respectively.
e.g., BC_L = C, AF_H = A

Exchange, Block Transfer, Block Search Groups

	Symbolic				Flags					Opcode		No. of	No. of M	No. of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
EX DE, HL	DE ↔ HL	•	•	X	•	X	•	•	•	11 101 011	EB	1	1	4	
EX AF, AF'	AF ↔ AF'	•	•	X	•	X	•	•	•	00 001 000	08	1	1	4	
EXX	BC ↔ BC' DE ↔ DE' HL ↔ HL'	•	•	X	•	X	•	•	•	11 011 001	D9	1	1	4	Register bank and auxiliary register bank exchange
EX (SP), HL	H ↔ (SP + 1) L ↔ (SP)	•	•	X	•	X	•	•	•	11 100 011	E3	1	5	19	
EX (SP), IX	IX_H ↔ (SP + 1) IX_L ↔ (SP)	•	•	X	•	X	•	•	•	11 011 101 11 100 011	DD E3	2	6	23	
EX (SP), IY	IY_H ↔ (SP + 1) IY_L ↔ (SP)	•	•	X	•	X	•	•	•	11 111 101 11 100 011	FD E3	2	6	23	
LDI	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC - 1	•	•	X	0	X	① ↕	0	•	11 101 101 10 100 000	ED A0	2	4	16	Load (HL) into (DE), increment the pointers and decrement the byte counter (BC)
LDIR	(DE) ← (HL) DE ← DE + 1 HL ← HL + 1 BC ← BC - 1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 10 110 000	ED B0	2 2	5 4	21 16	If BC ≠ 0 If BC = 0

NOTE: ① P/V flag is 0 if the result of BC - 1 = 0, otherwise P/V = 1

Exchange, Block Transfer, Block Search Groups (Continued)

	Symbolic				Flags					Opcode		No. of	No. of M	No. of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
LDD	(DE) ← (HL) DE ← DE - 1 HL ← HL - 1 BC ← BC - 1	•	•	X	0	X	① ↕	0	•	11 101 101 10 101 000	ED A8	2	4	16	
LDDR	(DE) ← (HL) DE ← DE - 1 HL ← HL - 1 BC ← BC - 1 Repeat until BC = 0	•	•	X	0	X	0	0	•	11 101 101 10 111 000	ED B8	2 2	5 4	21 16	If BC ≠ 0 If BC = 0
CPI	A - (HL) HL ← HL + 1 BC ← BC - 1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 100 001	ED A1	2	4	16	
CPIR	A - (HL) HL ← HL + 1 BC ← BC - 1 Repeat until A = (HL) or BC = 0	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 110 001	ED B1	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)
CPD	A - (HL) HL ← HL - 1 BC ← BC - 1	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 101 001	ED A9	2	4	16	
CPDR	A - (HL) HL ← HL - 1 BC ← BC - 1 Repeat until A = (HL) or BC = 0	↕	② ↕	X	↕	X	① ↕	1	•	11 101 101 10 111 001	ED B9	2 2	5 4	21 16	If BC ≠ 0 and A ≠ (HL) If BC = 0 or A = (HL)

NOTES: ① P/V flag is 0 if the result of BC - 1 = 0, otherwise P/V = 1
② Z flag is 1 if A = (HL), otherwise Z = 0.

8-Bit Arithmetic and Logical Group

	Symbolic				Flags					Opcode		No. of	No. of M	No. of T	
Mnemonic	Operation	S	Z		H		P/V	N	C	76 543 210	Hex	Bytes	Cycles	States	Comments
ADD A, r	A ← A + r	↕	↕	X	↕	X	V	0	↕	10 **000** r		1	1	4	r Reg. 000 B 001 C 010 D 011 E 100 H 101 L 111 A
ADD A, n	A ← A + n	↕	↕	X	↕	X	V	0	↕	11 **000** 110 ← n →		2	2	7	
ADD A, (HL)	A ← A + (HL)	↕	↕	X	↕	X	V	0	↕	10 **000** 110		1	2	7	
ADD A, (IX + d)	A ← A + (IX + d)	↕	↕	X	↕	X	V	0	↕	11 011 101 10 **000** 110 ← d →	DD	3	5	19	
ADD A, (IY + d)	A ← A + (IY + d)	↕	↕	X	↕	X	V	0	↕	11 111 101 10 **000** 110 ← d →	FD	3	5	19	
ADC A, s	A ← A + s + CY	↕	↕	X	↕	X	V	0	↕	**001**					s is any of r, n, (HL), (IX + d), (IY + d) as shown for ADD instruction. The indicated bits replace the **000** in the ADD set above.
SUB s	A ← A - s	↕	↕	X	↕	X	V	1	↕	**010**					
SBC A, s	A ← A - s - CY	↕	↕	X	↕	X	V	1	↕	**011**					
AND s	A ← A ∧ s	↕	↕	X	1	X	P	0	0	**100**					
OR s	A ← A ∨ s	↕	↕	X	0	X	P	0	0	**110**					
XOR s	A ← A ⊕ s	↕	↕	X	0	X	P	0	0	**101**					
CP s	A - s	↕	↕	X	↕	X	V	1	↕	**111**					
INC r	r ← r + 1	↕	↕	X	↕	X	V	0	•	00 r **100**		1	1	4	
INC (HL)	(HL) ← (HL) + 1	↕	↕	X	↕	X	V	0	•	00 110 **100**		1	3	11	
INC (IX + d)	(IX + d) ← (IX + d) + 1	↕	↕	X	↕	X	V	0	•	11 011 101 00 110 **100** ← d →	DD	3	6	23	
INC (IY + d)	(IY + d) ← (IY + d) + 1	↕	↕	X	↕	X	V	0	•	11 111 101 00 110 **100** ← d →	FD	3	6	23	
DEC m	m ← m - 1	↕	↕	X	↕	X	V	1	•	**101**					m is any of r, (HL), (IX + d), (IY + d) as shown for INC. DEC same format and states as INC. Replace **100** with **101** in opcode.

General-Purpose Arithmetic and CPU Control Groups

Mnemonic	Symbolic Operation	Flags								Opcode		No. of Bytes	No. of M Cycles	No. of T States	Comments
		S	Z		H		P/V	N	C	76 543 210	Hex				
DAA	Converts acc. content into packed BCD following add or subtract with packed BCD operands	↕	↕	X	↕	X	P	•	↕	00 100 111	27	1	1	4	Decimal adjust accumulator
CPL	A ← $\overline{A}$	•	•	X	1	X	•	1	•	00 101 111	2F	1	1	4	Complement accumulator (one's complement)
NEG	A ← 0 – A	↕	↕	X	↕	X	V	1	↕	11 101 101 01 000 100	ED 44	2	2	8	Negate acc. (two's complement)
CCF	CY ← $\overline{CY}$	•	•	X	X	X	•	0	↕	00 111 111	3F	1	1	4	Complement carry flag
SCF	CY ← 1	•	•	X	0	X	•	0	1	00 110 111	37	1	1	4	Set carry flag
NOP	No operation	•	•	X	•	X	•	•	•	00 000 000	00	1	1	4	
HALT	CPU halted	•	•	X	•	X	•	•	•	01 110 110	76	1	1	4	
DI *	IFF ← 0	•	•	X	•	X	•	•	•	11 110 011	F3	1	1	4	
EI *	IFF ← 1	•	•	X	•	X	•	•	•	11 111 011	FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	X	•	X	•	•	•	11 101 101 01 000 110	ED 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	X	•	X	•	•	•	11 101 101 01 010 110	ED 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	X	•	X	•	•	•	11 101 101 01 011 110	ED 5E	2	2	8	

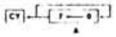
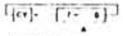
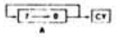
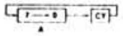
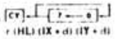
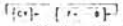
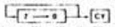
NOTES: IFF indicates the interrupt enable flip-flop
CY indicates the carry flip-flop
* indicates interrupts are not sampled at the end of EI or DI

16-Bit Arithmetic Group

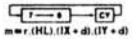
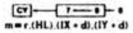
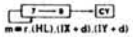
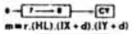
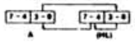
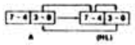
ADD HL, ss	HL ← HL + ss	•	•	X	X	X	•	0	↕	00 ss1 001		1	3	11	ss Reg 00 BC 01 DE 10 HL 11 SP
ADC HL, ss	HL ← HL + ss + CY	↕	↕	X	X	X	V	0	↕	11 101 101 01 ss1 010	ED	2	4	15	
SBC HL, ss	HL ← HL – ss – CY	↕	↕	X	X	X	V	1	↕	11 101 101 01 ss0 010	ED	2	4	15	
ADD IX, pp	IX ← IX + pp	•	•	X	X	X	•	0	↕	11 011 101 01 pp1 001	DD	2	4	15	pp Reg 00 BC 01 DE 10 IX 11 SP
ADD IY, rr	IY ← IY + rr	•	•	X	X	X	•	0	↕	11 111 101 00 rr1 001	FD	2	4	15	rr Reg 00 BC 01 DE 10 IY 11 SP
INC ss	ss ← ss + 1	•	•	X	•	X	•	•	•	00 ss0 011		1	1	6	
INC IX	IX ← IX + 1	•	•	X	•	X	•	•	•	11 011 101 00 100 011	DD 23	2	2	10	
INC IY	IY ← IY + 1	•	•	X	•	X	•	•	•	11 111 101 00 100 011	FD 23	2	2	10	
DEC ss	ss ← ss – 1	•	•	X	•	X	•	•	•	00 ss1 011		1	1	6	
DEC IX	IX ← IX – 1	•	•	X	•	X	•	•	•	11 011 101 00 101 011	DD 2B	2	2	10	
DEC IY	IY ← IY – 1	•	•	X	•	X	•	•	•	11 111 101 00 101 011	FD 2B	2	2	10	

NOTES: ss is any of the register pairs BC, DE, HL, SP
pp is any of the register pairs BC, DE, IX, SP
rr is any of the register pairs BC, DE, IY, SP

Rotate and Shift Group

RLCA		•	•	X	0	X	•	0	↕	00 000 111	07	1	1	4	Rotate left circular accumulator
RLA		•	•	X	0	X	•	0	↕	00 010 111	17	1	1	4	Rotate left accumulator
RRCA		•	•	X	0	X	•	0	↕	00 001 111	0F	1	1	4	Rotate right circular accumulator
RRA		•	•	X	0	X	•	0	↕	00 011 111	1F	1	1	4	Rotate right accumulator
RLC r		↕	↕	X	0	X	P	0	↕	11 001 011 00 000 r	CB	2	2	8	Rotate left circular register r
RLC (HL)		↕	↕	X	0	X	P	0	↕	11 001 011 00 000 110	CB	2	4	15	r Reg 000 B 001 C 010 D 011 E 100 H 101 L 111 A
RLC (IX + d)	 r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	11 011 101 11 001 011 – d – 00 000 110	DD CB	4	6	23	
RLC (IY + d)		↕	↕	X	0	X	P	0	↕	11 111 101 11 001 011 – d – 00 000 110	FD CB	4	6	23	
RL m	 m ≡ r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{010}$					Instruction format and states are as shown for RLC's. To form new opcode replace 000 or RLC's with shown code
RR m	 m ≡ r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{001}$					

Rotate and Shift Group (Continued)

Mnemonic	Symbolic Operation	Flags								Opcode		No. of Bytes	No. of M Cycles	No. of T States	Comments
		S	Z		H		P/V	N	C	76 543 210	Hex				
RR m	 m ≡ r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{011}$					
SLA m	 m ≡ r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{100}$					
SRA m	 m ≡ r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{101}$					
SRL m	 m ≡ r, (HL), (IX + d), (IY + d)	↕	↕	X	0	X	P	0	↕	$\boxed{111}$					
RLD		↕	↕	X	0	X	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	Rotate digit left and right between the accumulator and location (HL)
RRD		↕	↕	X	0	X	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	The content of the upper half of the accumulator is unaffected

Bit Set, Reset and Test Group

BIT b, r	Z ← $\overline{r}_b$	X	↕	X	1	X	X	0	•	11 001 011 01 b r	CB	2	2	8	r Reg 000 B 001 C 010 D 011 E 100 H 101 L 111 A
BIT b, (HL)	Z ← $\overline{(HL)}_b$	X	↕	X	1	X	X	0	•	11 001 011 01 b 110	CB	2	3	12	
BIT b, (IX + d)$_b$	Z ← $\overline{(IX+d)}_b$	X	↕	X	1	X	X	0	•	11 011 101 11 001 011 – d – 01 b 110	DD CB	4	5	20	
BIT b, (IY + d)$_b$	Z ← $\overline{(IY+d)}_b$	X	↕	X	1	X	X	0	•	11 111 101 11 001 011 – d – 01 b 110	FD CB	4	5	20	b Bit Tested 000 0 001 1 010 2 011 3 100 4 101 5 110 6 111 7
SET b, r	r_b ← 1	•	•	X	•	X	•	•	•	11 001 011 $\boxed{11}$ b r	CB	2	2	8	
SET b, (HL)	$(HL)_b$ ← 1	•	•	X	•	X	•	•	•	11 001 011 $\boxed{11}$ b 110	CB	2	4	15	
SET b, (IX + d)	$(IX+d)_b$ ← 1	•	•	X	•	X	•	•	•	11 011 101 11 001 011 – d – $\boxed{11}$ b 110	DD CB	4	6	23	
SET b, (IY + d)	$(IY+d)_b$ ← 1	•	•	X	•	X	•	•	•	11 111 101 11 001 011 – d – $\boxed{11}$ b 110	FD CB	4	6	23	
RES b, m	m_b ← 0 m ≡ r, (HL), (IX + d), (IY + d)	•	•	X	•	X	•	•	•	$\boxed{10}$					To form new opcode replace $\boxed{11}$ of SET b, s with $\boxed{10}$. Flags and time states for SET instruction.

NOTES: The notation m_b indicates bit b (0 to 7) or location m

Jump Group

JP nn	PC ← nn	•	•	X	•	X	•	•	•	11 000 011 – n – – n –	C3	3	3	10	
JP cc, nn	If condition cc is true PC ← nn, otherwise continue	•	•	X	•	X	•	•	•	11 cc 010 – n – – n –		3	3	10	cc Condition 000 NZ non-zero 001 Z zero 010 NC non-carry 011 C carry 100 PO parity odd 101 PE parity even 110 P sign positive 111 M sign negative
JR e	PC ← PC + e	•	•	X	•	X	•	•	•	00 011 000 – e–2 –	18	2	3	12	
JR C, e	If C = 0, continue	•	•	X	•	X	•	•	•	00 111 000 – e–2 –	38	2	2	7	If condition not met.
	If C = 1, PC ← PC + e											2	3	12	If condition is met.
JR NC, e	If C = 1, continue	•	•	X	•	X	•	•	•	00 110 000 – e–2 –	30	2	2	7	If condition not met.
	If C = 0, PC ← PC + e											2	3	12	If condition is met.
JP Z, e	If Z = 0 continue	•	•	X	•	X	•	•	•	00 101 000 – e–2 –	28	2	2	7	If condition not met.
	If Z = 1, PC ← PC + e											2	3	12	If condition is met.
JR NZ, e	If Z = 1, continue	•	•	X	•	X	•	•	•	00 100 000 – e–2 –	20	2	2	7	If condition not met.
	If Z = 0, PC ← PC + e											2	3	12	If condition is met.
JP (HL)	PC ← HL	•	•	X	•	X	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC ← IX	•	•	X	•	X	•	•	•	11 011 101 11 101 001	DD E9	2	2	8	

Jump Group (Continued)

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
JP (IY)	$PC \leftarrow IY$	•	•	X	•	X	•	•	•	11 111 101 11 101 001	FD E9	2	2	8	
DJNZ, e	$B \leftarrow B - 1$ If B = 0, continue	•	•	X	•	X	•	•	•	00 010 000 ← e−2 →	10	2	2	8	If B = 0
	If B ≠ 0, $PC \leftarrow PC + e$											2	3	13	If B ≠ 0

NOTES e represents the extension in the relative addressing mode
e is a signed two's complement number in the range < −126, 129 >
e − 2 in the opcode provides an effective address of pc + e as PC is incremented by 2 prior to the addition of e

Call and Return Group

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
CALL nn	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ $PC \leftarrow nn$	•	•	X	•	X	•	•	•	11 001 101 ← n → ← n →	CD	3	5	17	
CALL cc, nn	If condition cc is false continue, otherwise same as CALL nn	•	•	X	•	X	•	•	•	11 cc 100 ← n → ← n →		3	3	10	If cc is false
												3	5	17	If cc is true
RET	$PC_L \leftarrow (SP)$ $PC_H \leftarrow (SP+1)$	•	•	X	•	X	•	•	•	11 001 001	C9	1	3	10	
RET cc	If condition cc is false continue, otherwise same as RET	•	•	X	•	X	•	•	•	11 cc 000		1	1	5	If cc is false
												1	3	11	If cc is true
RETI	Return from interrupt	•	•	X	•	X	•	•	•	11 101 101 01 001 101	ED 4D	2	4	14	
RETN[1]	Return from non-maskable interrupt	•	•	X	•	X	•	•	•	11 101 101 01 000 101	ED 45	2	4	14	
RST p	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$ $PC_H \leftarrow 0$ $PC_L \leftarrow p$	•	•	X	•	X	•	•	•	11 t 111		1	3	11	

cc		Condition
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign positive
111	M	sign negative

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

NOTE [1]RETN loads $IFF_2 \leftarrow IFF_1$

Input and Output Group

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
IN A, (n)	$A \leftarrow (n)$	•	•	X	•	X	•	•	•	11 011 011 ← n →	DB	2	3	11	n to $A_0 - A_7$ Acc to $A_8 - A_{15}$
IN r, (C)	$r \leftarrow (C)$ if r = 110 only the flags will be affected	↕	↕	X	↕	X	P	0	•	11 101 101 01 r 000	ED	2	3	12	C to $A_0 - A_7$ B to $A_8 - A_{15}$
INI	$(HL) \leftarrow (C)$ $B \leftarrow B - 1$ $HL \leftarrow HL + 1$	X	① ↕	X	X	X	X	1	•	11 101 101 10 100 010	ED A2	2	4	16	C to $A_0 - A_7$ B to $A_8 - A_{15}$
INIR	$(HL) \leftarrow (C)$ $B \leftarrow B - 1$ $HL \leftarrow HL + 1$ Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 110 010	ED B2	2	5 (If B ≠ 0)	21	C to $A_0 - A_7$ B to $A_8 - A_{15}$
												2	4 (If R = 0)	16	
IND	$(HL) \leftarrow (C)$ $B \leftarrow B - 1$ $HL \leftarrow HL - 1$	X	① ↕	X	X	X	X	1	•	11 101 101 10 101 010	ED AA	2	4	16	C to $A_0 - A_7$ B to $A_8 - A_{15}$
INDR	$(HL) \leftarrow (C)$ $B \leftarrow B - 1$ $HL \leftarrow HL - 1$ Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 111 010	ED BA	2	5 (If B ≠ 0)	21	C to $A_0 - A_7$ B to $A_8 - A_{15}$
												2	4 (If B = 0)	16	
OUT (n), A	$(n) \leftarrow A$	•	•	X	•	X	•	•	•	11 010 011 ← n →	D3	2	3	11	n to $A_0 - A_7$ Acc. to $A_8 - A_{15}$
OUT (C), r	$(C) \leftarrow r$	•	•	X	•	X	•	•	•	11 101 101 01 r 001	ED	2	3	12	C to $A_0 - A_7$ B to $A_8 - A_{15}$
OUTI	$(C) \leftarrow (HL)$ $B \leftarrow B - 1$ $HL \leftarrow HL + 1$	X	① ↕	X	X	X	X	1	•	11 101 101 10 100 011	ED A3	2	4	16	C to $A_0 - A_7$ B to $A_8 - A_{15}$
OTIR	$(C) \leftarrow (HL)$ $B \leftarrow B - 1$ $HL \leftarrow HL + 1$ Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 110 011	ED B3	2	5 (If B ≠ 0)	21	C to $A_0 - A_7$ B to $A_8 - A_{15}$
												2	4 (If B = 0)	16	
OUTD	$(C) \leftarrow (HL)$ $B \leftarrow B - 1$ $HL \leftarrow HL - 1$	X	① ↕	X	X	X	X	1	•	11 101 101 10 101 011	ED AB	2	4	16	C to $A_0 - A_7$ B to $A_8 - A_{15}$

NOTE ① If the result of B − 1 is zero the Z flag is set, otherwise it is reset

Input and Output Group (Continued)

Mnemonic	Symbolic Operation	S	Z		Flags H		P/V	N	C	Opcode 76 543 210	Hex	No. of Bytes	No. of M Cycles	No. of T States	Comments
OTDR	$(C) \leftarrow (HL)$ $B \leftarrow B - 1$ $HL \leftarrow HL - 1$ Repeat until B = 0	X	1	X	X	X	X	1	•	11 101 101 10 111 011	ED	2	5 (If B ≠ 0)	21	C to $A_0 - A_7$ B to $A_8 - A_{15}$
												2	4 (If B = 0)	16	

Summary of Flag Operation

Instruction	D_7 S	Z		H		P/V	N	D_0 C	Comments
ADD A, s; ADC A, s	↕	↕	X	↕	X	V	0	↕	8-bit add or add with carry.
SUB s; SBC A, s; CP s; NEG	↕	↕	X	↕	X	V	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator.
AND s	↕	↕	X	1	X	P	0	0	Logical operations.
OR s, XOR s	↕	↕	X	0	X	P	0	0	
INC s	↕	↕	X	↕	X	V	0	•	8-bit increment.
DEC s	↕	↕	X	↕	X	V	1	•	8-bit decrement.
ADD DD, ss	•	•	X	X	X	•	0	↕	16-bit add.
ADC HL, ss	↕	↕	X	X	X	V	0	↕	16-bit add with carry.
SBC HL, ss	↕	↕	X	X	X	V	1	↕	16-bit subtract with carry.
RLA, RLCA, RRA, RRCA	•	•	X	0	X	•	0	↕	Rotate accumulator.
RL m; RLC m, RR m; RRC m, SLA m; SRA m, SRL m	↕	↕	X	0	X	P	0	↕	Rotate and shift locations.
RLD; RRD	↕	↕	X	0	X	P	0	•	Rotate digit left and right.
DAA	↕	↕	X	↕	X	P	•	↕	Decimal adjust accumulator.
CPL	•	•	X	1	X	•	1	•	Complement accumulator.
SCF	•	•	X	0	X	•	0	1	Set carry.
CCF	•	•	X	X	X	•	0	↕	Complement carry.
IN r (C)	↕	↕	X	0	X	P	0	•	Input register indirect.
INI, IND, OUTI; OUTD	X	↕	X	X	X	X	1	•	Block input and output. Z = 0 if B ≠ 0 otherwise Z = 0.
INIR; INDR, OTIR, OTDR	X	1	X	X	X	X	1	•	
LDI, LDD	X	X	X	0	X	↕	0	•	Block transfer instructions. P/V = 1 if BC ≠ 0, otherwise P/V = 0.
LDIR, LDDR	X	X	X	0	X	0	0	•	
CPI; CPIR, CPD; CPDR	X	↕	X	X	X	↕	1	•	Block search instructions. Z = 1 if A = (HL), otherwise Z = 0. P/V = 1 if BC ≠ 0, otherwise P/V = 0.
LD A, I, LD A, R	↕	↕	X	0	X	IFF	0	•	The content of the interrupt enable flip-flop (IFF) is copied into the P/V flag.
BIT b, s	X	↕	X	1	X	X	0	•	The state of bit b of location s is copied into the Z flag.

Symbolic Notation

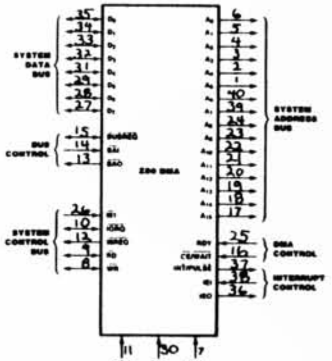
Symbol	Operation
S	Sign flag. S = 1 if the MSB of the result is 1.
Z	Zero flag. Z = 1 if the result of the operation is 0.
P/V	Parity or overflow flag. Parity (P) and overflow (V) share the same flag. Logical operations affect this flag with the parity of the result while arithmetic operations affect this flag with the overflow of the result. If P/V holds parity, P/V = 1 if the result of the operation is even, P/V = 0 if result is odd. If P/V holds overflow, P/V = 1 if the result of the operation produced an overflow.
H	Half-carry flag. H = 1 if the add or subtract operation produced a carry into or borrow from bit 4 of the accumulator.
N	Add/Subtract flag. N = 1 if the previous operation was a subtract.
H & N	H and N flags are used in conjunction with the decimal adjust instruction (DAA) to properly correct the result into packed BCD format following addition or subtraction using operands with packed BCD format.
C	Carry/Link flag. C = 1 if the operation produced a carry from the MSB of the operand or result.

Symbol	Operation
↕	The flag is affected according to the result of the operation.
•	The flag is unchanged by the operation.
0	The flag is reset by the operation.
1	The flag is set by the operation.
X	The flag is a "don't care."
V	P/V flag affected according to the overflow result of the operation.
P	P/V flag affected according to the parity result of the operation.
r	Any one of the CPU registers A, B, C, D, E, H, L.
s	Any 8-bit location for all the addressing modes allowed for the particular instruction.
ss	Any 16-bit location for all the addressing modes allowed for that instruction.
ii	Any one of the two index registers IX or IY.
R	Refresh counter.
n	8-bit value in range < 0, 255 >.
nn	16-bit value in range < 0, 65535 >.

Zilog

Z8410
Z80® DMA Direct Memory Access Controller

Features



Pin Description

A_0-A_{15}. *System Address Bus* (output, 3-state). Addresses generated by the DMA are sent to both source and destination ports (main memory or I/O peripherals) on these lines.

$\overline{\mathrm{BAI}}$. *Bus Acknowledge In* (input, active Low). Signals that the system buses have been released for DMA control. In multiple-DMA configurations, the $\overline{\mathrm{BAI}}$ pin of the highest priority DMA is normally connected to the Bus Acknowledge pin of the CPU. Lower-priority DMAs have their $\overline{\mathrm{BAI}}$ connected to the $\overline{\mathrm{BAO}}$ of a higher-priority DMA.

$\overline{\mathrm{BAO}}$. *Bus Acknowledge Out* (output, active Low). In a multiple-DMA configuration, this pin signals that no other higher-priority DMA has requested the system buses. $\overline{\mathrm{BAI}}$ and $\overline{\mathrm{BAO}}$ form a daisy chain for multiple-DMA priority resolution over bus control.

$\overline{\mathrm{BUSREQ}}$. *Bus Request* (bidirectional, active Low, open drain). **As an output, it sends requests for control of the system address bus, data bus and control bus to the CPU. As an input, when multiple DMAs are strung together in a priority daisy chain via $\overline{\mathrm{BAI}}$ and $\overline{\mathrm{BAO}}$, it senses when another DMA has requested the buses and causes this DMA to** refrain from bus requesting until the other DMA is finished. Because it is a bidirectional pin, there cannot be any buffers between this DMA and any other DMA. It can, however, have a buffer between it and the CPU because it is unidirectional into the CPU. A pull-up resistor is connected to this pin.

$\overline{\mathrm{CE}}/\overline{\mathrm{WAIT}}$. *Chip Enable and Wait* (input, active Low). Normally this functions only as a $\overline{\mathrm{CE}}$ line, but it can also be programmed to serve a $\overline{\mathrm{WAIT}}$ function. As a $\overline{\mathrm{CE}}$ line from the CPU, it becomes active when $\overline{\mathrm{WR}}$ and $\overline{\mathrm{IORQ}}$ are active and the I/O port address on the system address bus is the DMA's address, thereby allowing a transfer of control or command bytes from the CPU to the DMA. As a $\overline{\mathrm{WAIT}}$ line from memory or I/O devices, after the DMA has received a bus-request acknowledge from the CPU, it causes wait states to be inserted in the DMA's operation cycles thereby slowing the DMA to a speed that matches the memory or I/O device.

CLK. *System Clock* (input). Standard Z-80 single-phase clock at 2.5 MHz (Z-80 DMA) or 4.0 MHz (Z-80A DMA). For slower system clocks, a TTL gate with a pullup resistor may be adequate to meet the timing and voltage level specification. For higher-speed systems, use a clock driver with an active pullup to meet the V_{IH} specification and risetime requirements. In all cases there should be a resistive pullup to the power supply of 10K ohms (max) to ensure proper power when the DMA is reset.

D_0-D_7. *System Data Bus* (bidirectional, 3-state). **Commands from the CPU, DMA status, and data from memory or I/O peripherals are transferred on these lines.**

IEI. *Interrupt Enable In* (input, active High). **This is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.**

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this DMA. Thus, this signal blocks lower-priority devices from interrupting while a higher-priority device is being serviced by its CPU interrupt service routine.

$\overline{\mathrm{INT}}$/PULSE. *Interrupt Request* (output, active Low, open drain). This requests a CPU interrupt. The CPU acknowledges the interrupt by pulling its $\overline{\mathrm{IORQ}}$ output Low during an $\overline{\mathrm{M1}}$ cycle. It is typically connected to the $\overline{\mathrm{INT}}$ pin of the CPU with a pullup resistor and tied to all other $\overline{\mathrm{INT}}$ pins in the system. This pin can also be used to generate periodic pulses to an external device. It can be used this way only when the DMA is bus master (i.e., the CPU's $\overline{\mathrm{BUSREQ}}$ and $\overline{\mathrm{BUSACK}}$ lines are both Low and the CPU cannot see interrupts).

$\overline{\mathrm{IORQ}}$. *Input/Output Request* (bidirectional, active Low, 3-state). **As an input, this indicates that the lower half of the address bus holds a** valid I/O port **address** for transfer of control or status bytes from or to the CPU, respectively; **this DMA is the addressed port if its $\overline{\mathrm{CE}}$ pin and its $\overline{\mathrm{WR}}$ or $\overline{\mathrm{RD}}$ pins are simultaneously active. As an output, after the DMA has taken control of the system buses, it indicates that the 8-bit or 16-bit address bus holds a valid port address for another I/O device involved in a DMA transfer of data. When $\overline{\mathrm{IORQ}}$ and $\overline{\mathrm{M1}}$ are both active simultaneously, an interrupt acknowledge is indicated.**

$\overline{\mathrm{M1}}$. *Machine Cycle One* (input, active Low). Indicates that the current CPU machine cycle is an instruction fetch. It is used by the DMA to decode the return-from-interrupt instruction (RETI) (ED-4D) sent by the CPU. During two-byte instruction fetches, $\overline{\mathrm{M1}}$ is active as each opcode byte is fetched. An interrupt acknowledge is indicated when both $\overline{\mathrm{M1}}$ and $\overline{\mathrm{IORQ}}$ are active.

$\overline{\mathrm{MREQ}}$. *Memory Request* **(output, active Low, 3-state). This indicates that the address bus holds a valid address for a memory read or write operation. After the DMA has taken control of the system buses, it indicates a DMA transfer request from or to memory.**

$\overline{\mathrm{RD}}$. *Read* **(bidirectional, active Low, 3-state). As an input, this indicates that the CPU wants to read status bytes from the DMA's read registers. As an output, after the DMA has taken control of the system buses, it indicates a DMA-controlled read from a memory or I/O port address.**

RDY. *Ready* **(input, programmable active Low or High). This is monitored by the DMA to determine when a peripheral device associated with a DMA port is ready for a read or write operation. Depending on the mode of DMA operation (Byte, Burst or Continuous), the RDY line indirectly controls DMA activity by causing the $\overline{\mathrm{BUSREQ}}$ line to go Low or High.**

$\overline{\mathrm{WR}}$. *Write* **(bidirectional, active Low, 3-state). As an input, this indicates that the CPU wants to write control or command bytes to the DMA write registers. As an output, after the DMA has taken control of the system buses, it indicates a DMA-controlled write to a memory or I/O port address.**

Programming

The Z-80 DMA has two programmable fundamental states: (1) an enabled state, in which it can gain control of the system buses and direct the transfer of data between ports, and (2) a disabled state, in which it can initiate neither bus requests nor data transfers. When the DMA is powered up or reset by any means, it is automatically placed into the disabled state. Program commands can be written to it by the CPU in either state, but this automatically puts the DMA in the disabled state, which is maintained until an enable command is issued by the CPU. The CPU must program the DMA in advance of any data search or transfer by addressing it as an I/O port and sending a sequence of control bytes using an Output instruction (such as OTIR for the Z-80 CPU).

Writing. Control or command bytes **are written into one or more of the Write Register groups (WR0-WR6) by first writing to the base register byte in that group. All groups have base registers and most groups have additional associated registers. The associated registers in a group are sequentially accessed by first writing a byte to the base register containing** register-group identification and pointer bits (1's) to one or more of that base register's associated registers.

This is illustrated in Figure 8b. In this figure, the sequence in which associated registers within a group can be written to is shown by the vertical position of the associated registers. For example, if a byte written to the DMA contains the bits that identify WR0 (bits D0, D1 and D7), and also contains 1's in the bit positions that point to the associated "Port A Starting Address (low byte)" and "Port A Starting Address (high byte)," then the next two bytes written to the DMA will be stored in these two registers, in that order.

Reading. The Read Registers (RR0-RR6) are read by the CPU by addressing the DMA as an I/O port using an Input instruction (such as INIR for the Z-80 CPU). The readable bytes contain DMA status, byte-counter values, and port addresses since the last DMA reset. The registers are always read in a fixed sequence beginning with RR0 and ending with RR6. However, the register read in this sequence is determined by programming the Read Mask in WR6. The sequence of reading is initialized by writing an Initiate Read Sequence or Set Read Status command to WR6. After a Reset DMA, the sequence must be initialized with the Initiate Read Sequence command or a Read Status command. The sequence of reading all registers that are not excluded by the Read Mask register must be completed before a new Initiate Read Sequence or Read Status command.

Fixed-Address Programming. A special circumstance arises when programming a destination port to have a fixed address. The load command in WR6 only loads a fixed address to a port selected as the source, not to a port selected as the destination. Therefore, a fixed destination address must be loaded by temporarily declaring it a fixed-source address and subsequently declaring the true source as such, thereby implicitly making the other a destination.

The following example illustrates the steps in this procedure, assuming that transfers are to occur from a variable-address source (Port A) to a fixed-address destination (Port B):

1. Temporarily declare Port B as source in WR0.
2. Load Port B address in WR6.
3. Declare Port A as source in WR0.

Programming
(Continued)

4. Load Port A address in WR6.
5. Enable DMA in WR6.

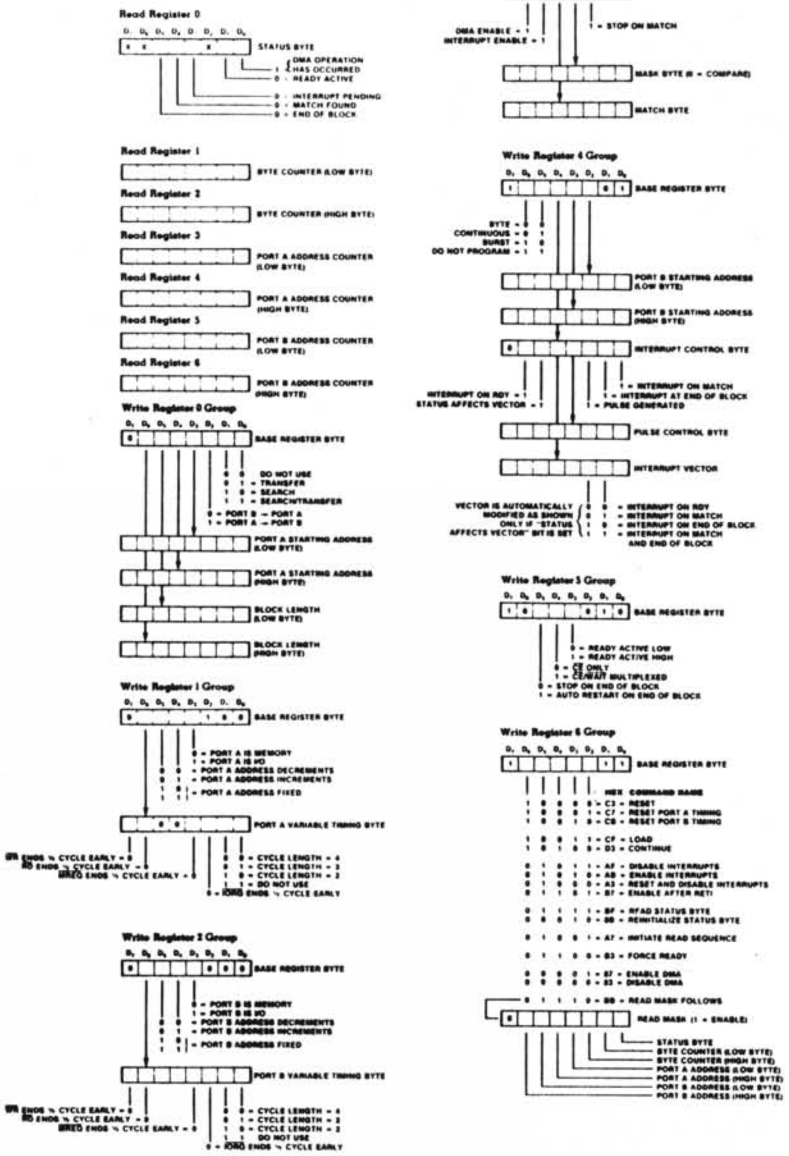
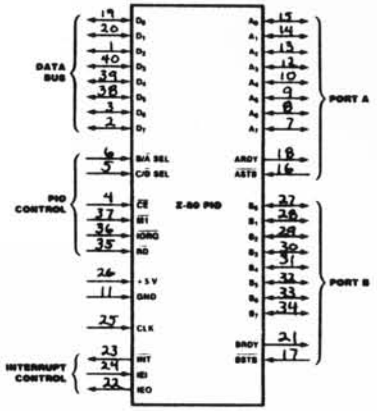

Read Register 0
STATUS BYTE
1 = DMA OPERATION HAS OCCURRED
0 = READY ACTIVE
0 = INTERRUPT PENDING
0 = MATCH FOUND
0 = END OF BLOCK
Read Register 1
BYTE COUNTER (LOW BYTE)
Read Register 2
BYTE COUNTER (HIGH BYTE)
Read Register 3
PORT A ADDRESS COUNTER (LOW BYTE)
Read Register 4
PORT A ADDRESS COUNTER (HIGH BYTE)
Read Register 5
PORT B ADDRESS COUNTER (LOW BYTE)
Read Register 6
PORT B ADDRESS COUNTER (HIGH BYTE)
Write Register 0 Group
BASE REGISTER BYTE
0 0 DO NOT USE
0 1 = TRANSFER
1 0 = SEARCH
1 1 = SEARCH/TRANSFER
0 = PORT B → PORT A
1 = PORT A → PORT B
PORT A STARTING ADDRESS (LOW BYTE)
PORT A STARTING ADDRESS (HIGH BYTE)
BLOCK LENGTH (LOW BYTE)
BLOCK LENGTH (HIGH BYTE)
Write Register 1 Group
BASE REGISTER BYTE
0 = PORT A IS MEMORY
1 = PORT A IS I/O
0 0 = PORT A ADDRESS DECREMENTS
0 1 = PORT A ADDRESS INCREMENTS
1 0 / 1 1 = PORT A ADDRESS FIXED
PORT A VARIABLE TIMING BYTE
WR ENDS ½ CYCLE EARLY = 0
RD ENDS ½ CYCLE EARLY = 0
MREQ ENDS ½ CYCLE EARLY = 0
0 0 = CYCLE LENGTH = 4
0 1 = CYCLE LENGTH = 3
1 0 = CYCLE LENGTH = 2
1 1 = DO NOT USE
0 = IORQ ENDS ½ CYCLE EARLY
Write Register 2 Group
BASE REGISTER BYTE
0 = PORT B IS MEMORY
1 = PORT B IS I/O
0 0 = PORT B ADDRESS DECREMENTS
0 1 = PORT B ADDRESS INCREMENTS
1 0 / 1 1 = PORT B ADDRESS FIXED
PORT B VARIABLE TIMING BYTE
WR ENDS ½ CYCLE EARLY = 0
RD ENDS ½ CYCLE EARLY = 0
MREQ ENDS ½ CYCLE EARLY = 0
0 0 = CYCLE LENGTH = 4
0 1 = CYCLE LENGTH = 3
1 0 = CYCLE LENGTH = 2
1 1 DO NOT USE
0 = IORQ ENDS ½ CYCLE EARLY
Write Register 3 Group
BASE REGISTER BYTE
1 = STOP ON MATCH
DMA ENABLE = 1
INTERRUPT ENABLE = 1
MASK BYTE (0 = COMPARE)
MATCH BYTE
Write Register 4 Group
BASE REGISTER BYTE
BYTE = 0 0
CONTINUOUS = 0 1
BURST = 1 0
DO NOT PROGRAM = 1 1
PORT B STARTING ADDRESS (LOW BYTE)
PORT B STARTING ADDRESS (HIGH BYTE)
INTERRUPT CONTROL BYTE
1 = INTERRUPT ON MATCH
1 = INTERRUPT AT END OF BLOCK
1 = PULSE GENERATED
INTERRUPT ON RDY = 1
STATUS AFFECTS VECTOR = 1
PULSE CONTROL BYTE
INTERRUPT VECTOR
VECTOR IS AUTOMATICALLY MODIFIED AS SHOWN ONLY IF "STATUS AFFECTS VECTOR" BIT IS SET
0 0 = INTERRUPT ON RDY
0 1 = INTERRUPT ON MATCH
1 0 = INTERRUPT ON END OF BLOCK
1 1 = INTERRUPT ON MATCH AND END OF BLOCK
Write Register 5 Group
BASE REGISTER BYTE
0 = READY ACTIVE LOW
1 = READY ACTIVE HIGH
0 = CE ONLY
1 = CE/WAIT MULTIPLEXED
0 = STOP ON END OF BLOCK
1 = AUTO RESTART ON END OF BLOCK
Write Register 6 Group
BASE REGISTER BYTE
HEX COMMAND NAME
1 0 0 0 0 = C3 = RESET
1 0 0 0 1 = C7 = RESET PORT A TIMING
1 0 0 1 0 = CB = RESET PORT B TIMING
1 0 0 1 1 = CF = LOAD
1 0 1 0 0 = D3 = CONTINUE
0 1 0 1 1 = AF = DISABLE INTERRUPTS
0 1 0 1 0 = AB = ENABLE INTERRUPTS
0 1 0 0 0 = A3 = RESET AND DISABLE INTERRUPTS
0 1 1 0 1 = B7 = ENABLE AFTER RETI
0 1 1 1 1 = BF = READ STATUS BYTE
0 0 0 1 0 = 8B = REINITIALIZE STATUS BYTE
0 1 0 0 1 = A7 = INITIATE READ SEQUENCE
0 1 1 0 0 = B3 = FORCE READY
0 0 0 0 1 = 87 = ENABLE DMA
0 0 0 0 0 = 83 = DISABLE DMA
0 1 1 1 0 = BB = READ MASK FOLLOWS
READ MASK (1 = ENABLE)
STATUS BYTE
BYTE COUNTER (LOW BYTE)
BYTE COUNTER (HIGH BYTE)
PORT A ADDRESS (LOW BYTE)
PORT A ADDRESS (HIGH BYTE)
PORT B ADDRESS (LOW BYTE)
PORT B ADDRESS (HIGH BYTE)

Figure 8b. Write Registers

Zilog

Z8420 Z80® PIO Parallel Input/Output Controller

Features



Pin Description

A_0-A_7. *Port A Bus* (bidirectional, 3-state). This 8-bit bus transfers data, status, or control information between Port A of the PIO and a peripheral device. A_0 is the least significant bit of the Port A data bus.

ARDY. *Register A Ready* (output, active High). The meaning of this signal depends on the mode of operation selected for Port A as follows:

Output Mode. This signal goes active to indicate that the Port A output register has been loaded and the peripheral data bus is stable and ready for transfer to the peripheral device.

Input Mode. This signal is active when the Port A input register is empty and ready to accept data from the peripheral device.

Bidirectional Mode. This signal is active when data is available in the Port A output register for transfer to the peripheral device. In this mode, data is not placed on the Port A data bus, unless $\overline{\text{ASTB}}$ is active.

Control Mode. This signal is disabled and forced to a Low state.

$\overline{\text{ASTB}}$. *Port A Strobe Pulse From Peripheral Device* (input, active Low). The meaning of this signal depends on the mode of operation selected for Port A as follows:

Output Mode. The positive edge of this strobe is issued by the peripheral to acknowledge the receipt of data made available by the PIO.

Input Mode. The strobe is issued by the peripheral to load data from the peripheral into the Port A input register. Data is loaded into the PIO when this signal is active.

Bidirectional Mode. When this signal is active, data from the Port A output register is gated onto the Port A bidirectional data bus. The positive edge of the strobe acknowledges the receipt of the data.

Control Mode. The strobe is inhibited internally.

B_0-B_7. *Port B Bus* (bidirectional, 3-state). This 8-bit bus transfers data, status, or control information between Port B and a peripheral device. The Port B data bus can supply 1.5 mA at 1.5 V to drive Darlington transistors. B_0 is the least significant bit of the bus.

B/$\overline{\text{A}}$. *Port B Or A Select* (input, High = B). This pin defines which port is accessed during a data transfer between the CPU and the PIO. A Low on this pin selects Port A; a High selects Port B. Often address bit A_0 from the CPU is used for this selection function.

BRDY. *Register B Ready* (output, active High). This signal is similar to ARDY, except that in the Port A bidirectional mode this signal is High when the Port A input register is empty and ready to accept data from the peripheral device.

$\overline{\text{BSTB}}$. *Port B Strobe Pulse From Peripheral Device* (input, active Low). This signal is similar to $\overline{\text{ASTB}}$, except that in the Port A bidirectional mode this signal strobes data from the peripheral device into the Port A input register.

C/$\overline{\text{D}}$. *Control Or Data Select* (input, High = C). This pin defines the type of data transfer to be performed between the CPU and the PIO. A High on this pin during a CPU write to the PIO causes the Z-80 data bus to be interpreted as a *command* for the port selected by the B/$\overline{\text{A}}$ Select line. A Low on this pin means that the Z-80 data bus is being used to transfer data between the CPU and the PIO. Often address bit A_1 from the CPU is used for this function.

$\overline{\text{CE}}$. *Chip Enable* (input, active Low). A Low on this pin enables the PIO to accept command or data inputs from the CPU during a write cycle or to transmit data to the CPU during a read cycle. This signal is generally decoded from four I/O port numbers for Ports A and B, data, and control.

CLK. *System Clock* (input). The Z-80 PIO uses the standard single-phase Z-80 system clock.

D_0-D_7. *Z-80 CPU Data Bus* (bidirectional, 3-state). This bus is used to transfer all data and commands between the Z-80 CPU and the Z-80 PIO. D_0 is the least significant bit.

IEI. *Interrupt Enable In* (input, active High). This signal is used to form a priority-interrupt daisy chain when more than one interrupt-driven device is being used. A High level on this pin indicates that no other devices of higher priority are being serviced by a CPU interrupt service routine.

Pin Description (Continued)

IEO. *Interrupt Enable Out* (output, active High). The IEO signal is the other signal required to form a daisy chain priority scheme. It is High only if IEI is High and the CPU is not servicing an interrupt from this PIO. Thus this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

INT. *Interrupt Request* (output, open drain, active Low). When $\overline{\text{INT}}$ is active the Z-80 PIO is requesting an interrupt from the Z-80 CPU.

$\overline{\text{IORQ}}$. *Input/Output Request* (input from Z-80 CPU, active Low). $\overline{\text{IORQ}}$ is used in conjunction with B/$\overline{\text{A}}$, C/$\overline{\text{D}}$, $\overline{\text{CE}}$, and $\overline{\text{RD}}$ to transfer commands and data between the Z-80 CPU and the Z-80 PIO. When $\overline{\text{CE}}$, $\overline{\text{RD}}$, and $\overline{\text{IORQ}}$ are active, the port addressed by B/$\overline{\text{A}}$ transfers data to the CPU (a read operation). Conversely, when $\overline{\text{CE}}$ and $\overline{\text{IORQ}}$ are active but $\overline{\text{RD}}$ is not, the port addressed by B/$\overline{\text{A}}$ is written into from the CPU with either data or control information, as specified by C/$\overline{\text{D}}$. Also, if $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are active simultaneously, the CPU is acknowledging an interrupt; the interrupting port automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

$\overline{\text{M1}}$. *Machine Cycle* (input from CPU, active Low). This signal is used as a sync pulse to control several internal PIO operations. When both the $\overline{\text{M1}}$ and $\overline{\text{RD}}$ signals are active, the Z-80 CPU is fetching an instruction from memory. Conversely, when both $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are active, the CPU is acknowledging an interrupt. In addition, $\overline{\text{M1}}$ has two other functions within the Z-80 PIO: it synchronizes the PIO interrupt logic; when $\overline{\text{M1}}$ occurs without an active $\overline{\text{RD}}$ or $\overline{\text{IORQ}}$ signal, the PIO is reset

$\overline{\text{RD}}$. *Read Cycle Status* (input from Z-80 CPU, active Low). If $\overline{\text{RD}}$ is active, or an I/O operation is in progress, $\overline{\text{RD}}$ is used with B/$\overline{\text{A}}$, C/$\overline{\text{D}}$, $\overline{\text{CE}}$, and $\overline{\text{IORQ}}$ to transfer data from the Z-80 PIO to the Z-80 CPU.

Programming

Mode 0, 1, or 2. *(Byte Input, Output, or Bidirectional).* Programming a port for Mode 0, 1, or 2 requires two words per port. These words are:

A Mode Control Word. Selects the port operating mode (Figure 6). This word may be written any time.

An Interrupt Vector. The Z-80 PIO is designed for use with the Z-80 CPU in interrupt Mode 2 (Figure 7). When interrupts are enabled, the PIO must provide an interrupt vector.

Mode 3. *(Bit Input/Output).* Programming a port for Mode 3 operation requires a control word, a vector (if interrupts are enabled), and three additional words, described as follows:

I/O Register Control. When Mode 3 is selected, the mode control word must be followed by another control word that sets the I/O control register, which in turn defines which port lines are inputs and which are outputs (Figure 8).

Interrupt Control Word. In Mode 3, handshake is not used. Interrupts are generated as a logic function of the input signal levels. The interrupt control word sets the logic conditions and the logic levels required for generating an interrupt. Two logic conditions or functions are available: AND (if all input bits change to the active level, an interrupt is triggered), and OR (if any one of the input bits changes to the active level, an interrupt is triggered). Bit D_6 sets the logic function, as shown in Figure 9. The active level of the input bits can be set either High or Low. The active level is controlled by Bit D_5.

Mask Control Word. This word sets the mask control register, allowing any unused bits to be masked off. If any bits are to be masked, then D_4 must be set. When D_4 is set, the next word written to the port must be a mask control word (Figure 10).

Interrupt Disable. There is one other control word which can be used to enable or disable a port interrupt. It can be used without changing the rest of the interrupt control word (Figure 11).

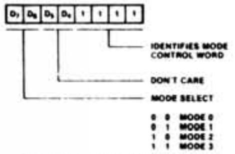


Figure 6. Mode Control Word

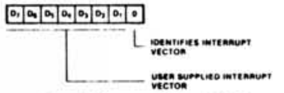


Figure 7. Interrupt Vector Word

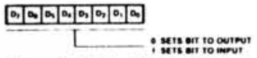


Figure 8. I/O Register Control Word

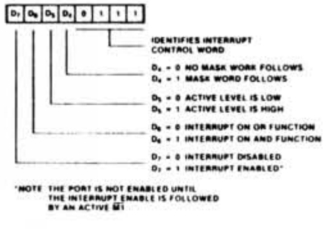

D7 D6 D5 D4 0 1 1 1
IDENTIFIES INTERRUPT CONTROL WORD
D4 = 0 NO MASK WORK FOLLOWS
D4 = 1 MASK WORD FOLLOWS
D5 = 0 ACTIVE LEVEL IS LOW
D5 = 1 ACTIVE LEVEL IS HIGH
D6 = 0 INTERRUPT ON OR FUNCTION
D6 = 1 INTERRUPT ON AND FUNCTION
D7 = 0 INTERRUPT DISABLED
D7 = 1 INTERRUPT ENABLED*
*NOTE THE PORT IS NOT ENABLED UNTIL THE INTERRUPT ENABLE IS FOLLOWED BY AN ACTIVE M1

Figure 9. Interrupt Control Word

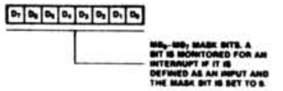

D7 D6 D5 D4 D3 D2 D1 D0
MB0-MB7 MASK BITS. A BIT IS MONITORED FOR AN INTERRUPT IF IT IS DEFINED AS AN INPUT AND THE MASK BIT IS SET TO 0.

Figure 10. Mask Control Word

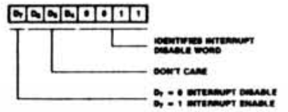
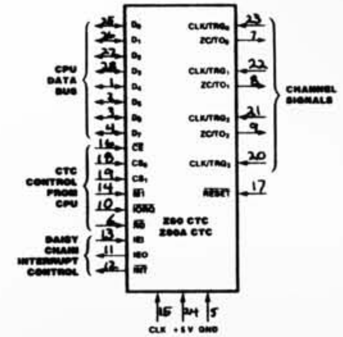


Figure 11. Interrupt Disable Word

Zilog

Z8430
Z80® CTC Counter/ Timer Circuit

Features



Pin Description

$\overline{\text{CE}}$. *Chip Enable* (input, active Low). When enabled the CTC accepts control words, interrupt vectors, or time constant data words from the data bus during an I/O write cycle; or transmits the contents of the down-counter to the CPU during an I/O read cycle. In most applications this signal is decoded from the eight least significant bits of the address bus for any of the four I/O port addresses that are mapped to the four counter-timer channels.

CLK. *System Clock* (input). Standard single-phase Z-80 system clock.

CLK/TRG$_0$-CLK/TRG$_3$. *External Clock/Timer Trigger* (input, user-selectable active High or Low). Four pins corresponding to the four Z-80 CTC channels. In counter mode, every active edge on this pin decrements the down-counter. In timer mode, an active edge starts the timer.

CS$_0$-CS$_1$. *Channel Select* (inputs active High). Two-bit binary address code selects one of the four CTC channels for an I/O write or read (usually connected to A_0 and A_1).

D$_0$-D$_7$. *System Data Bus* (bidirectional, 3-state). Transfers all data and commands between the Z-80 CPU and the Z-80 CTC.

IEI. *Interrupt Enable In* (input, active High). A High indicates that no other interrupting devices of higher priority in the daisy chain are being serviced by the Z-80 CPU.

IEO. *Interrupt Enable Out* (output, active High). High only if IEI is High and the Z-80 CPU is not servicing an interrupt from any Z-80 CTC channel. IEO blocks lower priority devices from interrupting while a higher priority interrupting device is being serviced.

$\overline{\text{INT}}$. *Interrupt Request* (output, open drain, active Low). Low when any Z-80 CTC channel that has been programmed to enable interrupts has a zero-count condition in its down-counter.

$\overline{\text{IORQ}}$. *Input/Output Request* (input from CPU, active Low). Used with $\overline{\text{CE}}$ and $\overline{\text{RD}}$ to transfer data and channel control words between the Z-80 CPU and the Z-80 CTC. During a write cycle, $\overline{\text{IORQ}}$ and $\overline{\text{CE}}$ are active and $\overline{\text{RD}}$ inactive. The Z-80 CTC does not receive a specific write signal; rather, it internally generates its own from the inverse of an active $\overline{\text{RD}}$ signal. In a read cycle, $\overline{\text{IORQ}}$, $\overline{\text{CE}}$ and $\overline{\text{RD}}$ are active; the contents of the down-counter are read by the Z-80 CPU. If $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are both true, the CPU is acknowledging an interrupt request, and the highest priority interrupting channel places its interrupt vector on the Z-80 data bus.

$\overline{\text{M1}}$. *Machine Cycle One* (input from CPU, active Low). When $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ are active, the Z-80 CPU is acknowledging an interrupt. The Z-80 CTC then places an interrupt vector on the data bus if it has highest priority, and if a channel has requested an interrupt ($\overline{\text{INT}}$).

$\overline{\text{RD}}$. *Read Cycle Status* (input, active Low). Used in conjunction with $\overline{\text{IORQ}}$ and $\overline{\text{CE}}$ to transfer data and channel control words between the Z-80 CPU and the Z-80 CTC.

$\overline{\text{RESET}}$. *Reset* (input active Low). Terminates all down-counts and disables all interrupts by resetting the interrupt bits in all control registers; the ZC/TO and the Interrupt outputs go inactive; IEO reflects IEI; D_0-D_7 go to the high-impedance state.

ZC/TO$_0$-ZC/TO$_2$. *Zero Count/Timeout* (output, active High). Three ZC/TO pins corresponding to Z-80 CTC channels 2 through 0 (Channel 3 has no ZC/TO pin). In both counter and timer modes the output is an active High pulse when the down-counter decrements to zero.

Programming

Each Z-80 CTC channel must be programmed prior to operation. Programming consists of writing two words to the I/O port that corresponds to the desired channel. The first word is a control word that selects the operating mode and other parameters; the second word is a time constant, which is a binary data word with a value from 1 to 256. A time constant word must be preceded by a channel control word.

After initialization, channels may be reprogrammed at any time. If updated control and time constant words are written to a channel during the count operation, the count continues to zero before the new time constant is loaded into the counter.

If the interrupt on any Z-80 CTC channel is enabled, the programming procedure should also include an interrupt vector. Only one vector is required for all four channels, because the interrupt logic automatically modifies the vector for the channel requesting service.

A control word is identified by a 1 in bit 0. A 0 in bit 2 indicates a time constant word is to follow. Interrupt vectors are always addressed to Channel 0, and identified by a 0 in bit 0.

Addressing. During programming, channels are addressed with the channel select pins CS_1 and CS_2. A 2-bit binary code selects the appropriate channel as shown in the following table.

Channel	CS_1	CS_0
0	0	0
1	0	1
2	1	0
3	1	1

Reset. The CTC has both hardware and software resets. The hardware reset terminates all down-counts and disables all CTC interrupts by resetting the interrupt bits in the control registers. In addition, the ZC/TO and Interrupt outputs go inactive, IEO reflects IEI, and D_0-D_7 go to the high-impedance state. All channels must be completely reprogrammed after a hardware reset.

The software reset is controlled by bit 1 in the channel control word. When a channel receives a software reset, it stops counting. When a software reset is used, the other bits in the control word also change the contents of the channel control register. After a software reset a new time constant word must be written to the same channel.

If the channel control word has both bits D_1 and D_2 set to 1, the addressed channel stops operating, pending a new time constant word. The channel is ready to resume after the new constant is programmed. In timer mode, if D_3 = 0, operation is triggered automatically when the time constant word is loaded.

Channel Control Word Programming. The channel control word is shown in Figure 5. It sets the modes and parameters described below.

Interrupt Enable. D_7 enables the interrupt, so that an interrupt output ($\overline{INT}$) is generated at zero count. Interrupts may be programmed in either mode and may be enabled or disabled at any time.

Operating Mode. D_6 selects either timer or counter mode.

Prescaler Factor. (Timer Mode Only). D_5 selects factor—either 16 or 256.

Trigger Slope. D_4 selects the active edge or slope of the CLK/TRG input pulses. Note that reprogramming the CLK/TRG slope during operation is equivalent to issuing an active edge. If the trigger slope is changed by a control word update while a channel is pending operation in timer mode, the result is the same as a CLK/TRG pulse and the timer starts. Similarly, if the channel is in counter mode, the counter decrements.

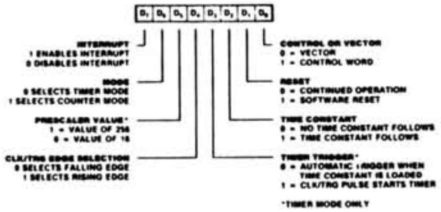


Figure 5. Channel Control Word

Programming (Continued)

Trigger Mode (Timer Mode Only). D_3 selects the trigger mode for timer operation. When D_3 is reset to 0, the timer is triggered automatically. The time constant word is programmed during an I/O write operation, which takes one machine cycle. At the end of the write operation there is a setup delay of one clock period. The timer starts automatically (decrements) on the rising edge of the second clock pulse (T_2) of the machine cycle following the write operation. Once started, the timer runs continuously. At zero count the timer reloads automatically and continues counting without interruption or delay, until stopped by a reset.

When D_3 is set to 1, the timer is triggered externally through the CLK/TRG input. The time constant word is programmed during an I/O write operation, which takes one machine cycle. The timer is ready for operation on the rising edge of the second clock pulse (T_2) of the following machine cycle. Note that the first timer decrement follows the active edge of the CLK/TRG pulse by a delay time of one clock cycle if a minimum setup time to the rising edge of clock is met. If this minimum is not met, the delay is extended by another clock period. Consequently, for immediate triggering, the CLK/TRG input must precede T_2 by one clock cycle plus its minimum setup time. If the minimum time is not met, the timer will start on the third clock cycle (T_3).

Once started the timer operates continuously, without interruption or delay, until stopped by a reset.

Time Constant to Follow. A 1 in D_2 indicates that the next word addressed to the selected channel is a time constant data word for the time constant register. The time constant word may be written at any time.

A 0 in D_2 indicates no time constant word is to follow. This is ordinarily used when the channel is already in operation and the new channel control word is an update. A channel will not operate without a time constant value. The only way to write a time constant value is to write a control word with D_2 set.

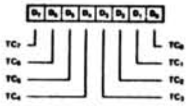


Figure 6. Time Constant Word

Software Reset. Setting D_1 to 1 causes a software reset, which is described in the Reset section.

Control Word. Setting D_0 to 1 identifies the word as a control word.

Time Constant Programming. Before a channel can start counting it must receive a time constant word from the CPU. During programming or reprogramming, a channel control word in which bit 2 is set must precede the time constant word to indicate that the next word is a time constant. The time constant word can be any value from 1 to 256 (Figure 6). Note that 00_{16} is interpreted as 256.

In timer mode, the time interval is controlled by three factors:

- The system clock period (ϕ)
- The prescaler factor (P), which multiplies the interval by either 16 or 256
- The time constant (T), which is programmed into the time constant register

Consequently, the time interval is the product of ϕ × P × T. The minimum timer resolution is 16 × ϕ (4 μs with a 4 MHz clock). The maximum timer interval is 256 × ϕ × 256 (16.4 ms with a 4 MHz clock). For longer intervals timers may be cascaded.

Interrupt Vector Programming. If the Z-80 CTC has one or more interrupts enabled, it can supply interrupt vectors to the Z-80 CPU. To do so, the Z-80 CTC must be pre-programmed with the most-significant five bits of the interrupt vector. Programming consists of writing a vector word to the I/O port corresponding to the Z-80 CTC Channel 0. Note that D_0 of the vector word is always zero, to distinguish the vector from a channel control word. D_1 and D_2 are not used in programming the vector word. These bits are supplied by the interrupt logic to identify the channel requesting interrupt service with a unique interrupt vector (Figure 7). Channel 0 has the highest priority.

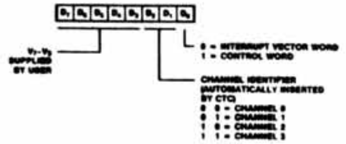
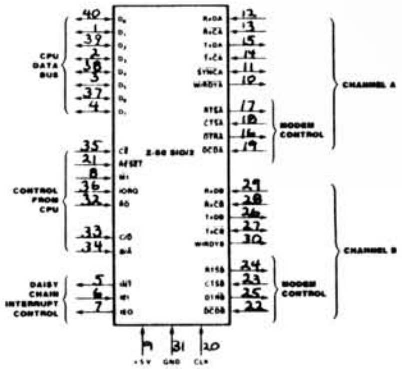


Figure 7. Interrupt Vector Word

Zilog

Z8440
Z80® SIO Serial Input/Output Controller

Features



Pin Description

Figures 1 through 6 illustrate the three pin configurations (bonding options) available in the SIO. The constraints of a 40-pin package make it impossible to bring out the Receive Clock ($\overline{\text{RxC}}$), Transmit Clock ($\overline{\text{TxC}}$), Data Terminal Ready ($\overline{\text{DTR}}$) and Sync ($\overline{\text{SYNC}}$) signals for both channels. Therefore, either Channel B lacks a signal or two signals are bonded together in the three bonding options offered:

- Z-80 SIO/2 lacks $\overline{\text{SYNCB}}$
- Z-80 SIO/1 lacks $\overline{\text{DTRB}}$
- Z 90 SIO/0 has all four signals, but $\overline{\text{TxCB}}$ and $\overline{\text{RxCB}}$ are bonded together

The first bonding option above (SIO/2) is the preferred version for most applications. The pin descriptions are as follows:

B/$\overline{\textbf{A}}$. *Channel A Or B Select* (input, High selects Channel B). This input defines which channel is accessed during a data transfer between the CPU and the SIO. Address bit A_0 from the CPU is often used for the selection function.

C/$\overline{\textbf{D}}$. *Control Or Data Select* (input, High selects Control). This input defines the type of information transfer performed between the CPU and the SIO. A High at this input during a CPU write to the SIO causes the information on the data bus to be interpreted as a command for the channel selected by B/$\overline{\text{A}}$. A Low at C/$\overline{\text{D}}$ means that the information on the data bus is data. Address bit A_1 is often used for this function.

$\overline{\textbf{CE}}$. *Chip Enable* (input, active Low). A Low level at this input enables the SIO to accept command or data input from the CPU during a write cycle or to transmit data to the CPU during a read cycle.

CLK. *System Clock* (input). The SIO uses the standard Z-80 System Clock to synchronize internal signals. This is a single-phase clock.

$\overline{\textbf{CTSA}}$, $\overline{\textbf{CTSB}}$. *Clear To Send* (inputs, active Low). When programmed as Auto Enables, a Low on these inputs enables the respective transmitter. If not programmed as Auto Enables, these inputs may be programmed as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these inputs and interrupts the CPU on both logic level transitions. The Schmitt-trigger buffering does not guarantee a specified noise-level margin.

$\textbf{D}_0$-$\textbf{D}_7$. *System Data Bus* (bidirectional, 3-state). The system data bus transfers data and commands between the CPU and the Z-80 SIO. D_0 is the least significant bit.

$\overline{\textbf{DCDA}}$, $\overline{\textbf{DCDB}}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if the SIO is programmed for Auto Enables; otherwise they may be used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow-risetime signals. The SIO detects pulses on these pins and interrupts the CPU on both logic level transitions. Schmitt-trigger buffering does not guarantee a specific noise-level margin.

$\overline{\textbf{DTRA}}$, $\overline{\textbf{DTRB}}$. *Data Terminal Ready* (outputs, active Low). These outputs follow the state programmed into Z-80 SIO. They can also be programmed as general purpose outputs.

In the Z-80 SIO/1 bonding option, $\overline{\text{DTRB}}$ is omitted.

IEI. *Interrupt Enable In* (input, active High). This signal is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

IEO. *Interrupt Enable Out* (output, active High). IEO is High only if IEI is High and the CPU is not servicing an interrupt from this SIO. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{\textbf{INT}}$. *Interrupt Request* (output, open drain, active Low). When the SIO is requesting an interrupt, it pulls $\overline{\text{INT}}$ Low.

$\overline{\textbf{IORQ}}$. *Input/Output Request* (input from CPU, active Low). $\overline{\text{IORQ}}$ is used in conjunction with B/$\overline{\text{A}}$, C/$\overline{\text{D}}$, $\overline{\text{CE}}$ and $\overline{\text{RD}}$ to transfer commands and data between the CPU and the SIO. When $\overline{\text{CE}}$, $\overline{\text{RD}}$ and $\overline{\text{IORQ}}$ are all active, the channel selected by B/$\overline{\text{A}}$ transfers data to the CPU (a read operation). When $\overline{\text{CE}}$ and $\overline{\text{IORQ}}$ are active but $\overline{\text{RD}}$ is inactive, the channel selected by B/$\overline{\text{A}}$ is written to by the CPU with either data or control information as specified by C/$\overline{\text{D}}$. If $\overline{\text{IORQ}}$ and $\overline{\text{M1}}$ are active simultaneously, the CPU is acknowledging an interrupt and the SIO automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

Pin Description (Continued)

$\overline{\textbf{M1}}$. *Machine Cycle* (input from Z-80 CPU, active Low). When $\overline{\text{M1}}$ is active and $\overline{\text{RD}}$ is also active, the Z-80 CPU is fetching an instruction from memory; when $\overline{\text{M1}}$ is active while $\overline{\text{IORQ}}$ is active, the SIO accepts $\overline{\text{M1}}$ and $\overline{\text{IORQ}}$ as an interrupt acknowledge if the SIO is the highest priority device that has interrupted the Z-80 CPU.

$\overline{\textbf{RxCA}}$, $\overline{\textbf{RxCB}}$. *Receiver Clocks* (inputs). Receive data is sampled on the rising edge of $\overline{\text{RxC}}$. The Receive Clocks may be 1, 16, 32 or 64 times the data rate in asynchronous modes. These clocks may be driven by the Z-80 CTC Counter Timer Circuit for programmable baud rate generation. Both inputs are Schmitt-trigger buffered (no noise level margin is specified).

In the Z-80 SIO/0 bonding option, $\overline{\text{RxCB}}$ is bonded together with $\overline{\text{TxCB}}$.

$\overline{\textbf{RD}}$. *Read Cycle Status* (input from CPU, active Low). If $\overline{\text{RD}}$ is active, a memory or I/O read operation is in progress. $\overline{\text{RD}}$ is used with B/$\overline{\text{A}}$, $\overline{\text{CE}}$ and $\overline{\text{IORQ}}$ to transfer data from the SIO to the CPU.

RxDA, RxDB. *Receive Data* (inputs, active High). Serial data at TTL levels.

$\overline{\textbf{RESET}}$. *Reset* (input, active Low). A Low $\overline{\text{RESET}}$ disables both receivers and transmitters, forces TxDA and TxDB marking, forces the modem controls High and disables all interrupts. The control registers must be rewritten after the SIO is reset and before data is transmitted or received.

$\overline{\textbf{RTSA}}$, $\overline{\textbf{RTSB}}$. *Request To Send* (outputs, active Low). When the RTS bit in Write Register 5 (Figure 14) is set, the $\overline{\text{RTS}}$ output goes Low. When the RTS bit is reset in the Asynchronous mode, the output goes High after the transmitter is empty. In Synchronous modes, the $\overline{\text{RTS}}$ pin strictly follows the state of the RTS bit. Both pins can be used as general-purpose outputs.

$\overline{\textbf{SYNCA}}$, $\overline{\textbf{SYNCB}}$. *Synchronization* (inputs/outputs, active Low). These pins can act either as inputs or outputs. In the asynchronous receive mode, they are inputs similar to $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$. In this mode, the transitions on these lines affect the state of the Sync/Hunt status bits in Read Register 0 (Figure 13), but have no other function. In the External Sync mode, these lines also act as inputs. When external synchronization is achieved, $\overline{\text{SYNC}}$ must be driven Low on the second rising edge of $\overline{\text{RxC}}$ after that rising edge of $\overline{\text{RxC}}$ on which the last bit of the sync character was received. In other words, after the sync pattern is detected, the external logic must wait for two full Receive Clock cycles to activate the $\overline{\text{SYNC}}$ input. Once $\overline{\text{SYNC}}$ is forced Low, it should be kept Low until the CPU informs the external synchronization detect logic that synchronization has been lost or a new message is about to start. Character assembly begins on the rising edge of $\overline{\text{RxC}}$ that immediately precedes the falling edge of $\overline{\text{SYNC}}$ in the External Sync mode.

In the internal synchronization mode (Monosync and Bisync), these pins act as outputs that are active during the part of the receive clock ($\overline{\text{RxC}}$) cycle in which sync characters are recognized. The sync condition is not latched, so these outputs are active each time a sync pattern is recognized, regardless of character boundaries.

In the Z-80 SIO/2 bonding option, $\overline{\text{SYNCB}}$ is omitted.

$\overline{\textbf{TxCA}}$, $\overline{\textbf{TxCB}}$. *Transmitter Clocks* (inputs). In asynchronous modes, the Transmitter Clocks may be 1, 16, 32 or 64 times the data rate; however, the clock multiplier for the transmitter and the receiver must be the same. The Transmit Clock inputs are Schmitt-trigger buffered for relaxed rise- and fall-time requirements (no noise level margin is specified). Transmitter Clocks may be driven by the Z-80 CTC Counter Timer Circuit for programmable baud rate generation.

In the Z-80 SIO/0 bonding option, $\overline{\text{TxCB}}$ is bonded together with $\overline{\text{RxCB}}$.

TxDA, TxDB. *Transmit Data* (outputs, active High). Serial data at TTL levels. TxD changes from the falling edge of $\overline{\text{TxC}}$.

$\overline{\textbf{W}}$/$\overline{\textbf{RDYA}}$, $\overline{\textbf{W}}$/$\overline{\textbf{RDYB}}$. *Wait/Ready A, Wait/Ready B* (outputs, open drain when programmed for Wait function, driven High and Low when programmed for Ready function). These dual-purpose outputs may be programmed as Ready lines for a DMA controller or as Wait lines that synchronize the CPU to the SIO data rate. The reset state is open drain.

Programming

The system program first issues a series of commands that initialize the basic mode of operation and then other commands that qualify conditions within the selected mode. For example, the asynchronous mode, character length, clock rate, number of stop bits, even or odd parity might be set first; then the interrupt mode; and finally, receiver or transmitter enable.

Both channels contain registers that must be programmed via the system program prior to operation. The channel-select input (B/$\bar{A}$) and the control/data input (C/$\bar{D}$) are the command-structure addressing controls, and are normally controlled by the CPU address bus. Figures 15 and 16 illustrate the timing relationships for programming the write registers and transferring data and status.

Read Registers. The SIO contains three read registers for Channel B and two read registers for Channel A (RR0-RR2 in Figure 13) that can be read to obtain the status information; RR2 contains the internally-modifiable interrupt vector and is only in the Channel B register set. The status information includes error conditions, interrupt vector and standard communications-interface signals.

To read the contents of a selected read register other than RR0, the system program must first write the pointer byte to WR0 in exactly the same way as a write register operation. Then, by executing a read instruction, the contents of the addressed read register can be read by the CPU.

The status bits of RR0 and RR1 are carefully grouped to simplify status monitoring. For example, when the interrupt vector indicates that a Special Receive Condition interrupt has occurred, all the appropriate error bits can be read from a single register (RR1).

Write Registers. The SIO contains eight write registers for Channel B and seven write registers for Channel A (WR0-WR7 in Figure 14) that are programmed separately to configure the functional personality of the channels; WR2 contains the interrupt vector for both channels and is only in the Channel B register set. With the exception of WR0, programming the write registers requires two bytes. The first byte is to WR0 and contains three bits (D_0-D_2) that point to the selected register; the second byte is the actual control word that is written into the register to configure the SIO.

WR0 is a special case in that all of the basic commands can be written to it with a single byte. Reset (internal or external) initializes the pointer bits D_0-D_2 to point to WR0. This implies that a channel reset must not be combined with the pointing to any register.

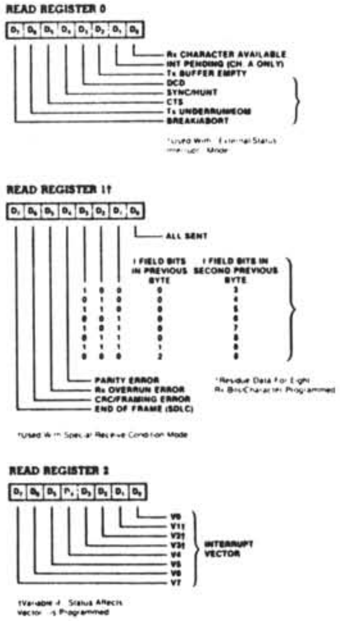


Figure 13. Read Register Bit Functions

Programming (Continued)

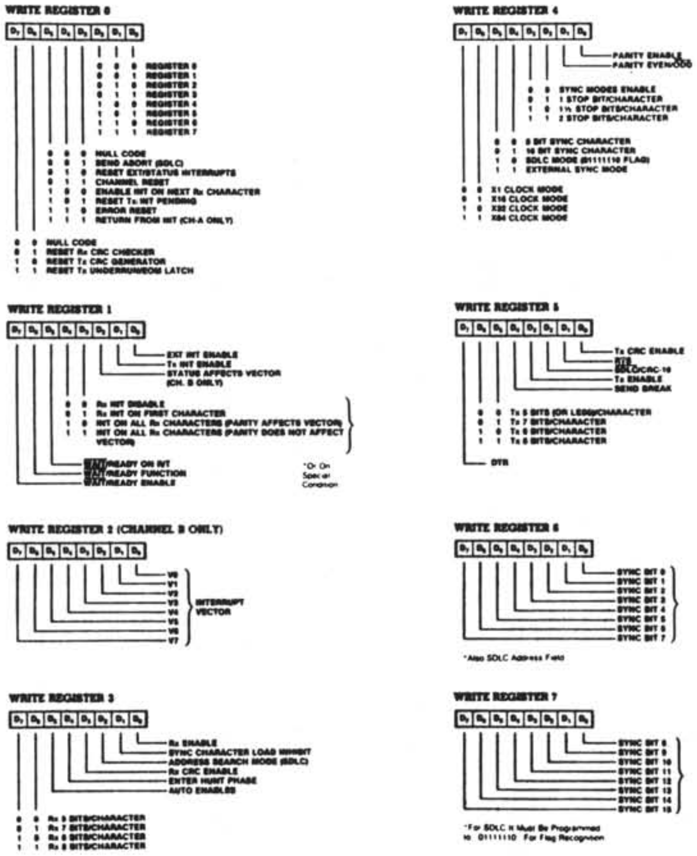


Figure 14. Write Register Bit Functions