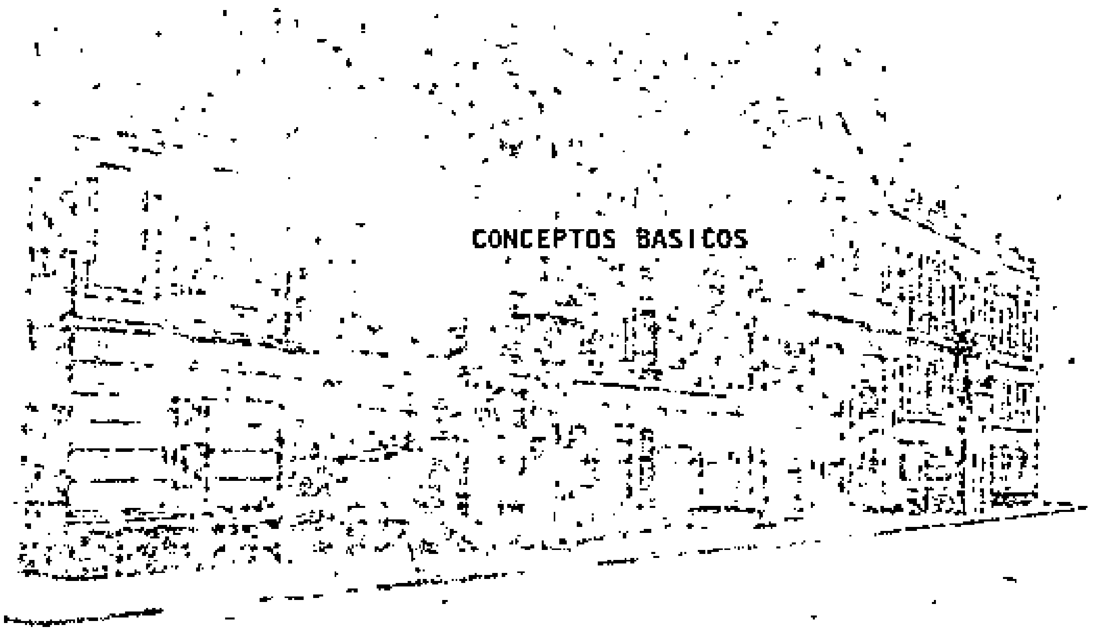
DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

INTRODUCCION A LOS MICROPROCESADORES (Z-80)



CONCEPTOS BASICOS

M.EN C. ARMANDO TORRES FENTANES

Marzo, 1982

CURSO MICROS

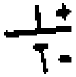
- CONCEPTOS BASICOS DE ELECTRONICA
- CONCEPTOS BASICOS DE ELECTRONICA DIGITAL
- CONCEPTOS DE PROGRAMACION
- LENGUAJE DE PROGRAMACION
- μ PROCESADOR M6800
- PERIFERICOS
- APLICACIONES
- LABORATORIO

CONCEPTOS BASICOS DE ELECTRONICA

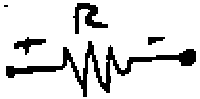
- PPIOS DE ELECTRONICA
- ELEMENTOS LOGICOS
- ALGEBRA BOOLEANA
- CIRCUITOS INTEGRADOS
- FAMILIAS LOGICAS
- MINIMIZACION F. BOOLEANAS
- SISTEMAS DE NUMERACION
- ARITMETICA DIGITAL PARA NUMEROS NO SIGNADOS
- ARITMETICA DIGITAL PARA NUMEROS SIGNADOS
- CODIGOS
- CIRC. DIGITALES BASICOS
- MEMORIAS
- MICROPROCESADORES

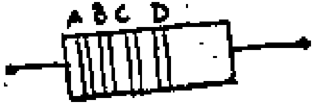
CONCEPTOS BASICOS

VOLTAJE – POTENCIAL QUE GENERA LA CORRIENTE
UNIDAD: [Volts] [V]
Ej.: BATERIAS, ACUMULADORES

BATERIAS (CARACT.)
- POTENCIAL
- BORNES + y –
- ENERGIA QUE ENTREGA
- SIMBOLO 

- PUEDEN CONECTARSE EN SERIE

RESISTENCIA (R) – OPOSICION AL FLUJO DE ELECTRONES
UNIDAD: [OHMS] [Ω]
$1000\Omega = 1K\Omega$
$1000 K\Omega = 1M\Omega$
SIMBOLO GRAFICO 

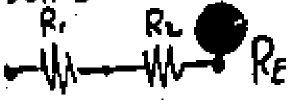
RESIST. CARBON


A: 1er DIGITO
B: 2° DIGITO
C: POTENCIA DE DIEZ X
D: TOLERANCIA

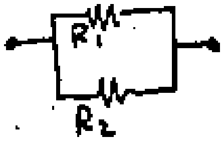
COLORES:

NEGRO	0	ORO	5%
CAFE	1	PLATA	10%
ROJO	2		
NARANJA	3		
AMAR.	4		
VERDE	5		
AZUL	6		
VIOLETA	7		
GRIS	8		

TIPOS DE CONEX.

a) SERIE


$R_{EQ} = R_1 + R$

b) PARALELO


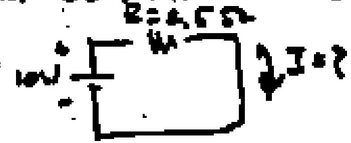
$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R}$

$R_{eq} < \min(R_1, R_2)$

CORRIENTE – FLUJO DE ELECTRONES
UNIDAD: AMPERES [A] < mA, μA

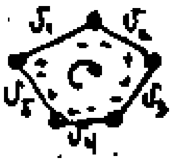
LEY DE OHM
$I = \frac{V}{R}$ ó $V = RI$

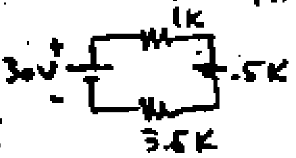
RANGO APROX. EN CIRC. DIG. < 1A



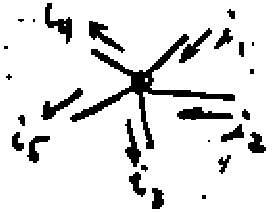
LEYES DE KIRCHHOFF

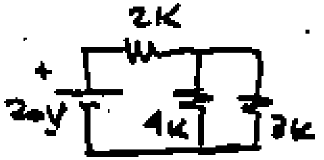
a) VOLTAJE $\Sigma V_i = 0$ EN UNA MALLA CERRADA





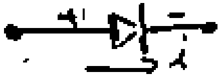
b) CORRIENTE $\Sigma I_i = 0$ EN UN NODO

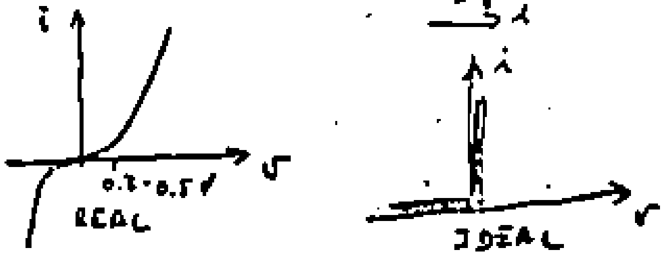


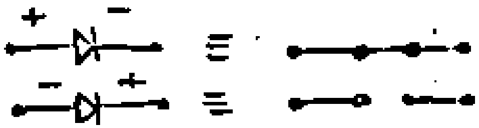


DIODOS: PERMITEN EL FLUJO DE CORRIENTE EN UN SOLO SENTIDO (SWITCH)

SIMBOLO GRAFICO





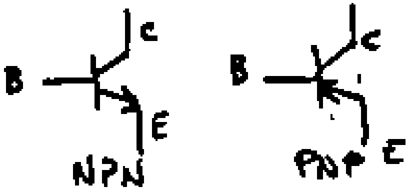


TRANSISTOR: DISPOSITIVO CREADO CON 3 CAPA DE SEMICONDUCTORES

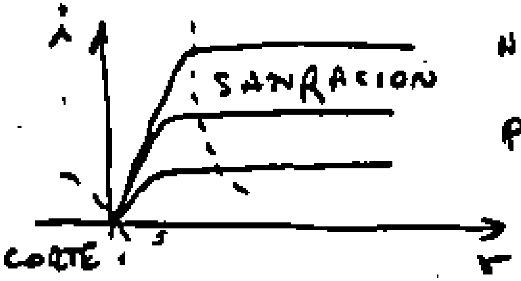
USOS: - FTE DE CORRIENTE
- AMPLIF. DE CORR.
- INVERSOR
- OTROS

APLIC. SIST. DIGITALES: SWITCH

SIMBOLO GRAFICO



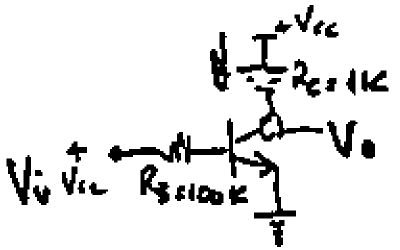
$i_C = \beta i_B$



NPN - Polariz. Posit.

PNP - Polariz. Negat.

FUNCIONAMIENTO



Vi	Vo

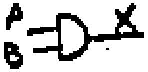
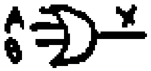
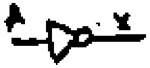
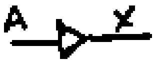
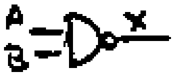
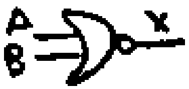
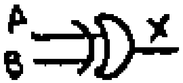
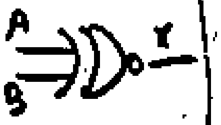
VOLTAJE	N. LOGICO
0V	0
+5V	1

ELEMENTOS LOGICOS

CARACTERISTICA: MANEJAN SEÑALES BINARIAS (0 y 1)

COMPUERTAS LOGICAS: DISPOSITIVOS ELECTRONICOS QUE DAN UNA SALIDA COMO FUNCION DE UNA O VARIAS ENTRADAS. TODAS LAS VAR. DE ENTRADA Y SALIDA SON BINARIAS

TABLAS DE VERDAD: REPRESENTACION EN FORMA TABULAR DE LA RELACION ENTRE LAS VAR. DE ENTRADA Y LAS DE SALIDA (PARA TODAS LAS COMBINACIONES DE LAS VAR. DE ENTRADA).

NOMBRE	MBOLO	F. ALGEB AKA	T. DE VERDA
AND		$X = AB$	A B | X 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$X = A+B$	A B | X 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
INVERSOR COMPLEM.)		$X = A' = \overline{A}$	A | X 0 | 1 1 | 0
BUFFER		X=A, amplifica corriente	A | X 0 | 0 1 | 1
NAND		$X = (AB)'$	A B | X 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$X = (A+B)'$	A B | X 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
XOR f. n n		$X = A \oplus B$ $= A'B + AB'$	A B | X 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
XNOR f. par		$X = A \odot B$ $= A'B' + AB$	A B | Y 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

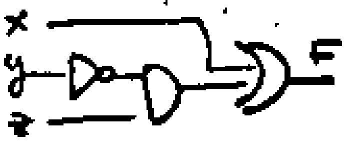
ALGEBRA BOOLEANA = TRATA CON VARIABLES INARI
Y LAS OPERACIONES LOGICAS:
AND
OR
COMPLEMENTU

FUNCION BOOLEANA = ES UNA EXPRESION ALGEBRAI
QUE CONTIENE:
- VARIABLES BINARIAS
- SIMBOLOS DE F. LOGICAS
- ADQUIERE DOS VALORES 1 ó 0

Ejemplo

$$F = x + y'z$$

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

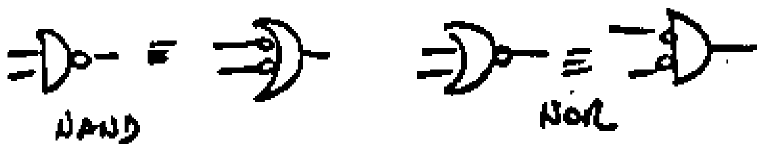


TEOREMAS DEL ALGEBRA BOOLEANA

1) $x+0=x$ 2) $x \cdot 0 = 0$
3) $x+1=1$ 4) $x \cdot 1 = x$
5) $x+x=x$ 6) $x \cdot x = x$
7) $x+x'=1$ 8) $x \cdot x' = 0$
9) $x+y=y+x$ 10) $xy=yx$
11) $x+(y+z)=(x+y)+z$ 12) $x(yz)=(xy)z$
13) $x(y+z)=xy+xz$ 14) $x+yz=(x+y)(x+z)$
15) $(x+y)'=x'y'$ 16) $(xy)'=x'+y'$
17) $(x')'=x$

$$xy' + y = x + y$$
$$f = f$$
$$x + \bar{x}y = x + y$$

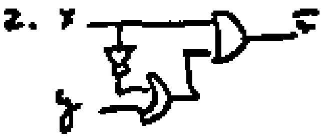
DE LAS LEYES DE MORGAN



EJEMPLO:

1. $f = [(A'+B)\cdot B']'$

2.



CIRCUITOS INTEGRADOS (IC, CHIPS

QUÉ SON: SEMICONDUCTORES CON ELEMENTOS MONTADOS EN UN PAQUETE DE CERAMICA Y PATAS EXTERNAS

VENTAJAS:
- REDUCC. DE TAMAÑO
- REDUCC. DE COSTO
- REDUCC. DE CONSUMO DE POTENCIA
- MAYOR CONFIABILIDAD
- VELOCIDAD + ALTA
- MENOS CONEXIONES EXTERNAS

TIPOS:
LINEALES: AMPLIF. OPERACIONALES, COMPARADORES, REGULADORES DE VOLTAJE
DIGITALES: COMPUERTAS, FF, REGISTROS, CONTADORES, SUMADORES, ALU, MEMORIAS

FAMILIAS LOGICAS +USADAS:
RTL
DTL
TTL
ECL
MOS
CMOS

PARAMETROS MAS IMPORTANTES:
FAN IN
FAN OUT
DISIPACION DE POTENCIA (POWER DISSIPATION)
RETRASO (PROPAGATION DELAY)
MARGEN DE RUIDO (NOISE MARGIN)

TTL

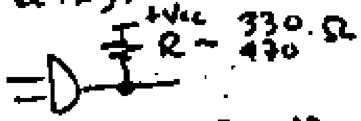
- FAN OUT ~ 10
- NOISE MARGIN ~ 0.4V
- ENTRADA ABIERTA ≡ **ALTO**

VERSIONES

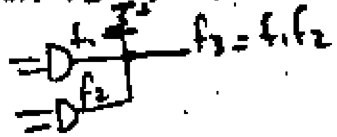
		ns	mW
ESTANDAR	TTL	10	10
LOW POWER	LTTL	33	1
HIGH SPEED	HTTL	6	22
SHOTKY /	STTL	3	19
LOW POWER SHOTKY	LSTTL	1.5	2

CONFIGURACIONES A LA SALIDA PARA C/VERSION

OPEN COLLECTOR: REQUIERE RESISTOR EXTERNO A LA SALIDA CONECTADA A +VCC

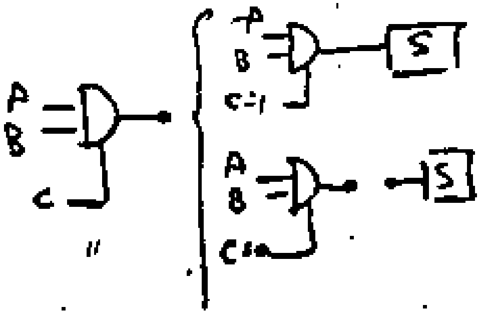


PERMITE WIRE-AND



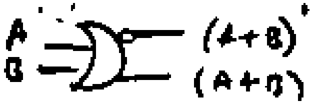
TOTEM POLE: SALIDA ESTANDAR POCO RETRASO, MAY FANOUT

TRISTATE: 3 ESTADOS DE SALIDA
- * 1
- * 0
- * ALTA IMPEDANCIA

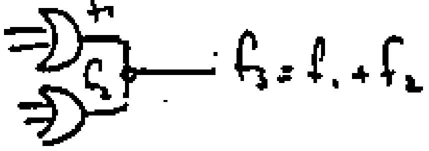


ECL

- TRABAJAN EN NO SATURACION
- RETRASO ~ 2 ns
- DISIPACION — 25 mW
- FAN OUT > 25
- NOISE MARGIN = 0.2 V



- PERMITE FORMAR WIRE-OR



MOS

- PMOS - Polariz. negat.
- NMOS - Polariz. posit.
- NO COMPATIBLE CON TTL
- RETRASO > ECL y TTL
- ALTA DENSIDAD
- BAJO CONSUMO DE POTENCIA (10 uW)
- USOS: MEMORIAS, ALU

CMOS

- COMPATIBLE CON TTL
- PMOS + NMOS
- FAN OUT > 50
- DISIPACION ~10 nW
- DELAY ~ 25 ns

CALA DE ITEGRACION

SSI	< 10
MSI	(10, 100)
LSI	(100, 1000)
VLSI	> 1000

MINIMIZACION

formas canonicas

- Σ de Π : $f(x_1,\dots,x_n) = \underbrace{(x_1\bar{x}_2\dots x_n)}_{\text{minitérminos}} + (x_1\dots x_n) + \dots$
- Π de Σ : $f(x_1,\dots,x_n) = \underbrace{(x_1+x_2+\dots+\bar{x}_n)}_{\text{maxitérminos}} \cdot (\bar{x}_1+\dots+x_n) \cdot$
- f = Σ de minitérminos para los cuales f = 1
- f = Π de maxitérminos para los cuales f = 0

F	x y z	minitérminos		maxitérminos	
0	0 0 0	$x'y'z'$	m_0	$x+y+z$	M_0
1	0 0 1	⋮			
0	0 1 0				
0	0 1 1			⋮	
1	1 0 0	⋮			
0	1 0 1				
0	1 1 0				
1	1 1 1	xyz	m_7	$x'+y'+z'$	M_7

$$M_i = \bar{m}_i$$

$$f = \Sigma(1,4,7) = m_1 + m_4 + m_7$$

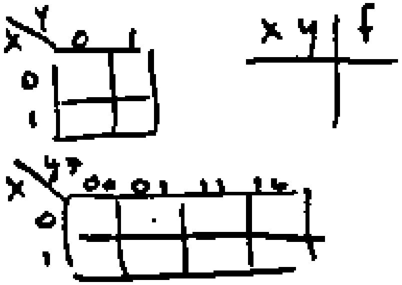
$$f = \Pi(0,2,3,5,6) = M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$$

METODOS DE REDUCCION

- ALGEBRA BOOLEANA
- MAPAS DE KARNAUGH
- QUINE Mc CLUSKEY.

MAPAS DE KARNAUGH

- REPRESENTACION GRAFICA DE TABLA DE VERDAD
- TABLERO CON 2^n CUADROS SI HAY n VARIABLES DE ENTRADA
- CUADRO ≡ MINITERMINO
- VALOR CUADRO ≡ VALOR DE LA FUNCION PARA EL MINITERMINO
- UTIL HASTA PARA 5 VAR
- LOS CUADROS SE ARREGLAN DE MANERA QUE SOLO DIFIERAN EN UN BIT DOS CUADROS ADYACENTES.
- EJEMPLO

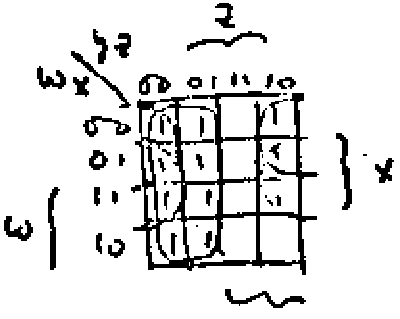


METODO

- TRAZAR TABLERO CON 2^n CUADROS PARA n VAR. DE ENTRADA
- NUMERAR CUADROS EN FORMA TAL QUE LOS ADYASCENTES DIFIERAN EN UN BIT
- LLENAR TABLERO
- AGRUPAR LOS UNOS FORMANDO CONJUNTOS QUE SEAN LO MAS GRANDE POSIBLE. # ELEMENTOS EN EL CONJUNTO = 2^i
- ESCOGER MINIMA CANTIDAD DE CONJUNTOS QUE CUBRA TODOS LOS UNOS (IMPLICANTES PRIMOS)
- ENCONTRAR LA REPRESENTACION ALGEBRAICA DE LOS I.P.
- $f = \Sigma$ DE I.P.

EJEMPLO

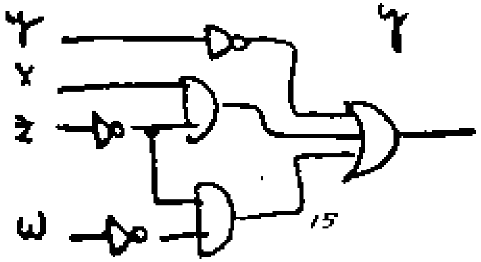
$f(w,x,y,z) = \Sigma\,(0,1,2,4,5,6,8,9,12,13,14$



wx \ yz	00	01	11	10
00	1	1		1
01	1	1		1
11	1	1		1
10	1	1		

y

$f = \bar{y} + y\bar{z} + \bar{w}\bar{z}$



SISTEMAS DE NUMERACION

RAZON: COMPUTADORA DIGITAL MANEJA SEÑALES BINARIAS (1 ó 0)

SIST. DE NUM. DE BASE M

- EMPLEA M DIGITOS $(0,\ldots,M-1)$ PARA REPRESENTAR LOS VALORES NUMERICOS
- EJEMPLO

* DECIMAL	0 - 9
* BINARIO	0, 1
* OCTAL	0 - 7
* HEXA	0 - F

CONVERSION BASE M A BASE 10

$$a_n a_{n-1} \cdots a_0 \cdot a_{-1} a_{-2} \cdots a_{-n\,[m]} = \sum_{i=-n}^{n} a_i * m^i$$

- EJEMPLO

$1010110.01_{[2]}$

$7624.3_{[8]}$

REPRESENTACION PARA VALORES DIGITALES

- INFORMACION DE TIPO BINARIO
- ELEMENTO BASICO DE INFORMACION = 1 DIGITO BINARIO = BIT
- 1 BYTE = 8 BITS
- 1 PALABRA = { 1 BYTES, 2 ", 3 ", 4
- SIST. BINARIO { 0, 1
- SIST. OCTAL { 0, 1, 2, 3, 4, 5, 6, 7
- SIST. HEXADEC. { 0, 1, .., 9, A, B, C, D, E, F

CONVERSION BINARIA, OCTAL, HEXADEC.

$$\begin{tikzcd}
 & \text{BINARIA} \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\text{OCTAL} & & \text{HEXADECIMAL}
\end{tikzcd}$$

a) BINARIA → OCTAL $2^{\textcircled{3}} = 8$

- DIVIDIR # BIN. EN PERIODOS DE 3

$\leftarrow \; \cdot \; \rightarrow$

- OBTENER # DEC. EQUIVALENTE EN CADA PERIODO

101 011 101 011.11011

b) OCTAL → BINARIA

- REEMPLAZAR CADA DIGITO OCTAL POR SU EQUIVALENTE BINARIO

$5072.41_{(8)}$

c) BINARIA → HEXADECIMAL $2^{\textcircled{4}} = 16$

- DIVIDIR # BIN. EN PERIODOS DE 4

$\leftarrow \; \cdot \; \rightarrow$

- OBTENER # DEC. EQUIVALENTE EN CADA PERIODO

$11010\ 1110\ 1011.11011_{(2)}$

d) HEXADEC. → BINARIO

- REEMPLAZAR CADA DIGITO HEXADECIMAL POR SU EQUIVALENTE BINARIO

ABC.E9

CONVERSION DE BASE 10 A BASE M

- SEPARAR # DECIMAL EN . TE ENTE
  Y FRACCIONARIA

$$A_{[10]} = A_E + A_F$$

a) PARTE ENTERA$_{[10]}$ → PARTE ENTERA$_{[M]}$

- DIVIDIR SUCESIVAMENTE A_E POR
- LOS RESIDUOS OBTENIDOS SON LOS DIGITOS EN BASE M DE LA PARTE ENTERA.

$$\begin{array}{r|l|l}
m & \underline{A_E} & \text{RESIDUOS} \\
m & \underline{N_1} & B_0 \\
m & \underline{N_2} & B_1 \\
 & \vdots & \\
m & \underline{N_n} & B_{n-1} \\
 & 0 & B_n
\end{array} \uparrow$$

LEER
ESTA
DIRECC

$$B_{E_{(m)}} = B_n B_{n-1} \ldots B_0$$

$41_{[10]} \rightarrow [2], [8], [16]$

b) PARTE FRACC.$_{(10)}$ → PARTE FRACC.

- MULTIPLICAR SUCESIVAMENTE A
  LAS FRACCIONES RESULTANTE(S) P
  M.
- LAS PARTES ENTERAS QUE S
  OBTENGAN CORRESPONDEN
  LOS DIGITOS EN BASE 'M'
  LA PARTE FRACCIONARIA

LEER EN ESTA DIRECC.

$$\begin{array}{r} A_F \\ \times M \\ \hline B_{-1} \cdot N_{-1} \\ \times M \\ \hline B_{-2} \cdot N_{-2} \\ \times M \\ \hline B_{-3} \cdot N_{-3} \end{array}$$

$$B_{F[M]} = \cdot B_{-1} B_{-2} B_{-3} \ldots$$

$$.6875_{[10]} \rightarrow [2], [8], [16]$$

$$- \ B_{[M]} = B_E \cdot B_F$$

ARITMETICA DIGITAL (NUMEROS NO SIGNADOS

SUMA

BASE 10

$$\begin{array}{rl} 1 & \leftarrow C_{i+1} \\ 174 & \leftarrow A_i \\ 229 & \leftarrow B_i \\ \hline 403 & \leftarrow S_i \end{array}$$

$S_i = A_i + B_i + C_i - n(M)$

$S_i \in [0, M-1]$

$C_{i+1} = n$

BASE 2

$0+0=0$

$0+1=1$

$1+0=1$

$1+1=10$

$$\begin{array}{r} 111010 \\ +\,011101 \\ \hline \end{array}$$

BASE 8

$$\begin{array}{r} 7425 \\ +\,6237 \\ \hline \end{array}$$

RESTA

$A - B = A + (-B)$ ← # SIGNADO

MULTIPLIC.

BASE 10

$$\begin{array}{rl} 1 & \leftarrow CP_i \\ 16 & \leftarrow A_i \\ \times\,12 & \leftarrow B_i \\ \hline 32 & \leftarrow P_i \\ 16 & \\ \hline 192 & \end{array}$$

$P_i = A_i B_i + CP_i - n(M)$

$P_i \in [0, M-1]$

$CP_{i+1} = n$

BASE 2

$0 \times 0 = 0$

$0 \times 1 = 0$

$1 \times 0 = 0$

$1 \times 1 = 1$

$$\begin{array}{r} 101101 \\ \times\,101 \\ \hline \end{array}$$

BASE 8

$$\begin{array}{r} 527 \\ \times\,37 \\ \hline 4541 \\ 2005 \\ \hline 24611 \end{array}$$

DIVISION

BASE 10

```
      ┌──── COi
      ↓
     68
  5 ⟌340    A
  ↑  -30
  B   ───
       40
  Ri   0
```

A/B

$CO_i = \begin{cases} 0 , & A < B \\ \max(n) \ni n \times B < A \end{cases}$

$R_i = A - n \times B$

NUMERO QUE DEBE SER SUMADO A n×B PARA OBTENER A.

BASE 2

$\frac{0}{0}$, $\frac{1}{0}$ > NO TIENE SENTIDO

$\frac{0}{1} = 0$ RESTO 0

$\frac{1}{1} = 1$ RESTO 0

EJEMPLO

```
            11
      110 ⟌10010
           -110
           ────
            110
           -110
           ────
            000
```

21

ARITMETICA BINARIA (NUMEROS SIGNADOS)

* SE REQUIERE UN BIT EXTRA PARA REPRE-SENTAR EL SIGNO

TIPOS DE REPRESENTACIONES
- MAGNITUD Y SIGNO
- 1'COMPLEMENTO
- 2'COMPLEMENTO

BIT DE SIGNO EN ∀ REPRESENTACION
- 1 ⇒ −
- 0 ⇒ +

1'COMPLEMENTO DE # BINARIO
- COMPLEMENTAR TODOS LOS BITS DE LA PALABRA 1 → 0, 0 → 1

```
10111.01  # BINARIO
01000.10  1'COMPL.
```

2'COMPLEMENTO DE # BINARIO
- SE OBTIENE SUMANDO '1' AL 1'COMPLEMENTO DEL NUMERO BINARIO

```
   10111.01  # BINARIO
   01000.10  1'COMPL.
 +        1
 ──────────
   01000.11  2'COMPL.
```

- COMPLEMENTAR BITS A PARTIR DEL 1er BIT = 1 (DE DER A IZQ.)

22

REPRESENTACION DE #S SIGNADOS EN 2'COMPLEMENTO CON 'n' BITS

- # POSITIVO
  - REPRESENTAR MAGNITUD EN FORMA BINARIA EMPLEANDO n-1 BITS
  - HACER BIT DEL SIGNO (n) = 0
- # NEGATIVO
  - REPRESENTAR MAGNITUD EN FORMA BINARIA EMPLEANDO n-1 BITS
  - HACER BIT DE SIGNO (n) = 0
  - OBTENER 2'COMPLEMENTO DE LOS '<u>n</u>' BITS
- PARA CONOCER LA MAGNITUD DE UN # NEGATIVO (BIT DE SIGNO = 1), OBTENER EL 2' COMPLEMENTO DE LOS 'n' BITS Y LEER LA MAGNITUD EN LOS N-1 BITS
- EJEMPLO
  EMPLEAR 4 BITS PARA ±5
- RANGO DE VALORES REPRESENTABLES

$$-2^{n-1} \le x \le 2^{n-1}-1$$

SUMA
S=A+B

- SUMAR LOS DOS NUMEROS (INCLUYENDO B DE SIGNO), DESECHAR EL BIT DE ACAR QUE SE OBTENGA AL SUMAR LOS BITS DE SIGNO (CARRY OUT)
- RESULTADO ES <u>VALIDO</u> <u>CUANDO</u>:
  - a) <u>NI</u> ENTRA <u>NI</u> SALE CARRY DEL BI DE SIGNO
  - b) ENTRA <u>Y</u> SALE CARRY DEL BIT DE SIGNO
- EJEMPLO

$$\begin{array}{r}5\\+6\\\hline\end{array}\quad\begin{array}{r}6\\+(-5)\\\hline\end{array}\quad\begin{array}{r}-6\\+5\\\hline\end{array}\quad\begin{array}{r}-6\\+\;-5\\\hline\end{array}$$

RESTA
R=A-B

$$R = A - B = A + (-B)$$

MULTIPLIC.
M=AxB

- OBTENER PRODUCTO DE |A|·|B|
- OBTENER 2' COMPLEMENTO DEL RESULTADO CUANDO
  $A>0, B<0$
  $A<0, B>0$
- EL PRODUCTO TENDRA 2n BITS DE MAGNITUD

DIVISION
D=A/B

- OBTENER EL COCIENTE |A|/|B|
- OBTENER 2' COMPLEMENTO DEL RESULTADO CUANDO
  $A>0, B<0$
  ó $A<0, B>0$

CODIGOS

DEFINICION: FORMA DE REPRESENTAR LA INFORMACION CON SIMBOLOS DIFERENTES

REPRESENTACION DE #S DECIMALES

- CODIGO BINARIO (BCD)

0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

* BCD ≠ REPRES. BINARIA

* EJEMPLO

985_{10}

$\underbrace{1001}\ \underbrace{1000}\ \underbrace{0101}$

$49_{10} = 110001_2 = 01001001_{BCD}$

* PARA SUMAR AGREGAR 6_{BCD} A CADA RESULTADO QUE SEA INVALIDO

```
60  0110 0000
55  0101 0101
   ----------
    1011 0101
    0110
  -----------
  1 0001 0101  = 115
```

* VENTAJAS: REDUCE CONVERSIONES NECESARIAS E/S

* DESV.: LAS OPERACIONES SON MAS LENTAS

REPRESENTACION DE CARACTERES

EBCDIC
- Extended Binary Code International Corporation (I
- 8 BITS/CHAR

ASCII
- American Standard Code for Information Interchange
- Emplea 7 bits
- Generalmente se agrega 1 bit de paridad (+ signif.
- Permite representar
  - 128 caract. (may. y min
  - 95 símbolos gráficos
  - 23 formateadores
  - 10 comunicación y statu

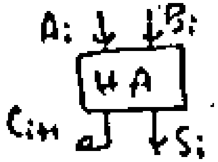
CAR	ASCII
A	100 0001
B	
C	
D	
E	
F	
G	
H	
I	
J	
K	
L	
M	
N	
O	100 1111
P	101 0000
Q	
R	
S	
T	
U	
V	
W	
X	
Y	
Z	[illegible]

26

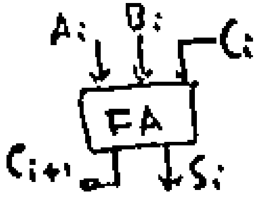
CAR	ASCII
0	011 0000
1	
2	
3	
4	
5	
6	
7	
8	
9	011 1001
blanco	010 0000
.	1110
(	1000
+	1011
$	0100
*	1010
)	1001
-	1101
/	1111
,	010 1100
=	011 1101

CIRCUITOS DIGITALES BASICOS

MEDIO SUMADOR (HALF ADDER)

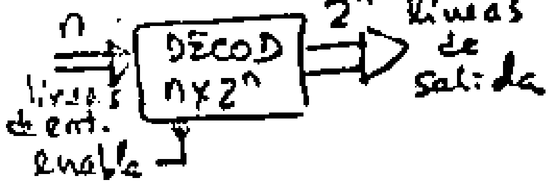


SUMADOR COMPLETO (FULL ADDER)

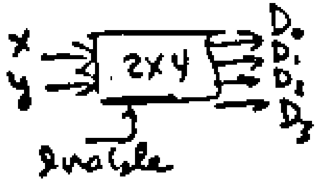


DECODIFICADOR

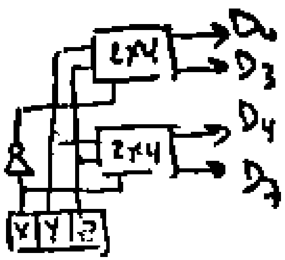
- CONVIERTE INFORMACION BINARIA DE UN CODIGO A OTRO
- SELECCION DE DISPOSITIVOS

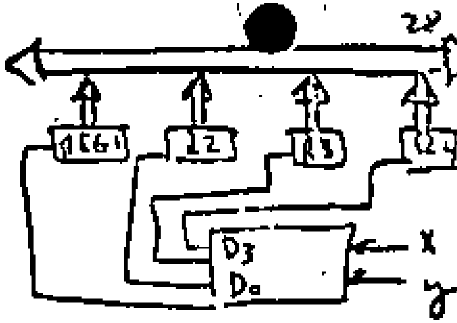


- SOLO UNA LINEA DE SALIDA ADQUIERE VALOR UNITARIO PARA C/COMBINACION DE LAS LINEAS DE ENTRADA
- LOS HAY DE 2X4, 3X8, 4X16,...
- CON ENABLE SE PUEDEN UNIR DOS PARA GENERAR OTRO DE MAYOR CAPACIDAD



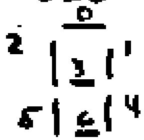
X	Y	D_0	D_1	D_2	D_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1



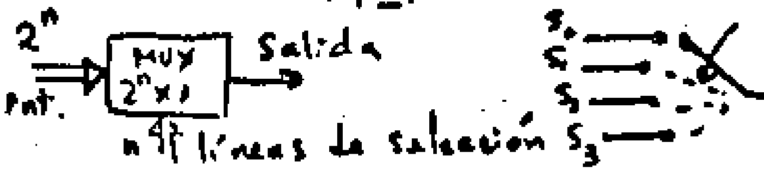


PROBLEMA:

CONSTRUIR UN CIRC. CON DECODIF. QUE CONVIERTA CODIGO BCD EN CODIGO DE 7 SEGMENTOS

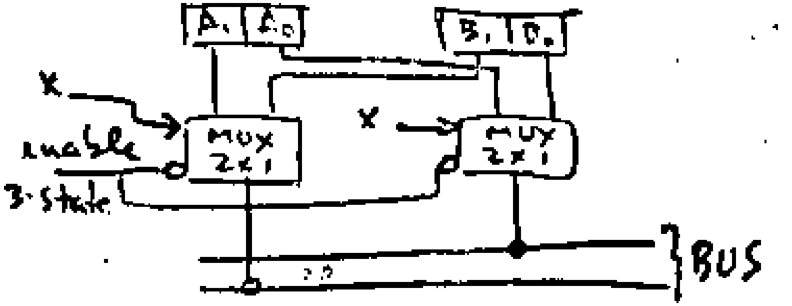


MULTIPLEXORES



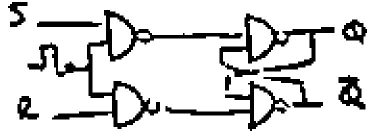
USOS

- SELECC. INFORMACION DE ENTRADA PROVENIENTE DE DIFERENTES REGISTROS
- MULTIPLEXAR SEÑALES BINARIAS
- CONVERSION PARALELO SERIE
- IMPLEMENTAR FUNCIONES BOOLEAN



FLIP-FLOPS

- CIRC. DIGITALES CON MEMORIA
  $Q(t) = f(S_t, R_t, Q_{t-1})$



- UNIDAD BASICA DE MEMORIA (1FF = 1BIT)
- TIPOS

RS síncrono

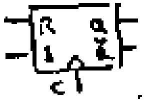
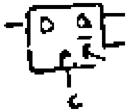


TABLA TRANSICION

S	R	Q_{t+1}	
0	0	Q_t	
0	1	0	clear
1	0	1	set
1	1	?	

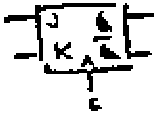
D



D	Q_{t+1}	
0	0	clear
1	1	set

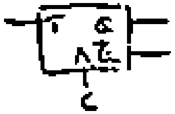
$R = \bar{S}$
$D = S$

J K



J	K	Q_{t+1}	
0	0	Q_t	
0	1	0	clear
1	0	1	set
1	1	$\bar{Q}_t$	

T



T	Q_{t+1}
0	Q_t
1	$\bar{Q}_t$

$J = K$

- TABLAS DE EXCITACION

Q_t	Q_{t+1}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

Q_t	Q_{t+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

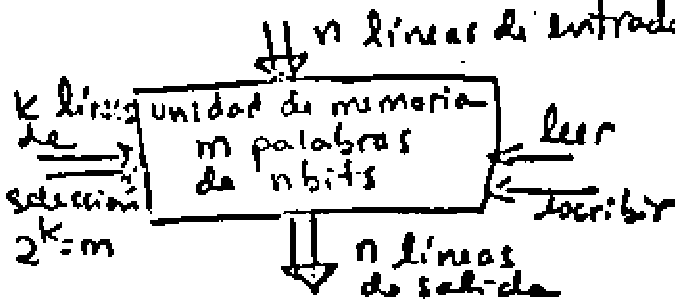
Q_t	Q_{t+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

Q_t	Q_{t+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

MEMORIAS

DEFINICION: CONJUNTO DE FF DE ALMACENAMIENTO JUNTO CON LOS CIRCUITOS ASOCIADOS NECESARIOS PARA LA XFERENCIA DE INFORMACION DE E y/o S.

ESTRUCTURA



$2^k = m$

TIPOS (ACCESO ALEATORIO)

- CORE
  - NUCLEOS MAGNETICOS
  - NO VOLATILES
  - LENTAS Y VOLUMINOSAS
- MOS
  - VOLATILES
  - RAPIDAS Y COMPACTAS
  - TIPOS
    - RAM
      - LEEN Y ESCRIBEN
      - ESTATICAS
      - DINAMICAS ó requieren REFRESH
    - ROM
      - LECTURA
      - INFORMACION ES GRABADA POR FABRICANTE
    - PROM
      - LECTURA
      - INFO. LA GRABA EL USUARIO
    - EPROM
      - PROM QUE PUEDE BORRARSE CON UV

PARAMETROS MAS RELEVANTES

TIEMPO DE ACCESO: Tiempo que transcurre entre la llegada da la Señal de lectura y que la información este disponible a la salida

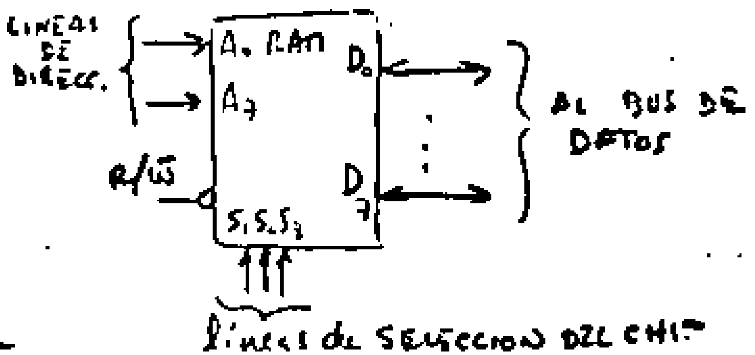
TIEMPO DE CICLO = TIEMPO DE ACCESO + $\underbrace{\text{TPO. RESTAURACION}}_{\text{PARA CORE.}}$

RAM – 100 ns : 400 ns

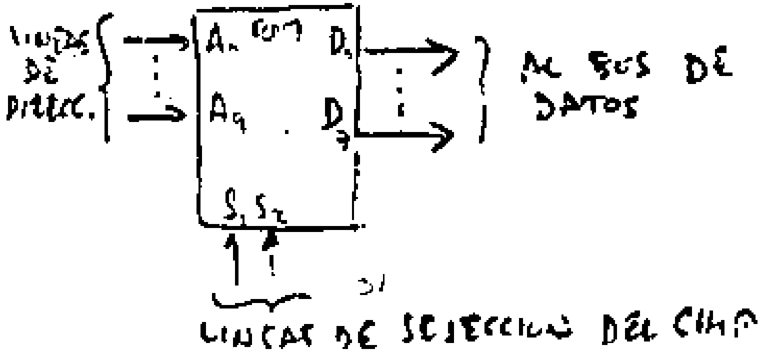
CORE – 1 μs

RAM

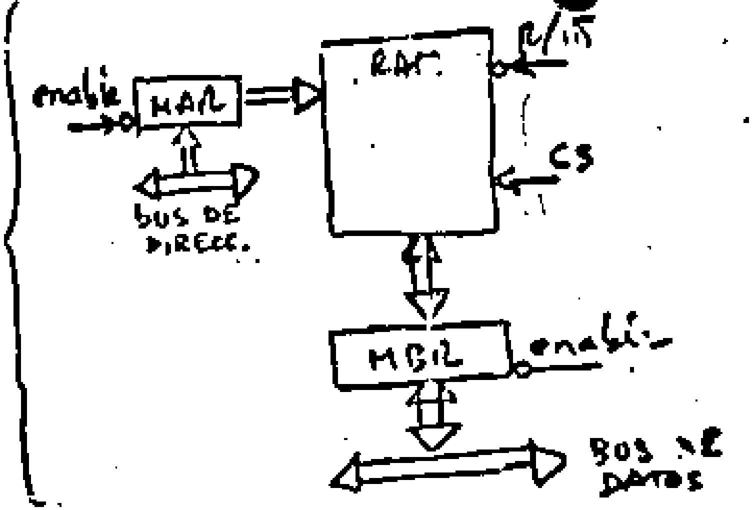
– TÍPICA : 128 × 8



– TÍPICA : 1024 × 8



OPERACIONES DE UNA UNIDAD DE MEMORIA



EJEMPLO DE CONEXIONES

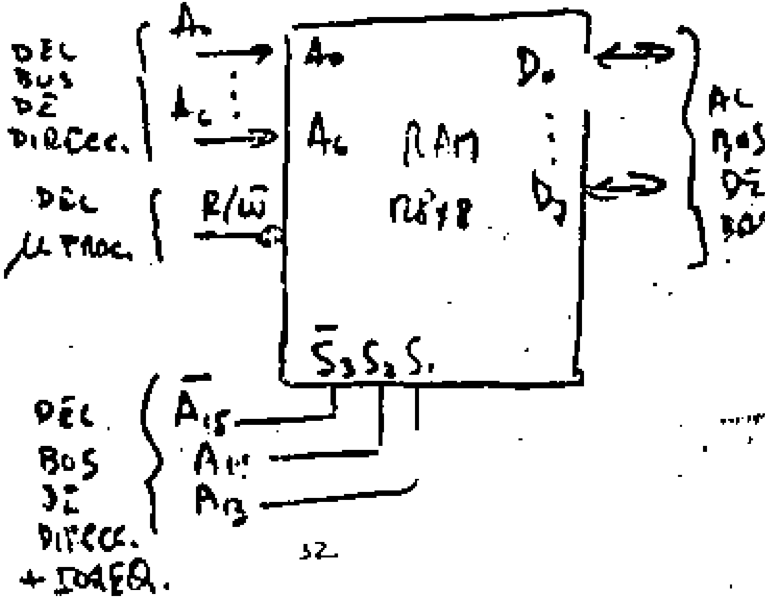
– DIRECC. DE RAM:

1110 0000 0000 0000 $E000_{16}$

1110 0000 0111 1111 $E07F_{16}$

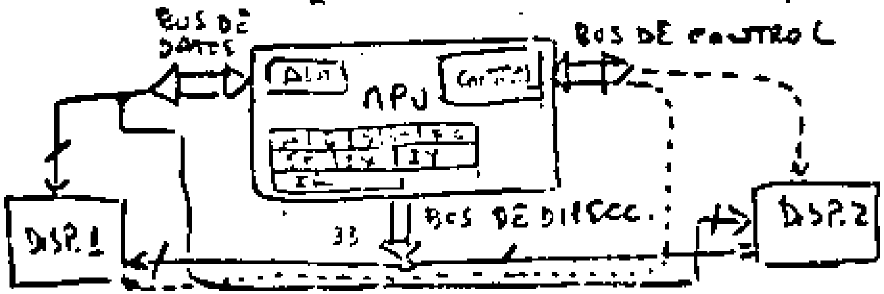
– BUS DE DATOS : 8 BITS

– BUS DE DIRECC. : 16 BITS

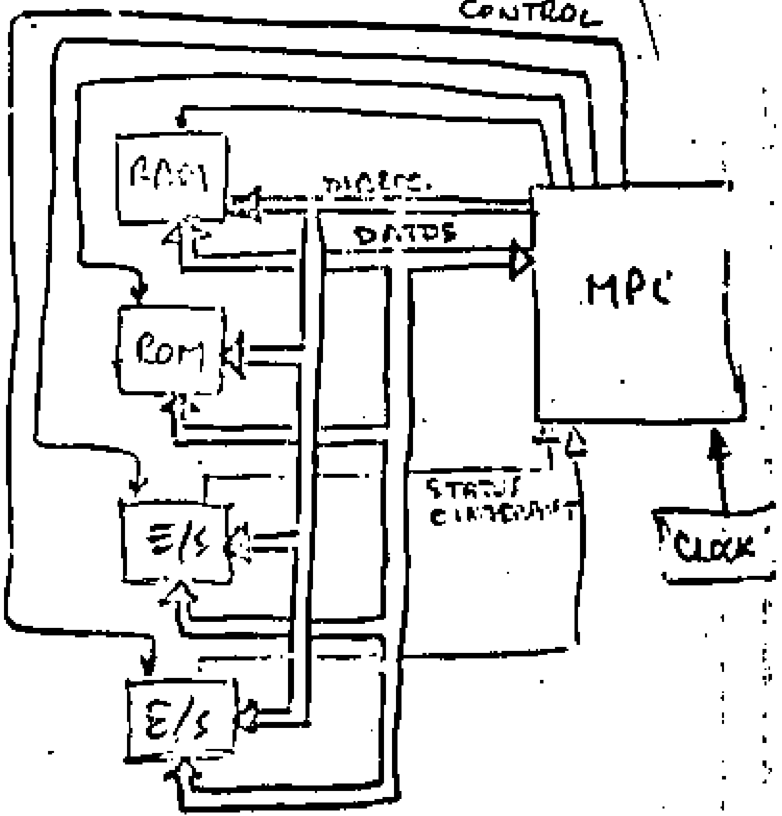


MICROPROCESADOR (MPU)

- EJECUTA OPERACIONES ← ARITMETICAS, LOGICAS, CONTROL
- GENERA LAS SEÑALES NECESARIAS PARA ACTIVAR PERIFERICOS Y SINCRONIZAR ACTIVIDADES
- REQUIERE DE UN RELOJ
- GENERA LAS DIRECCIONES
- COORDINA EL PROCESAMIENTO DE LA INFORMACION
- # LINEAS DE DIRECC ~ 16 (TIPICO)
- # LINEAS DE DATOS ~ 8 (TIPICO)
- # LINEAS DE CONTROL ~ VARIABLE
  - IORQ
  - INT
  - NMI
  - R/W
  - ACK
  - IRQ
- ELEMENTOS {
  - ALU
  - REGISTROS DE USO GRAL. { [illegible]
  - REGISTRO DE STATUS { F
  - REGISTROS INDICE { IX, IY
  - APUNTADOR DE STACK { SP
  - CONTADOR DE INSTRUCC. { PC
  - REGISTRO DE INSTRUCC. { IR



CONFIGURACION TIPICA



DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

INTRODUCCION A LOS MICROPROCESADORES (Z-80)

MODOS DE DIRECCIONAMIENTO

Marzo, 1982

IV.- MODOS DE DIRECCIONAMIENTO

1.- ESQUEMAS DE DIRECCIONAMIENTO.

La unidad central de proceso (CPU) en las computadoras debe realizar las siguientes funciones:

- Obtener y traer de memoria primaria al CPU la siguiente instrucción a ejecutar.
- Entender los operandos, esto es, definir la localización de los operandos necesarios para ejecutar la instrucción y traerlos al CPU.
- Ejecutar la instrucción.

Para llevar a cabo las funciones anteriores el CPU debe contar con la siguiente información:

- El código de operación de la instrucción a ejecutar.
- Las direcciones de los operandos y la del resultado.
- La dirección de la siguiente instrucción a ejecutar.

Existen diferentes soluciones que satisfacen los requerimientos anteriores, los cuales determinan la arquitectura de los procesadores que las utilizan.

Se supondrán operaciones aritméticas en las que se tienen dos operandos y un resultado ya que son las que proporcionan el caso más general.

a) Máquinas de "3+1" direcciones

El formato de instrucción en este esquema de direccionamiento contiene todos los elementos necesitados por el CPU para realizar sus funciones.

Un posible formato de instrucción se muestra en la figura IV.1

CODIGO DE OPERAC.	DIRECCION PRIMER OPERANDO	DIRECCION SEGUNDO OPERANDO	DIRECCION RESULTADO	DIRECCION DE LA SIGUIENTE INSTRUCCION

Palabra de memoria

FIG. IV1

En este caso se tienen cinco campos en el formato de instrucción: Uno para el código de operación que sirve para indicar el tipo de operación a realizar (suma, resta, multiplicación, etc.), tres campos para las direcciones de los operandos y resultado de las operaciones, un campo para indicar la dirección de la siguiente instrucción a ejecutar.

Las instrucciones para ésta máquina podrían ser escritas en forma simbólica en la siguiente forma: ADD A, B, C, D donde ADD representa el código de operación suma y A, B, C y D son nombres simbólicos asignados a localidades de memoria.

Suponiendo que existen las instrucciones suma (ADD), substracción (SUB) y multiplicación (MUL), entonces una posible traducción de la expresión A=(B*C)-(D*E) en FORTRAN a lenguaje simbólico en la máquina de 3+1 direcciones sería:

```
L1:   MUL   B, C, T1, L3
L3:   MUL   D, E, T2, L7
L7:   SUB   T2, T1, A, L8
L8:   Siguiente Instrucción
```

donde T1 y T2 representan localidades temporales usadas para guardar resultados aritméticos intermedios.

Las conclusiones más importantes en este esquema son:

Los programas no necesitan estar almacenados en memoria en forma secuencial ya que el campo de dirección de la siguiente instrucción permite conocer donde fueron almacenados.

Debido a que cada instrucción contiene en forma explícita tres direcciones, no es necesario tener en el CPU hardware para guardar los resultados de las operaciones.

b) Máquinas de "3" direcciones

Considerando que los programas se escriben secuencialmente y que por consiguiente es muy lógico almacenarlos en este mismo orden, se llega a un nuevo esquema de direccionamiento en el cual se sustituyen todos los campos de dirección de la siguiente instrucción por un solo registro dentro del procesador que lleva en forma secuencial y automáticamente la dirección de la siguiente instrucción a ejecutar. Un posible formato de instrucción se muestra en la fig. IV.2 .

Dirección de la sig. inst.

Registro en el procesador

Código de operac.	Dirección primer operando	Dirección segundo operando	Dirección resultado

Palabra n de memoria

FIG. IV.2

Utilizando este esquema de direccionamiento la expresión A=(B*C)-(D*E) en FORTRAN, quedaría expresada como:

```
MUL  B, C, T1
MUL  D, E, T2
SUB  T2, T1, A
Siguiente instrucción
```

Donde se ha suprimido la dirección de la siguiente instrucción ya que ésta es llevada en forma secuencial y automática por un registro del procesador conocido como contador del programa (PC).

Con el esquema de 3 direcciones se logra aprovechar la memoria en forma más eficiente y reducir la longitud de palabra lo que redunda directamente en los costos de la misma.

c) Máquinas de "2" direcciones.

En las operaciones aritméticas no siempre es necesario guardar el resultado en una localidad de memoria y preservar los operandos, por lo que se puede pensar en utilizar uno de ellos para guardar el resultado una vez que la operación se ha efectuado. Las consideraciones anteriores llevan a presentar un posible formato de instrucción en esta máquina, mostrado en la figura IV.3

DIR. DE LA SIG. INST. A EJECUTAR

REG. EN EL PROC.

COD. OP.	DIR. P. OP.	DIR. SEG. OP.

Palabra n de memoria

FIG. IV.3

En este esquema se usará la dirección del segundo operando como la dirección del resultado una vez que la operación se haya efectuado, por lo que el segundo operando será destruido. Así pues la expresión A=(B*C)-(D*E) en FORTRAN, quedaría:

```
MUL  B,C
MUL  D,E
SUB  E,C
ADD  C,A
```

La eliminación del campo de dirección del resultado permite reducir la longitud de la palabra de memoria y los costos de la misma, lo que permite usar este esquema en máquinas medianas y chicas.

d) Máquinas de "1" dirección

Este esquema de direccionamiento permite eliminar de todas las instrucciones el campo de dirección de uno de los operando y sustituirlo por un registro dentro del procesador, el cual contendrá a uno de los operandos. A este registro se le conoce como acumulador. - El formato de instrucción para la máquina de 1 dirección se muestra en la figura IV.4

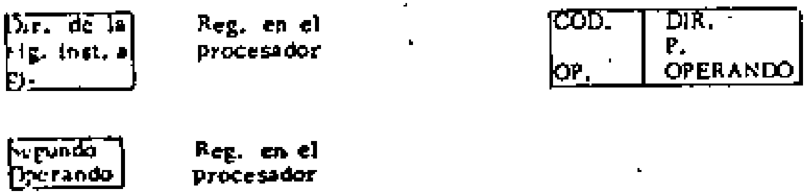


FIG. IV.4

Lo anterior implica la creación de instrucciones que permitan cargar el acumulador con el segundo operando (LAC) y depositar el contenido del acumulador en memoria (DAC).

Es importante hacer notar que todas las operaciones se llevan a cabo implícitamente contra el acumulador y que éste contendrá el resultado de la operación efectuada. La expresión A=(B*C)-(D*E) en FORTRAN, podría traducirse a:

```
LAC   D
MUL   E
DAC   T1
LAC   B
MUL   C
SUB   T1
DAC   A
```

Este esquema de direccionamiento ha sido ampliamente implementado en una gran mayoría de las minicomputadoras, como por ejemplo: PDP-8, -- PDP-15, IBM-1130, IBM-7090 y CDC 3600.

e) Máquinas de "0" direcciones

Este esquema de direccionamiento solo utiliza el campo de código de operación, por lo que es necesario contar con algún mecanismo que implícitamente permita conocer los operandos.

El mecanismo anterior se implementa usando una pila ó stack, el cual se puede pensar como un conjunto de localidades contiguas de

memoria accesadas usando una disciplina UEPS (últimas entradas, primeras salidas). De lo anterior se concluye que en cada momento se tendrá disponible el elemento que se encuentre en el tope del stack.

El formato de instrucción para este esquema de direccionamiento se encuentra en la figura IV.5

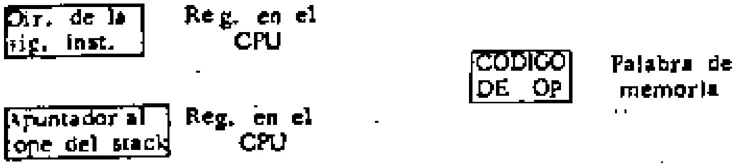


FIG. IV.5

Es necesario contar con instrucciones que permitan meter elementos de memoria al stack (PUSH) y sacar elementos del stack a memoria (POP).

La expresión A=(B*C)-(D*E) en FORTRAN, podría expresarse como:

FIG. IV-6

PUSH D

PUSH E

MUL

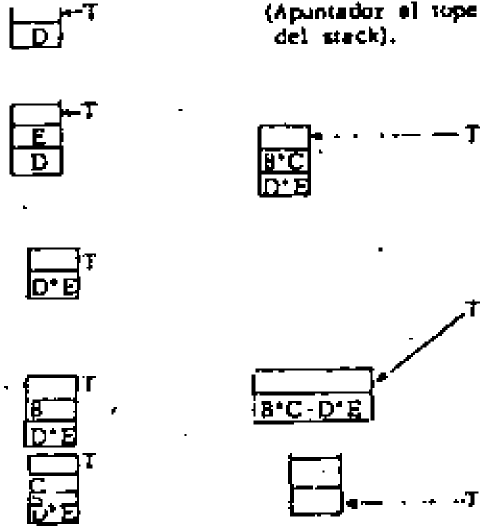
PUSH B

PUSH C

MUL

SUB

POP A



En la fig. IV.6 se ilustra el estado del stack después de cada una de las inst. anteriores.

Se puede concluir que el conjunto de instrucciones de la máquina no está formado solamente por instrucciones de cero direcciones ya que también se requieren instrucciones de una dirección para meter y sacar elementos al stack.

Se requiere un registro en el procesador que apunte al tope del stack y se elimine el acumulador ya que el resultado de las operaciones -- también quedará en el stack.

2.- METODOS DE DIRECCIONAMIENTO

En las máquinas de una sola dirección el formato de las instrucciones que hace referencia a memoria consta de dos campos: el campo de código de operación y el campo de dirección del operando. Si suponemos que el campo de dirección consta de n bits, entonces la máxima capacidad de memoria direccionable será 2^n localidades. Lo anterior puede resultar bastante drástico en el caso de las minicomputadoras ya que, por lo general tienen palabras de 12 ó 16 bits y si se asignan cuatro de ellos al campo de código de operación solo se pueden direccionar $2^8 = 256$ localidades de memoria en el caso de palabras de 12 bits ó $2^{12} = 4096$ localidades de memoria en el caso de palabras de 16 bits, lo cual resulta insuficiente para la gran mayoría de las aplicaciones.

Lo anterior ha ocasionado diferentes modos de direccionamiento, en los cuales el campo de dirección sirve para calcular la dirección efectiva del operando, logrando una mayor capacidad de memoria direccionable.

a) Inmediato

En este caso el operando puede estar contenido directamente en el campo de dirección ó en la localidad de memoria siguiente a la instrucción.

Será necesario dedicar un bit de la palabra para saber como se debe interpretar la instrucción.

b) Directo

Existe direccionamiento directo cuando el campo de dirección de la instrucción contiene la dirección del operando ó cuando este campo combinado con algún registro ó palabra de memoria generan la dirección del operando.

b.1) Usando página cero

Uno de los esquemas más comunes de organización de memoria, divide ésta en n páginas de longitud fija, donde n dependerá del tamaño de la memoria y del tamaño de las páginas.

Las máquinas que usan estos esquemas generalmente usan la página cero con propósitos especiales, como son: manejo de interrupciones, traps, localidades autoincrementables, etc.

La forma de indicar si el contenido del campo de dirección se refiere a la página cero, es usando un bit para este propósito, p. ej. si este bit es cero el campo de dirección apunta a una localidad en la página cero.

b.2) Usando página actual

Si el bit de página está en uno, se asume que el campo de dirección apunta a una localidad en la página en la que se encuentra la instrucción. A esta página se le conoce como página actual.

La dirección del operando se determina sumando los bits de orden superior del PC al campo de dirección de la instrucción.

b.3) Relativo al PC

En este modo de direccionamiento el contenido del campo de dirección de la instrucción, interpretado como un entero con signo, se suma al PC para obtener la dirección del operando.

b.4) Relativo a un registro índice

El contenido del campo de dirección de la instrucción, interpretado como un entero con signo, se suma al contenido de un registro índice para obtener la dirección del operando. En caso de existir más de un registro índice es preciso asignar los bits necesarios para su identificación.

c) Indirecto

En el direccionamiento indirecto el campo de dirección de la instrucción contiene un apuntador a la dirección del operando ó este campo combinado con algún registro ó palabra de memoria genera un apuntador a la dirección del operando.

Mediante un bit en la instrucción se puede saber si el direccionamiento usado es directo ó indirecto.

c.1) Usando página cero

El campo de dirección de la instrucción apunta a una localidad en la página cero. A su vez ésta localidad contiene la dirección del operando.

c.2) Usando página actual

El campo de dirección de la instrucción apunta a una localidad en la página actual. Esta localidad contiene la dirección del operando.

c.3) Relativo al PC

El contenido del campo de dirección de la instrucción, interpretado como un entero con signo, se suma al PC para obtener la dirección del apuntador al operando.

c.4) Relativo a un registro índice

El contenido del campo de dirección de la instrucción, interpretado como un entero con signo, se suma al contenido de un registro índice para obtener la dirección del apuntador al operando.

La combinación de todos los métodos de direccionamiento anteriores con registros de propósito general, permiten lograr modos de direccionamiento bastante poderosos. Cuando se usan los registros de propósito general, el campo de dirección de la instrucción especifica que registro se usa y como se interpreta la información que contiene.

3.- DIRECCIONAMIENTO EN Z-80

El microprocesador Z-80 es una máquina de una dirección en la que los diferentes modos de direccionamiento son usados por grupos de instrucciones y no se aplican de una forma general a todo el conjunto de instrucciones.

a) Implícito

En este modo de direccionamiento el operando no se define en forma explícita ya que el formato de instrucción es fijo y en los códigos de operación se especifica implícitamente sobre qué registros del procesador actúan las instrucciones, por lo que el usuario no puede alterarlo de ninguna manera.
Los grupos de instrucciones, que utilizan este modo de direccionamiento son: carga de 8 bits; carga de 16 bits; intercambio, transferencia de bloques y búsqueda; aritméticas de propósito general y control del CPU.

Ejemplos 1.

b) Inmediato

El operando se encuentra en la localidad de memoria siguiente a la instrucción y se considera que forma parte de la misma. Los valores de los operandos inmediatos en ningún caso podrán exceder la capacidad de representación de un byte. Este modo de direccionamiento se utiliza cuando se desean realizar operaciones con valores constantes.

Los grupos de instrucciones que utilizan este modo de direccionamiento son: carga de 8 bits; aritméticas y lógicas de 8 bits y entrada/salida.

Ejemplos 2.

c) Inmediato extendido

El operando se encuentra en los dos bytes (16 bits) siguientes al código de operación de la instrucción. El primer byte contiguo al código de operación es el menos significativo y el siguiente es el más significativo.

Este modo de direccionamiento es usado por algunas instrucciones de carga de 16 bits.

Ejemplos 3.

d) Registro

El formato de instrucción contiene un campo de dirección de operando donde se especifica cual de los registros del CPU será utilizado como operando.

Los grupos de instrucciones que utilizan este modo de direccionamiento son: carga de 8 bits; carga de 16 bits; aritméticas y lógicas de 8 bits; aritméticas y lógicas de 16 bits; rotaciones y desplazamientos; encendido y apagado de bits; entrada/salida.

Ejemplos 4.

e) Registro indirecto

En este modo de direccionamiento un par de registros (16 bits) contiene la dirección de memoria en la que se encuentra el operando.

Es utilizado por los grupos de instrucciones de carga de 8 bits; intercambio, transferencia de bloques y búsqueda; rotaciones y desplazamientos; prendido y apagado de bits; saltos, llamadas y regreso de subrutinas; entrada/salida.

Ejemplos 5.

f) Extendido

La dirección del operando está contenida dentro del campo de operando de la instrucción. El campo de dirección tiene una longitud de 16 bits por lo que la máxima capacidad de memoria direccionable es de 64 K bytes.

Este modo de direccionamiento es utilizado por los grupos de instrucciones de carga de 8 bits; carga de 16 bits; saltos, llamadas y regreso de subrutinas.

Ejemplos 6.

g) Modificado de página cero

En este modo de direccionamiento el campo de dirección del operando se refiere a una localidad de memoria dentro de la página cero. Este campo de dirección consta de 3 bits y para su correcta interpretación se multiplica por 08H, obteniéndose de esta forma la referencia a las localidades deseadas.

Este modo de direccionamiento se utiliza exclusivamente por la instrucción RST.

Ejemplos 7.

h) Relativo

La dirección del operando se determina sumando al contador del programa al contenido del byte siguiente al código de operación de la instrucción.

El desplazamiento anterior se interpretará como un número en complemento a dos, con lo que se logra un rango de direccionamiento de -126 a +129 localidades relativas al contador del programa.

Este modo de direccionamiento es usado por el grupo de instrucciones de salto, llamada y regreso de subrutinas.

Ejemplos 8.

i) Indexado

La dirección del operando se determina sumando al registro de índice especificado el contenido del byte de desplazamiento.

El desplazamiento se interpreta como una cantidad en complemento a dos, con lo que se logra un rango de direccionamiento de -128 a +127 localidades relativas al registro de índice.

Los grupos de instrucciones que utilizan este modo de direccionamiento son: carga de 8 bits; aritméticas y lógicas de 8 bits; rotaciones y desplazamientos; encendido y apagado de bits; saltos, llamada y regreso de subrutinas.

Ejemplos 9.

j) Bit

Este modo de direccionamiento permite prender o apagar un bit dentro de un operando seleccionado, usando los modos antes descritos.

Ejemplos 10.

ING. LUIS G. CORDERO BORBOA
AGOSTO-79

E J E M P L O S

Se asumirá que todos los ejemplos siguientes utilizan el sistema de numeración hexadecimal.

Ejemplos 1.

```
          ; MODOS DE DIRECCIONAMIENTO DEL MICROPROCESADOR Z-80
          ; PROGRAMA GRABADO EN CASSETE CON EL NOMBRE DE  -
          ; [illegible]
          ;
          ;
          ; DIRECCIONAMIENTO IMPLICITO.
          ;
0000 ED5F         LD      A,R
          ;
          ; CARGA EN EL REGISTRO A EL CONTENIDO DEL REGISTRO
          ; DE REFRESCAMIENTO R.
          ;
          ;
0002 2F           CPL
          ;
          ; REALIZA EL COMPLEMENTO LOGICO DEL CONTENIDO DEL
          ; ACUMULADOR Y LO DEJA EN EL MISMO REGISTRO
          ;
          ;
   DD23           INC     IX
          ;
          ; EL CONTENIDO DEL REGISTRO DE INDICE IX SE IN--
          ; CREMENTA EN UNO
          ;
          ;*
```

Ejemplos 2.

```
          ;
          ; DIRECCIONAMIENTO INMEDIATO
          ;
   C634           ADD     A,34H
          ;
          ; SUMA AL CONTENIDO DEL REGISTRO ACUMULADOR A, EL
          ; DATO 34H Y DEJA EL RESULTADO EN EL MISMO RE--
          ; GISTRO
          ;
          ;
0007 E610         AND     10H
          ;
          ; REALIZA LA OPERACION LOGICA AND ENTRE EL CONTE--
          ; NIDO DEL REGISTRO A Y EL DATO 10H DEJANDO EL -
          ; RESULTADO EN EL MISMO REGISTRO
          ;
          ;
```

Ejemplos 3.

```
              ;
              ; DIRECCIONAMIENTO INMEDIATO EXTENDIDO
              ;
0009 FD212020         LD      IY,2020H
              ;
              ; CARGA EN EL REGISTRO DE INDICE IY EL DATO 2020H
              ;
              ;
000D 213F12           LD      HL,123FH
              ;
              ; CARGA EL REGISTRO PAR HL CON EL DATO 123FH
              ;
              ;
```

Ejemplos 4.

```
          ;
          ; DIRECCIONAMIENTO DE REGISTRO
          ;
0010 4F           LD      C,A
          ;
          ; CARGA EL REGISTRO C CON EL CONTENIDO DEL REGIS-
          ; TRO A
          ;
          ;
0011 80           ADD     A,B
          ;
          ; SUMA AL CONTENIDO DEL REGISTRO A EL CONTENIDO
          ; DEL REGISTRO B Y DEJA EL RESULTADO EN EL REGIS-
          ; TRO A
          ;
          ;
0012 ED52         SBC     HL,DE
          ;
          ; SUBSTRAE DEL CONTENIDO DEL REGISTRO HL, EL CONTE-
          ; NIDO DE LOS REGISTROS DE Y ACARREO CY, DEJANDO
          ; EL RESULTADO EN EL REGISTRO HL
          ;
          ;
```

Ejemplos 5.

```
                ;
                ; DIRECCIONAMIENTO DE REGISTRO INDIRECTO
                ;
0014 0A                 LD      A,(BC)
                ;
                ; CARGA EL REGISTRO A CON EL CONTENIDO DE LA LO-
                ; CALIDAD DE MEMORIA APUNTADA POR EL REGISTRO PAR
                ; BC
                ;
                ;
0015 34                 INC     (HL)
                ;
                ; INCREMENTA EN UNO EL CONTENIDO DE LA LOCALIDAD
                ; DE MEMORIA APUNTADA POR EL REGISTRO PAR HL
                ;
                ;
0016 12                 LD      (DE),A
                ;
                ; DEPOSITA EL CONTENIDO DEL ACUMULADOR EN LA LOCA-
                ; LIDAD DE MEMORIA APUNTADA POR EL REGISTRO PAR DE
                ;
                ;
```

Ejemplos 6.

```
                ;
                ; DIRECCIONAMIENTO EXTENDIDO
                ;
0017 3A2010             LD      A,(1020H)
                ;
                ; CARGA EL ACUMULADOR CON EL CONTENIDO DE LA LOCA-
                ; LIDAD DE MEMORIA 1020H
                ;
                ;
001A FD220400           LD      (0004H),IY
                ;
                ; DEPOSITA EL CONTENIDO DEL REGISTRO DE INDICE EN
                ; LAS LOCALIDADES DE MEMORIA 0004H (BYTE BAJO) Y
                ; 0005H (BYTE ALTO).
                ;
                ;
```

Ejemplos 7.

```
                ;
                ; DIRECCIONAMIENTO MODIFICADO DE PAGINA CERO
                ;
001E CF                 RST     08H
                ;
                ; EFECTUA UN SALTO INCONDICIONAL A LA LOCALIDAD DE
                ; MEMORIA 08H DESPUES DE HABER GUARDADO EN EL --
                ; STACK EL CONTENIDO DEL CONTADOR DEL PROGRAMA
                ;
                ;
```

Ejemplos 8.

```
                ;
                ; DIRECCIONAMIENTO RELATIVO
                ;
001F 2804               JR      Z,25H
                ;
                ; SI LA BANDERA Z=1, AL CONTADOR DEL PROGRAMA SE LE
                ; SUMA EL VALOR 04H CON LO QUE SE EFECTUARA UN SAL-
                ; TO A LA LOCALIDAD DE MEMORIA 25H
                ; SI LA BANDERA Z=0 SE CONTINUARA EJECUTANDO LA SI-
                ; GUIENTE INSTRUCCION DEL PROGRAMA
                ;
                ;
0021 30F4               JP      NC,17H
                ;
                ;SI LA BANDERA C=0, AL CONTADOR DEL PROGRAMA SE LE
                ; SUMA EL VALOR F4H CON LO QUE SE EFECTUARA UN SAL-
                ; TO A LA LOCALIDAD DE MEMORIA 17H
                ; SI LA BANDERA C=1 SE CONTINUARA EJECUTANDO LA -
                ; SIGUIENTE INSTRUCCION DEL PROGRAMA
                ;
                ;L
```

Ejemplos 9.

```
                 ;
                 ; DIRECCIONAMIENTO  INDEXADO
                 ;
0023 FD364313            LD      (IY+43H),13H
                 ;
                 ; EL  DESPLAZAMIENTO  43H  SE  SUMA  AL CONTENIDO DEL RE-
                 ; GISTRO  IY  PARA DETERMINAR, LA  DIRECCION  EFECTIVA A
                 ; DONDE  SE  DEPOSITARA  EL  DATO  13H
                 ;
                 ;
0027 DD8621              ADD     A,(IX+21H)
                 ;
                 ; EL  DESPLAZAMIENTO  21H  SE  SUMA  AL  CONTENIDO  DEL
                 ; REGISTRO  IX  PARA  DETERMINAR  LA  DIRECCION  DEL  O-
                 ; PERANDO  QUE SERA  SUMADO  AL  REGISTRO  A  EL  RESUL-
                 ; TADO  QUEDA  EN  EL  REGISTRO  A
                 ;
                 ;
002A DD3407              INC     (IX+07H)
                 ;
                 ; EL  DESPLAZAMIENTO  07H  SE  SUMA  AL  CONTENIDO  DEL
                 ; REGISTRO  IX  PARA  DETERMINAR  LA  DIRECCION  DE  LA
                 ; LOCALIDAD  DE  MEMORIA  CUYO  CONTENIDO  SE  INCREMEN-
                 ; TA  EN  UNO.
                 ;
                 ;
```

Ejemplos 10.

```
                 ;
                 ; DIRECCIONAMIENTO  DE  BIT.
                 ;
002D CBC7                SET     0000H,A
                 ;
                 ; ENCIENDE  EL  BIT  0  DEL  REGISTRO  A
                 ;
                 ;
002F CBAE                RES     05H,(HL)
                 ;
                 ; APAGA  EL  BIT  5  DE  LA  LOCALIDAD  DE  MEMORIA  DI-
                 ; RECCIONADA  POR  EL  REGISTRO  HL.
```

ING. LUIS G. CORDERO BORBOA
AGOSTO-79

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

INTRODUCCION A LOS MICROPROCESADORES (Z-80)

PROGRAMA DE RELOJ DIGITAL

Marzo, 1982

Programa de Reloj Digital

Reloj del sistema = 1.9968 MHz
1.9968 MHz /256=7800 Hz; contando 200 ciclos por interrupción y 39 - interrupciones por segundo para incrementar segundos.

Programa Principal

```
2000    ED 5E           IM 2            Modo de interrupción 2
        3E 20           LDA, 20         Byte más significativo
        ED 47           LDI, A          Registro I
        3E 1A           LDA, 1A         Byte menos significativo
        D3 84           OUT (84),A      Canal 0 CTC
        3F A5           LDA,A5          Permitir int. .reloj/256
        D3 85           OUT(85),A       Canal 1 CTC
        3E C8           LDA,C8          Constante de tiempo = 200
2010    D3 85           OUT (85), A
        D9              EXX             Intercambiar registros
        06 27           LDB,39          Número de iterrupciones
        D9              EXX
        FB              EI              Se permiten interrupciones
        C3 9F 00        JP RESTAR       Monitor despliegue
201A    2020                            Vector de interrupción
```

Programa de servicio de interrupciones

```
201C                                    Segundos en BCD
201D                                    Minutos en BCD
201E                                    Horas en BCD

2020  08                EX AF,A'F'      Intercambiar registros
      D9                EXX
      05                DEC B           Contar interrupciones
      20 5C             JR NZ RET       Continuar
      06 27             LD B,27         39 interrupciones por seg.
      3A 1C 20          LD A,(201C)     Segundos
      3C                INC A
      27                DAA             Ajuste decimal
      FE 60             CP 60           60 segundos.
      38 26             JR C            No
2030  AF                XOR A           Si-borrar A
```

```
      32 1C 20                LD (201C),A
      3A 1D 20                LDA,(201D)       Minutos
      3C                      INC A
      27                      DAA
      FE 60                   CP 60            60 minutos ?
      38 14                   JR C             No
      AF                      XOR A            Si-borrar A
      32 1D 20                LD (201D),A
2041  3A 1E 20                LDA,(201E)       Horas
      3C                      INC A
      27                      DAA
      FE 13                   CP 13            13 horas ?
      38 02                   JR C             No.
204A  3E 01                   LDA,01
      32 1E 20                LD (201E),A
      18 08                   JR 08
2051  32 1D 20                LD (201D),A
      18 03                   JR 03
      32 1C 20                LD (201C),A
      3A 1C 20                LDA,(201C)       Segundos
      DD 21 FB 23             LD IX,23FB       LED buffer No.5
2060  CD 86 20                CALL FMT         Format para LED buffer
      3A 1D 20                LDA, (201D )     Minutos
      DD 21 F9 23             LD IX, 23F9      LED buffer No. 3
      CD 86 20                CALL FMT
      3A 1E 20                LDA, (201E )     Horas
2070  DD 21 F7 23             LD IX,23E7       LED buffer No. 1
      CD 86 20                CALL FMT
      3A F7 23                LDA,(23E7)       LED buffer No. 1=cero ?
      20 05                   JR NZ,05
      3E 10                   LDA,10           Apagar LED No. 1
      32 F7 23                LD (23F7),A
2081  08                 RET  EX AF,A'F'       Intercambiar registros
      D9                      EXX
      FB                      EI               Permitir interrupciones
      ED 4D                   RETI             Retorno
2086  4F                 FMT  LD C,A           Subrutina para format
      E6 0F                   AND 0F           4 bits menos significativo
```

```
     DD 77 01        LD (IX+1),A     LED buffer
     CB 39           SRL C           4 bits más significativo
     CB 39           SRL C
2090 CB 39           SRL C
     CB 39           SRL C
     DD 71 00        LD (IX+0),C     LED buffer
     C9              RET             Retorno
```

Programa para Control   un Motor de Paso Superior HS-25

Cuenta Inicial: Registros BC , Cuenta Final: Registros DE

Bobinas del motor conectados a PIO PB0-PB3 por amplificador de potencia -(transistores Darlington).

```
2000  3E 0F                  LD A,0F          Modo Salidas
      D3 83                  OUT (83),A
      62           COMP      LD H,D
      6B                     LD L,E
      AF                     XOR A            Borrar acarreo
      ED 42                  SBC HL,BC
      FA 11 20               JP M 2011        Dirección negativa ?
      28 25                  JR Z STOP        Igual?
      03                     INC BC           Positiva
      18 01                  JR 01
2011  0B                     DEC BC           Negativa
      79                     LD A,C           Byte menos significativo
      E6 03                  AND 03           Dos bits
      62                     LD H, D
      6B                     LD L,E
      EB                     EX DE,HL         Guardar cuenta final
      16 00                  LD D,0
      5F                     LD E,A
      DD 21 39 20            LD IX,TABLA      Rotacion de bobinas
      DD 19                  ADD IX,DE        Posición en la tabla
2021  DD 7E 00               LD A, (IX+0)     Codigo de bobinas
      D3 81                  OUT (81),A       PIO lado B
      EB                     EX DE, HL        Reponer cuenta en DE
      26 01                  LD H,1           Byte más significativo
      2E 00                  LD L,0           Byte menos significativo
      2D                     DEC L            De espera
      20 FD                  JR NZ ESP
      25                     DEC H
      20 FA                  JR NZ ESP
2031  18 D1                  JR COMP          Otro paso
      AF           STOP      XOR A            Salidas Cero
      D3 81                  OUT (81),A
      C3 AE 00               JP RESTRI        Retorno al monitor
2039  0A 06 05 09 TABLA
```

Programa para Contar Interrupciones

Interrupciones: pulsos hacia arriba en ASTRB. Despliegue número 1-99 en los LEDs.

```
2200    ED 5E              IM 2
        3E F8              LDA F8              Byte menos significativo
        D3 82              OUT (82),A          del vector de inter.
        3E 4F              LDA, 4F             Modo entradas
        D3 82              OUT(82),A           Lado A control
        3E 87              LDA,87              Se permiten interrupciones
        D3 82              OUT (82),A
        3E 07              LDA, 07             Byte más significativo
2210    ED 47              LDI,A               en registro I
        FB                 EI                  MPU acepta interrupciones
        C3 9F 00           JP RESTAR           Retorno al monitor

07F8    06 23                                  Vector en ROM

2306    C3 20 22           JP 2220

2220    08                 EX AF,A'F'          Intercambiar registros
        D9                 EXX
        3A FC 23           LDA, (DISMEM + 5)   LED Buffer No. 6
        E6 0F              AND 0F
        3C                 INC A
        FE 0A              CP A                Diez o más?
        38 0F              JR C, CONT          No.
        AF                 XOR A               Si- borrar A
        32 FC 23           LD(DISMEM + 5),A
2230    3A FB 23           LDA,(DISMEM + 4)    Incrementar 10's
        E6 0F              AND 0F
        3C                 INC A
        32 FB 23           LD(DISMEM + 4),A
        18 03              JR 03
        32 FC 23           LD(DISMEM +5) ,A
        21 00 80           LD HL,8000          Esperar desconexion
2241    2D           ESP   DEC L
        20 FD              JR NZ ESP
```

```
25                 DEC H
20 FA              JR NZ ESP
08                 EX AF,A' F'         Reponer registros
D9                 EXX
FB                 EI                  Permitir interrupciones
ED 4D              RTI                 Retorno
```

Programa para contar frecuencia de una señal externa con CTC

Contar ondas en Canal 1 CLK/TRG por 1/10 seg. usando canal 3 con 4 interrupciones, cada una de $195*256/1.9968x10^6$ = .025 seg.
Para probarlo, usar canal 0 salida ZC.

Programa Principal

```
2060    3E 05          LDA,05            Reloj /16, no interrup.
        D3 84          OUT(84),A         Canal 0
        3E 80          LDA,80            128*16, ciclos =975 Hz
        D3 84          OUT (84),A
2068    ED 5E          IM 2
        3E 20          LDA,20            Byte más significativo
        ED 47          LDI,A             Registro I
        3E 88          LDA,88            Byte menos significativo
2070    D3 84          OUT (84),A        Se escribe en canal 0
        3E A5          LDA,A5            Reloj/256, permitir  int.
        D3 87          OUT (87),A        Canal 3
        3E C3          LDA,C3            Constante de tiempo = 195
        D3 87          OUT (87),A
        3E 55          LDA,55            Modo contador, no int.
        D3 85          OUT (85),A        Canal 1
        AF             XOR A             Borrar A
        D3 85          OUT (85),A        Constante de tiempo
2081    32 FC 23       LD (DISMEM +5),A  Cero en LED No. 6
        D9             EXX               Cargar registros altern.
        0E 00          LDC,00            Cuenta inicial
        06 04          LD B,04           Cuatro interrupciones
        D9             EXX
        FB             EI                Permitir interrupciones
        C3 F4 00       JP DISUP          Monitor despleque
208E    90 20                            Vector de int. canal 3
```

Programa de servicio de interrupciones

```
2090    08                   EX AF,A'F'
        D9                   EXX
        05                   DEC B                 Contar interrupciones
        20 41                JR NZ RET
        06 04                LD B, 04
        DB 85                IN A,(85)             Cuenta nueva
        5F                   LD E, A               Guardar en E
        57                   LD D, A               Contar con D
        AF                   XOR A                 Borrar A y DISMEM
        32 F9 23             LD (DISMEM +2),A
        32 FA 23             LD (DISMEM +3),A
20A2    32 FB 23             LD (DISMEM +4),A
        18 2A                JR CMPR
        3A FB 23     REPT    LD A,(DISMEM +4)
        3C                   INC A
        FE 0A                CP A                  Diez o mayor?
        38 1E                JR C                  No.-- continuar
        AF                   XOR A                 Si --borrar A
20B0    32 FB 23             LD (DISMEM +4),A
20B3    3A FA 23             LD A,(DISMEM +3)
        3C                   INC A
        FE 0A                CP A                  Dies o mayor?
        38 0D                JR C                  No - continuar
        AF                   XOR A                 Si - borrar A
        32 FA 23             LD (DISMEM +3),A
        3A F9 23             LD A, (DISMEM +2)
20C2    3C                   INC A
        32 F9 23             LD (DISMEM +2),A
        18 08                JR 08
        32 FA 23             LD(DISMEM +3),A
        18 03                JR 03
        32 FB 23             LD (DISMEM +4),A
20D0    14                   INC D
        7A           CMPR    LDA,D
        B9                   CP C                  Igual a cuenta previa?
        20 D2                JR NZ,REPT            Repetir
        4B                   LD C,E                Guardar nueva cuenta
        08           RET     EX AF,A'F'            Intercambiar registros
        D9                   EXX
        FB                   EI                    Permitir interrupciones
        ED 4D                RTI                   Retorno
```

Programa para Síntesis de Musica con Z-80 Starter Kit

Salida: onda cuadrata en U14 pata 2 proveniendo del CTC canal 1.
La melodia se almacena en una tabla de frecuencias (periodos de tonos - en unidades de 16 micro segundos) y duraciones en unidades de 33 milisegundos. Una frecuencia de 00 en la tabla indica silencio.

Programa Principal

```
2200    21 04 22     COM   LD  HL, TABLA       Dirección de la tabla
        7E          RECOM  LD A,(HL)           Nota
        B7                 OR A                Checar para cero
        CC 20 22           CALL Z, SILEN
        C4 28 22           CALL, NZ, NOTA
        23                 INC HL
        56                 LD D,(HL)           Duración
        CD 30 22           CALL ESP            Subrutina de espera
2210    CD 20 22           CALL SILEN          Pausa entre notas
        16 01              LD D, 01
        CD 30 22           CALL ESP
        23                 INC HL              Siguiente nota
        18 E8              JR RECOM
```

Subrutina de silencio

```
2220    3E 03       SILEN LD A, 03             Apagar CTC
        D3 85             OUT (85),A           Canal 1
        AF                XOR A                Regresar con cero
        C9                RET                  Retorno
```

Subrutina de Nota

```
2228    3E 05    NOTA    LD A, 05              Dividir por 16
        D3 85            OUT (85),A            Canal 1
        7E               LDA,(HL)              Sacar período
        D3 85            OUT (85),A
        C9               RET                   Retorno
```

Subrutina de Espera

```
2230      06 10          ESP   LD B, 10            16 veces por el bucle
          1E 00                LD E, 00
          1D           BUCLE   DEC E
          20 FD                JR NZ BUCLE
          10 FB                DJNZ BUCLE          Registro B
          15                   DEC D               Duración de la tabla
          20 F4                JR NZ ESP
          C9                   RET                 Retorno
```

Tabla de la melodia

```
2240  9F 08 D5 04  D5  04 C8  08 D5  10  A9 08  9F 10 00 00 00 00
```

Programa para Salida Audio — PIO PAO

440 Hz (=la) = 1.13636 mseg. por 1/2 ciclo
Reloj del Starter Kit= 1.9968 MHz
2269.1 ciclos = 1.13636 mseg.

```
2260    3E 0F             LDA 0F          Modo de salidas
        D3 82             OUT (82),A      Canal A control
        AF                XOR A           Registro A = 0
        D3 80     REP     OUT(80),A       Canal A datos
        06 8C             Ld B,8C         140 veces por el bucle
        05                DEC B
        20 FD             JR NZ
        3C                INC A           Cambiar bit 0
        18 F6             JR REP
```

Cuenta : 16*139 + 11 + 11 + 7 +4 +12 =2269 ciclos.

Programa para Salida Audio --CTC Canal 1

Starter Kit ZC1 = pata 8 del CTC = pulsos (1 microseg.) a 878.9 Hz con constante de tiempo de 142*16 = 2272 ciclos del reloj. U14 pata 3= onda cuadrata a 439.4Hz.

```
2270    3E 05         LDA,05          Timer modo, ÷ 16, no int.
        D3 85         OUT(85),A       Canal 1 control
        3E 8E         LDA,8E          Constante de tiempo
        D3 85         OUT(85),A
        C3 F4 00      JP 00F4         Retorno al monitor
```

Programa para contar ángulos usando convertidor incremental

Rutina de servicio de interrupciones

```
2000    08            EX AF, A'F'       Interrupción lado A
        DB 81         IN A, (81)        Leer lado B
        18 03         JR 03
2005    08            EX AF, A'F'       Interrupción lado B
        DB 80         IN A,(80)         Leer lado A
        D9            EXX               Guardar registros
        E6 C0         AND C0            Examinar bits 6,7
        B8            CP B              B=valor previo
        FA 1420       JP M 2014         Negativo ?
        28 09         JR Z              Cero?
2011    13            INC DE            Ajustar ángulo +
        18 01         JR 01
2014    1B            DEC DE            Ajustar ángulo -
        47            LD B,A            Guardar bits 6,7
        ED 53 9C 20   LD(209C),DE       Guardar ángulos
        08            EX AF,A'F'        Intercambiar registros
        D9            EXX
        FB            EI                Permitir interrupciones
        ED 40         RETI              Retorno
```

Programa Principal

```
2020    ED 5E         IM 2
        3E 9B         LDA,9B            Byte menos significativo
```

```
      D3 82          OUT(82),A          Vector lado A
      3E 9A          LDA, 9A
      D3 83          OUT (83),A         Vecotr lado B
      3E 4F          LDA,4F             Modo entradas
      D3 82          OUT(82),A          Lado A
      D3 83          OUT(83),A          Lado B
2030  3E 87          LDA,87             Permitir interrupciones
      D3 82          OUT(82),A          Lado A
      D3 83          OUT(83),A          Lado B
      3E 20          LDA,20             Byte más significativo
      ED 47          LDI, A             Registro I
      FB             EI                 Permitir interrupciones
      DD 21 A0 20 FMT LD IX, 20A0       LED buffer
      3A 9D 20       LD A,(209D)        Byte más significativo
2042  E6 0F          AND 0F             4 bits menos signif.
      DD 77 01       LD (IX+1),A
      3A 9D 20       LD A,(209D)
204A  CB 3F          SRL A              Correr bits 4-7
      CB 3F          SRL A
      CB 3F          SRL A
2050  CB 3F          SRL A
      DD 77 00       LD(IX),A           4 bits más significativo
      3A 9C 20       LDA,(209C)         Byte menos significativo
      E6 0F          AND 0F
      DD 77 03       LD (IX+3),A        Bits menos significativo
      3A 9C 20       LDA,(209C)
2060  CB 3F          SRL A
      CB 3F          SRL A
      CB 3F          SRL A
      CB 3F          SRL A
      DD 77 02       LD (IX+2),A        4 bits más signif.
      21 A0 20       LD HL,20A0         LED buffer
      06 08          LD B,08            LED No. 3
2070  5E        DES  LD E,(HL)
      16 00          LD D,O
      3E 00          LD A, O
      D3 8C          OUT (DIGLH),A      Apagar LEDs
      DD 21 A6 07    LD IX, 07A6        Tabla de segmentos
      DD 19          ADD IX,DE          Posición en la tabla
```

```
      DD 7E 00       LD A,(IX+0)        Codigo de segmentos
2080  D3 88          OUT (SEGLH),A      LED segmentos
      78             LD A,B             B indica LED número
      D3 8C          OUT (DIGLH),A      LED catodos
      1E 2D          LD E,2D            0.6 mseg. espera
      1D             DEC E
      3E 00          LD A,O
      BB             CP E
      20 FA          JR NZ
      3E 01          LD A,01
      B8             CP B               Terminado?
2090  28 A9          JR Z,FMT           Recomenzar
      23             INC HL             Continuar
      CB 38          SRL B              Siguiente LED
      18 D9          JR DES
      00
2098  00 20                             Puerto A vector
209A  05 20                             Puerto B vector
```

DIVISION DE EDUCACION CONTINUA
FACULTAD DE INGENIERIA U.N.A.M.

INTRODUCCION A LOS MICROPROCESADORES (Z-80)

DATOS TECNICOS

Marzo, 1982

1

Z80™-PIO Z80™A-PIO

Product Specification

The Zilog Z-80 product line is a complete set of microcomputer components, development systems and support software. The Z-80 microcomputer component set includes all of the circuits necessary to build high-performance microcomputer systems with virtually no other logic and a minimum number of low cost standard memory elements.

The Z-80 Parallel I/O (PIO) Interface Controller is a programmable, two port device which provides TTL compatible interfacing between peripheral devices and the Z80-CPU. The Z80-CPU configures the Z80-PIO to interface with standard peripheral devices such as tape punches, printers, keyboards, etc.

Structure

- N-Channel Silicon Gate Depletion Load technology
- 40 Pin DIP
- Single 5 volt supply
- Single phase 5 volt clock
- Two independent 8-bit bidirectional peripheral interface ports with "handshake" data transfer control

Features

- Interrupt driven "handshake" for fast response
- Any one of the following modes of operation may be selected for either port:
  Byte output
  Byte input
  Byte bidirectional bus (available on Port A only)
  Bit Mode
- Programmable interrupts on peripheral status conditions.
- Daisy chain priority interrupt logic included to provide for automatic interrupt vectoring without external logic.
- Eight outputs are capable of driving Darlington transistors.
- All inputs and outputs fully TTL compatible.

PIO Architecture

A block diagram of the Z80-PIO is shown in figure 1. The internal structure of the Z80-PIO consists of a Z80-CPU bus interface, internal control logic, Port A I/O logic, Port B I/O logic, and interrupt control logic. A typical application might use Port A as the data transfer channel and Port B for the status and control monitoring.

The Port I/O logic is composed of 6 registers with "handshake" control logic as shown in figure 2. The registers include: an 8-bit input register, an 8-bit output register, a 2-bit mode control register, an 8-bit mask register, an 8-bit input/output select register, and a 2-bit mask control register. The last three registers are used only when the port has been programmed to operate in the bit mode.

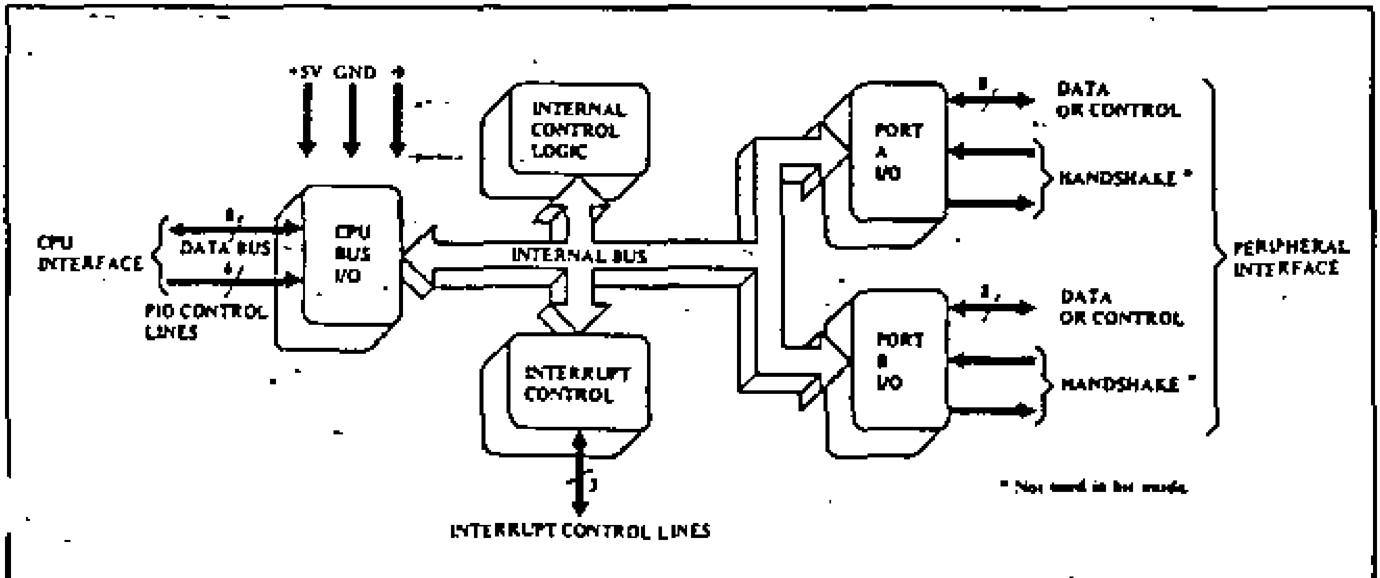


FIGURE 1
PIO BLOCK DIAGRAM

™ Z80 is a registered trademark of Zilog, Inc.

Mode Control Register—2 bits, loaded by CPU to select the operating mode: byte output, byte input, byte bidirectional bus or bit mode.

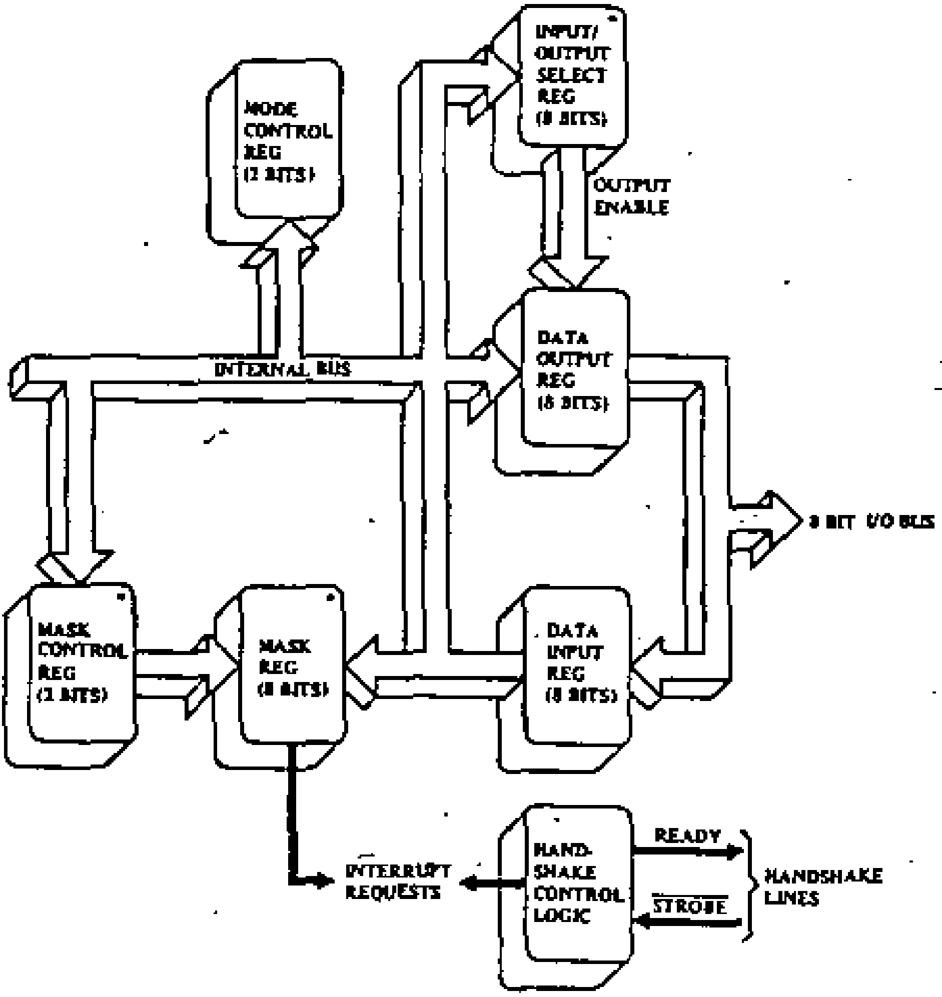
Data Output Register—8 bits, permits data to be transferred from the CPU to the peripheral.

Data Input Register—8 bits, accepts data from the peripheral for transfer to the CPU.

Mask Control Register—2 bits, loaded by the CPU to specify the active state (high or low) of any peripheral device interface pins that are to be monitored and, if an interrupt should be generated when all unmasked pins are active (AND condition) or, when any unmasked pin is active (OR condition).

Mask Register—8 bits, loaded by the CPU to determine which peripheral device interface pins are to be monitored for the specified status condition.

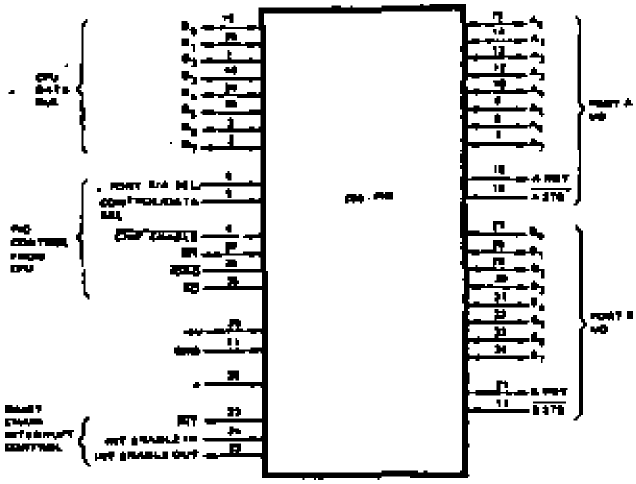
Input/Output Select Register—8 bits, loaded by the CPU to allow any pin to be an output or an input during bit mode operation.



* Used in the bit mode only to allow generation of an interrupt if the peripheral I/O pins go to the specified state.

FIGURE 2
A TYPICAL PORT I/O BLOCK DIAGRAM

Z80-PIO Pin Description



D_7-D_0 Z80-CPU Data Bus (bidirectional, tristate)

B/A Sel Port B or A Select (input, active high)

C/D Sel Control or Data Select (input, active high)

$\overline{CE}$ Chip Enable (input, active low)

Φ System Clock (input)

$\overline{M1}$ Machine Cycle One Signal from CPU (input, active low)

$\overline{IORQ}$ Input/Output Request from Z80-CPU (input, active low)

$\overline{RD}$ Read Cycle Status from the Z80-CPU (input, active low)

IEI Interrupt Enable In (input, active high)

IEO Interrupt Enable Out (output, active high). IEI and IEO form a daisy chain connection for priority interrupt control.

$\overline{INT}$ Interrupt Request (output, open drain, active low)

A_0-A_7 Port A Bus (bidirectional, tristate)

$\overline{A\ STB}$ Port A Strobe Pulse from Peripheral Device (input, active low)

A RDY Register A Ready (output, active high)

B_0-B_7 Port B Bus (bidirectional, tristate)

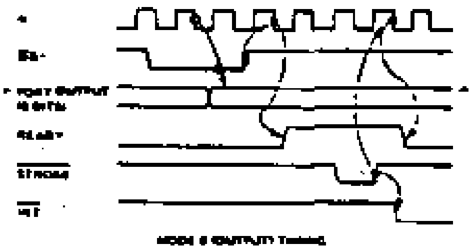
$\overline{B\ STB}$ Port B Strobe Pulse from Peripheral Device (input, active low)

B RDY Register B Ready (output, active high)

Timing Waveforms

OUTPUT MODE

An output cycle is always started by the execution of an output instruction by the CPU. The $\overline{WR}$ pulse from the CPU latches the data from the CPU data bus into the selected port's output register. The write pulse sets the ready flag after a low going edge of Φ, indicating data is available. Ready stays active until the positive edge of the strobe line is received indicating that data was taken by the peripheral. The positive edge of the strobe pulse generates an $\overline{INT}$ if the interrupt enable flip flop has been set and if this device has the highest priority.

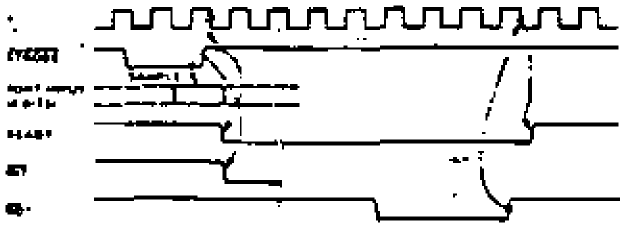


MODE 0 (OUTPUT) TIMING

WR* = RD · CE · C/D · IORQ

INPUT MODE

When $\overline{STROBE}$ goes low data is loaded into the selected port input register. The next rising edge of strobe activates $\overline{INT}$ if interrupt enable is set and this is the highest priority requesting device. The following falling 1ge of Φ resets Ready to an inactive state, indicating that e input register is full and cannot accept any more data until the CPU completes a read. When a read is complete the positive edge of $\overline{RD}$ will set Ready at the next low going transition of Φ. At this time new data can be loaded into the PIO.

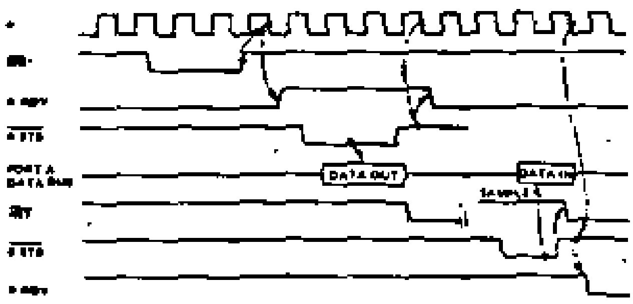


RD* = RD · CE · C/D · IORQ MODE 1 (INPUT) TIMING

Timing Waveforms (continued)

BIDIRECTIONAL MODE

This is a combination of modes 0 and 1 using all four handshake lines and the 8 Port A I/O lines. Port B must be set to the Bit Mode. The Port A handshake lines are used for output control and the Port B lines are used for input control. Data is allowed out onto the Port A bus only when A STB is low. The rising edge of this strobe can be used to latch the data into the peripheral.

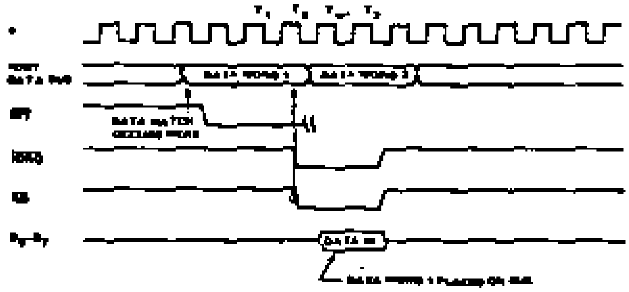


WR* = $\overline{RD}$ · CE · $\overline{C/D}$ · IORQ

BIT MODE

The bit mode does not utilize the handshake signals and a normal port write or port read can be executed at any time. When writing, the data will be latched into the output registers with the same timing as the output mode.

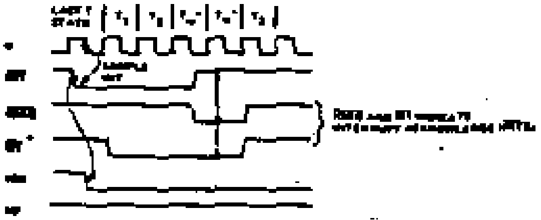
When reading the PIO, the data returned to the CPU will be composed of output register data from those port data lines assigned as outputs and input register data from those port data lines assigned as inputs. The input register will contain data which was present immediately prior to the falling edge of $\overline{RD}$. An interrupt will be generated if interrupts from the port are enabled and the data on the port data lines satisfy the logical equation defined by the 8-bit mask and 2-bit mask control registers.



* Timing Diagram Refers to Bit Mode Read

INTERRUPT ACKNOWLEDGE

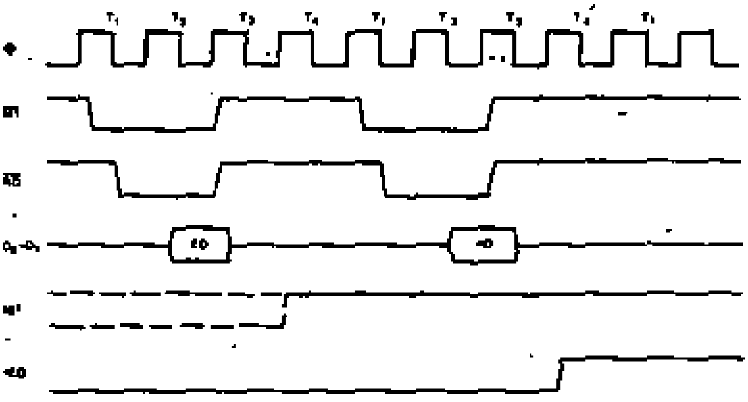
During $\overline{M1}$ time, peripheral controllers are inhibited from changing their interrupt enable status, permitting the $\overline{INT}$ Enable signal to ripple through the daisy chain. The peripheral with IEI high and IEO low during $\overline{INTA}$ will place a preprogrammed 8-bit interrupt vector on the data bus at this time. IEO is held low until a return from interrupt (RETI) instruction is executed by the CPU while IEI is high. The 2-byte RETI instruction is decoded internally by the PIO for this purpose.



RETURN FROM INTERRUPT CYCLE

If a Z80 peripheral device has no interrupt pending and is not under service, then its IEO=IEI. If it has an interrupt under service (i.e., it has already interrupted and received an interrupt acknowledge) then its IEO is always low, inhibiting lower priority chips from interrupting. If it has an interrupt pending which has not yet been acknowledged, IEO will be low unless an "ED" is decoded as the first byte of a two byte opcode. In this case, IEO will go high until the next opcode byte is decoded, whereupon it will again go low. If the second byte of the opcode was a "4D" then the opcode was an RETI instruction.

After an "ED" opcode is decoded, only the peripheral device which has interrupted and is currently under service will have its IEI high and its IEO low. This device is the highest priority device in the daisy chain which has received an interrupt acknowledge. All other peripherals have IEI=IEO. If the next opcode byte decoded is "4D", this peripheral device will reset its "interrupt under service" condition.



LOAD INTERRUPT VECTOR

The Z80-CPU requires an 8-bit interrupt vector be supplied by the interrupting device. The CPU forms the address for the interrupt service routine of the port using this vector. During an interrupt acknowledge cycle the vector is placed on the Z-80 data bus by the highest priority device requesting service at that time. The desired interrupt vector is loaded into the PIO by writing a control word to the desired port of the PIO with the following format.

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	V2	V1	0

signifies this control word is an interrupt vector

SELECTING AN OPERATING MODE

When selecting an operating mode, the 2-bit mode control register is set to one of four values. These two bits are the most significant bits of the register, bits 7 and 6; bits 5 and 4 are not used while bits 3 through 0 are all set to 1111 to indicate "set mode."

D7	D6	D5	D4	D3	D2	D1	D0
M1	M0	X	X	1	1	1	1

mode word

signifies mode word to be set

X=unused bit

Mode	M_1	M_0
Output	0	0
Input	0	1
Bidirectional	1	0
Bit	1	1

MODE 0 active indicates that data is to be written from the CPU to the peripheral.

MODE 1 active indicates that data is to be read from the peripheral to the CPU.

MODE 2 allows data to be written to or read from the peripheral device.

MODE 3 is intended for status and control applications. When selected, the next control word must set the I/O Register to indicate which lines are to be input and which lines are to be output.

I/O = 1 sets bit to input.
I/O = 0 sets bit to output.

D7	D6	D5	D4	D3	D2	D1	D0
I/O_7	I/O_6	I/O_5	I/O_4	I/O_3	I/O_2	I/O_1	I/O_0

INTERRUPT CONTROL

Bit 7 = 1 interrupt enable is set—allowing interrupt to be generated.

Bit 7 = 0 indicates the enable flag is reset and interrupts may not be generated.

Bits 6,5,4 are used in the bit mode interrupt operations; otherwise they are disregarded.

Bits 3,2,1,0 signify that this command word is an interrupt control word.

D7	D6	D5	D4	D3	D2	D1	D0
Enable Interrupt	AND/OR	High/Low	Mask follows	0	1	1	1

used in Mode 3 only

signifies interrupt control word

If the "mask follows" bit is high (D4 = 1), the next control word written to the port must be the mask.

D7	D6	D5	D4	D3	D2	D1	D0
MB_7	MB_6	MB_5	MB_4	MB_3	MB_2	MB_1	MB_0

Only those port lines whose mask bit is a 0 will be monitored for generating an interrupt.

The interrupt enable flip-flop of a port may be set or reset without modifying the rest of the interrupt control word by the following command.

D7	D6	D5	D4	D3	D2	D1	D0
Int Enable	X	X	X	0	0	1	1

TA = 0° C to 70° C, Vcc = +5 V ± 5%, unless otherwise noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c $t_W(\Phi H)$ $t_W(\Phi L)$ t_r, t_f	Clock Period Clock Pulse Width, Clock High Clock Pulse Width, Clock Low Clock Rise and Fall Times	400 170 170	[1] 2000 2000 30	nsec nsec nsec nsec	
	t_h	Any Hold Time for Specified Set-Up Time	0		nsec	
CS, $\overline{CE}$ ETC.	$t_{S\Phi}(CS)$	Control Signal Set-Up Time to Rising Edge of Φ During Read or Write Cycle	280		nsec	
D_0-D_7	$t_{DR}(D)$ $t_{S\Phi}(D)$ $t_{DI}(D)$ $t_F(D)$	Data Output Delay from Falling Edge of $\overline{RD}$ Data Set-Up Time to Rising Edge of Φ During Write or $\overline{M1}$ Cycle Data Output Delay from Falling Edge of $\overline{IORQ}$ During INTA Cycle Delay to Floating Bus (Output Buffer Disable Time)	 50	430 340 160	nsec nsec nsec nsec	[2] C_L = 50 pf [3]
IEI	$t_S(IEI)$	IEI Set-Up Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	140		nsec	
IEO	$t_{DH}(IO)$ $t_{DL}(IO)$ $t_{DM}(IO)$	IEO Delay Time from Rising Edge of IEI IEO Delay Time from Falling Edge of IEI IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring Just Prior to $\overline{M1}$) See Note A.		210 190 300	nsec nsec nsec	[5] [5] C_L = 50 pf [5]
$\overline{IORQ}$	$t_{S\Phi}(IR)$	$\overline{IORQ}$ Set-Up Time to Rising Edge of Φ During Read or Write Cycle	250		nsec	
$\overline{M1}$	$t_{S\Phi}(M1)$	$\overline{M1}$ Set-Up Time to Rising Edge of Φ During INTA or $\overline{M1}$ Cycle. See Note B.	210		nsec	
$\overline{RD}$	$t_{S\Phi}(RD)$	$\overline{RD}$ Set-Up Time to Rising Edge of Φ During Read or $\overline{M1}$ Cycle	240		nsec	
A_0-A_7, B_0-B_7	$t_S(PD)$ $t_{DS}(PD)$ $t_F(PD)$ $t_{DI}(PD)$	Port Data Set-Up Time to Rising Edge of $\overline{STROBE}$ (Mode 1) Port Data Output Delay from Falling Edge of $\overline{STROBE}$ (Mode 2) Delay to Floating Port Data Bus from Rising Edge of $\overline{STROBE}$ (Mode 2) Port Data Stable from Rising Edge of $\overline{IORQ}$ During WR Cycle (Mode 0)	260	 230 200 200	nsec nsec nsec nsec	 [5] C_L = 50 pf [5]
$\overline{ASTB}$, $\overline{BSTB}$	$t_W(ST)$	Pulse Width, $\overline{STROBE}$	150 [4]		nsec nsec	
$\overline{INT}$	$t_D(IT)$ $t_D(IT3)$	$\overline{INT}$ Delay Time from Rising Edge of $\overline{STROBE}$ $\overline{INT}$ Delay Time from Data Match During Mode 3 Operation		490 420	nsec nsec	
ARDY, BRDY	$t_{DH}(RY)$ $t_{DL}(RY)$	Ready Response Time from Rising Edge of $\overline{IORQ}$ Ready Response Time from Rising Edge of $\overline{STROBE}$		t_c+ 460 t_c+ 400	nsec nsec	[5] C_L = 50 pf [5]

A. 2.5 t_c > (N-2) $t_{DL}(IO)$ + $t_{DM}(IO)$ + $t_S(IEI)$ + TTL Buffer Delay, if any

B. $\overline{M1}$ must be active for a minimum of 2 clock periods to reset the PIO.

[1] $t_c = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$

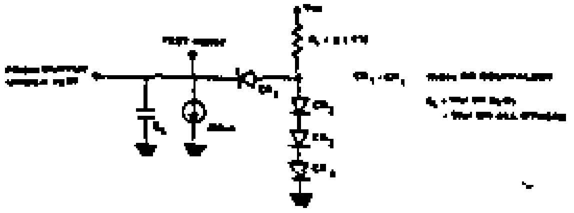
[2] Increase $t_{DR}(D)$ by 10 nsec for each 50 pf increase in loading up to 200 pf max.

[3] Increase $t_{DI}(D)$ by 10 nsec for each 50 pf increase in loading up to 200 pf max.

[4] For Mode 2: $t_W(ST) > t_S(PD)$

[5] Increase these values by 2 nsec for each 10 pf increase in loading up to 100 pf max.

Output load circuit.



Capacitance

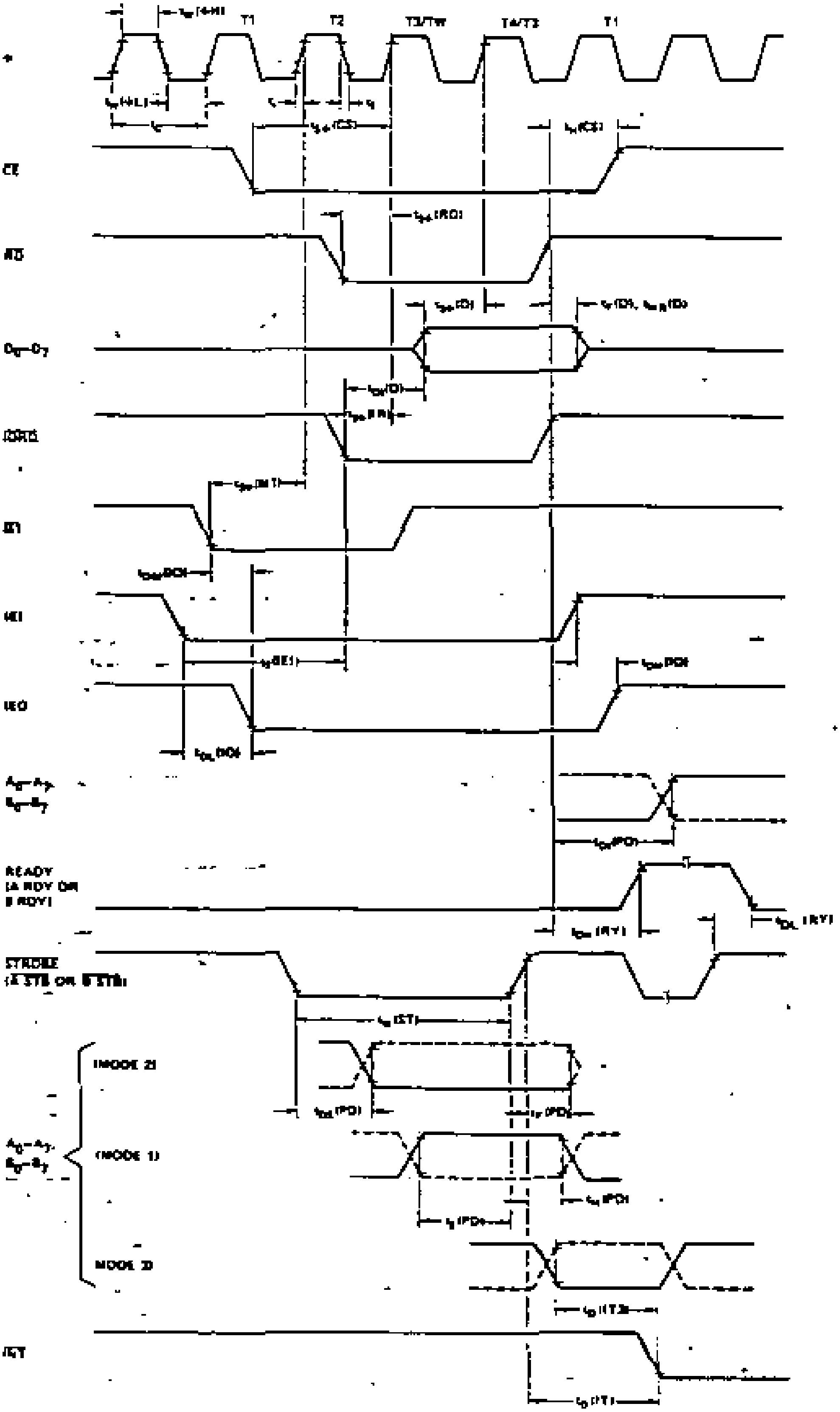
TA = 25° C, f = 1 MHz

Symbol	Parameter	Max.	Unit	Test Condition
C_Φ	Clock Capacitance	10	pF	Unmeasured Pins
C_{IN}	Input Capacitance	5	pF	Returned to Ground
C_{OUT}	Output Capacitance	10	pF	

A.C. Timing Diagram

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	4.2V	0.8V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV	= +0.5V



Absolute Maximum Ratings

Temperature Under Bias	Specified operating range.
Storage Temperature	−65° C to +150° C
Voltage On Any Pin With Respect To Ground	−0.3 V to +7 V
Power Dissipation	6 W

*Comment

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Note: All AC and DC characteristics remain the same for the military grade parts except I_{CC}

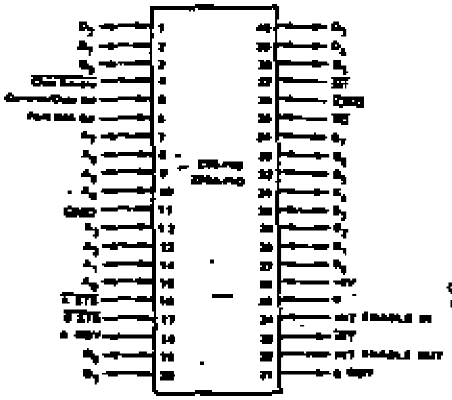
I_{CC} = 130 mA.

Z80-PIO and Z80A-PIO D.C. Characteristics

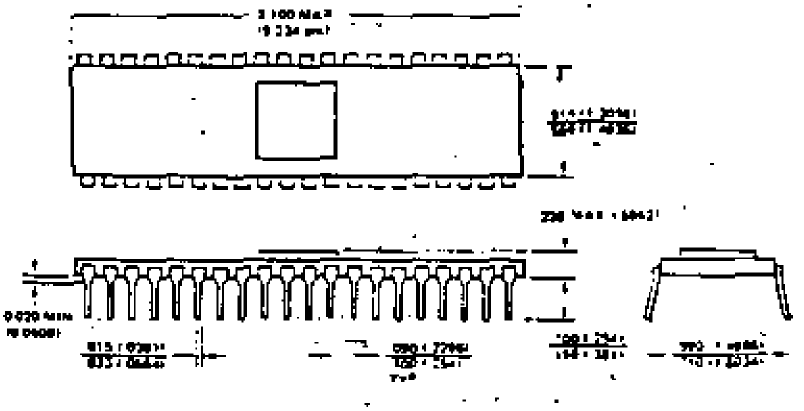
TA = 0° C to 70° C, Vcc = 5 V ± 5% unless otherwise specified

Symbol	Parameter	Min.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	−0.3	.45	V	
V_{IHC}	Clock Input High Voltage	Vcc-.6	Vcc+.3	V	
V_{IL}	Input Low Voltage	−0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	Vcc	V	
V_{OL}	Output Low Voltage		0.4	V	I_{OL} = 2.0 mA
V_{OH}	Output High Voltage	2.4		V	I_{OH} = 250 μA
I_{CC}	Power Supply Current		70	mA	
I_{LI}	Input Leakage Current		±10	μA	V_{IN} = 0 to Vcc
I_{LOH}	Tri-State Output Leakage Current in Float		10	μA	V_{OUT} = 2.4 to Vcc
I_{LOL}	Tri-State Output Leakage Current in Float		−10	μA	V_{OUT} = 0.4 V
I_{LD}	Data Bus Leakage Current in Input Mode		±10	μA	0 < V_{IN} < Vcc
I_{OHD}	Darlington Drive Current	−1.5		mA	V_{OH} = 1.5 V Port B Only

Package Configuration



Package Outline



*Dimensions for metric system are in parentheses

TA = 0° C to 70° C, Vcc = +5 V ± 5%, unless otherwise noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c $t_w(\Phi H)$ $t_w(\Phi L)$ t_r, t_f	Clock Period Clock Pulse Width, Clock High Clock Pulse Width, Clock Low Clock Rise and Fall Times	250 105 105	[1] 2000 2000 30	nsec nsec nsec nsec	
	t_h	Any Hold Time for Specified Set-Up Time	0		nsec	
$\overline{CS}$, CE ETC.	$t_{S\Phi}(CS)$	Control Signal Set-Up Time to Rising Edge of Φ During Read or Write Cycle	145		nsec	
D_0-D_7	$t_{DR}(D)$ $t_{S\Phi}(D)$ $t_{DI}(D)$ $t_F(D)$	Data Output Delay From Falling Edge of $\overline{RD}$ Data Set-Up Time to Rising Edge of Φ During Write or $\overline{M1}$ Cycle Data Output Delay from Falling Edge of $\overline{IORQ}$ During INTA Cycle Delay to Floating Bus (Output Buffer Disable Time)	 50	380 250 110	nsec nsec nsec nsec	[2] C_L = 50 pf [3]
IEI	$t_S(IEI)$	IEI Set-Up Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	140		nsec	
IEO	$t_{DH}(IO)$ $t_{DL}(IO)$ $t_{DM}(IO)$	IEO Delay Time from Rising Edge of IEI IEO Delay Time from Falling Edge of IEI IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring Just Prior to $\overline{M1}$). See Note A.		160 130 190	nsec nsec nsec	[5] [5] C_L = 50 pf [5]
$\overline{IORQ}$	$t_{S\Phi}(IR)$	$\overline{IORQ}$ Set-Up Time to Rising Edge of Φ During Read or Write Cycle	115		nsec	
$\overline{M1}$	$t_{S\Phi}(M1)$	$\overline{M1}$ Set-Up Time to Rising Edge of Φ During INTA or $\overline{M1}$ Cycle. See Note B	90		nsec	
$\overline{RD}$	$t_{S\Phi}(RD)$	$\overline{RD}$ Set-Up Time to Rising Edge of Φ During Read or $\overline{M1}$ Cycle	115		nsec	
A_0-A_7, B_0-B_7	$t_S(PD)$ $t_{DS}(PD)$ $t_F(PD)$ $t_{DI}(PD)$	Port Data Set-Up Time to Rising Edge of $\overline{STROBE}$ (Mode 1) Port Data Output Delay from Falling Edge of $\overline{STROBE}$ (Mode 2) Delay to Floating Port Data Bus from Rising Edge of $\overline{STROBE}$ (Mode 2) Port Data Stable from Rising Edge of $\overline{IORQ}$ During WR Cycle (Mode 0)	230	 210 180 180	nsec nsec nsec nsec	 [5] C_L = 50 pf [5]
$\overline{ASTB}$, $\overline{BSTB}$	$t_W(ST)$	Pulse Width, $\overline{STROBE}$	150 [4]		nsec nsec	
$\overline{INT}$	$t_D(IT)$ $t_D(IT3)$	$\overline{INT}$ Delay Time from Rising Edge of $\overline{STROBE}$ $\overline{INT}$ Delay Time from Data Match During Mode 3 Operation		440 380	nsec nsec	
ARDY, BRDY	$t_{DH}(RY)$ $t_{DL}(RY)$	Ready Response Time from Rising Edge of $\overline{IORQ}$ Ready Response Time from Rising Edge of $\overline{STROBE}$		t_c+ 460 t_c+ 360	nsec nsec	[5] C_L = 50 pf [5]

A. $2.5\ t_c > (N-2)\ t_{DL}(IO) + t_{DM}(IO) + t_S(IEI)$ + TTL Buffer Delay, if any

B. $\overline{M1}$ must be active for a minimum of 2 clock periods to reset the PIO.

[1] $t_c = t_w(\Phi H) + t_w(\Phi L) + t_r + t_f$

[2] Increase $t_{DR}(D)$ by 10 nsec for each 50 pf increase in loading up to 200 pf max.

[3] Increase $t_{DI}(D)$ by 10 nsec for each 50 pf increase in loading up to 200 pf max.

[4] For Mode 2: $t_W(ST) > t_S(PD)$

[5] Increase these values by 2 nsec for each 10 pf increase in loading up to 100 pf max.

ZILOG SALES OFFICES

EASTERN REGION
Zilog, Inc.
[illegible]1 Totten Pond Road
Waltham, MA 02154
TEL 617 890-0640
TWX 710 324 1974

MIDWESTERN REGION
Zilog, Inc.
1701 Woodfield Place
Suite 417
Schaumburg, IL 60195
TEL 312 885-8080
TWX 910 291 1064

WESTERN REGION
Zilog, Inc.
1815 Via el Prado
Redondo Beach, CA 90277
TEL 213 540-7749

ZILOG EUROPEAN HEADQUARTERS
Zilog (UK) Ltd.
Nicholson House
Maidenhead
Berks
England
TEL (0628) 36131/2/3
TWX 848-609

ZILOG U.S. DISTRIBUTORS

Western Region

Intermark Electronics
1802 E. Carnegie Avenue
Santa Ana, CA 92705
TEL 714 540 1322
TWX 910 595 1583

Intermark Electronics
4040 Sorrento Valley Blvd.
San Diego, CA 92121
TEL 714 279 5200
714 453 9005
TWX 910 335 1515

Intermark Electronics
1020 Stewart Drive
Sunnyvale, CA 94086
TEL 408 738 1111
TWX 910 339 9312

R.V. Weatherford Co.
6921 San Fernando Road
Glendale, CA 91201
TEL 212 849 3451
TWX 910 498 2223

R.V. Weatherford Co.
1550 Babbitt Avenue
Anaheim, CA 92805
TEL 714 634 9600
TWX 910 593 1334

Weatherford Co.
[illegible] Hillview Ave.
[illegible] Industrial Park
Palo Alto, CA 94304
TEL 415 493 5373

Sterling Electronics
5608 6th Avenue South
Seattle, WA 98108
TEL 206 762 9100
TLX 32 9652

Sterling Electronics
5608 6th Avenue South
Seattle, WA 98108
TEL 206 762 9100
TWX 32 9652

R.V. Weatherford Co.
1095 East Third Street
Pomona, CA 91766
TEL 714 623 1261
TWX 910 581 3811

R.V. Weatherford Co.
3311 W. Earll Drive
Phoenix, AZ 85017
TEL 602 272 7144
TWX 910 951 0636

Mountain

Century Electronics
121 Elizabeth, N.E.
Albuquerque, NM 87123
TEL 505 292 0625
TWX 910 989 0625

Century Electronics
2150 South 300 West
Salt Lake City, UT 84115
TEL 801 487 8551
TWX 910 925 5686

Century Electronics
8155 West 48th Avenue
Wheatridge, CO 80033
TEL 303 424 1985
TWX 910 938 0393

R.V. Weatherford Company
3901 South Mariposa
Englewood, CO 80110
TEL 303 761 5432
TWX 910 933 0173

Eastern

Hallmark Electronics
4739 Commercial Drive
Huntsville, AL 35805
TEL 205 837 8700
TWX 810 726 2187

Hallmark Electronics
1302 West McNab Road
Fort Lauderdale, FL 33309
TEL 305 971 9280
TWX 510 956 9720

Hallmark Electronics
7233 Lake Ellenor Drive
Orlando, FL 32809
TEL 305 855 4020
TWX 810 850 0183

Hallmark Electronics
3355 Amberton Drive
Baltimore, MD 21227
TEL 301 796 9300
TWX 710 862 1942

Hallmark Electronics
1208 Front Street
Building K
Raleigh, NC 27609
TEL 919 832 4465
TWX 510 928 1831

Hallmark Electronics
Pike Industrial Park
Huntingdon Valley, PA
TEL 215 355 7300
TWX 510 667 1750

Quay Corporation
P.O. Box 386
Freehold, NJ 07728
TEL 201 681 8700

Summit
916 Main Street
Buffalo, NY 14202
TEL 716 884 3450

Midwestern

Hallmark Electronics
180 Crossen Avenue
Elk Grove Village, IL 60076
TEL 312 475 8450
TWX 910 223 3645

Hallmark Electronics
11870 West 91st Street
Congleton Industrial Park
Shawnee Mission, KS 66214
TEL 913 888 4747
TWX 910 749 6620

Hallmark Electronics
9201 Penn Avenue South
Suite 10
Bloomington, MN 55431
TEL 612 884 9056
TWX 910 576 3187

Hallmark Electronics
13789 Rider Trail
Earth City, MO 63045
TEL 314 291 5350
TWX 910 760 0671

Hallmark Electronics
6969 Worthington-Galena Road
Worthington, OH 43085
TEL 614 846 1882

Hallmark Electronics
4846 S. 83rd. Road E. Avenue
Tulsa, OK 74145
TEL 918 835 8458
TWX 910 845 2290

Hallmark Electronics
3100-A Industrial Terrace
Austin, TX 78758
TEL 512 837 2841
TWX 910 874 2031

Hallmark Electronics
9333 Forest Lane
Dallas, TX 75231
TEL 214 231 6101
TWX 910 867 4721

Hallmark Electronics
8000 W. Glenn
Houston, TX 77063
TWX 910 881 2711

Hallmark Electronics
237 South Curtis
West Allis, WI 53214
TEL 414 476 1270
TWX 910 262 3186

Ordering Information

C — Ceramic
P — Plastic
S — Standard 5V ± 5% 0° to 70° C
E — Extended 5V ± 5% −40° to 85°C
M — Military 5V ± 10% −55° to 125°C

Example:

Z80-PIO CS (Ceramic—Standard range)

10460 Bubb Road, Cupertino, California 95014
Telephone: (408) 446-4666 TWX 910-338-7621

Printed in U.S.A.

11

Z80-DMA
Z80A-DMA

Product Specification

OCTOBER 1977

PRELIMINARY

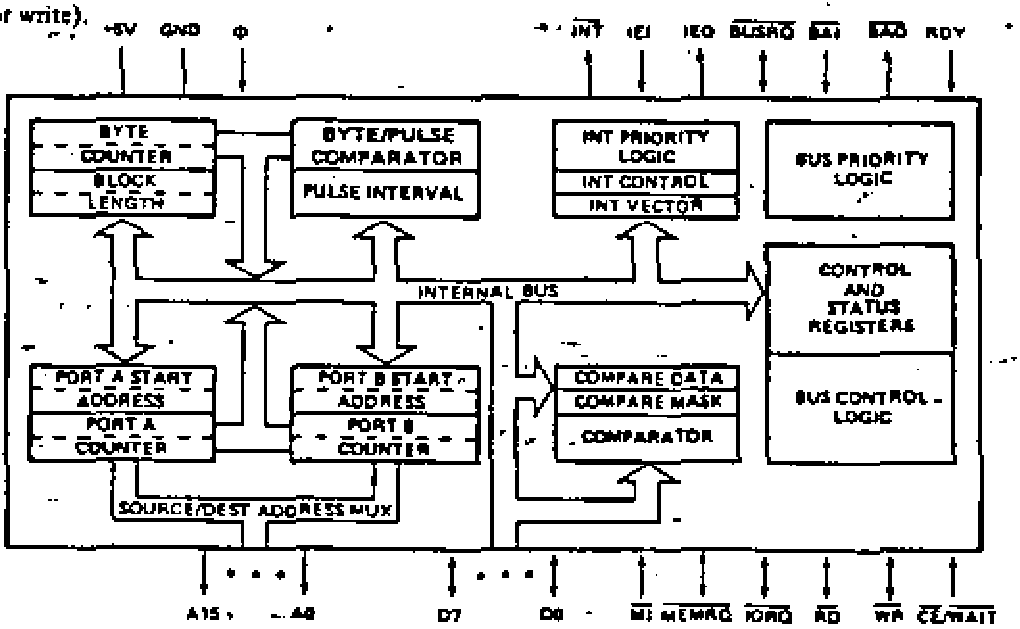
Zilog's Z80 microcomputer product line includes a third generation LSI component set, development systems and support software. The component set includes all the logic circuits necessary for the user to build high performance microcomputer systems with virtually no external logic and a minimal number of standard low-cost memory components. The Z80-DMA (Direct Memory Access) circuit is a programmable single-channel device which provides all address, timing and control signals to effect the transfer of blocks of data between two ports within a Z80-CPU based system. These ports may be either system main memory or any system peripheral I/O device. The DMA can also search a block of data for a particular byte (bit maskable), with or without a simultaneous transfer.

Structure

- N-channel Silicon Gate Depletion Load Technology
- 40 Pin DIP
- Single 5 volt supply
- Single phase 5 volt clock
- Single channel, two port

eatures

- Three classes of operation:
  - —Transfer Only
  - —Search Only
  - —Search-Transfer
- Address and Block Length Registers, fully buffered. Values for next operation may be loaded without disturbing current values.
- Dual addresses generated during a transfer (one for read port and one for write).
- Programmable data transfers and searches, automatically incrementing or decrementing the port addresses from programmed starting addresses (they can also remain fixed).
- Four modes of operation:
  - —Byte-at-a-time: One byte transferred per request
  - —Burst: Continues as long as ports are ready
  - —Continuous: Locks out CPU until operation complete
  - —Transparent: Steals refresh cycles
- Timing may be programmed to match the speed of any port.
- Interrupts on Match Found, End of Block, or Ready, may be programmed.
- An entire previous operation may be repeated automatically or on command. (Auto restart or Load)
- The DMA can signal when a specified number of bytes has been transferred, without halting transfer.
- Multiple DMA's easily configured for rotating priority.
- The channel may be enabled, disabled or reset under software control.
- Complete channel status upon program (CPU) request.
- Up to 1.25 megabyte Search or Transfer Rate.
- Daisy-chain priority interrupt and bus acknowledge included to provide automatic interrupt vectoring and bus request control, without need for additional external logic.
- TTL compatible inputs and outputs
- The CPU can read current Port counters, Byte counter, or Status Register. A mask byte can be set which defines which registers can be accessed during read operations.



DMA Internal Block Diagram

Fig.1

[illegible]

DMA Architecture

A block diagram of the Z80 DMA is shown in Figure 1. The internal structure consists of the following circuitry:

Bus Interface: provides driver and receiver circuitry to interface to the Z80-CPU Bus.

- *Control Logic and Registers*: set the class, mode and other basic control parameters of the DMA.
- *Address, Byte Count and Pulse Circuitry*: generates the proper port addresses for the read and write operations, with provisions for incrementing or decrementing the address. When zero bytes remain to be handled, the byte count circuitry sets a flag in the status register. Pulse circuitry generates a pulse each time the byte counter lower 8-bits equal the pulse reg.
- *Timing Circuitry*: allows the user to completely specify the read/write timing for both of the channels' addressed ports.
- *Match Circuitry*: holds the match byte and a mask byte which allows for the comparison of only certain bits within the byte. If a match is encountered during a Search or Transfer, this circuitry sets a flag in the status register.
- *$\overline{INT}$ and $\overline{BUSRQ}$ Circuitry*: includes a control register which specifies the conditions under which the DMA can generate an interrupt; priority encoding logic to select between the generation of an $\overline{INT}$ or $\overline{BUSRQ}$ output under these conditions; and an interrupt vector register for automatic vectoring to the interrupt service routine.
- *Status Register*: holds current status of DMA.

Register Description

The following DMA-internal registers are available to the programmer:

- *Control Registers:* Hold DMA control information: such as, when to initiate an interrupt or pulse, what mode or class of operation to perform, etc. (Write Only) (8 Bits)
- *Timing Registers*: Hold read/write timing parameters for the two ports. (Write Only) (8 bits)
- *Interrupt Vector Register:* Holds the 8-bit vector that the DMA will put onto the data bus after receiving an $\overline{IORQ}$ during an interrupt acknowledge sequence if it is the highest priority device requesting an interrupt. (This register is readable only during interrupt acknowledge cycles.) (Read/Write) (8 bits)
- *Block Length Register:* Contains total block length of data to be searched and/or transferred. (Write Only) (16 bits)
- *Byte Counter:* Counts number of bytes transferred (or searched). On a Load or Continue the Byte Counter is reset to zero. Thereafter, each byte transfer operation increments it until it matches the contents of the Block Length Register, at which time End of Block is set in the status register and operation is suspended if programmed. Also if so programmed the DMA will generate an interrupt. (Read Only) (16 bits)

  Compare Register: Holds the byte for which a match is being sought in Search operations. (Write Only) (8 bits)
- *Mask Register:* Holds the 8 bit mask to determine which bits in the compare register are to be examined for a match. (Write Only) (8 bits)
- *Starting Address Registers (Port A and Port B)* Hold the starting addresses (upper and lower 8 bits) for the two ports involved in Transfer operations. In Search Only operations, only one port address would have to be specified. Only memory starting addresses require both upper and lower 8-bits; I/O ports are generally addressed with only the lower 8-bits, and in this case the address contained in the register is a generally fixed address. (Write Only) (16 bits each)
- *Address Counters (Port A and Port B)*: These counters are loaded with the contents of the corresponding Starting Address Registers whenever Searches or Transfers are initiated with a Load or Contunue. They are incremented, decremented or remain fixed, as programmed. (Read Only) (16 bits each)
- *Pulse Control Register:* Holds program-supplied length (in bytes) of block after which the DMA will provide a signal pulse on the $\overline{INT}$ pin. (Since this occurs while both $\overline{BUSRQ}$ and $\overline{BUSAK}$ are active, the CPU will not interpret this as an interrupt request. Instead, the signal is used to communicate with a peripheral I/O device.) (Write Only) (8 bits)
- *Status Register:* Match, End of Block, Ready Active, Interrupt Pending, and Write Address Valid bits indicate these functions when set. (Read Only) (8 bits)

Modes of Operation

The DMA may be programmed for one of four modes of operation. (See Command Byte 2B).

- *Byte at a time*: control is returned to the CPU after each one-byte cycle.
- *Burst*: operation continues as long as the DMA's RDY input is active, indicating that the relevant port is ready. Control returns to the CPU when RDY is inactive or at end of block or a match if so programmed.
- *Continuous*: the entire Search and/or Transfer of a block of data is completed before control is returned to CPU.
- *Transparent*: DMA operation occurs during normal memory refresh times without visible loss of CPU time.

Classes of Operation

The DMA has three classes of operation: Transfer only, Search Only and a combined Search-Transfer. (See Command Byte 1A.)

During a Transfer, data is first read from one port and then written to the other port, byte by byte. (The DMA's two ports are termed Port A and Port B.) The ports may be programmed to be either system main memory or peripheral I/O devices. Thus, a block of data might be written from a peripheral to another; or it might be written from one area in main memory to another; or from a peripheral to main memory.

During a Search, data is read only, and compared byte by byte against two DMA-internal registers, one of which contains a match byte and the other an optional mask byte which allows only certain bits to be compared. If any byte of searched data matches, a DMA-internal status bit is set; if programmed to do so, the DMA will then suspend operation and/or generate an interrupt.

The third class of operation is a combined Search-Transfer. In such an operation a block of data is transferred as described above until a match is found; then, as in a Search Only operation, the transfer may be suspended and/or an intern rated.

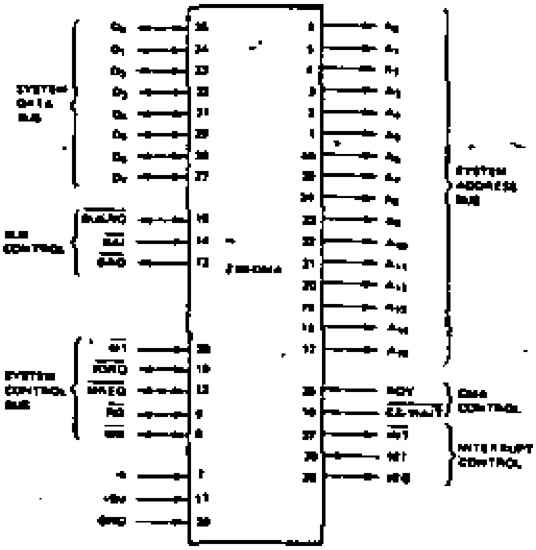
Addressing

The DMA's addressing of ports is either fixed or sequential, incrementing or decrementing from a starting address. The length of the operation (number of bytes) is specified by the programmed contents of a block length register. The DMA can address block lengths of up to 64K bytes. During a transfer two separate port addresses are generated, one during the Read cycle and one during the Write cycle.

Operating Sequence

Once the DMA has been programmed it may be "Enabled" (command byte 2d or 2a). In the enabled condition when Ready goes active the DMA will request the bus by bringing $\overline{BUSRQ}$ low. The CPU will acknowledge this with a $\overline{BUSACK}$ which will normally be attached to $\overline{BAI}$. When the DMA receives $\overline{BAI}$ it will start its programmed operation releasing $\overline{BUSRQ}$ to a "high" state when it is through.

Z80-DMA Pin Description



A_0–A_{15} System Address Bus. All sixteen of these pins are used by the DMA to address system main memory or an I/O port (output)

D_0–D_7 System Data Bus. Commands from the CPU, DMA status and data from memory or peripherals are transferred on these tristate pins (input/output)

+5V Power

GND Ground

Φ System clock (input)

$\overline{M1}$ Machine cycle One signal from CPU (input)

$\overline{IORQ}$ Input/Output Request to and from the System Bus (input/output)

$\overline{MREQ}$ Memory REQuest to the System Bus (input/output)

$\overline{RD}$ ReaD to and from the System Bus (input/output)

$\overline{WR}$ WRite to and from the System Bus (input/output)

$\overline{CE}/\overline{WAIT}$ Chip Enable: may also be programmed to be $\overline{WAIT}$ during time when $\overline{BAI}$ is low (input)

$\overline{BUSRQ}$ BUS ReQuest. Requests control of the CPU Address Bus, Data Bus and Status/Control Bus (input/output, open drain)

$\overline{BAI}$ Bus Acknowledge In. Signals that the system buses have been released for DMA control (input)

$\overline{BAO}$ Bus Acknowledge Out. $\overline{BAI}$ and $\overline{BAO}$ form a daisy-chain connection for system-wide priority bus control (output)

$\overline{INT}$ INTerrupt request (output, open drain)

IEI Interrupt Enable In (input)

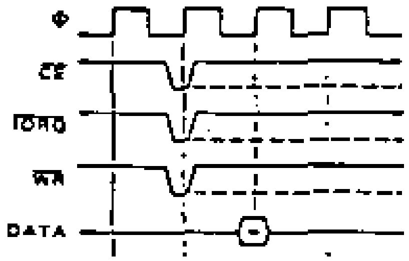
IEO Interrupt Enable Out. IEI and IEO form a daisy-chain connection for system-wide priority interrupt control (output)

RDY ReaDY is monitored by the DMA to determine when a peripheral device associated with a DMA port is ready for a read or write operation (input, programmable as active high or low)

DMA Timing Waveforms

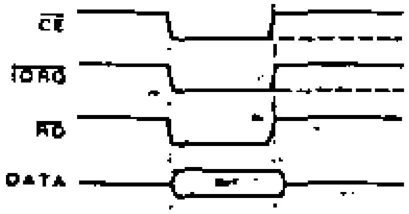
DMA Command Write Cycle

Illustrated here is the timing associated with a command byte or control byte being written to the DMA which is to be loaded into internal registers. Z80 Output instructions satisfy this timing.



DMA Register Read Cycle

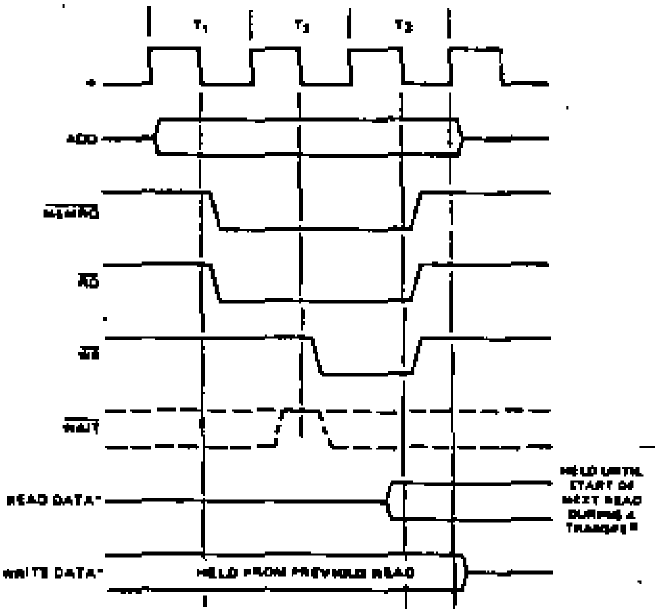
This timing is used when a read operation is performed on the DMA to access the contents of the Status Register, Address Counter or other readable registers. Z80 Input instructions satisfy this timing.



STD Memory Timing

This timing is exactly the same as used by the Z80-CPU to access system main memory, either in a Read or Write operation. The DMA will default to this timing after a power-on reset, or when a Reset or Reset Timing command is written to it; and unless otherwise programmed, will use this timing during all Transfer or Search operations involving system main memory. During the memory Read portion of a transfer cycle, data is latched in the DMA on the negative edge of Φ during T_3 and held into the following Write cycle. During the memory Write portion of a transfer cycle, data is held from the previous Read cycle and released at the end of the present cycle.

NOTE: The DMA is normally programmed for a 3 T-cycle duration in memory transactions. But $\overline{WAIT}$ is sampled during the negative transition of T_2, and if it is low, T_2 will be extended another T-cycle, during which $\overline{WAIT}$ will again be sampled. The duration of a memory transaction cycle may thus be indefinitely extended.

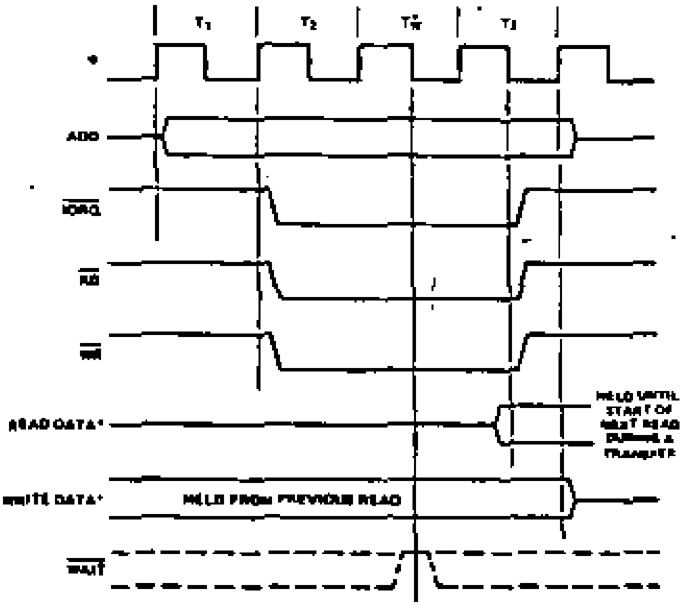


*LATCHED BY DMA ON BUS DURING A TRANSFER

STD Peripheral Timing

This timing is identical to the Z80-CPU's Read/Write timing to I/O peripheral devices. The DMA will default to this timing after a power-on reset, or when a Reset or Reset Timing command is written to it; and unless otherwise programmed, will use this timing during all Transfer or Search operations involving I/O peripherals. During the I/O Read of a transfer cycle, data is latched on the negative edge of Φ during T_3 and is then held into the Write cycle. During an I/O Write, data is held from the previous Read cycle until the end of the Write cycle.

NOTE: If $\overline{WAIT}$ is low during the negative transition of T_W*, then T_W* will be extended another T-cycle and $\overline{WAIT}$ will again be sampled. The duration of a peripheral transaction cycle may thus be indefinitely extended.

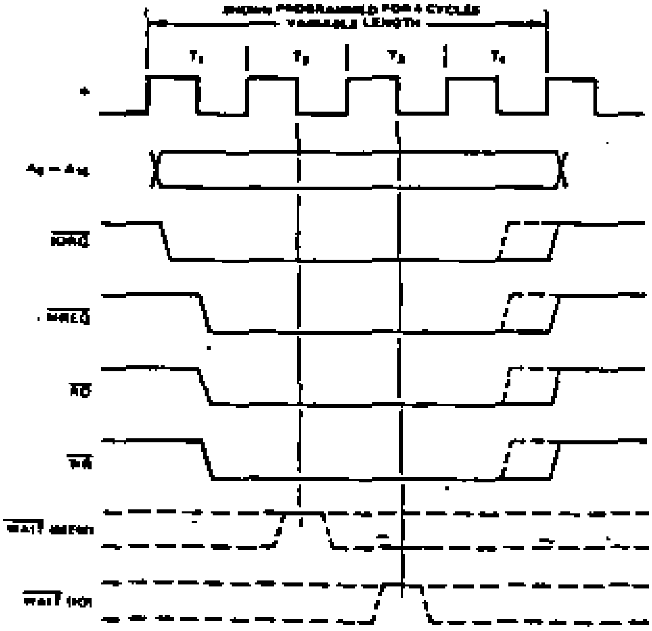


*LATCHED BY DMA ON BUS DURING A TRANSFER

Variable Cycle

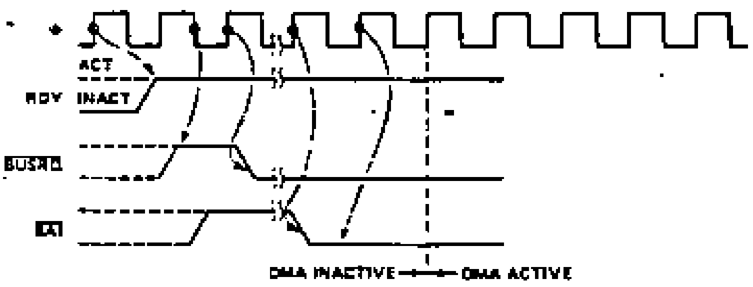
The Variable feature of the DMA allows the user to program the DMA's memory or peripheral transaction timing to values different than given above in the standard default diagrams. This permits the designer to tailor his timing to the particular requirements of his system components, and maximizes the data transfer rate while eliminating external signal conditioning logic. Cycle length can be one to four T-cycles (more if $\overline{WAIT}$ is used). Signal timing can be varied as shown. During transfer, data will be latched by the DMA on the clock edge using the rising edge of $\overline{RD}$ and will be held on the data lines until the end of the following Write cycle.

(See Timing Control Byte, page 7.)



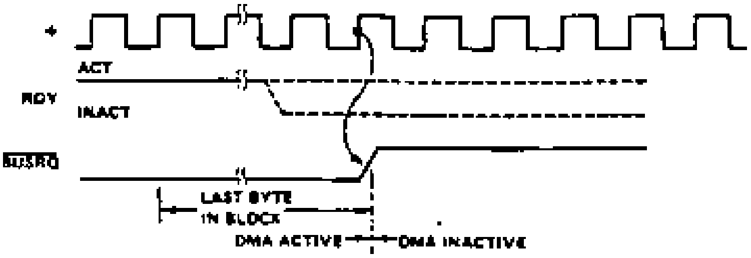
DMA Bus Request and Acceptance for /te-at-a-Time, Burst, and Continuous Mode

Ready is sampled on every rising edge of Φ. When it is found to be active, the following rising edge of Φ generates $\overline{BUSRQ}$. After receiving $\overline{BUSRQ}$ the CPU will grant a $\overline{BUSAK}$ which will be connected to $\overline{BAI}$ either directly or through the Bus Acknowledge Daisy Chain. When a low is detected on $\overline{BAI}$ (sampled on every rising edge of Φ), the next rising edge of Φ will start an active DMA cycle.



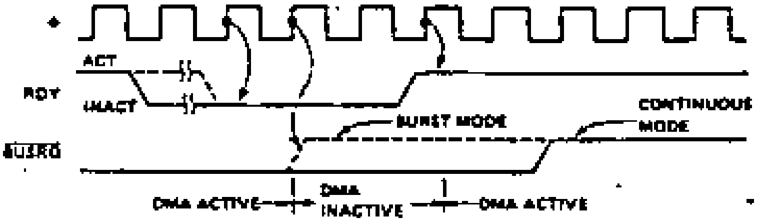
DMA Bus Release at End of Block for Burst or Continuous Mode

Timing for End of Block and DMA not programmed for Auto-restart.



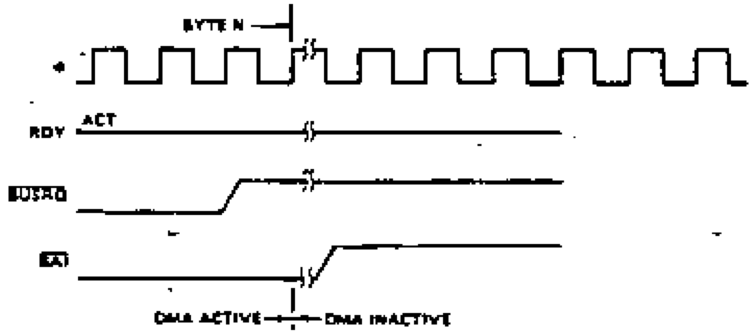
DMA Bus Release with 'Ready' for Burst and Continuous Mode

The DMA will relinquish the bus after RDY has gone inactive (Burst mode) or after an End of Block or a Match is found (Continuous mode). With RDY inactive, the DMA in Continuous mode is inactive but maintains control of the bus ($\overline{BUSRQ}$ low) until the cycle is resumed when RDY goes active.



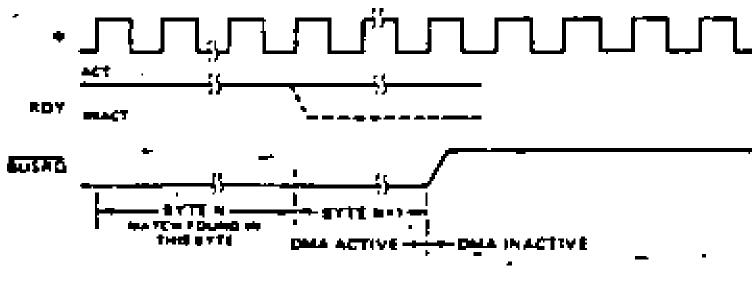
DMA Bus Release for Byte-at-a-Time Mode

In the Byte mode the DMA will release $\overline{BUSRQ}$ on the rising edge of Φ prior to the end of each Read cycle in Search Only or each Write cycle in a Transfer, regardless of the state of RDY. The next bus request will come after both $\overline{BUSRQ}$ and $\overline{BAI}$ have returned high.



DMA Bus Release with Match for Burst or Continuous Modes

When a Match is found and the DMA is programmed to stop on Compare, the DMA performs an operation on the next byte and then releases bus.



Reading from the DMA Internal Registers

Seven registers are available on the DMA for reading. They are: 8 bits of the status register, the upper and lower 8 bits of the block length register, and two port address registers.

These are available to be read sequentially: status, BLK Lower, BLK Upper, Port A Address lower, Port A Address Upper, Port B Address lower, Port B Address upper. An internal pointer points to each register in turn as each READ is accomplished. If a register is not to be read, it may be excluded by programming a 0 in the Read Byte. The internal pointer will skip any register not programmed with a 1 in the Read Byte. After a Reset or a Load, Reset RD must be given to set the internal pointer pointing to the first register programmed to be read by the Read Byte. After RD Status, the pointer will be pointing to the status register regardless of the Read Byte and the next read will be from the status register. The following read will be from the register pointed to before RD Status.

Programming the DMA

Previous sections of this specification have indicated the various functions and modes of the DMA. The diagrams and charts below will show how the DMA is programmed to select among these functions and modes and to adapt itself to the requirements of the user system. More detailed programming information is available in the *Z80-DMA Technical manual.*

The Z80-DMA chip may be in an "enable" state, in which it can gain control of the system buses and direct the transfer of data between its ports, or in a "disable" state, when it cannot gain control of the bus. Program commands can be written to it in either state, but writing a command to it automatically puts it in the disable state, which is maintained until an enable command is issued to the DMA. The CPU must program it in advance of any data search or transfer by addressing it as an I/O port and sending it a sequence of 8-bit command bytes via the system data bus using Output instructions. When the DMA is powered up or reset by any means, the DMA will automatically be placed into a disable state, in which it can initiate neither bus requests nor data transfers nor interrupts.

The command bytes contain information to be loaded into the DMA's control and other registers and/or information to alter the state of the chip, such as an Enable Interrupt command. The command structure is designed so that certain bits in some commands can be set to alert the DMA to expect the next byte written to it to be for a particular internal register.

The following diagrams and charts give the function of each bit in the six different command bytes. Two of these are defined as being from Group 1, and are termed command bytes 1A and 1B. These Group 1 commands contain the most basic DMA set-up information. The other four are categorized as Group 2, and are termed commands 2A, 2B, 2C and 2D. Group 2 words specify more detailed set-up information.

Command Byte 1A

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	BLOCK LENGTH UPPER FOLLOWS	BLOCK LENGTH LOWER FOLLOWS	PORT A STARTING ADDRESS UPPER FOLLOWS	PORT A STARTING ADDRESS LOWER FOLLOWS	SOURCE PORT	CLASS CONTROL C_1	CLASS CONTROL C_0

Specifies Group 1

Byte 1A cannot be 00

C_1	C_0	Function
0	0	Not allowed. (Command Byte 1B)
0	1	Transfer Only.
1	0	Search Only.
1	1	Search and Transfer.

D_2 = 1 Port A is read from, Port B is written to (unless the Search Only Mode has been selected, in which case Port B is never addressed).

D_2 = 0 Port B is read from, Port A is written to (unless the Search Only Mode has been selected, in which case Port A is never addressed).

Command Byte 1B

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
0	TIMING BYTE FOLLOWS	ADDRESS FIXED	ADDRESS INCREMENTS DECREMENTS	I/O OR MEMORY	PORT A OR B	0	0

Specifies Group 1

Specifies Byte 1B

D_4 = 1	Address for this port increments after each byte.
D_4 = 0	Address for this port decrements after each byte.
D_3 = 1	This port addresses an I/O peripheral.
D_3 = 0	This port addresses main memory.
D_2 = 1	This word programs Port A.
D_2 = 0	This word programs Port B.

Command Byte 2A

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	ENABLE CHIP	ENABLE INTERRUPT	MATCH BYTE FOLLOWS	MASK BYTE FOLLOWS	STOP ON COMPARE	0	0

Specifies Group 2

Specifies Byte 2A

Command Byte 2B

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	MODE M_1	MODE M_0	INTERRUPT CONTROL BYTE FOLLOWS	PORT A UPPER ADDRESS FOLLOWS	PORT B LOWER ADDRESS FOLLOWS	1	1

Specifies Group 2 — Specifies Byte 2B

M_1	M_0	Mode
0	0	Byte
0	1	Continuous
1	0	Burst
1	1	Transparent

Command Byte 2C

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	NOT USED	AUTOMATIC RESTART	$\overline{WAIT}$ MULTIPLEXED	READY ACTIVE HIGH	NOT USED	1	0

Specifies Group 2 — Specifies Byte 2C

$D_5 = 1$ Automatically repeats entire operation when end of block is reached.

$D_5 = 0$ No affect.

$D_4 = 1$ $\overline{CE}$ and $\overline{WAIT}$ multiplexed on same pin.

$D_4 = 0$ $\overline{CE}$ only.

$D_3 = 1$ Ready active high.

$D_3 = 0$ Ready active low.

Command Byte 2D

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
1	f_4	f_3	f_2	f_1	f_0	1	1

Specifies Group 2 — Specifies Byte 2D

Hex	f_4	f_3	f_2	f_1	f_0	
C3	1	0	0	0	0	Reset
C7	1	0	0	0	1	Reset Port A Timing
CB	1	0	0	1	0	Reset Port B Timing
CF	1	0	0	1	1	Load
D3	1	0	1	0	0	Continue
AB	0	1	0	1	0	Enable Int
AF	0	1	0	1	1	Disable Int
A3	0	1	0	0	0	Reset Int
87	0	0	0	0	1	Enable DMA
83	0	0	0	0	0	Disable DMA
BB	0	1	1	1	0	Read Byte Follows
A7	0	1	0	0	1	Reset RD
BF	0	1	1	1	1	RD Status
B3	0	1	1	0	0	Force Ready
B7	0	1	1	0	1	Enable After RETI
8B	0	0	0	1	0	Reset Status

Command Byte 2D Summary

Reset: Resets all interrupt circuitry, disables interrupts and bus req. logic.

Reset Timing A or B: Resets timing for Port A or B to standard Z80-CPU timing.

Load: Zeros Byte Counter and loads Starting Address for both Ports.

Continue: Resets byte counter only. Addresses continue from present location.

Enable Interrupt: Permits interrupt to occur.

Disable Interrupt: Inhibits interrupt from occurring.

Reset Interrupt: Resets and disables all interrupt circuits (similar to RETI).

Enable DMA, Disable DMA: Overall enable or disable for all operations except interrupts: does not reset any functions.

Read Byte Follows: Next write to DMA will contain a mask to program which readable registers are to be read.

Reset RD: Next read will be from 1st register set as readable by response mask.

RD Status: Next read will be from status register.

Force Ready: Ready will be considered active regardless of the state of external RDY pin. Used for Mem-Mem operations where no RDY signal is needed.

Enable after RETI: DMA will not request bus until after it has received a RETI.

RST Status: Resets Match and End of Block status bits.

Read Byte

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
NOT USED	PORT B UPPER ADDR	PORT B LOWER ADDR	PORT A UPPER ADDR	PORT A LOWER ADDR	BYTE UPPER COUNT	BYTE LOWER COUNT	STATUS

A "1" in any bit position enables that register to be read.

Interrupt Control Byte

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
NO EFFECT	INTERRUPT ON RDY	STATUS AFFECTS INTERRUPT VECTOR	INTERRUPT VECTOR FOLLOWS	PULSE COUNT FOLLOWS	PULSE GENERATED	INTERRUPT ON MATCH FOUND	INTERRUPT AT END OF BLOCK

A "1" in a bit position selects the option.

Timing Control Byte

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
$\overline{WR}$ END	$\overline{RD}$ END	NOT USED	NOT USED	$\overline{MREQ}$ END	$\overline{IORQ}$ END	T_1	T_0

T_1	T_0	Cycle Length
0	0	4
0	1	3
1	0	2
1	1	1

A "0" in D_5, D_3, D_6, or D_7 will cause the corresponding control signal to end ½ clock time before the end of the cycle. Note: the total operation (Read and Write in Transfer or Read in Search) must be at least 2 cycles long.

Mask Byte

A zero in a given bit position will cause a compare to be performed between that bit position in the compare word register and the same bit positjon in the data being read.

Match Byte

Up to an 8-bit word to be compared to $D_0 - D_7$ during a read. See MASK BYTE.

Status Byte (Status Bits Active—Low)

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0
NOT USED	NOT USED	END OF BLK	MATCH	INT. PENDING	NOT USED	READY ACTIVE	WRITE ADDRESS START

Pulse Count

This 8-bit word is loaded into a register. At the completion of each operation, the register is compared with the lower 8-bits of the byte counter. When it compares, the $\overline{INT}$ line is pulsed (but no interrupt is generated).

Interrupt Vector

This 8-bit byte is supplied to the CPU during Interrupt acknowledge if the DMA is the highest priority interrupting device.

If bit 5 of the Interrupt Control Byte (see p. 7) has been set and the DMA has been programmed to interrupt on a given status condition then D_1 and D_2 of the vector will be modified as follows:

Vector Bits	D_2	D_1	
	0	0	INT on RDY
	0	1	Match
	1	0	End of Blk
	1	1	Match, End of Blk

DMA Programming Example

The following example will show how the DMA may be programmed to transfer data from a peripheral (Port A) to memory (Port B). The table of bytes may be stored in memory and transferred to the DMA with an output instruction such as an OTIR.

Port A	Data Flow	Port B
Peripheral Address "X"5H	⟶ Block Length 1000H Bytes	Memory Starting Address 1050H

READY from the peripheral is active high
Memory address increments on each write

		D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	HEX
1	Command Byte 1a Sets the DMA to receive Block length and Port A address and sets direction of transfer	0 Group 1	1 Blk Length Upper Follows	1 Blk Length Lower Follows	0 No Port A Upper Addr Follows	1 Port A Lower Addr Follows	1 A→B	0 1a Transfer No Search	1	6D
2	Port A Address Lower 8-bits	0	0	0	0	0	1	0	1	01
3	Block Length Lower 8-bits	0	0	0	0	0	0	0	0	00
4	Block Length Upper 8-bits	0	0	0	1	0	0	0	0	10
5	Command Byte 1b Defines Port A as peripheral with fixed addresses	0 Group 1	0 No Timing Follows	1 Fixed Addresses	X	1 Port is IO	1 This is Port "A"	0 1b	0	
6	Command Byte 1b Defines Port B as a memory with incrementing addresses	0 Group 1	0 No Timing Follows	0 Address Changes	1 Address Increments	0 Port is Memory	0 This is Port "B"	0 1b	0	14
7	Command Byte 2b Sets mode to burst, sets DMA to expect Port B starting address	1 Group 2	1 Burst Mode	0	0 No Int Cont Byte Follows	1 Port B Upper Addr Follows	1 Port B Lower Addr Follows	0 2b	1	CD
8	Port B Address Lower 8-bits	0	1	0	1	0	0	0	0	50
9	Port B Address Upper 8-bits	0	0	0	1	0	0	0	0	10
10	Command Byte 2c Sets Ready Active High	1 Group 2	X	0 No Auto Restart	0 No wait States	1 Rdy Active High	X	1 2c	0	
11	Command Byte 2d loads starting addresses and resets block counter	1 Group 2	1	0	0 Load	1	1	1 2A	1	CF
12	Command Byte 2d Enables DMA to start operation	1 Group 3	0	0	0 ENABLE DMA	0	1	1 2d	1	87

To reload the same addresses and block length for a subsequent operation, only two bytes are needed.

1. Command byte 2d Reloads port addresses and block length	11001111 Load	CF
2. Command byte 2d Enables DMA	10001011 Enable DMA	87

Absolute Maximum Ratings

Temperature Under Bias	Specified operating range.
rage Temperature	−65°C to +150°C
tage On Any Pin with Respect to Ground	−0.3V to +7V
Power Dissipation	1.5W

Note: All AC and DC characteristics remain the same for the military grade parts except I_{CC}.

I_{CC} = 200 mA.

*Comment

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Z80-DMA D.C. Characteristics

T_A = 0°C to 70°C, V_{CC} = 5V ±5% unless otherwise specified

Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	−0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}−.6		V_{CC}+.3	V	
V_{IL}	Input Low Voltage	−0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = −250 µA
I_{CC}	Power Supply Current			150	mA	t_C = 400 nsec
I_{LI}	Input Leakage Current			10	µA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float			10	µA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float			−10	µA	V_{OUT} = 0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	µA	0 < V_{IN} < V_{CC}

Z80A-DMA D.C. Characteristics

T_A = 0°C to 70°C, V_{CC} = 5V ±5% unless otherwise specified

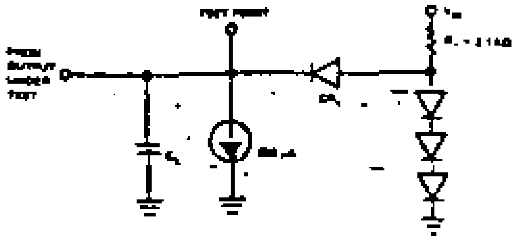
Symbol	Parameter	Min.	Typ.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	−0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC}−.6		V_{CC}+.3	V	
V_{IL}	Input Low Voltage	−0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{CC}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL} = 2 mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = −250 µA
I_{CC}	Power Supply Current		90	200	mA	t_C = 250nsec
I_{LI}	Input Leakage Current			10	µA	V_{IN} = 0 to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float			10	µA	V_{OUT} = 2.4 to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float			−10	µA	V_{OUT} = 0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	µA	0 < V_{IN} < V_{CC}

Capacitance

T_A = 25°C, f = 1 MHz

Symbol	Parameter	Max.	Unit	Test Condition
C_ϕ	Clock Capacitance	35	pF	Unmeasured Pins
C_{IN}	Input Capacitance	5	pF	Returned to Ground
C_{OUT}	Output Capacitance	10	pF	

Load Circuit for Output

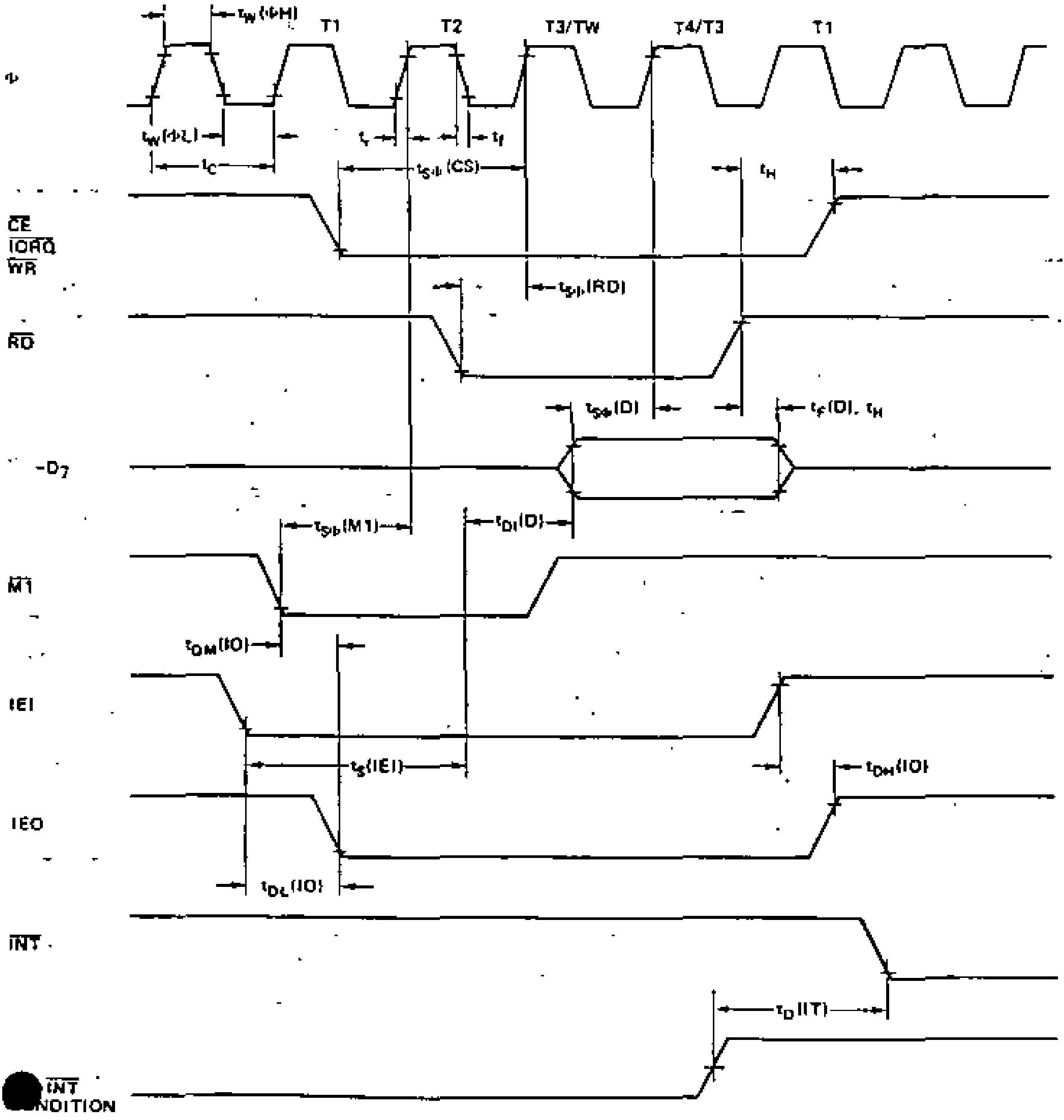


A.C. Timing Diagrams

and Z80A as a Peripheral Device (Inactive State)

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	4.2V	0.8V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV	±0.5V



DMA as a Peripheral Device (Inactive State).
0°C to 70°C, Vcc = +5V±5%, Unless Otherwise Noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c $t_{w(\Phi H)}$ $t_{w(\Phi L)}$ $t_{r,f}$	Clock Period Clock Pulse Width, Clock High Clock Pulse Width, Clock Low Clock Rise and Fall Times	400 170 170	[1] 2000 2000 30	nsec nsec nsec nsec	
	t_H	Any Hold Time for Specified Setup Time	0		nsec	
$\overline{CE}$, $\overline{WR}$ $\overline{IORQ}$	$t_{S\Phi(CS)}$	Control Signal Setup Time to Rising Edge of Φ During Write Cycle ($\overline{IORQ}$, $\overline{WR}$, $\overline{CE}$)	280		nsec	
D_{0-7}	$t_{DR(D)}$ $t_{S\Phi(D)}$ $t_{DI(D)}$ $t_{F(D)}$	Data Output Delay from Falling Edge of $\overline{RD}$ Data Setup Time to Rising Edge of Φ During Write or $\overline{M1}$ Cycle Data Output Delay from Falling Edge of $\overline{IORQ}$ During INTA Cycle Delay to Floating Bus (Output Buffer Disable Time)	 50	430 340 160	nsec nsec nsec nsec	[2] C_L = 50pF [3]
IEI	$t_{S(IEI)}$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	140		nsec	
IEO	$t_{DH(IO)}$ $t_{DL(IO)}$ $t_{DM(IO)}$	IEO Delay Time from Rising Edge of IEI IEO Delay Time from Falling Edge of IEI IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring Just Prior to $\overline{M1}$). See Note A.		210 190 300	nsec nsec nsec	C_L = 50pF
$\overline{M1}$	$t_{S\Phi(M1)}$	$\overline{M1}$ Setup Time to Rising Edge of Φ During INTA or $\overline{M1}$ Cycle. See Note B	210		nsec	
$\overline{RD}$	$t_{S\Phi(RD)}$	$\overline{RD}$ Setup Time to Rising Edge of Φ During $\overline{M1}$ Cycle	240		nsec	
$\overline{INT}$	$t_{D(IT)}$	$\overline{INT}$ Delay Time from Condition Causing $\overline{INT}$. $\overline{INT}$ generated only when DMA is inactive.		500	nsec	
BAO	$t_{DH(BO)}$ $t_{DL(BO)}$	BAO Delay from Rising Edge of BAI BAO Delay from Falling Edge of BAI	150 150	200 200	nsec nsec	

[1] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

[2] Increase $t_{DR(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

[3] Increase $t_{DI(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

A. 2.5 t_c > (N−2) $t_{DL(IO)}$ + $t_{DM(IO)}$ + $t_{S(IEI)}$ + TTL Buffer Delay, if any

Z80A-DMA as a Peripheral Device (Inactive State).
T_A = 0°C to 70°C, Vcc = +5V±5%, Unless Otherwise Noted

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c $t_{w(\Phi H)}$ $t_{w(\Phi L)}$ $t_{r,f}$	Clock Period Clock Pulse Width, Clock High Clock Pulse Width, Clock Low Clock Rise and Fall Times	250 105 105	[1] 2000 2000 30	nsec nsec nsec nsec	
	t_H	Any Hold Time for Specified Setup Time	0		nsec	
$\overline{CS}$	$t_{S\Phi(CS)}$	Control Signal Setup Time to Rising Edge of Φ During Write Cycle	145		nsec	
D_{0-7}	$t_{DR(D)}$ $t_{S\Phi(D)}$ $t_{DI(D)}$ $t_{F(D)}$	Data Output Delay from Falling Edge of $\overline{RD}$ Data Setup Time to Rising Edge of Φ During Write or $\overline{M1}$ Cycle Data Output Delay from Falling Edge of $\overline{IORQ}$ During INTA Cycle Delay to Floating Bus (Output Buffer Disable Time)	 50	380 250 110	nsec nsec nsec nsec	[2] C_L = 50pF [3]
IEI	$t_{S(IEI)}$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	140		nsec	
IEO	$t_{DH(IO)}$ $t_{DL(IO)}$ $t_{DM(IO)}$	IEO Delay Time from Rising Edge of IEI IEO Delay Time from Falling Edge of IEI IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring Just Prior to $\overline{M1}$). See Note A.		160 130 190	nsec nsec nsec	C_L = 50pF
$\overline{M1}$	$t_{S\Phi(M1)}$	$\overline{M1}$ Setup Time to Rising Edge of Φ During INTA or $\overline{M1}$ Cycle. See Note B	90		nsec	
$\overline{RD}$	$t_{S\Phi(RD)}$	$\overline{RD}$ Setup Time to Rising Edge of Φ During $\overline{M1}$ Cycle	115		nsec	
$\overline{INT}$	$t_{D(IT)}$	$\overline{INT}$ Delay Time from Condition Causing $\overline{INT}$. $\overline{INT}$ generated only when DMA is inactive.		900	nsec	
BAO	$t_{DH(BO)}$ $t_{DL(BO)}$	BAO Delay from Rising Edge of BAI BAO Delay from Falling Edge of BAI	150 150	200 200	nsec nsec	

[1] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

[2] Increase $t_{DR(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

[3] Increase $t_{DI(D)}$ by 10 nsec for each 50pF increase in loading up to 200pF max.

A. 2.5 t_c > (N−2) $t_{DL(IO)}$ + $t_{DM(IO)}$ + $t_{S(IEI)}$ + TTL Buffer Delay, if any

780-DMA as a Bus Controller (Active State)
A = 0°C to 70°C, Vcc = +5V±5%, Unless Otherwise Noted.

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c $t_{w(\Phi H)}$ $t_{w(\Phi L)}$ $t_{r,f}$	Clock Period Clock Pulse Width, Clock High Clock Pulse Width, Clock Low Clock Rise and Fall Time	4 180 180	[12] 2000 2000 30	μsec nsec nsec nsec	
A_{0-15}	$t_{D(AD)}$ $t_{F(AD)}$ t_{acm} t_{aci} t_{ca} t_{caf}	Address Output Delay Delay to Float Address Stable Prior to $\overline{MREQ}$ (Memory Cycle) Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle) Address Stable from $\overline{RD}$ or $\overline{WR}$ Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	 [1] [2] [3] [4]	145 110	nsec nsec nsec nsec nsec nsec	 C_L = 50pF D D D D
D_{0-7}	$t_{D(D)}$ $t_{F(D)}$ $t_{S\Phi(D)}$ $t_{S\overline{\Phi}(D)}$ t_{dcm} t_{dci} t_{cdf}	Data Output Delay Delay to Float During Write Cycle Data Setup Time to Rising Edge of Clock During Read When Rising Edge Ends $\overline{RD}$ Data Setup Time to Falling Edge of Clock During Read When Falling Edge Ends $\overline{RD}$ Data Stable Prior to $\overline{WR}$ (Memory Cycle) Data Stable Prior to $\overline{WR}$ (I/O Cycle) Data Stable From $\overline{WR}$	 50 60 [5] [6] [7]	230 90	nsec nsec nsec nsec nsec nsec nsec	 C_L = 200pF D D D
	t_H	Any Hold Time for Setup Time	0		nsec	
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$ $t_{DH\Phi(MR)}$ $t_{DH\overline{\Phi}(MR)}$ $t_{DL\Phi(MR)}$ $t_{w(\overline{MRL})}$ $t_{w(\overline{MRH})}$	$\overline{MREQ}$ Delay from Falling Edge of Clock, $\overline{MREQ}$ Low $\overline{MREQ}$ Delay from Rising Edge of Clock, $\overline{MREQ}$ High $\overline{MREQ}$ Delay from Falling Edge of Clock, $\overline{MREQ}$ High $\overline{MREQ}$ Delay from Falling Edge of Clock, $\overline{MREQ}$ Low Pulse Width, $\overline{MREQ}$ Low Pulse Width, $\overline{MREQ}$ High	 [8] [9]	100 100 100 100	nsec nsec nsec nsec nsec nsec	 C_L = 50pF D D
$\overline{IORQ}$	$t_{DL\Phi(IR)}$ $t_{DL\overline{\Phi}(IR)}$ $t_{DH\Phi(IR)}$ $t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay from Rising Edge of Clock, $\overline{IORQ}$ Low $\overline{IORQ}$ Delay from Falling Edge of Clock, $\overline{IORQ}$ Low $\overline{IORQ}$ Delay from Rising Edge of Clock, $\overline{IORQ}$ High $\overline{IORQ}$ Delay from Falling Edge of Clock, $\overline{IORQ}$ High		90 110 100 110	nsec nsec nsec nsec	C_L = 50pF
$\overline{RD}$	$t_{DL\Phi(RD)}$ $t_{DL\overline{\Phi}(RD)}$ $t_{DH\Phi(RD)}$ $t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay from Rising Edge of Clock, $\overline{RD}$ Low $\overline{RD}$ Delay from Falling Edge of Clock, $\overline{RD}$ Low $\overline{RD}$ Delay from Rising Edge of Clock, $\overline{RD}$ High $\overline{RD}$ Delay from Falling Edge of Clock, $\overline{RD}$ High		100 130 100 110	nsec nsec nsec nsec	C_L = 50pF
$\overline{WR}$	$t_{DL\Phi(WR)}$ $t_{DL\overline{\Phi}(WR)}$ $t_{DH\overline{\Phi}(WR)}$ $t_{DH\Phi(WR)}$ $t_{w(\overline{WRL})}$	$\overline{WR}$ Delay from Rising Edge of Clock, $\overline{WR}$ Low $\overline{WR}$ Delay from Falling Edge of Clock, $\overline{WR}$ Low $\overline{WR}$ Delay from Falling Edge of Clock, $\overline{WR}$ High $\overline{WR}$ Delay from Rising Edge of Clock, $\overline{WR}$ High Pulse Width, $\overline{WR}$ Low	 [10]	80 90 100 100	nsec nsec nsec nsec nsec	C_L = 50pF
$\overline{WAIT}$	$t_{s(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{BUSRQ}$	$t_{D(BQ)}$	$\overline{BUSRQ}$ Delay Time from Rising Edge of Clock	100		nsec	
	t_{FC}	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		100	nsec	

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 75$
[2] $t_{aci} = t_c - 80$
[3] $t_{ca} = t_{w(\Phi L)} + t_r - 40$
[4] $t_{caf} = t_{w(\Phi L)} + t_r - 60$

[5] $t_{dcm} = t_c - 180$
[6] $t_{dci} = t_{w(\Phi L)} + t_r - 180$
[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 80$

[8] $t_{w(MRL)} = t_c - 40$
[9] $t_{w(MRH)} = t_c - 40$ Std. CPU Timing
$t_{w(MRH)} = t_{w(\Phi H)} + t_f - 30$ Variable Cycle

[10] $t_{w(WR)} = t_c - 40$ Std. CPU Timing
$t_{w(WR)} = t_{w(\Phi H)} + t_f - 30$ Variable Cycle

NOTES.

A. Data should be enabled onto the DMA data bus when $\overline{RD}$ is active.

B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.

C. Output Delay vs. Loaded Capacitance
TA = 70°C Vcc = +5V±5%
[1] ΔC_L = −100pF ($A_0 - A_{15}$ and Control Signals), add 30 nsec to timing shown.

D. During Standard CPU Timing

A-DMA as a Bus Controller (Active State)
= 0°C to 70°C, Vcc = +5V±5%, Unless Otherwise Noted.

SIGNAL	SYMBOL	PARAMETER	MIN	MAX	UNIT	COMMENTS
Φ	t_c $t_{w(\Phi H)}$ $t_{w(\Phi L)}$ $t_{r,f}$	Clock Period Clock Pulse Width, Clock High Clock Pulse Width, Clock Low Clock Rise and Fall Time	.25 110 110 	[12] 2000 2000 30	μsec nsec nsec nsec	
A_{0-15}	$t_{D(AD)}$ $t_{F(AD)}$ t_{acm} t_{aci} t_{ca} t_{caf}	Address Output Delay Delay to Float Address Stable Prior to $\overline{MREQ}$ (Memory Cycle) Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle) Address Stable from $\overline{RD}$ or $\overline{WR}$ Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	 [1] [2] [3] [4]	110 90 	nsec nsec nsec nsec nsec nsec	 C_L = 50pF D D D D
D_{0-7}	$t_{D(D)}$ $t_{F(D)}$ $t_{S\Phi(D)}$ $t_{S\overline{\Phi}(D)}$ t_{dcm} t_{dci} t_{cdf}	Data Output Delay Delay to Float During Write Cycle Data Setup Time to Rising Edge of Clock During Read When Rising Edge Ends $\overline{RD}$ Data Setup Time to Falling Edge of Clock During Read When Falling Edge Ends $\overline{RD}$ Data Stable Prior to $\overline{WR}$ (Memory Cycle) Data Stable Prior to $\overline{WR}$ (I/O Cycle) Data Stable From $\overline{WR}$	 35 50 [5] [6] [7]	180 	nsec nsec nsec nsec nsec nsec nsec	 C_L = 200pF D D D
	t_H	Any Hold Time for Setup Time		0	nsec	
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$ $t_{DH\Phi(MR)}$ $t_{DH\overline{\Phi}(MR)}$ $t_{w(\overline{MRL})}$ $t_{w(\overline{MRH})}$	$\overline{MREQ}$ Delay from Falling Edge of Clock, $\overline{MREQ}$ Low $\overline{MREQ}$ Delay from Rising Edge of Clock, $\overline{MREQ}$ High $\overline{MREQ}$ Delay from Falling Edge of Clock, $\overline{MREQ}$ High Pulse Width, $\overline{MREQ}$ Low Pulse Width, $\overline{MREQ}$ High	 [8] [9]	75 75 75 	nsec nsec nsec nsec nsec	 C_L = 50pF D D
$\overline{IORQ}$	$t_{DL\Phi(IR)}$ $t_{DH\Phi(IR)}$ $t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay from Rising Edge of Clock, $\overline{IORQ}$ Low $\overline{IORQ}$ Delay from Rising Edge of Clock, $\overline{IORQ}$ High $\overline{IORQ}$ Delay from Falling Edge of Clock, $\overline{IORQ}$ High		75 80 80	nsec nsec nsec	C_L = 50pF
$\overline{RD}$	$t_{DL\Phi(RD)}$ $t_{DL\overline{\Phi}(RD)}$ $t_{DH\Phi(RD)}$ $t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay from Rising Edge of Clock, $\overline{RD}$ Low $\overline{RD}$ Delay from Falling Edge of Clock, $\overline{RD}$ Low $\overline{RD}$ Delay from Rising Edge of Clock, $\overline{RD}$ High $\overline{RD}$ Delay from Falling Edge of Clock, $\overline{RD}$ High		75 85 75 80	nsec nsec nsec nsec	C_L = 50pF
$\overline{WR}$	$t_{DL\Phi(WR)}$ $t_{DL\overline{\Phi}(WR)}$ $t_{DH\overline{\Phi}(WR)}$ $t_{DH\Phi(WR)}$ $t_{w(\overline{WRL})}$	$\overline{WR}$ Delay from Rising Edge of Clock, $\overline{WR}$ Low $\overline{WR}$ Delay from Falling Edge of Clock, $\overline{WR}$ Low $\overline{WR}$ Delay from Falling Edge of Clock, $\overline{WR}$ High $\overline{WR}$ Delay from Rising Edge of Clock, $\overline{WR}$ High Pulse Width, $\overline{WR}$ Low	 [10]	80 80 80 80 	nsec nsec nsec nsec nsec	C_L = 50pF
$\overline{WAIT}$	$t_{s(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{BUSRQ}$	$t_{D(BQ)}$	$\overline{BUSRQ}$ Delay Time from Rising Edge of Clock	100		nsec	
	$t_{F(C)}$	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		80	nsec	

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 75$
[2] $t_{aci} = t_c - 80$
[3] $t_{ca} = t_{w(\Phi L)} + t_r - 40$
[4] $t_{caf} = t_{w(\Phi L)} + t_r - 60$

[5] $t_{dcm} = t_c - 180$
[6] $t_{dci} = t_{w(\Phi L)} + t_r - 180$
[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 80$

[8] $t_{w(MRL)} = t_c - 40$
[9] $t_{w(MRH)} = t_c - 40$ Std. CPU Timing
$t_{w(MRH)} = t_{w(\Phi H)} + t_f - 30$ Variable 1 Cycle

[10] $t_{w(WR)} = t_c - 40$ Std. CPU Timing
$t_{w(WR)} = t_{w(\Phi H)} + t_f - 30$ Variable 1 Cycle.

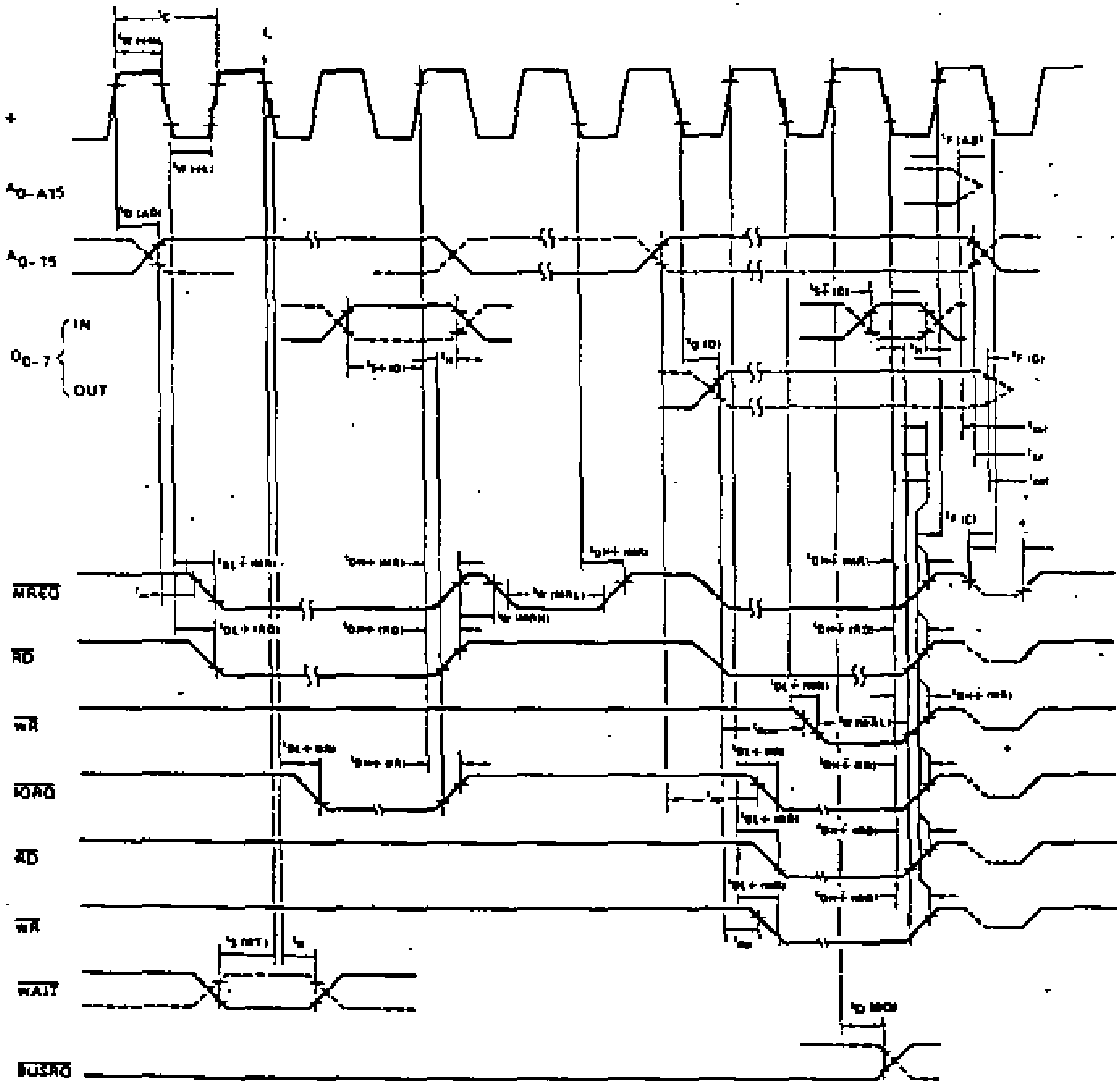
NOTES.
A. Data should be enabled onto the DMA data bus when $\overline{RD}$ is active.
B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.
C. Output Delay vs. Loaded Capacitance
TA = 70°C Vcc = +5V±5%
[1] ΔC_L = +100pF (A_0–A_{15} and Control Signals), add 30 nsec to timing shown.
D. During Standard CPU Timing

A.C. Timing Diagrams

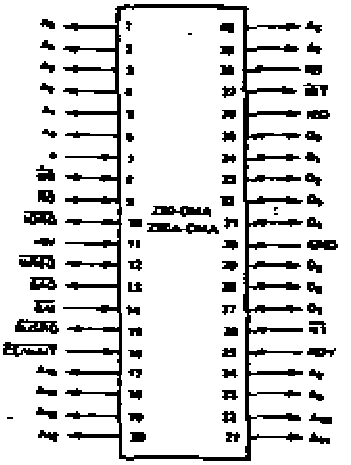
Z80 and Z80A as a Bus Controller (Active State)

Timing measurements are made at the following voltages, unless otherwise specified:

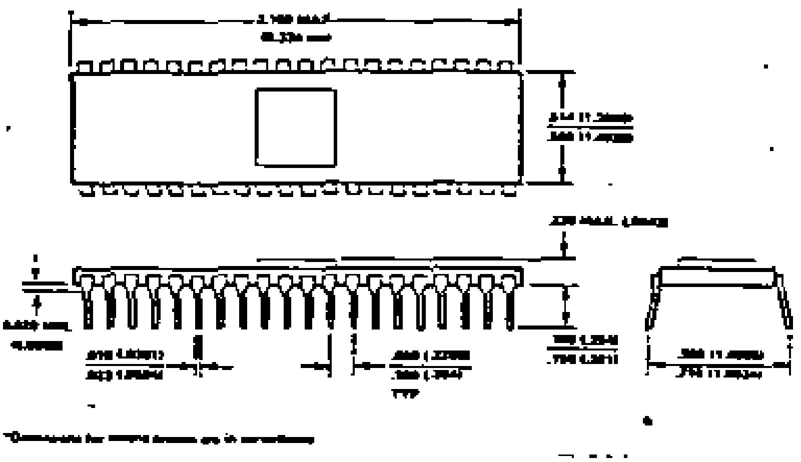
	"1"	"0"
CLOCK	4.2V	0.8V
OUTPUT	2.0V	0.8V
INPUT	2.0V	0.8V
FLOAT	ΔV	= +0.5V



Package Configuration



Package Outline



EASTERN

Hallmark Electronics
4739 Commercial Drive
Huntsville, AL 35805
TEL 205 837 8700
TWX 810 726 2187

Hallmark Electronics
1302 West McNab Road
Fort Lauderdale, FL 33309
TEL 305 971 9280
TWX 510 956 9720

lmark Electronics
3 Lake Ellenor Drive
rlando, FL 32809
TEL 305 855 3020
TWX 810 850 0183

Hallmark Electronics
3355 Amberton Drive
Baltimore MD 21227
TEL 301 796 9300
TWX 710 862 1942

Hallmark Electronics
1208 Front Street
Building K
Raleigh, NC 27609
TEL 919 832 4465
TWX 510 928 1831

Hallmark Electronics
Pike Industrial Park
Huntingdon Valley, PA
TEL 215 355 7300
TWX 510 667 1750

Summit
916 Main Street
Buffalo, NY 14202
TEL 716 884 3450

Wilshire Electronics
2554 State Street
Hamden, CT 06517
TEL 203 281 1166
TWX 800 922 1734

Wilshire Electronics
1855 New Highway
Farmingdale, LI, NY 11735
TEL 516 293 5775
TWX 212 895 8707

Wilshire Electronics
One Wilshire Road
Burlington, MA 01803
TEL 617 272 8200
TWX 710 332 6359

Wilshire Electronics
1111 Paulison Avenue
Clifton, NJ 07015
TEL 201 340 1900
TWX 710 989 7052

MIDWESTERN

Hallmark Electronics
180 Crossen Avenue
Elk Grove Village, IL 60076
TEL 312 437 8800
TWX 910 223 3645

Hallmark Electronics
11870 West 91st Street
Congleton Industrial Park
Shawnee Mission, KS 66214
TEL 913 888 4747
TWX 910 749 6620

Hallmark Electronics
9201 Penn Avenue South
Suite 10
Bloomington, MN 55435
TEL 612 884 9056
TWX 910 576 3187

Hallmark Electronics
13789 Rider Trail
Earth City, MO 63045
TEL 314 291 5350
TWX 910 760 0671

Hallmark Electronics
6969 Worthington Galena Road
Worthington, OH 43085
TEL 614 846 1882

Hallmark Electronics
4846 S. 83rd E. Avenue
Tulsa, OK 74145
TEL 918 835 8458
TWX 910 845 2290

Hallmark Electronics
3100—A Industrial Terrace
Austin, TX 78758
TEL 512 837 2841
TWX 910 874 2031

Hallmark Electronics
9333 Forest Lane
Dallas, TX 75222
TEL 214 234-7400
TWX 910 867 4721

Hallmark Electronics
8000 W. Glenn
Houston, TX 77063
TEL 713 781 6100
TWX 910 881 2711

Hallmark Electronics
237 South Curtis
West Allis, WI 53214
TEL 414 476 1270
TWX 910 262 3186

RM Electronics
4860 South Division
Kentwood, MI 49508
TEL 616 531 9300
TWX 810 273 8779

MOUNTAIN

Century Electronics
121 Elizabeth, NE
Albuquerque, NM 87123
TEL 505 292 2700
TWX 910 989 0625

Century Electronics
2150 South 300 West
Salt Lake City, UT 84115
TEL 801 487 8551
TWX 910 925 5686

Century Electronics
8155 West 48th Avenue
Wheatridge, CO 80033
TEL 303 424 1985
TWX 910 938 0393

R. V. Weatherford Co.
3905 South Mariposa
Englewood, CO 80110
TEL 303 761 5432
TWX 910 933 0173

WESTERN

Intermark Electronics
1802 E. Carnegie Avenue
Santa Ana, CA 92705
TEL 714 540 1322
TWX 910 595 1583

Intermark Electronics
4040 Sorrento Valley Blvd.
San Diego, CA 92121
TEL 714 279 5200
714 453 9005
TWX 910 335 1515

Intermark Electronics
1020 Stewart Drive
Sunnyvale, CA 94086
TEL 408 738 1111
TWX 910 339 9312

R.V. Weatherford Co.
6921 San Fernando Road
Glendale, CA 91201
TEL 213 849 3451
TWX 910 498 2223

R.V. Weatherford Co.
1550 Babbitt Avenue
Anaheim, CA 92805
TEL 714 634 9600
TWX 910 593 1334

R.V. Weatherford Co.
1095 East Third Street
Pomona, CA 91765
TEL 714 623 1261
TWX 910 581 3811

R.V. Weatherford Co.
3240 Hillview Avenue
Stanford Industrial Park
Palo Alto, CA 94304
TEL 415 493 5373

R.V. Weatherford Co.
3311 W. Earll Drive
Phoenix, AZ 85017
TEL 602 272 7144
TWX 910 951 0636

Sterling Electronics
5608 6th Avenue South
Seattle, WA 98108
TEL 206 762 9100
TLX 32-9652

CANADA

Future Electronics
5647 Ferrier Street
Montreal, Quebec,
CANADA H4P 2K5

Ordering Information

C – Ceramic
P – Plastic
S – Standard 5V ±5%, 0° to 70°C
E – Extended 5V ± 5% -40° to 85°C
M – Military 5V ±10% -55° to 125°C

Example:

Z80-DMA CS (Ceramic–Standard range)

Printed in U.S.A.

Z80®-CTC
Z80®A-CTC

Product Specification

APRIL 1978

The Zilog Z80 product line is a complete set of microcomputer components, development systems and support software. The Z80 microcomputer component set includes all of the circuits necessary to build high-performance microcomputer systems with virtually no other logic and a minimum number of low cost standard memory elements.

The Z80-Counter Timer Circuit (CTC) is a programmable, four channel device that provides counting and timing functions for the Z80-CPU. The Z80-CPU configures the Z80-CTC's four independent channels to operate under various modes and conditions as required.

Structure

- N-Channel Silicon Gate Depletion Load Technology
- 28 Pin DIP
- Single 5 volt supply
- Single phase 5 volt clock
- Four independent programmable 8-bit counter/16-bit timer channels

Features

- Each channel may be selected to operate in either a counter mode or timer mode.
- Programmable interrupts on counter or timer states.
- A time constant register automatically reloads the down counter at zero and the cycle is repeated.
- Readable down counter indicates number of counts-to-go until zero.
- Selectable 16 or 256 clock prescaler for each timer channel.
- Selectable positive or negative trigger may initiate timer operation.
- Three channels have zero count/timeout outputs capable of driving Darlington transistors.
- Daisy chain priority interrupt logic included to provide for automatic interrupt vectoring without external logic.
- All inputs and outputs fully TTL compatible.
- Outputs directly compatible with Z80-SIO.

CTC Architecture

A block diagram of the Z80-CTC is shown in figure 1. The internal structure of the Z80-CTC consists of a Z80-CPU bus interface, internal control logic, four counter channels, and interrupt control logic. Each channel has an interrupt vector for automatic interrupt vectoring, and interrupt priority is determined by channel number with channel Ø having the highest priority.

The channel logic is composed of 2 registers, 2 counters and control logic as shown in figure 2. The registers include an 8-bit time constant register and an 8-bit channel control register. The counters include an 8-bit readable down counter and an 8-bit prescaler. The prescaler may be programmed to divide the system clock by either 16 or 256.

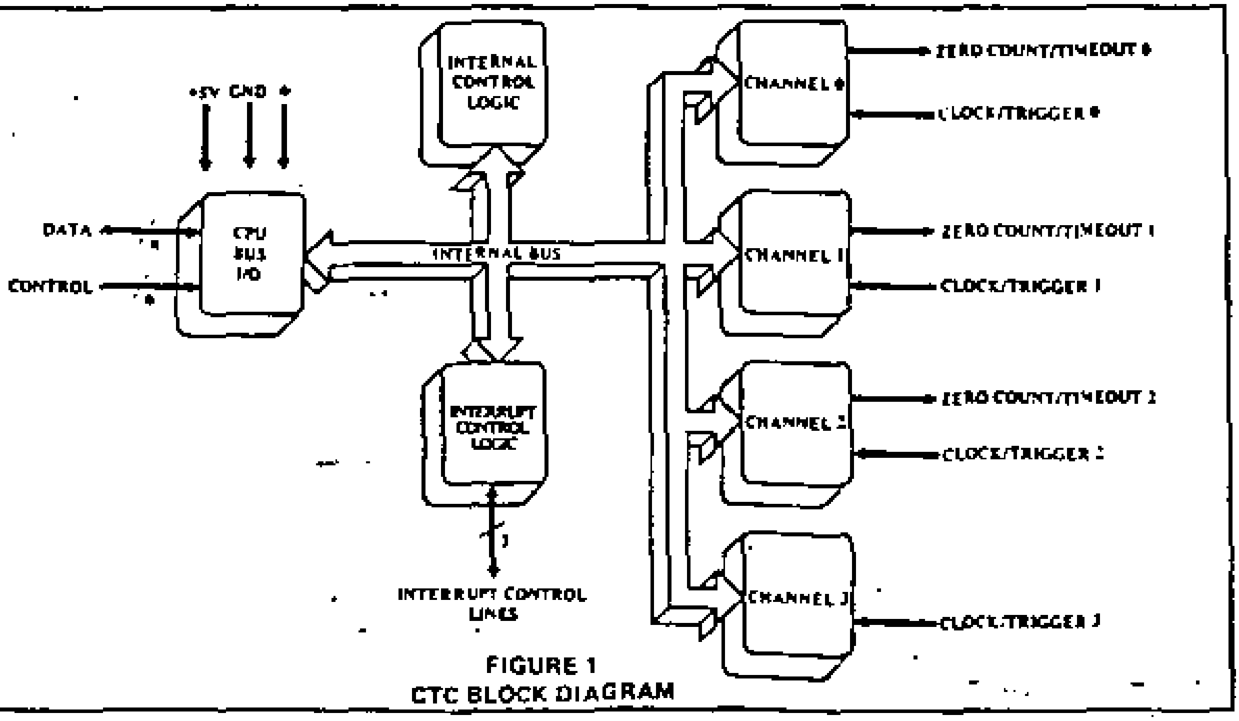


FIGURE 1
CTC BLOCK DIAGRAM

Constant Register – 8 bits, loaded by the CPU to initialize and re-load Down Counter at a count of zero.

Channel Control Register – 8 bits, loaded by the CPU to select the mode and conditions of channel operation.

Down Counter – 8 bits, loaded by the Time Constant Register under program control and automatically at a count of zero. At any time, the CPU can read the number of counts-to-go until a zero count. This counter is decremented by the prescaler in timer mode and CLK/TRIG in counter mode.

Prescaler – 8 bit counter, divides system clock by 16 or 256 for decrementing Down Counter. It is used in timer mode only.

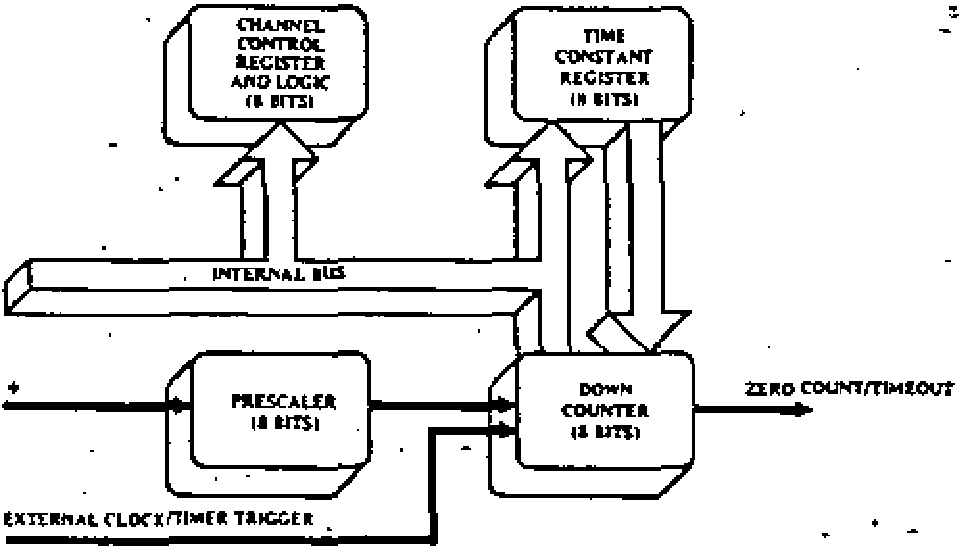
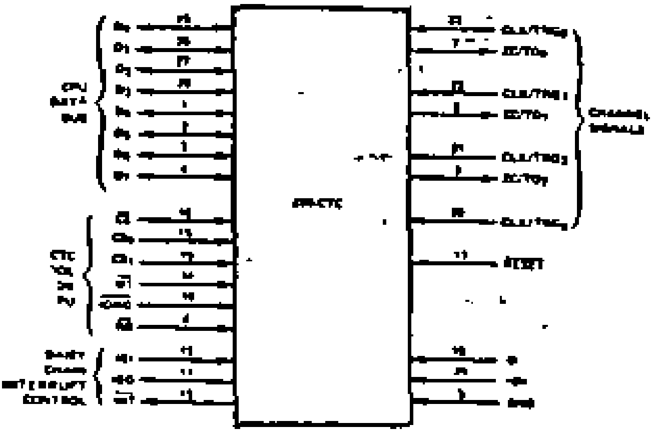

CHANNEL CONTROL REGISTER AND LOGIC (8 BITS)
TIME CONSTANT REGISTER (8 BITS)
INTERNAL BUS
Φ
PRESCALER (8 BITS)
DOWN COUNTER (8 BITS)
ZERO COUNT/TIMEOUT
EXTERNAL CLOCK/TIMER TRIGGER

FIGURE 2
CHANNEL BLOCK DIAGRAM

Z80-CTC Pin Description



CLK/TRG0 Channel 0 External Clock or Timer Trigger (Input).

CLK/TRG1 Channel 1 External Clock or Timer Trigger (Input)

CLK/TRG2 Channel 2 External Clock or Timer Trigger (Input)

CLK/TRG3 Channel 3 External Clock or Timer Trigger (Input)

ZC/TO0 Channel 0 Zero Count or Timeout (output, active high)

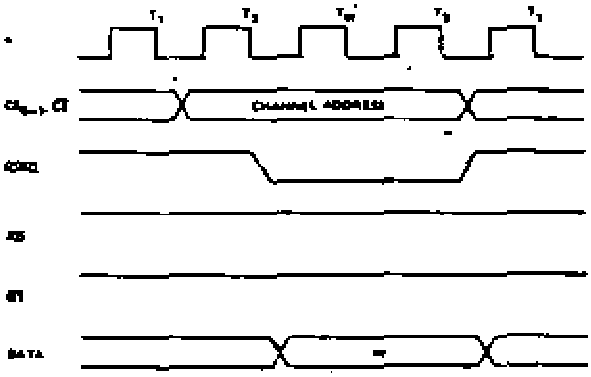
Z80-CTC Pin Description (continued)

ZC/TO_1	Channel 1 Zero Count or Timeout (output, active high)
ZC/TO_2	Channel 2 Zero Count or Timeout (output, active high)
$CS_1 - CS_\emptyset$	Channel Select (input, active high). These form a 2-bit binary address of the channel to be accessed.
$D_7 - D_\emptyset$	Z80-CPU Data Bus (bidirectional, tristate)
$\overline{CE}$	Chip Enable (input, active low)
Φ	System Clock (input)
$\overline{M1}$	Machine Cycle One Signal from Z80-CPU (input, active low)
$\overline{IORQ}$	Input/Output Request from Z80-CPU (input, active low)
$\overline{RD}$	Read Cycle Status from the Z80-CPU (input, active low)
IEI	Interrupt Enable In (input, active high)
IEO	Interrupt Enable Out (output, active high). IEI and IEO form a daisy chain connection for priority interrupt control
$\overline{INT}$	Interrupt Request (output, open drain, active low)
$\overline{RESET}$	RESET stops all channels from counting and resets channel interrupt enable bits in all control registers. During reset time $ZC/TO_{\emptyset-2}$ and $\overline{INT}$ go to the inactive states, IEO reflects the state of IEI, and the data bus output drivers go to the high impedance state (input, active low)

Timing Waveforms

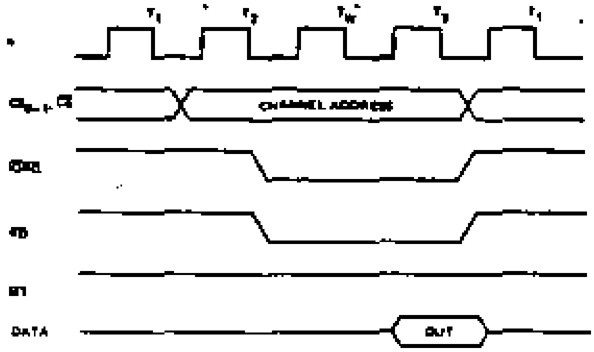
CTC WRITE CYCLE

Illustrated here is the timing for loading a channel control word, time constant and interrupt vector. No wait states are allowed for writing to the CTC other than the automatically inserted (T_W*). Since the CTC does not receive a specific write signal, it internally generates its own from the lack of an $\overline{RD}$ signal.



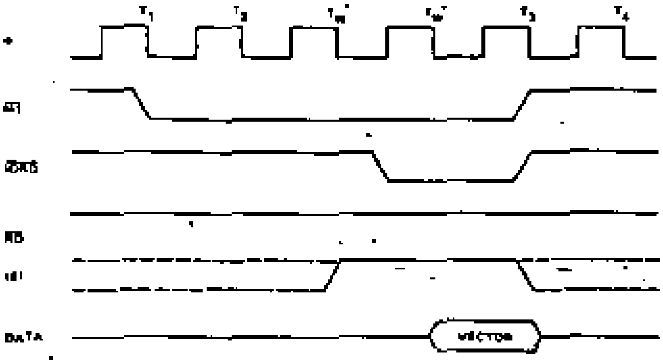
CTC READ CYCLE

Illustrated here is the timing for reading a channel's Down Counter when in Counter Mode. The value read onto the data bus reflects the number of external clock's rising edges prior to the rising edge of cycle (T_2). No wait states are allowed for reading the CTC other than the automatically inserted (T_W*).



INTERRUPT ACKNOWLEDGE CYCLE

Some time after an interrupt is requested by the CTC, the CPU will send out an interrupt acknowledge ($\overline{M1}$ and $\overline{IORQ}$). During this time the interrupt logic of the CTC will determine the highest priority channel which is requesting an interrupt. To insure that the daisy chain enable lines stabilize, channels are inhibited from changing their interrupt request status when $\overline{M1}$ is active. If the CTC Interrupt Enable Input (IEI) is active, then the highest priority interrupting channel places the contents of its interrupt vector register onto the Data Bus when $\overline{IORQ}$ goes active. Additional wait cycles are allowed.

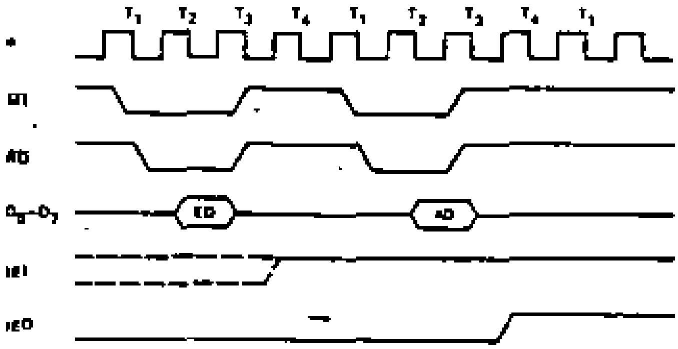


RETURN FROM INTERRUPT CYCLE

If a Z80 peripheral device has no interrupt pending and is not under service, then its IEO = IEI. If it has an interrupt under service (i.e. it has already interrupted and received an interrupt acknowledge) then its IEO is always low, inhibiting lower priority chips from interrupting. If it has an interrupt pending which has not yet been acknowledged, IEO will be low unless an "ED" is decoded as the first byte of a two byte opcode. In this case, IEO will go high until the next opcode byte is decoded, whereupon it will again go low. If the second byte of the opcode was a "4D" then the opcode was an RETI instruction.

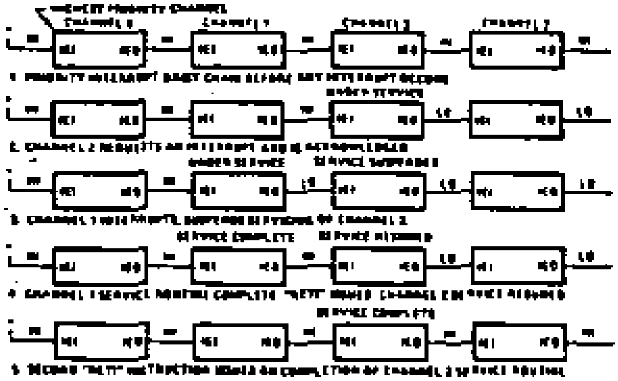
After an "ED" opcode is decoded, only the peripheral device which has interrupted and is currently under service will have its IEI high and its IEO low. This device is the highest priority device in the daisy chain which has received an interrupt acknowledge. All other peripherals have IEI = IEO. If the next opcode byte decoded is "4D", this peripheral device will reset its "interrupt under service" condition.

Wait cycles are allowed in the $\overline{M1}$ cycles.



DAISY CHAIN INTERRUPT SERVICING

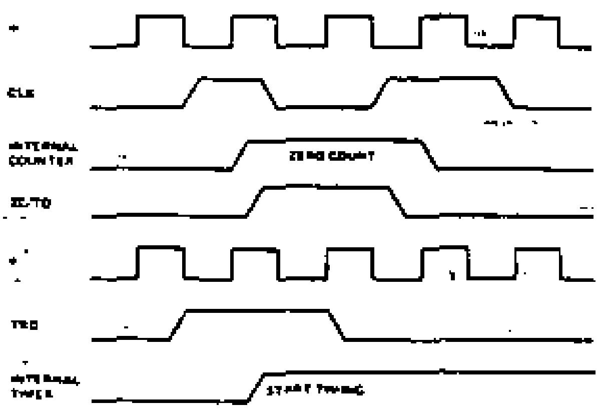
Illustrated at right is a typical nested interrupt sequence ch may occur in the CTC. In this sequence channel 2 rrupts and is granted service. While this channel is being serviced, higher priority channel 1 interrupts and is granted service. The service routine for the higher priority channel is completed and a RETI instruction is executed to indicate to the channel that its routine is complete. At this time the service routine of lower priority channel 2 is completed.


HIGHEST PRIORITY CHANNEL
CHANNEL 0
CHANNEL 1
CHANNEL 2
CHANNEL 3
IEI
IEO
1. PRIORITY INTERRUPT DAISY CHAIN BEFORE ANY INTERRUPT OCCURS
2. CHANNEL 2 REQUESTS AN INTERRUPT AND IS ACKNOWLEDGED
UNDER SERVICE
SERVICE SUSPENDED
3. CHANNEL 1 INTERRUPTS, SUSPENDING SERVICE OF CHANNEL 2
SERVICE COMPLETE
SERVICE RESUMED
4. CHANNEL 1 SERVICE ROUTINE COMPLETE, "RETI" ISSUED, CHANNEL 2 SERVICE RESUMED
SERVICE COMPLETE
5. SECOND "RETI" INSTRUCTION ISSUED ON COMPLETION OF CHANNEL 2 SERVICE ROUTINE

CTC COUNTING AND TIMING

In the counter mode the rising or falling edge of the CLK input causes the counter to be decremented. The edge is detected totally asynchronously and must have a minimum CLK pulse width. However, the counter is synchronous with Φ therefore a setup time must be met when it is desired to have the counter decremented by the next rising edge of Φ.

In the timer mode the prescaler may be enabled by a rising or falling edge on the TRG input. As in the counter mode, the edge is detected totally asynchronously and must have a minimum TRG pulse width. However, when timing is to with respect to the next rising edge of Φ a setup time be met. The prescaler counts rising edges of Φ.



SELECTING AN OPERATING MODE

When selecting a channel's operating mode, bit Ø is set to 1 to indicate this word is to be stored in the channel control register.

D7	D6	D5	D4	D3	D2	D1	D0
INTERRUPT ENABLE	MODE	RANGE	SLOPE	TRIGGER	LOAD TIME CONSTANT	RESET	1
		TIMER (TIMER MODE ONLY)		TIMER (TIMER MODE ONLY)			

Bit 7 = Ø Channel interrupts disabled.

Bit 7 = 1 Channel interrupts enabled to occur every time Down Counter reaches a count of zero. Setting Bit 7 does not let a preceding count of zero cause an interrupt.

Bit 6 = Ø Timer Mode – Down counter is clocked by the prescaler. The period of the counter is:

$t_c \cdot P \cdot TC$

t_c = system clock period

P = prescale of 16 or 256

TC = 8 bit binary programmable time constant (256 max)

Bit 6 = 1 Counter Mode – Down Counter is clocked by external clock. The prescaler is not used.

Bit 5 = Ø Timer Mode Only–System clock Φ is divided by 16 in prescaler.

Bit 5 = 1 Timer Mode Only–System clock Φ is divided by 256 in prescaler.

Bit 4 = Ø Timer Mode – negative edge trigger starts timer operation.
Counter Mode – negative edge decrements the down counter.

Bit 4 = 1 Timer Mode – positive edge trigger starts timer operation.
Counter Mode – positive edge decrements the down counter.

Bit 3 = Ø Timer Mode Only – Timer begins operation on the rising edge of T_2 of the machine cycle following the one that loads the time constant.

Bit 3 = 1 Timer Mode Only – External trigger is valid for starting timer operation after rising edge of T_2 of the machine cycle following the one that loads the time constant. The Prescaler is decremented 2 clock cycles later if the setup time is met, otherwise 3 clock cycles.

Bit 2 = Ø No time constant will follow the channel control word. One time constant must be written to the channel to initiate operation.

Bit 2 = 1 The time constant for the Down Counter will be the next word written to the selected channel. If a time constant is loaded while a channel is counting, the present count will be completed before the new time constant is loaded into the Down Counter.

Bit 1 = Ø Channel continues counting.

Bit 1 = 1 Stop operation. If Bit 2 = 1 channel will resume operation after loading a time constant, otherwise a new control word must be loaded.

LOADING A TIME CONSTANT

An 8-bit time constant is loaded into the Time Constant register following a channel control word with bit 2 set. All zeros indicate a time constant of 256.

D7	D6	D5	D4	D3	D2	D1	D0
TC7	TC6	TC5	TC4	TC3	TC2	TC1	TC0

LOADING AN INTERRUPT VECTOR

The Z80-CPU requires that an 8-bit interrupt vector be supplied by the interrupting channel. The CPU forms the address for the interrupt service routine of the channel using this vector. During an interrupt acknowledge cycle the vector is placed on the Z80 Data Bus by the highest priority channel requesting service at that time. The desired interrupt vector is loaded into the CTC by writing into channel Ø with a zero in DØ. D_7-D_3 contain the stored interrupt vector, D_2 and D_1 are not used in loading the vector. When the CTC responds to an interrupt acknowledge, these two bits contain the binary code of the highest priority channel which requested the interrupt and D_0 contains a zero since the address of the interrupt service routine starts at an even byte. Channel Ø is the highest priority channel.

D7	D6	D5	D4	D3	D2	D1	D0
V7	V6	V5	V4	V3	X	X	0

A.C. Characteristics

T_A = 0° C to 70° C, Vcc = +5 V ± 5%, unless otherwise noted

Signal	Symbol	Parameter	Min	Max	Unit	Comments
Φ	t_C	Clock Period	400	[1]	ns	
	$t_W(\Phi H)$	Clock Pulse Width, Clock High	170	2000	ns	
	$t_W(\Phi L)$	Clock Pulse Width, Clock Low	170	2000	ns	
	t_r, t_f	Clock Rise and Fall Times		30	ns	
	t_H	Any Hold Time for Specified Setup Time	0		ns	
CS, $\overline{CE}$, etc.	$t_{S\Phi}(CS)$	Control Signal Setup Time to Rising Edge of Φ During Read or Write Cycle	160		ns	
D_0-D_7	$t_{DR}(D)$	Data Output Delay from Rising Edge of $\overline{RD}$ During Read Cycle		480	ns	[2]
	$t_{S\Phi}(D)$	Data Setup Time to Rising Edge of Φ During Write or M1 Cycle	60		ns	
	$t_{DI}(D)$	Data Output Delay from Falling Edge of IORQ During INTA Cycle		340	ns	[2]
	$t_F(D)$	Delay to Floating Bus (Output Buffer Disable Time)		230	ns	
IEI	$t_S(IEI)$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	200		ns	
IEO	$t_{DH}(IO)$	IEO Delay Time from Rising Edge of IEI		220	ns	[3]
	$t_{DL}(IO)$	IEO Delay Time from Falling Edge of IEI		190	ns	[3]
	$t_{DM}(IO)$	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring just Prior to $\overline{M1}$)		300	ns	[3]
$\overline{IORQ}$	$t_{S\Phi}(IR)$	$\overline{IORQ}$ Setup Time to Rising Edge of Φ During Read or Write Cycle	250		ns	
$\overline{M1}$	$t_{S\Phi}(M1)$	$\overline{M1}$ Setup Time to Rising Edge of Φ During INTA or M1 Cycle	210		ns	
	$t_{S\Phi}(RD)$	$\overline{RD}$ Setup Time to Rising Edge of Φ During Read or M1 Cycle	240		ns	
$\overline{INT}$	$t_{DCK}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of CLK/TRG		$2t_C(\Phi) + 200$		Counter Mode
	$t_{D\Phi}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of Φ		$t_C(\Phi) + 200$		Timer Mode
CLK/TRG_{0-3}	$t_C(CK)$	Clock Period	$2t_C(\Phi)$			Counter Mode
	t_r, t_f	Clock and Trigger Rise and Fall Times		50		
	$t_S(CK)$	Clock Setup Time to Rising Edge of Φ for Immediate Count	210			Counter Mode
	$t_S(TR)$	Trigger Setup Time to Rising Edge of Φ for Enabling of Prescaler on Following Rising Edge of Φ	210			Timer Mode
	$t_W(CTH)$	Clock and Trigger High Pulse Width	200			Counter and Timer Modes
	$t_W(CTL)$	Clock and Trigger Low Pulse Width	200			Counter and Timer Modes
ZC/TO_{0-2}	$t_{DH}(ZC)$	ZC/TO Delay Time from Rising Edge of Φ, ZC/TO High		190		Counter and Timer Modes
	$t_{DL}(ZC)$	ZC/TO Delay Time from Falling Edge of Φ, ZC/TO Low		190		Counter and Timer Modes

Notes:

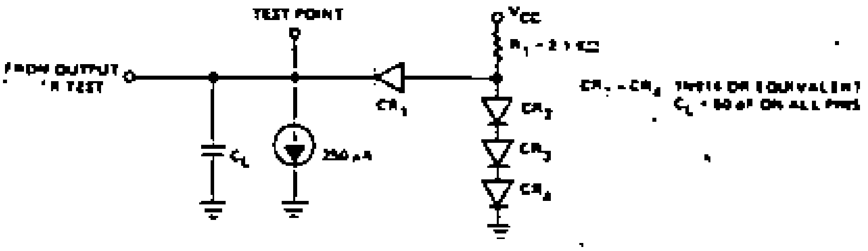
[1] $t_C = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$.

[2] Increase delay by 10 nsec for each 50 pF increase in loading, 200 pF maximum for data lines and 100 pF for control lines.

[3] Increase delay by 2 nsec for each 10 pF increase in loading, 100 pF maximum

[4] $\overline{RESET}$ must be active for a minimum of 3 clock cycles.

OUTPUT LOAD CIRCUIT

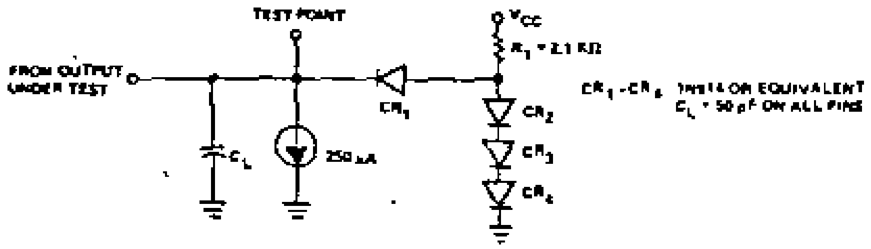


TA = 0° C to 70° C, Vcc = +5 V ± 5%, unless otherwise noted

Signal	Symbol	Parameter	Min	Max	Unit	Comments
Φ	t_C	Clock Period	250	[1]	ns	
	$t_W(\Phi H)$	Clock Pulse Width, Clock High	105	2000	ns	
	$t_W(\Phi L)$	Clock Pulse Width, Clock Low	105	2000	ns	
	t_r, t_f	Clock Rise and Fall Times		30	ns	
	t_H	Any Hold Time for Specified Setup Time	0		ns	
CS, $\overline{CE}$, etc	$t_{S\Phi}(CS)$	Control Signal Setup Time to Rising Edge of Φ During Read or Write Cycle	60		ns	
D_0-D_7	$t_{DR}(D)$	Data Output Delay from Falling Edge of $\overline{RD}$ During Read Cycle		380	ns	[2]
	$t_{S\Phi}(D)$	Data Setup Time to Rising Edge of Φ During Write or M1 Cycle	50		ns	
	$t_{DI}(D)$	Data Output Delay from Falling Edge of $\overline{IORG}$ During INTA Cycle		160	ns	[2]
	$t_F(D)$	Delay to Floating Bus (Output Buffer Disable Time)		110	ns	
IEI	$t_S(IEI)$	IEI Setup Time to Falling Edge of $\overline{IORQ}$ During INTA Cycle	140		ns	
IEO	$t_{DH}(IO)$	IEO Delay Time from Rising Edge of IEI		160	ns	[3]
	$t_{DL}(IO)$	IEO Delay Time from Falling Edge of IEI		130	ns	[3]
	$t_{DM}(IO)$	IEO Delay from Falling Edge of $\overline{M1}$ (Interrupt Occurring just Prior to $\overline{M1}$)		190	ns	[3]
$\overline{IORQ}$	$t_{S\Phi}(IR)$	$\overline{IORQ}$ Setup Time to Rising Edge of Φ During Read or Write Cycle	115		ns	
$\overline{M1}$	$t_{S\Phi}(M1)$	$\overline{M1}$ Setup Time to Rising Edge of Φ During INTA or M1 Cycle	90		ns	
$\overline{RD}$	$t_{S\Phi}(RD)$	$\overline{RD}$ Setup Time to Rising Edge of Φ During Read or M1 Cycle	115		ns	
$\overline{INT}$	$t_{DCK}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of CLK/TRG		$2t_C(\Phi) + 140$		Counter Mode
	$t_{D\Phi}(IT)$	$\overline{INT}$ Delay Time from Rising Edge of Φ		$t_C(\Phi) + 140$		Timer Mode
CLK/TRG_{0-3}	$t_C(CK)$	Clock Period	$2t_C(\Phi)$			Counter
	t_r, t_f	Clock and Trigger Rise and Fall Times		50		
	$t_S(CK)$	Clock Setup Time to Rising Edge of Φ for Immediate Count	210			Counter Mode
	$t_S(TR)$	Trigger Setup Time to Rising Edge of Φ for enabling of Prescaler on Following Rising Edge of Φ	210			Timer
	$t_W(CTH)$	Clock and Trigger High Pulse Width	200			Counter and Timer Modes
	$t_W(CTL)$	Clock and Trigger Low Pulse Width	200			Counter and Timer Modes
ZC/TO_{0-2}	$t_{DH}(ZC)$	ZC/TO Delay Time from Rising Edge of Φ, ZC/TO High		190		Counter and Timer M
	$t_{DL}(ZC)$	ZC/TO Delay Time from Falling Edge of Φ, ZC/TO Low		190		Counter and Timer

Notes: [1] $t_C = t_W(\Phi H) + t_W(\Phi L) + t_r + t_f$.
[2] Increase delay by 10 nsec for each 50 pF increase in loading, 200 pF maximum for data lines and 100 pF for control lines.
[3] Increase delay by 2 nsec for each 10 pF increase in loading, 100 pF maximum.
[4] $\overline{RESET}$ must be active for a minimum of 3 clock cycles.

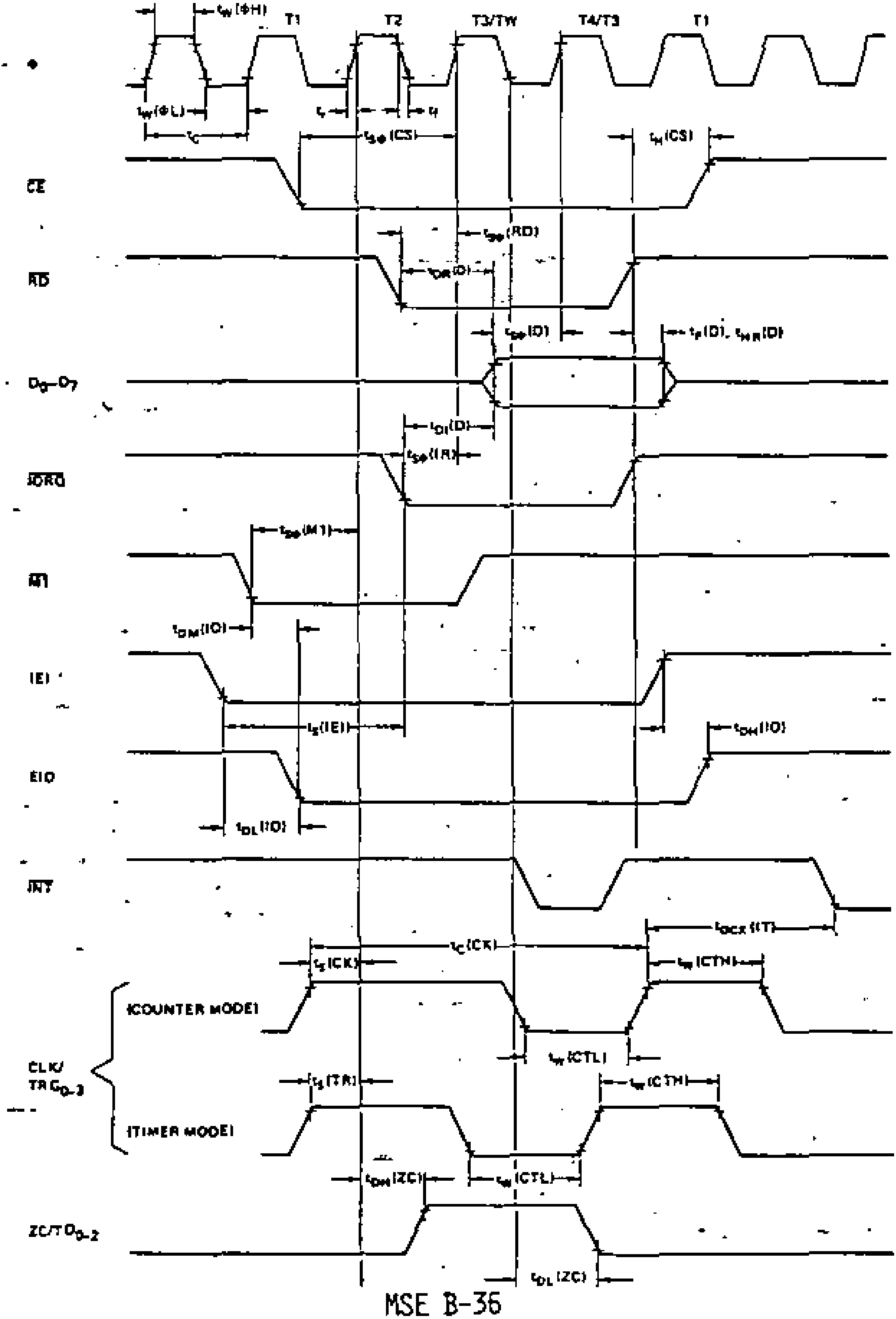
OUTPUT LOAD CIRCUIT



A.C. Timing Diagram

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	V_{CC} - .6V	.45V
OUTPUT	2.0V	.8V
INPUT	2.0V	.8V
FLOAT	ΔV	±0.5V



Absolute Maximum Ratings

Temperature Under Bias	0° C to 70° C
Storage Temperature	-65° C to +150° C
Voltage On Any Pin With Respect To Ground	-0.3 V to +7 V
Power Dissipation	0.6W

*Comment

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

D.C. Characteristics

TA = 0° C to 70° C, Vcc = 5 V ± 5% unless otherwise specified

Z80–CTC

Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	.45	V	
V_{IHC}	Clock Input High Voltage (1)	$V_{CC} - .6$	$V_{CC} + .3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = 2$ mA
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250$ µA
I_{CC}	Power Supply Current		120	mA	$T_C = 400$ nsec
I_{LI}	Input Leakage Current		10	µA	$V_{IN} = 0$ to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float		10	µA	$V_{OUT} = 2.4$ to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float		-10	µA	$V_{OUT} = 0.4V$
I_{OHD}	Darlington Drive Current	-1.5		mA	$V_{EXT} = 1.5V$ $R_{EXT} = 390\Omega$

Z80A–CTC

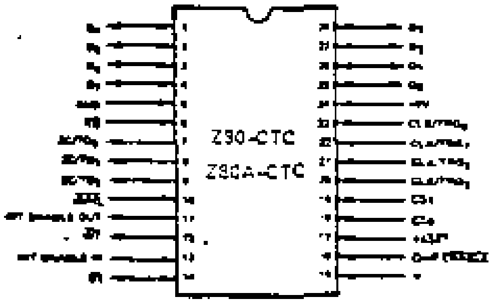
Symbol	Parameter	Min	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3	.45	V	
V_{IHC}	Clock Input High Voltage (1)	$V_{CC} - .6$	$V_{CC} + .3$	V	
V_{IL}	Input Low Voltage	-0.3	0.8	V	
V_{IH}	Input High Voltage	2.0	V_{CC}	V	
V_{OL}	Output Low Voltage		0.4	V	$I_{OL} = 2$ mA
V_{OH}	Output High Voltage	2.4		V	$I_{OH} = -250$ µA
I_{CC}	Power Supply Current		120	mA	$T_C = 250$ nsec
I_{LI}	Input Leakage Current		10	µA	$V_{IN} = 0$ to V_{CC}
I_{LOH}	Tri-State Output Leakage Current in Float		10	µA	$V_{OUT} = 2.4$ to V_{CC}
I_{LOL}	Tri-State Output Leakage Current in Float		-10	µA	$V_{OUT} = 0.4V$
I_{OHD}	Darlington Drive Current	-1.5		mA	$V_{EXT} = 1.5V$ $R_{EXT} = 390\Omega$

Capacitance

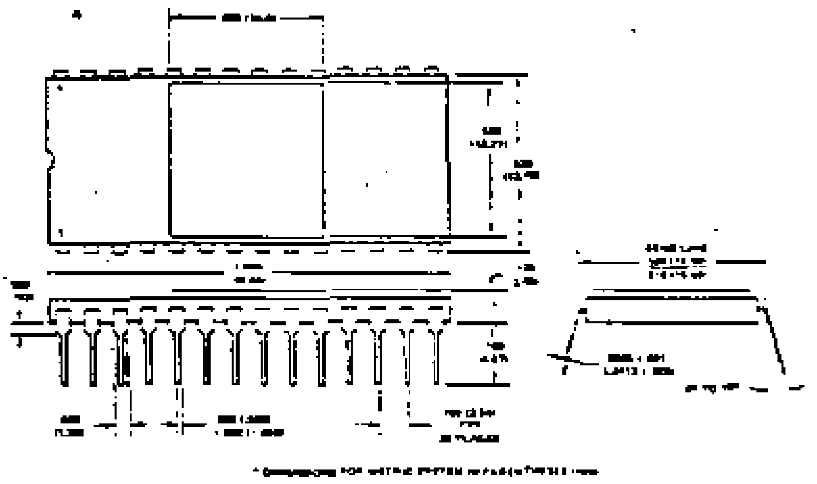
TA = 25° C, f = 1 MHz

Symbol	Parameter	Max.	Unit	Test Condition
C_ϕ	Clock Capacitance	20	pF	Unmeasured Pins
C_{IN}	Input Capacitance	5	pF	Returned to Ground
C_{OUT}	Output Capacitance	10	pF	

Package Configuration



Package Outline



EASTERN

Hallmark Electronics
4739 Commercial Drive
Huntsville, AL 35805
TEL 205 837 8700
TWX 810 726 2187

Hallmark Electronics
1302 West McNab Road
Fort Lauderdale, FL 33309
TEL 305 971 9280
TWX 510 956 9720

[illegible] Electronics
[illegible] Lake Ellenor Drive
[illegible]ndo, FL 32809
TEL 305 855 4020
TWX 810 850 0183

Hallmark Electronics
3335 Amberton Drive
Baltimore, MD 21227
TEL 301 796 9300
TWX 710 862 1942

Hallmark Electronics
1208 Front Street
Building K
Raleigh, NC 27609
TEL 919 832 4465
TWX 510 928 1831

Hallmark Electronics
Pike Industrial Park
Huntingdon Valley, PA 19006
TEL 215 355 7300
TWX 510 667 1750

Summit
916 Main Street
Buffalo, NY 14202
TEL 716 884 3450

Wilshire Electronics
2554 State Street
Hamden, CT 06517
TEL 203 281 1166
TWX 800 922 1734

Wilshire Electronics
1855 New Highway
Farmingdale, L.I., NY 11735
TEL 516 293 5775
TWX 212 895 8707

Wilshire Electronics
One Wilshire Road
Burlington, MA 01803
TEL 617 272 8200
TWX 710 332 6359

Wilshire Electronics
1111 Paulison Avenue
Clifton, NJ 07015
TEL 201 340 1900
TWX 710 989 7052

MIDWESTERN

Hallmark Electronics
180 Crossen Avenue
Elk Grove Village, IL 60076
TEL 312 437 8800
TWX 910 223 3645

Hallmark Electronics
11870 West 91st Street
Congleton Industrial Park
Shawnee Mission, KS 66214
TEL 913 888 4747
TWX 910 749 6620

Hallmark Electronics
9201 Penn Avenue South
Suite 10
Bloomington, MN 55435
TEL 612 884 9056
TWX 910 576 3187

Hallmark Electronics
13789 Rider Trail
Earth City, MO 63045
TEL 314 291 5350
TWX 910 760 0671

Hallmark Electronics
6969 Worthington Galena Road
Worthington, OH 43085
TEL 614 846 1882

Hallmark Electronics
4846 S. 83rd E. Avenue
Tulsa, OK 74145
TEL 918 835 8458
TWX 910 845 2290

Hallmark Electronics
3100-A Industrial Terrace
Austin, TX 78758
TEL 512 837 2841
TWX 910 874 2031

Hallmark Electronics
9333 Forest Lane
Dallas, TX 75231
TEL 214 234 7300
TWX 910 867 4721

Hallmark Electronics
8000 Westglen
Houston, TX 77063
TEL 713 781 6100
TWX 910 881 2711

Hallmark Electronics
237 South Curtis
West Allis, WI 53214
TEL 414 476 1270
TWX 910 262 3186

RM Electronics
4860 South Division
Kentwood, MI 49508
TEL 616 531 9300
TWX 810 273 8779

RM Electronics
47 Chestnut Lane
Westmont, Illinois 60559
TEL 312 323 9670

MOUNTAIN

R.V. Weatherford Company
3905 South Mariposa
Englewood, CO 80110
TEL 303 761 5432
TWX 910 933 0173

WESTERN

Intermark Electronics
1802 E. Carnegie Avenue
Santa Ana, CA 92705
TEL 714 540 1322
TWX 910 595 1583

Intermark Electronics
4040 Sorrento Valley Blvd.
San Diego, CA 92121
TEL 714 279 5200
714 453 9005
TWX 910 335 1515

Intermark Electronics
1020 Stewart Drive
Sunnyvale, CA 94086
TEL 408 738 1111
TWX 910 339 9312

R.V. Weatherford Co.
6921 San Fernando Road
Glendale, CA 91201
TEL 213 849 3451
TWX 910 498 2223

R.V. Weatherford Co.
1550 Babbitt Avenue
Anaheim, CA 92805
TEL 714 634 9600
TWX 910 593 1334

R.V. Weatherford Co.
1095 East Third Street
Pomona, CA 91765
TEL 714 623 1261
TWX 910 581 3811

R.V. Weatherford Co.
3240 Hillview Avenue
Stanford Industrial Park
Palo Alto, CA 94304
TEL 415 493 5373

R.V. Weatherford Co.
3311 W. Earll Drive
Phoenix, AZ 85017
TEL 602 272 7144
TWX 910 951 0636

Sterling Electronics
5608 6th Avenue South
Seattle, WA 98108
TEL 206 762 9100
TLX 32-9652

Western Microtechnology
977 Benecia Avenue
Sunnyvale, CA 94086
TEL 408 737 1660

CANADA

Future Electronics
5647 Ferrier Street
Montreal, Quebec,
CANADA H4P 2K5
TEL 514 735 5775
TWX 610 421 3251

EASTERN REGION
Zilog, Inc.
76 Treble Cove Road
No. Billerica, MA 01862
TEL 617 667 2179
TWX 710 347 6660

MIDATLANTIC REGION
Zilog, Inc.
P.O. Box 92
Bergenfield, NJ 07621
TEL 201 385 9158
TWX 710 991 9771

MIDWESTERN REGION
Zilog, Inc.
1701 Woodfield Place
Suite 417
Schaumburg, IL 60195
TEL 312 885 8080
TWX 910 291 1064

SOUTHWESTERN REGION
Zilog, Inc.
17982 Sky Park Circle
Suite C
Irvine, CA 92714
TEL 714 549 2891
TWX 910 595 2803

EUROPEAN HDQTS
Zilog (UK) Ltd
Nicholson House
Maidenhead, Berkshire
England
TEL (0628) 36131 2 3
TWX 848 609

10460 Bubb Road, Cupertino, California 95014

Telephone: (408) 446-4666 TWX: 910-338-7621

03-0031-02

Printed in U.S.A.

MSE B-38

Z80®-SIO Z80A-SIO

Product Specification

AUGUST 1978

The Z80-SIO (Serial Input/Output) is a dual-channel multi-function peripheral component designed to satisfy a wide variety of serial data communications requirements in microcomputer systems. Its basic function is a serial-to-parallel, parallel-to-serial converter/controller, but—within that role—it is configurable by systems software so its "personality" can be optimized for a given serial data communications application.

The Z80-SIO is capable of handling asynchronous formats, synchronous byte-oriented protocols such as IBM Bisync, and synchronous bit-oriented protocols such as HDLC and SDLC. This versatile device can also be used to support virtually any other serial protocol for applications other than data communications (cassette or floppy disk interfaces, for example).

The Z80-SIO can generate and check CRC codes in any synchronous mode and can be programmed to check data integrity in various modes. The device also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

N-channel silicon-gate depletion-load technology

40-pin DIP

Single 5 V power supply

Single-phase 5 V clock

All inputs and outputs TTL compatible

Two independent full-duplex channels

Data rates in synchronous or isosynchronous modes:

- 0–550K bits/second with 2.5 MHz system clock rate
- 0–880K bits/second with 4.0 MHz system clock rate

Receiver data registers quadruply buffered; transmitter doubly buffered.

Asynchronous features:

- 5, 6, 7 or 8 bits/character

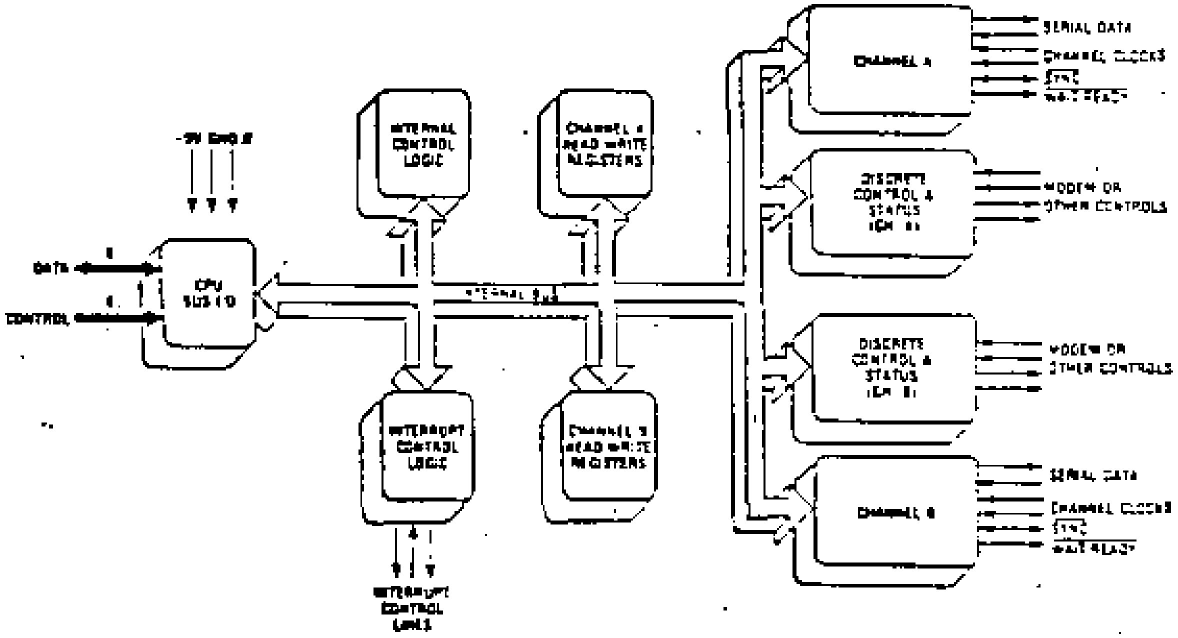


Figure 1. Z80-SIO Block Diagram

- 1, 1½ or 2 stop bits
- Even, odd or no parity
- ×1, ×16, ×32 and ×64 clock modes
- Break generation and detection
- Parity, overrun and framing error detection

Binary synchronous features:

- Internal or external character synchronization
- One or two sync characters in separate registers
- Automatic sync character insertion/deletion
- CRC generation and checking

HDLC and SDLC features:

- Abort sequence generation and detection
- Automatic zero insertion and deletion
- Automatic flag insertion between messages
- Address field recognition
- Support for one to eight bits/character
- Valid receive messages protected from overrun
- CRC generation and checking

Interrupt features:

- Daisy-chain interrupt logic provides automatic interrupt vectoring with no external logic
- Programmable interrupt vector
- Status Affects Interrupt Vector mode for fast interrupt processing

CRC-16 or CRC-CCITT block frame check

Separate modem control inputs and outputs for both channels

Modem status can be monitored

D_0–D_7. *System Data Bus* (bidirectional, 3-state). The system data bus transfers data and commands between the CPU and the Z80-SIO. D_0 is the least significant bit.

B/$\overline{A}$. *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 Z80-SIO. Address bit A_0 from the CPU is often used for the selection function.

C/$\overline{D}$. *Control Or Data Select* (input, High selects Control). This input defines the type of information transfer performed between the CPU and the Z80-SIO. A High at this input during a CPU write to the Z80-SIO causes the information on the data bus to be interpreted as a command for the channel selected by B/$\overline{A}$. A Low at C/D means that the information on the data bus is data. Address bit A_1 is often used for this function.

$\overline{CE}$. *Chip Enable* (input, active Low). A Low level at this input enables the Z80-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.

ϕ. *System Clock* (input). The Z80-SIO uses the standard Z80 System Clock to synchronize internal signals. This is a single-phase clock.

$\overline{M1}$. *Machine Cycle One* (input from Z80-CPU, active Low). When $\overline{M1}$ is active and $\overline{RD}$ is also active, the Z80-CPU is fetching an instruction from memory; when $\overline{M1}$ is active while $\overline{IORQ}$ is active, the Z80-SIO accepts $\overline{M1}$

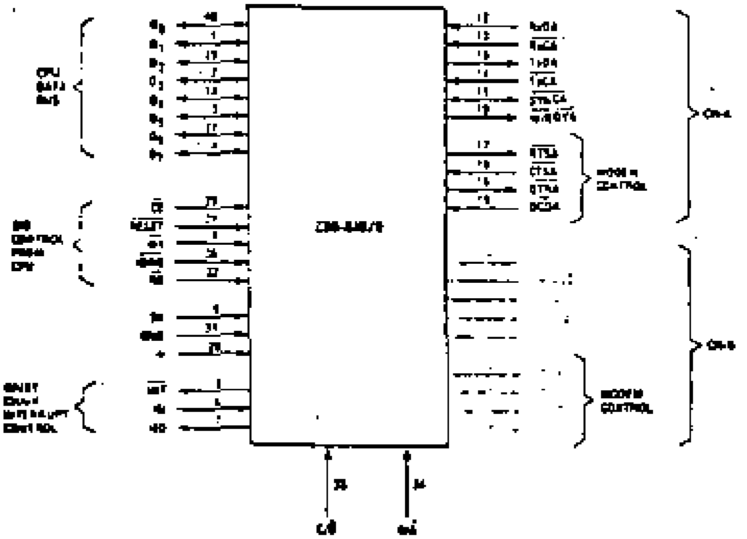


Figure 2. Z80-SIO/0 Pin Configuration

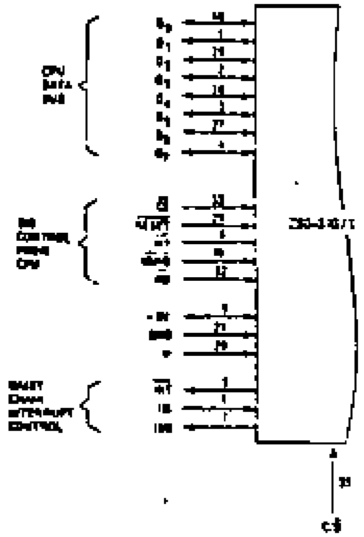


Figure 3.

and $\overline{IORQ}$ as an interrupt acknowledge if the Z80-SIO is the highest priority device that has interrupted the Z80-CPU.

$\overline{IORQ}$. *Input/Output Request* (input from CPU, active Low). $\overline{IORQ}$ is used in conjunction with B/$\overline{A}$, C/$\overline{D}$, $\overline{CE}$ and $\overline{RD}$ to transfer commands and data between the CPU and the Z80-SIO. When $\overline{CE}$, $\overline{RD}$ and $\overline{IORQ}$ are all active, the channel selected by B/$\overline{A}$ transfers data to the CPU (a read operation). When $\overline{CE}$ and $\overline{IORQ}$ are active, but $\overline{RD}$ is inactive, the channel selected by B/$\overline{A}$ is written to by the CPU with either data or control information as specified by C/$\overline{D}$. As mentioned previously, if $\overline{IORQ}$ and $\overline{M1}$ are active simultaneously, the CPU is acknowledging an interrupt and the Z80-SIO automatically places its interrupt vector on the CPU data bus if it is the highest priority device requesting an interrupt.

$\overline{RD}$. *Read Cycle Status.* (input from CPU, active Low). If $\overline{RD}$ is active, a memory or I/O read operation is in progress. $\overline{RD}$ is used with B/$\overline{A}$, $\overline{CE}$ and $\overline{IORQ}$ to transfer data from the Z80-SIO to the CPU.

$\overline{RESET}$. *Reset* (input, active Low). A Low $\overline{RESET}$ 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 Z80-SIO is reset and before data is transmitted or received.

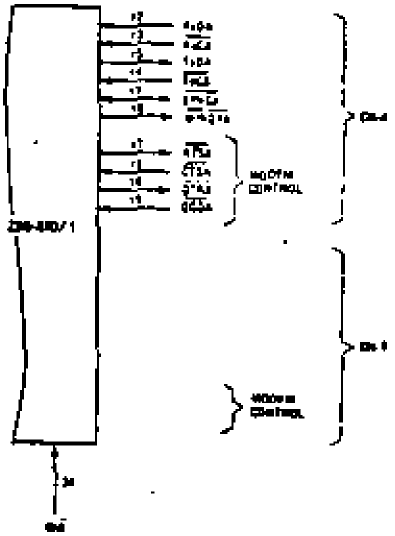
IEI. *Interrupt Enable In* (input, active High). This signal is used with IEO to form a priority daisy chain when there is more than one interrupt-driven device. A High on this line indicates that no other device of higher priority is being serviced by a CPU interrupt service routine.

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 Z80-SIO. Thus, this signal blocks lower priority devices from interrupting while a higher priority device is being serviced by its CPU interrupt service routine.

$\overline{INT}$. *Interrupt Request* (output, open drain, active Low). When the Z80-SIO is requesting an interrupt, it pulls $\overline{INT}$ Low.

$\overline{W/RDYA}$, $\overline{W/RDYB}$. *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 Z80-SIO data rate. The reset state is open drain.

$\overline{CTSA}$, $\overline{CTSB}$. *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 Z80-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.



Z80-SIO/1 Pin Configuration

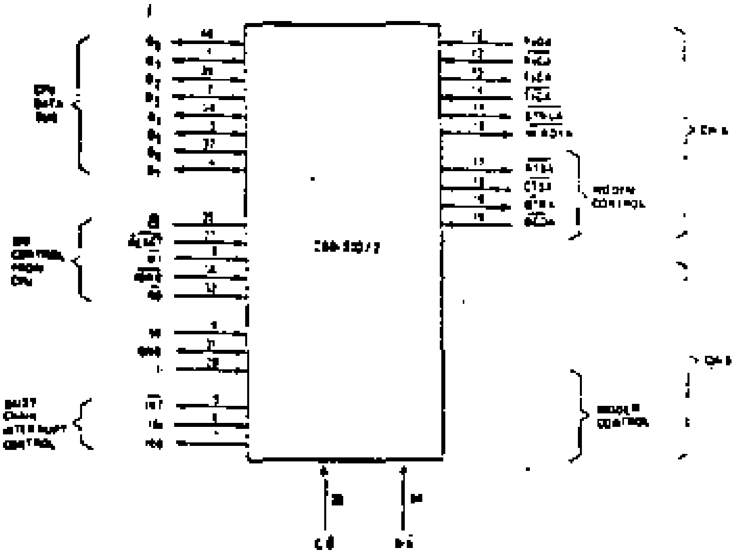


Figure 4. Z80-SIO/2 Pin Configuration

$\overline{\text{DCDA}}$, $\overline{\text{DCDB}}$. *Data Carrier Detect* (inputs, active Low). These pins function as receiver enables if the Z80-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 Z80-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.

RxDA, RxDB. *Receive Data* (inputs, active High).

TxDA, TxDB. *Transmit Data* (outputs, active High).

$\overline{\text{RxCA}}$, $\overline{\text{RxCB}}$. *Receiver Clocks* (inputs). Receive data is sampled on the rising edge of 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 Z80-CTC Counter Timer Circuit for programmable baud rate generation. Both inputs are Schmitt-trigger buffered (no noise level margin is specified). See the following section for bonding options.

$\overline{\text{TxCA}}$, $\overline{\text{TxCB}}$. *Transmitter Clocks* (inputs). TxD changes on the falling edge of $\overline{\text{TxC}}$. 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 the Z80-CTC Counter Timer Circuit for programmable baud rate generation. See the following section for bonding options.

$\overline{\text{RTSA}}$, $\overline{\text{RTSB}}$. *Request To Send* (outputs, active Low). When the RTS bit 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.

$\overline{\text{DTRA}}$, $\overline{\text{DTRB}}$. *Data Terminal Ready* (outputs, active Low). See note on bonding options. These outputs follow the state programmed into the DTR bit. They can also be programmed as general-purpose outputs.

$\overline{\text{SYNC A}}$, $\overline{\text{SYNC B}}$. *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 RR0. 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 RxC after that rising edge of 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 is wise to keep it Low until the CPU informs the external sync logic that synchronization has been lost or a new message is about to start. Character assembly begins on the rising edge of 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.

The constraints of a 40-pin package make it impossible to bring out the Receive Clock, Transmit Clock, Data Terminal Ready and Sync signals for both channels. Therefore, Channel B must sacrifice a signal or have two signals bonded together. Since user requirements vary, three bonding options are offered:

- Z80-SIO/0 has all four signals, but $\overline{\text{TxCB}}$ and $\overline{\text{RxCB}}$ are bonded together (Fig. 2).
- Z80-SIO/1 sacrifices DTRB and keeps $\overline{\text{TxCB}}$, $\overline{\text{RxCB}}$ and $\overline{\text{SYNCB}}$ (Fig. 3).
- Z80-SIO/2 sacrifices SYNCB and keeps $\overline{\text{TxCB}}$, $\overline{\text{RxCB}}$ and $\overline{\text{DTRB}}$ (Fig. 4).

The device internal structure includes a Z80-CPU interface, internal control and interrupt logic, and two full-duplex channels. Each channel contains read and write registers, and discrete control and status logic that provides the interface to modems or other external devices.

The read and write register group includes five 8-bit control registers, two sync-character registers and two status registers. The interrupt vector is written into an additional 8-bit register (Write Register 2) in Channel B that may be read through Read Register 2 in Channel B. The registers for both channels are designated in the text as follows:

WR0-WR7 — Write Registers 0 through 7
RR0-RR2 — Read Registers 0 through 2

The bit assignment and functional grouping of each register is configured to simplify and organize the programming process. Table 1 lists the functions assigned to each read or write register.

RR0	Transmit/Receive buffer status, interrupt status and external status
RR1	Special Receive Condition status
RR2	Modified interrupt vector (Channel B only)

Read Register Functions

WR0	Register pointers, CRC initialize, initialization commands for the various modes, etc.
WR1	Transmit/Receive interrupt and data transfer mode definition.
WR2	Interrupt vector (Channel B only)
WR3	Receive parameters and control
WR4	Transmit/Receive miscellaneous parameters and modes
WR5	Transmit parameters and controls
WR6	Sync character or SDLC address field
WR7	Sync character or SDLC flag

Write Register Functions

Table 1. Functional Assignments of Read and Write Registers

The logic for both channels provides formats, synchronization and validation for data transferred to and from the channel interface. The modem control inputs Clear to Send (CTS) and Data Carrier Detect (DCD) are monitored by the discrete control logic under program control. All the modem control signals are general purpose in nature and can be used for functions other than modem control.

For automatic interrupt vectoring, the interrupt control logic determines which channel and which device within the channel has the highest priority. Priority is fixed with Channel A assigned a higher priority than Channel B; Receive, Transmit and External/Status interrupts are prioritized in that order within each channel.

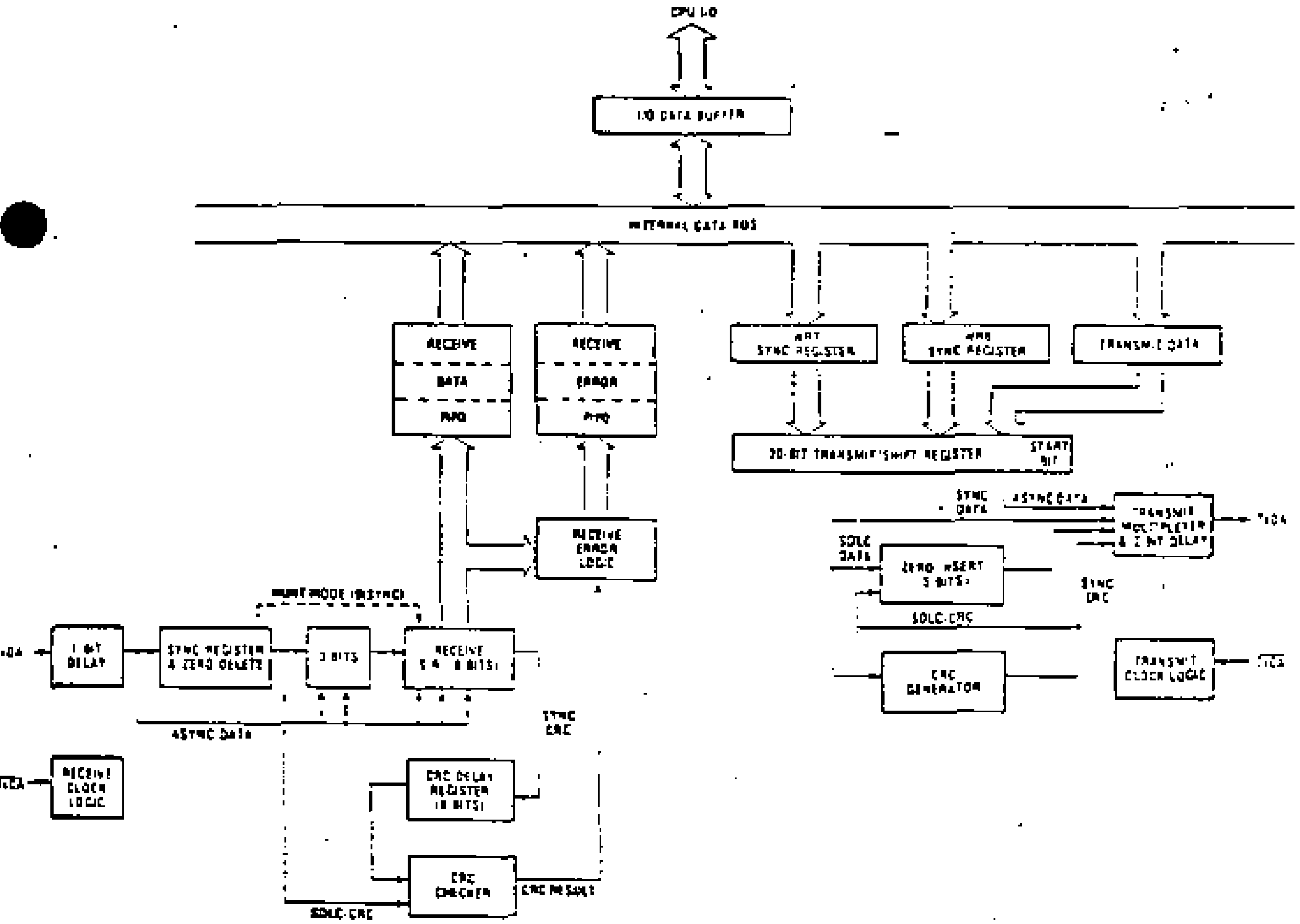


Figure 5. Transmit and Receive Data Path

The transmit and receive data path illustrated for Channel A in Figure 5 is identical for both channels. The receiver has three 8-bit buffer registers in a FIFO arrangement in addition to the 8-bit receive shift register. This scheme creates additional time for the CPU to service an interrupt at the beginning of a block of high-speed data. Incoming data is routed through one of several paths (data or CRC) depending on the selected mode and—in Asynchronous modes—the character length.

The transmitter has an 8-bit transmit data register that is loaded from the internal data bus, and a 20-bit transmit shift register that can be loaded from the sync character buffers (WR6 and WR7) or from the transmit data register. Depending on the operational mode, outgoing data is routed through one of four main paths before it is transmitted from the Transmit Data Output (TxD).

The functional capabilities of the Z80-SIO can be described from two different points of view: as a data communications device, it transmits and receives serial data, and meets the requirements of various data communications protocols; as a Z80 family peripheral, it interacts with the Z80-CPU and other Z80 peripheral circuits, and shares the data, address and control busses, as well as being a part of the Z80 interrupt structure. As a peripheral to other microprocessors, the Z80-SIO offers valuable features such as non-vectored interrupts, polling and simple handshake capability.

The first part of the following functional description describes the interaction between the CPU and Z80-SIO; the second part introduces its data communications capabilities.

The Z80-SIO offers the choice of Polling, Interrupt (vectored or non-vectored) and Block Transfer modes to transfer data, status and control information to and from the CPU. The Block Transfer mode can be implemented under CPU or DMA control.

Polling. There are no interrupts in the Polled mode. Status registers RR0 and RR1 are updated at appropriate times for each function being performed (for example, CRC Error status valid at the end of the message). All the interrupt modes of the Z80-SIO must be disabled to operate the device in a polled environment.

While in its Polling sequence, the CPU examines the status contained in RR0 for each channel; the RR0 status bits serve as an acknowledge to the Poll inquiry. The two RR0 status bits D_0 and D_2 indicate that a data transfer is needed. The status also indicates Error or other special status conditions (see "Z80-SIO Programming"). The Special Receive Condition status contained in RR1 does not have to be read in a Polling sequence because the status bits in RR1 must be accompanied by a Receive Character Available status in RR0.

Interrupts. The Z80-SIO offers an elaborate interrupt scheme to provide fast interrupt response in real-time applications. Channel B registers WR2 and RR2 contain the interrupt vector that points to an interrupt service routine in the memory. To service operations in both channels and to eliminate the necessity of writing a status analysis routine, the Z80-SIO can modify the interrupt vector in RR2 so it points directly to one of eight interrupt service routines. This is done under program control by setting a program bit (WR1, D_2) in Channel B called "Status Affects Vector." When this bit is set, the interrupt vector in WR2 is modified according to the assigned priority of the various interrupting conditions. The table in the Write Register 1 description (Z80-SIO Programming section) shows the modification details.

Transmit interrupts, Receive interrupts and External/Status interrupts are the main sources of interrupts. Each interrupt source is enabled under program control with Channel A having a higher priority than Channel B, and with Receiver, Transmit and External/Status interrupts prioritized in that order within each channel. When the Transmit interrupt is enabled, the CPU is interrupted by the transmit buffer becoming empty. (This implies that the transmitter must have had a data character written into it so it can become empty.) When enabled, the receiver can interrupt the CPU in one of three ways:

- Interrupt on the first received character
- Interrupt on all received characters
- Interrupt on a Special Receive condition

Interrupt On First Character is typically used with the Block Transfer mode. Interrupt On All Receive Characters has the option of modifying the interrupt vector in the event of a parity error. The Special Receive Condition interrupt can occur on a character or message basis (End Of Frame interrupt in SDLC, for example). The Special Receive condition can cause an interrupt only if the Interrupt On First Receive Character or Interrupt On All Receive Characters mode is selected. In Interrupt On First Receive Character, an interrupt can occur from Special Receive conditions (except Parity Error) after the first receive character interrupt (example: Receive Overrun interrupt).

The main function of the External/Status interrupt is to monitor the signal transitions of the $\overline{CTS}$, $\overline{DCD}$ and $\overline{SYNC}$ pins; however, an External/Status interrupt is also caused by a Transmit Underrun condition or by the detection of a Break (Asynchronous mode) or Abort (SDLC mode) sequence in the data stream. The interrupt caused by the Break/Abort sequence has a special feature that allows the Z80-SIO to interrupt when the Break/Abort sequence is detected or terminated. This feature facilitates the proper termination of the current message, correct initialization of the next message, and he accurate timing of the Break/Abort condition in external logic.

PU/DMA Block Transfer. The Z80-SIO provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers (Z80-DMA or other designs). The Block Transfer mode uses the $\overline{\text{WAIT}}$/$\overline{\text{READY}}$ output in conjunction with the Wait/Ready bits of Write Register 1. The $\overline{\text{WAIT}}$/$\overline{\text{READY}}$ output can be defined under software control as a $\overline{\text{WAIT}}$ line in the CPU Block Transfer mode or as a $\overline{\text{READY}}$ line in the DMA Block Transfer mode.

To a DMA controller, the Z80-SIO READY output indicates that the Z80-SIO is ready to transfer data to or from memory. To the CPU, the WAIT output indicates that the Z80-SIO is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle. The programming of bits 5, 6 and 7 of Write Register 1 and the logic states of the $\overline{\text{WAIT}}$/$\overline{\text{READY}}$ line are defined in the Write Register 1 description (Z80-SIO Programming section).

In addition to the I/O capabilities previously discussed, the Z80-SIO provides two independent full-duplex channels that can be programmed for use in Asynchronous, Synchronous and SDLC (HDLC) modes. These different modes are provided to facilitate the implementation of commonly used data communications protocols. The following is a short description of the data communications protocols supported by the Z80-SIO. A more detailed explanation of these modes can be found in the *Z80-SIO Technical Manual*.

Asynchronous Modes. The Z80-SIO offers transmission and reception of five to eight bits per character, plus optional even or odd parity. The transmitter can supply one, one and a half or two stop bits per character and can provide a break output at any time. The receiver break detection logic interrupts the CPU only at the start and end of a received break. Reception is protected from spikes by a transient spike rejection mechanism that checks the signal one-half a bit time after a Low level is detected on the Receive Data input. If the Low does not persist—as in the case of a transient—the character assembly process is not started.

Framing errors and overrun errors are detected and buffered together with the partial character on which they occurred. Vectored interrupts allow fast servicing of error conditions using dedicated routines. Furthermore, a built-in checking process avoids interpreting a framing error as a new start bit: a framing error results in the addition of one-half a bit time to the point at which the search for the next start bit is begun.

The Z80-SIO does not require symmetric Transmit and Receive Clock signals—a feature that allows it to be used with a Z80-CTC or any other clock source. The transmitter and receiver can handle data at a rate of 1, 1/16, 1/32 or 1/64 of the clock rate supplied to the Receive and Transmit Clock inputs.

In Asynchronous modes, the $\overline{\text{SYNC}}$ pin may be programmed for an input that can be used for functions such as monitoring a ring indicator.

Synchronous Modes. The Z80-SIO supports both byte-oriented and bit-oriented synchronous communication. Synchronous byte-oriented protocols can be handled in several modes that allow character synchronization with an 8-bit sync character (Monosync), any 16-bit sync pattern (Bisync), or with an external sync signal. Leading sync characters can be removed without interrupting the CPU. CRC checking for synchronous byte-oriented modes is delayed by one character time so the CPU may disable CRC checking on specific characters. This permits implementation of protocols such as IBM Bisync.

Both CRC-16 ($X^{16}+X^{15}+X^{2}+1$) and CCITT ($X^{16}+X^{12}+X^{5}+1$) error checking polynomials are supported. In all non-SDLC modes, the CRC generator is initialized to 0's; in SDLC modes, it is initialized to 1's. (This means that the Z80-SIO cannot generate or check CRC for IBM-compatible soft-sectored disks.) The Z80-SIO also provides a feature that automatically transmits CRC data when no other data is available for transmission. This allows very high-speed transmissions under DMA control with no need for CPU intervention at the end of a message. When there is no data or CRC to send in Synchronous modes, the transmitter inserts 8- or 16-bit sync characters regardless of the programmed character length. Since the CPU can read status information from the Z80-SIO, it can determine the type of transmission (data, CRC or sync characters) that is taking place at any time.

The Z80-SIO supports synchronous bit-oriented protocols such as SDLC and HDLC by performing automatic flag sending, zero insertion and CRC generation. A special command can be used to abort a frame in transmission. The Z80-SIO automatically transmits the CRC and trailing flag when the transmit buffer becomes empty. An interrupt warns the CPU of this status change so an abort may be issued if a transmitter underrun has occurred. One to eight bits per character can be sent, which allows transmission of a message exactly as received with no prior information about the character structure in the information field of a frame.

The receiver automatically synchronizes on the leading flag of a frame and provides a synchronization signal that can be programmed to interrupt. In addition, an interrupt on the first received character or on every character can be selected. The receiver automatically deletes all zeroes inserted by the transmitter during character assembly. It also calculates and automatically checks the CRC to validate frame transmission. At the end of transmission, the status of a received frame is available in the status registers. The receiver can be programmed to search for frames addressed to only a specified user-selectable address or to a global broadcast address. In this mode, frames that do not match the user-selected or broadcast address are ignored. The Address Search mode provides for a single-byte address recognizable by the hardware. The number of address bytes be extended under software control.

The Z80-SIO can be conveniently used under DMA control to provide high-speed reception. The Z80-SIO can interrupt the CPU when the first character of a message is received. The CPU then enables the DMA to transfer the message to memory. The Z80-SIO then issues an End Of Frame interrupt and the CPU checks the status of the received message. Thus, the CPU is freed for other service while the message is being received. A similar scheme allows message transmission under DMA control.

To program the Z80-SIO, 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 are first set, then the interrupt mode and, finally, receiver or transmitter enable. The WR4 parameters must be issued before any other parameters are issued in the initialization routine.

Both channels contain command registers that must be programmed via the system program prior to operation. The Channel Select input (B/A) and the Control/Data input (C/D) are the command structure addressing controls, and are normally controlled by the CPU address bus. Figure 8 illustrates the timing relationships for programming the write registers, and transferring data and status.

The Z80-SIO contains eight registers (WR0–WR7) in each channel that are programmed separately by the system program to configure the functional personality of the channels. With the exception of WR0, programming the write registers requires two bytes. The first byte 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 Z80-SIO.

WR0 is a special case in that all the basic commands (CMD_0–CMD_2) can be accessed with a single byte. Reset (internal or external) initializes the pointer bits D_0–D_2 to point to WR0.

. Z80-SIO contains three registers, RR0-RR2 (Figure 6), that can be read to obtain the status information for each channel (except for RR2 — Channel B only). 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 an input 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).

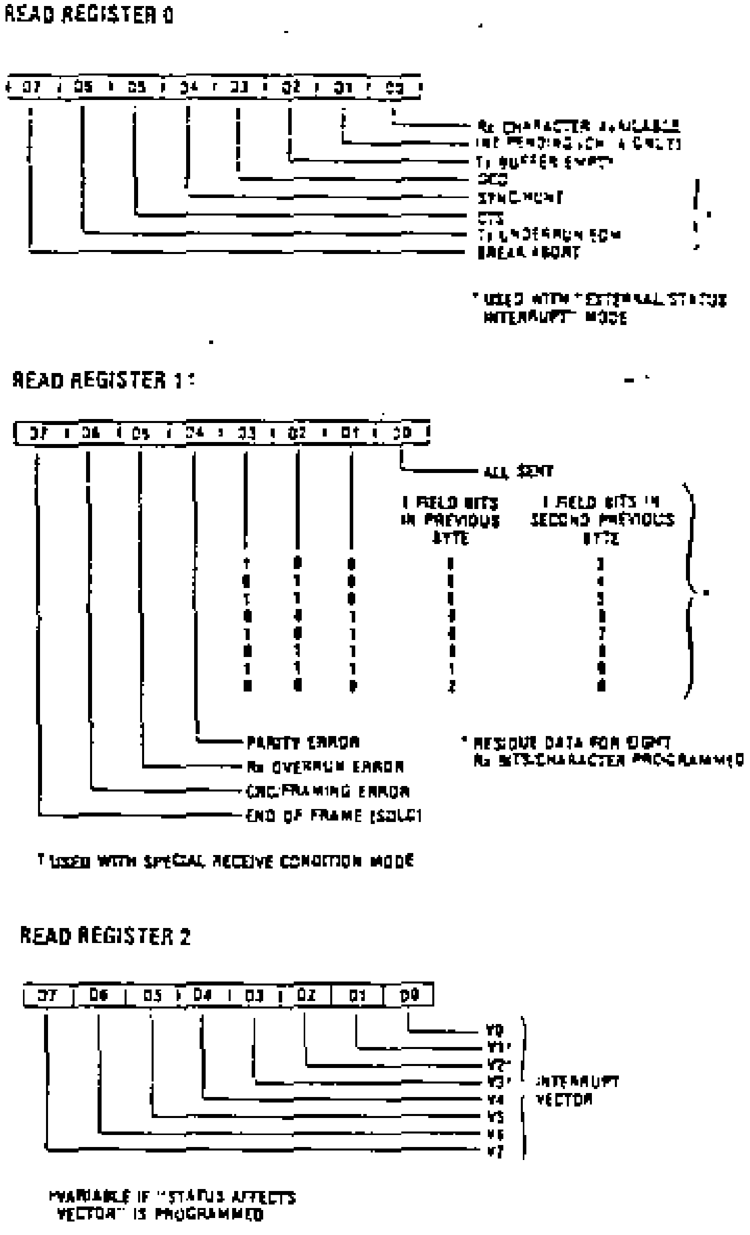


Figure 6. Read Register Bit Functions

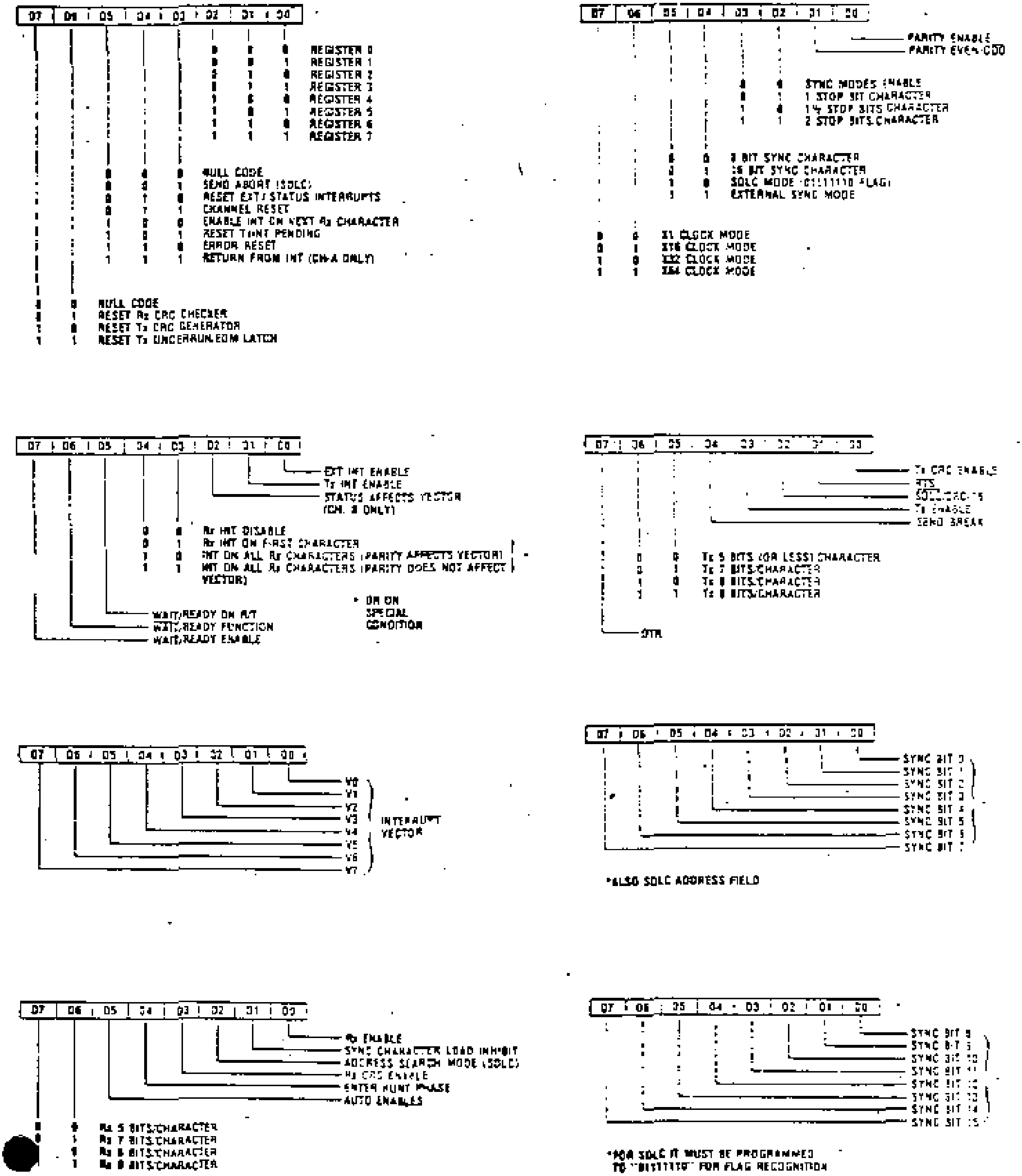


Figure 7. Write Register Bit Functions

d Cycle. The timing signals generated by a 0-CPU input instruction to read a Data or Status byte from the Z80-SIO are illustrated in Figure 8a.

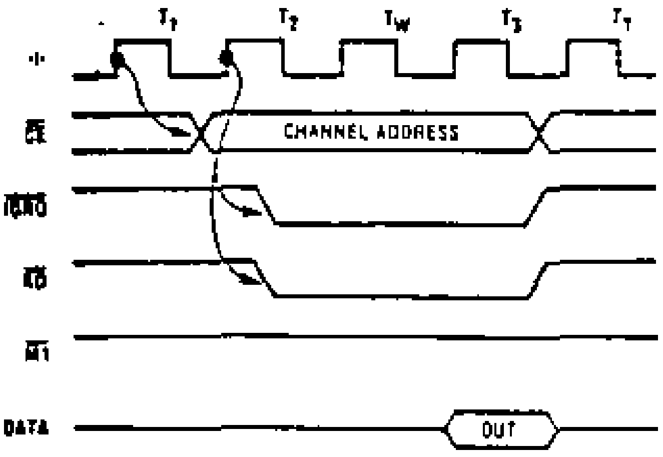


Figure 8a. Read Cycle

Interrupt Acknowledge Cycle. After receiving an Interrupt Request signal ($\overline{INT}$ pulled Low), the 0-CPU sends an Interrupt Acknowledge signal i and $\overline{IORQ}$ both Low). The daisy-chained interrupt circuits determine the highest priority interrupt requestor. The IEI of the highest priority peripheral is terminated High. For any peripheral that has no interrupt pending or under service, IEO = IEI. Any peripheral that does have an interrupt pending or under service forces its IEO Low.

To insure stable conditions in the daisy chain, all interrupt status signals are prevented from changing while $\overline{M1}$ is Low. When $\overline{IORQ}$ is Low, the highest priority interrupt requestor (the one with IEI High) places its interrupt vector on the data bus and sets its internal interrupt-under-service latch.

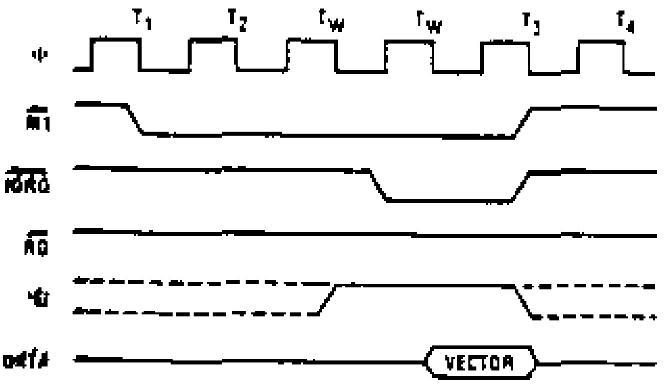


Figure 8c. Interrupt Acknowledge Cycle

Write Cycle. Figure 8b illustrates the timing and data signals generated by a Z80-CPU output instruction to write a Data or Control byte into the Z80-SIO.

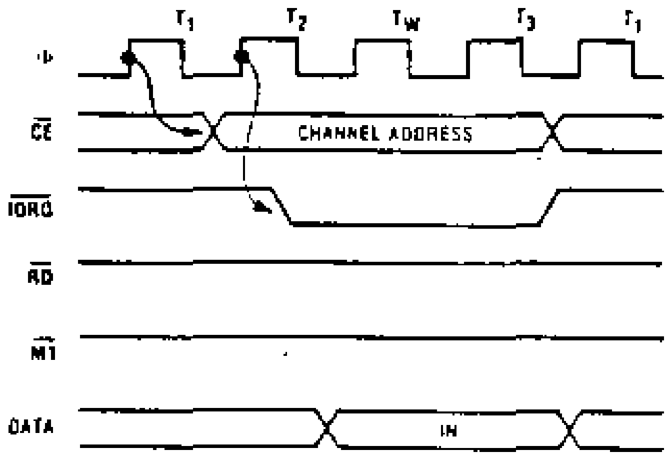


Figure 8b. Write Cycle

Return From Interrupt Cycle. Normally, the Z80-CPU issues a RETI (RETurn from interrupt) instruction at the end of an interrupt service routine. RETI is a 2-byte opcode (ED-4D) that resets the interrupt-under-service latch to terminate the interrupt that has just been processed. This is accomplished by manipulating the daisy chain in the following way.

The normal daisy chain operation can be used to detect a pending interrupt; however, it cannot distinguish between an interrupt under service and a pending unacknowledged interrupt of a higher priority. Whenever "ED" is decoded, the daisy chain is modified by forcing High the IEO of any interrupt that has not yet been acknowledged. Thus the daisy chain identifies the device presently under service as the only one with an IEI

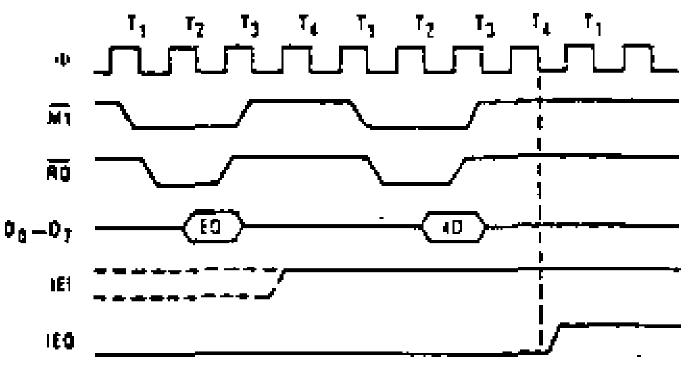


Figure 8d. Return from Interrupt Cycle

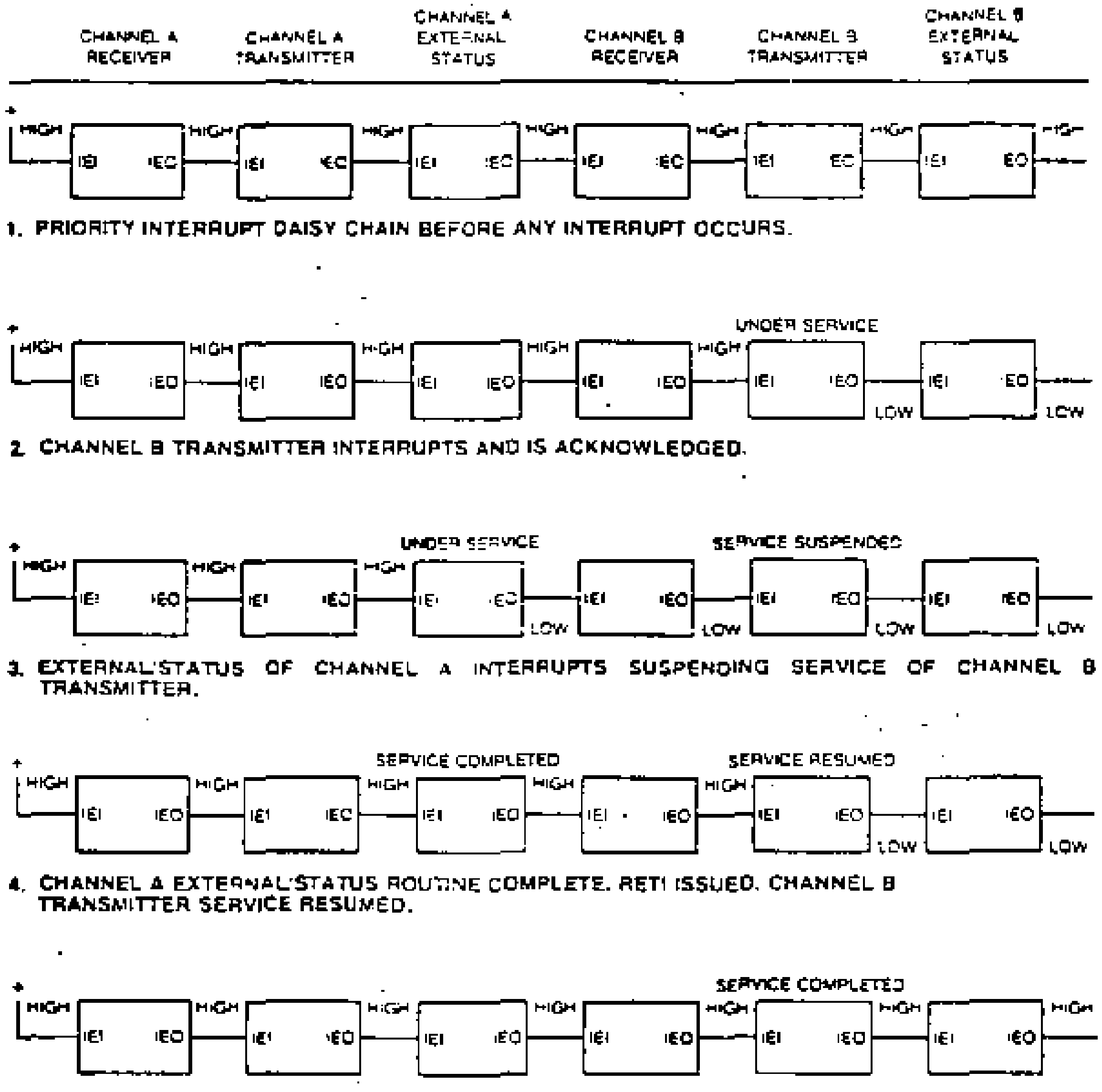
High and an IEO Low. If the next opcode byte is "4D," the interrupt-under-service latch is reset.

The ripple time of the interrupt daisy chain (both the High-to-Low and the Low-to-High transitions) limits the number of devices that can be placed in the daisy chain. Ripple time can be improved with carry-look-ahead, or by extending the interrupt acknowledge cycle. For further information about techniques for increasing the number of daisy-chained devices, refer to Zilog Application Note 03-0041-01 (*The Z80 Family Program Interrupt Structure*).

Figure 9 illustrates the daisy chain configuration of interrupt circuits and their behavior with nested interrupts (an interrupt that is interrupted by another with a higher priority).

Each box in the illustration could be a separate external Z80 peripheral circuit with a user-defined order of interrupt priorities. However, a similar daisy chain structure also exists inside the Z80-SIO, which has six interrupt levels with a fixed order of priorities.

The case illustrated occurs when the transmitter of Channel B interrupts and is granted service. While this interrupt is being serviced, it is interrupted by a higher priority interrupt from Channel A. The second interrupt is serviced and—upon completion—a RETI instruction is executed or a RETI command is written into the Z80-SIO, resetting the interrupt-under-service latch of the Channel A interrupt. At this time, the service routine for Channel B is resumed. When it is completed, another RETI instruction is executed to complete the interrupt service.



1. PRIORITY INTERRUPT DAISY CHAIN BEFORE ANY INTERRUPT OCCURS.

2. CHANNEL B TRANSMITTER INTERRUPTS AND IS ACKNOWLEDGED.

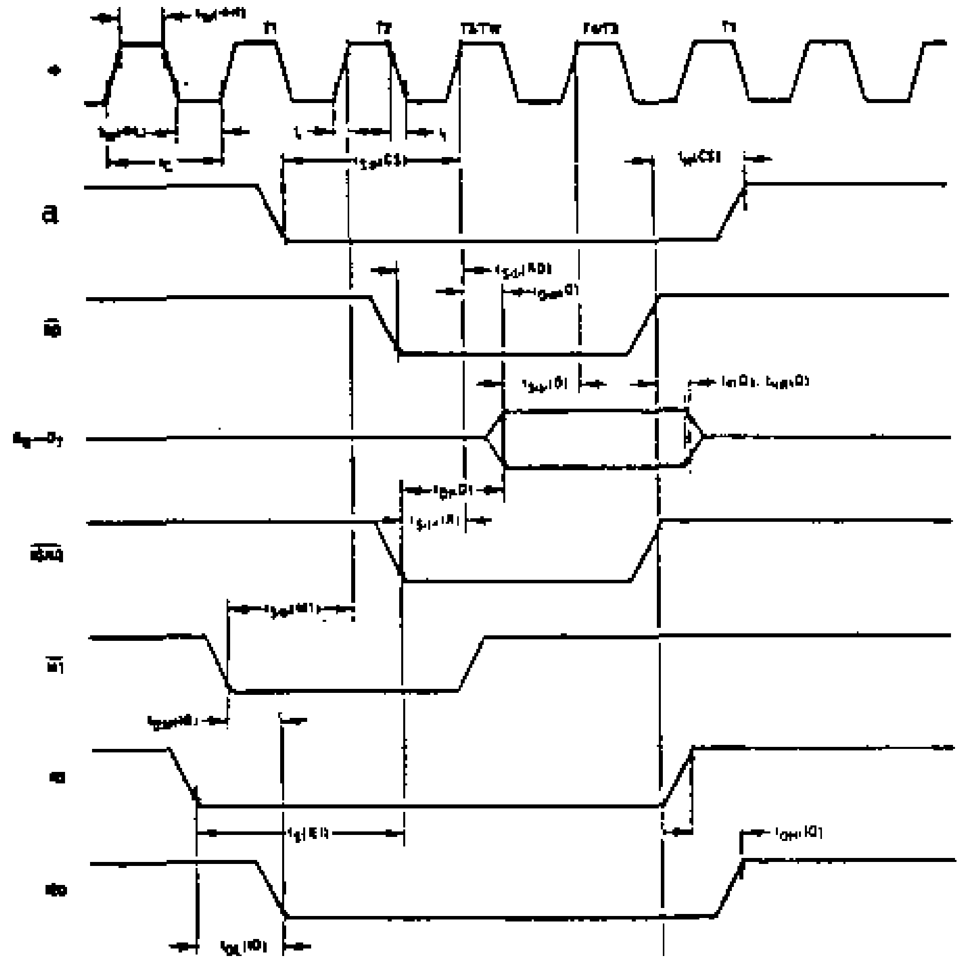
3. EXTERNAL/STATUS OF CHANNEL A INTERRUPTS SUSPENDING SERVICE OF CHANNEL B TRANSMITTER.

4. CHANNEL A EXTERNAL/STATUS ROUTINE COMPLETE. RETI ISSUED. CHANNEL B TRANSMITTER SERVICE RESUMED.

5. CHANNEL B TRANSMITTER SERVICE ROUTINE COMPLETE. SECOND RETI ISSUED.

Figure 9. Typical Interrupt Sequence

$T_A = 0°C, V_{CC} = +5V, \pm 5\%$

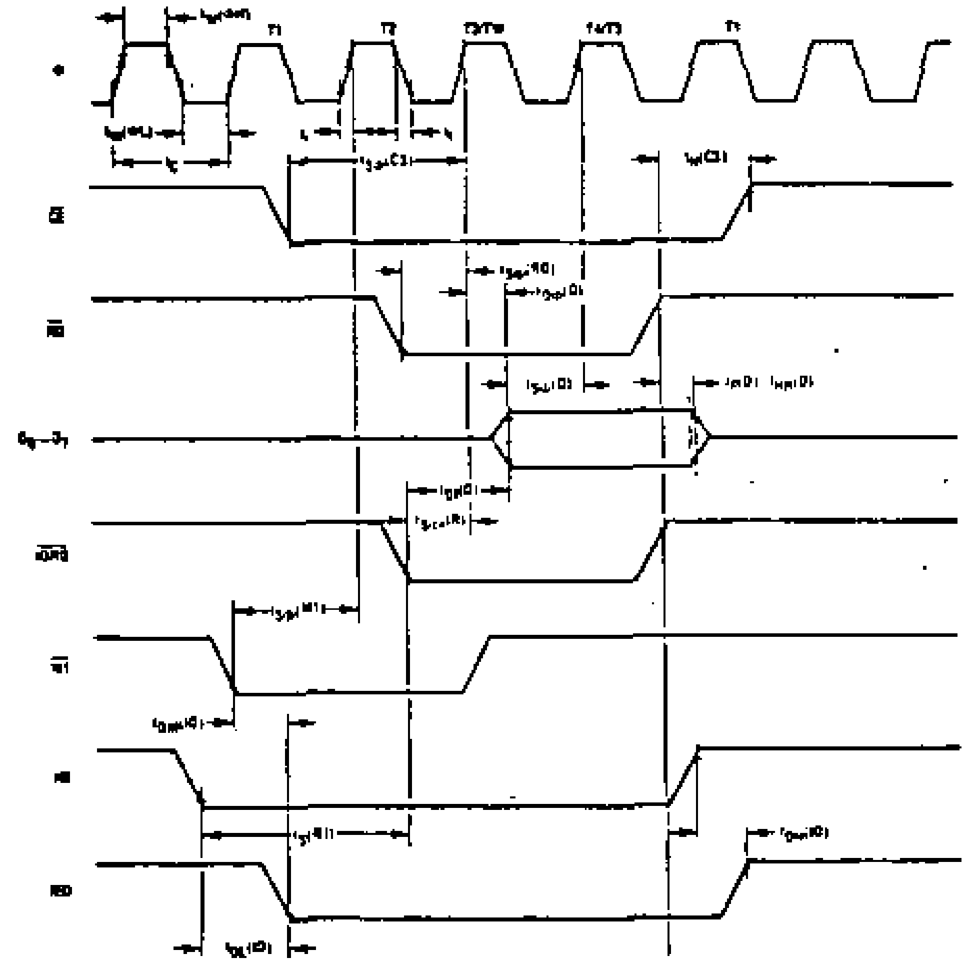


			Z80-SIO		Z80A-SIO		
Signal	Symbol	Parameter	Min	Max	Min	Max	Unit
Φ	t_c	Clock Period	400	4000	250	4000	ns
	$t_w(ΦH)$	Clock Pulse Width, clock HIGH	170	2000	105	2000	ns
	$t_w(ΦL)$	Clock Pulse Width, clock LOW	170	2000	105	2000	ns
	t_r, t_f	Clock Rise and Fall Times	0	30	0	30	ns
	t_h	Any Unspecified Hold Time for setup times specified below	0		0		ns
$\overline{CE}$, B/$\overline{A}$, C/$\overline{D}$, $\overline{IORQ}$	$t_{SΦ}(CS)$	Control Signal Setup Time to rising edge of Φ during Read or Write Cycle	160		145		ns
D_0-D_7	$t_{DR}(D)$	Data Output Delay from rising edge of Φ during Read Cycle		240		220	ns
	$t_{SΦ}(D)$	Data Setup Time to rising edge of Φ during Write or M1 Cycle	50		50		ns
	$t_{DI}(D)$	Data Output Delay from falling edge of $\overline{IORQ}$ during INTA Cycle		340		160	ns
	$t_F(D)$	Delay to Floating Bus (output buffer disable time)		230		110	ns
IEI	$t_S(IEI)$	IEI Setup Time to falling edge of $\overline{IORQ}$ during INTA Cycle	200		140		ns
IEO	$t_{DH}(IO)$	IEO Delay Time from rising edge of IEI (after ED decode)		150		100	ns
	$t_{DL}(IO)$	IEO Delay Time from falling edge of IEI		150		100	ns
	$t_{DM}(IO)$	IEO Delay Time from falling edge of $\overline{M1}$ (interrupt occurring just prior to M1)		300		190	ns
$\overline{M1}$	$t_{SΦ}(M1)$	$\overline{M1}$ Setup Time to rising edge of Φ during INTA or M1 Cycle	210		90		ns
$\overline{RD}$	$t_{SΦ}(RD)$	$\overline{RD}$ Setup Time to rising edge of Φ during Read or M1 Cycle	240		115		ns

*If WAIT from the SIO is to be used, $\overline{CE}$, $\overline{IORQ}$, C/$\overline{D}$ and $\overline{M1}$ must be valid for as long as the Wait condition is to persist.

Figure 9. Typical Interrupt Sequence

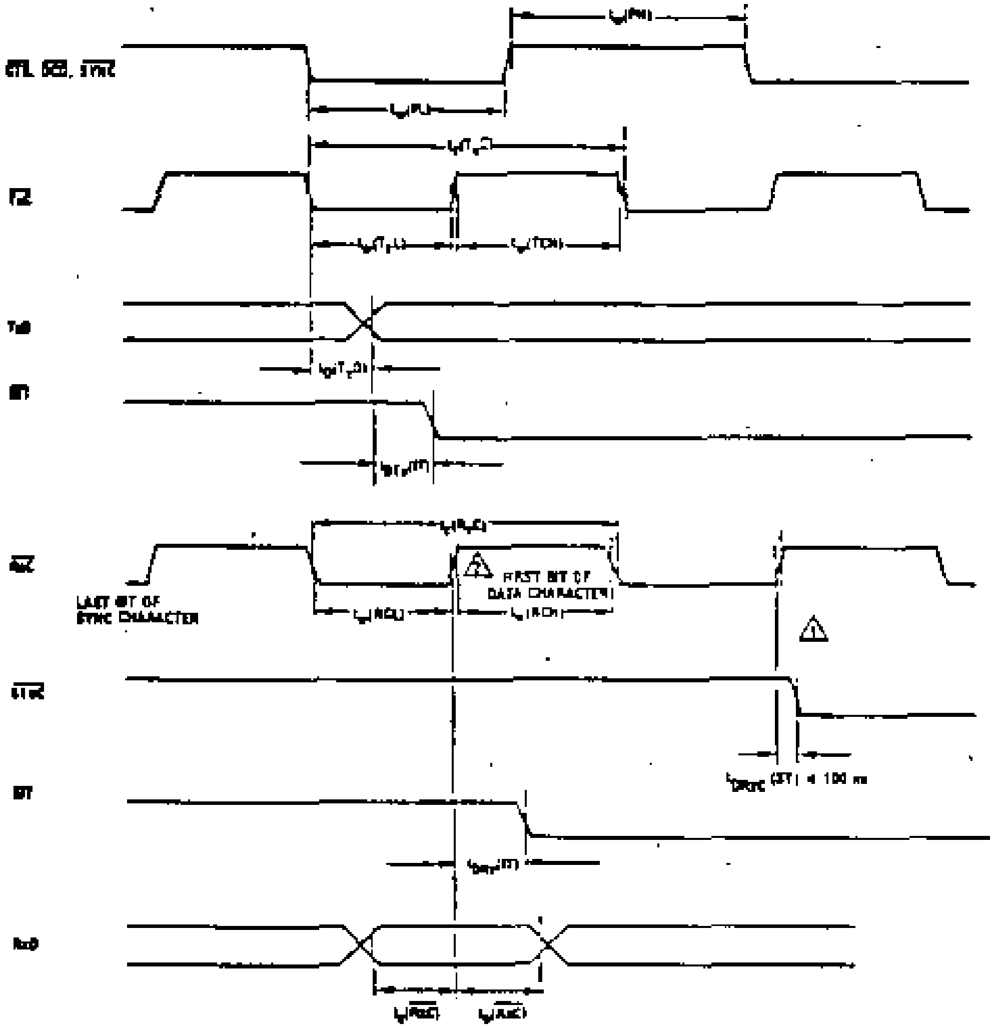
$T_A = 0°C, V_{CC} = +5V, \pm 5\%$



Signal	Symbol	Parameter	Z80-SIO Min	Z80-SIO Max	Z80A-SIO Min	Z80A-SIO Max	Unit
Φ	tc(Φ)	Clock Period	400	4000	250	4000	ns
	tw(ΦH)	Clock Pulse Width, clock HIGH	170	2000	105	2000	ns
	tw(ΦL)	Clock Pulse Width, clock LOW	170	2000	105	2000	ns
	tr, tf	Clock Rise and Fall Times	0	30	0	30	ns
	th	Any Unspecified Hold Time for setup times specified below	0		0		ns
CE, B/A C/D, IORQ	tsΦ(CS)	Control Signal Setup Time to rising edge of Φ during Read or Write Cycle	160		145		ns
D0-D7	tdR(D)	Data Output Delay from rising edge of Φ during Read Cycle		240		220	ns
	tsΦ(D)	Data Setup Time to rising edge of Φ during Write or M1 Cycle	50		50		ns
	tdI(D)	Data Output Delay from falling edge of IORQ during INTA Cycle		340		160	ns
	tF(D)	Delay to Floating Bus (output buffer disable time)		230		110	ns
IEI	tsIEI(IO)	IEI Setup Time to falling edge of IORQ during INTA Cycle	200		140		ns
IEO	tdH(IO)	IEO Delay Time from rising edge of IEI (after ED decode)		150		100	ns
	tdL(IO)	IEO Delay Time from falling edge of IEI		150		100	ns
	tdM(IO)	IEO Delay Time from falling edge of M1 (interrupt occurring just prior to M1)		300		190	ns
M1	tsM1	M1 Setup Time to rising edge of Φ during INTA or M1 Cycle	210		90		ns
RD	tsRD	RD Setup Time to rising edge of Φ during Read or M1 Cycle	240		115		ns

*If WAIT from the SIO is to be used, CE, IORQ, C/D and M1 must be valid for as long as the Wait condition is to persist.

Figure 9. Typical Interrupt Sequence

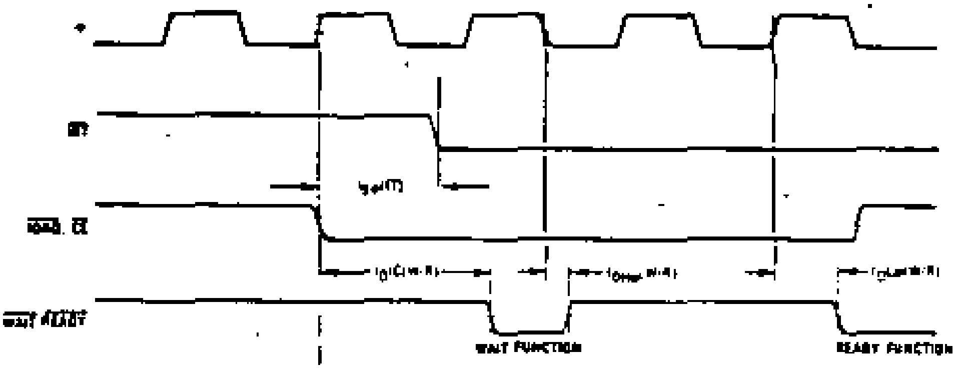


NOTES:

1. The SYNC input must be driven Low on the rising edge of RxC delayed two complete clock cycles from the last bit of the sync character.
2. Data character assembly begins on the first Receive Clock cycle after the last bit of the sync character is received.

Signal	Symbol	Parameter	Z80-SIO Min	Z80-SIO Max	Z80A-SIO Min	Z80A-SIO Max	Unit
INT	tdRC(IT)	INT Delay Time from rising edge of RxC	10	13	10	13	Φ periods
	tdRI(IT)	INT Delay Time from Middle of Next Data Bit	5	9	5	9	Φ periods
CTSA, CTSB DCDA, DCDB SYNCA, SYNCB	tw(PH)	Minimum HIGH Pulse Width for latching state into Register and generating interrupt	200		200		ns
	tw(PL)	Minimum LOW Pulse Width for latching state into Register and generating interrupt	200		200		ns
SYNCA, SYNCB	tdRC(SY)	Sync Pulse Delay Time from rising edge of RxC, Output Modes	4	7	4	7	Φ periods
	tsRC(SY)	Sync Pulse Delay Time from rising edge of RxC, External Sync Mode		100		100	ns
TxCA, TxCB	tc(TxC)	Transmit Clock Period	400	∞	400	∞	ns
	tw(TCH)	Transmit Clock Pulse Width, clock HIGH	180	∞	180	∞	ns
	tw(TCL)	Transmit Clock Pulse Width, clock LOW	180	∞	180	∞	ns
TxDA, TxDB*	td(TxD)	TxD Output Delay from falling Edge of TxC (x1 Clock Mode)		400		300	ns
RxCA, RxCB	tc(RxC)	Receive Clock Period	400	∞	400	∞	ns
	tw(RCH)	Receive Clock Pulse Width, clock HIGH	180	∞	180	∞	ns
	tw(RCL)	Receive Clock Pulse Width, clock LOW	180	∞	180	∞	ns
RxDA, RxDB*	ts(RxC)	Setup Time to rising edge of RxC (x1 mode)	0		0		ns
	th(RxC)	Hold Time from rising edge of RxC (x1 mode)	140		140		ns

*In all modes, the system clock rate must be at least 4.5 times the maximum data rate. RESET must be active a minimum of one complete Φ cycle.



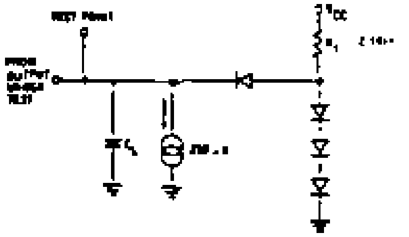
			Z80-SIO		Z80A-SIO		
Signal	Symbol	Parameter	Min	Max	Min	Max	Unit
INT	$t_{dIO}(IT)$	INT Delay Time from rising edge of Φ		200		200	ns
	$t_{dIC}(W/R)$	WAIT/READY Delay Time from IORQ or CE in Wait Mode		140		130	ns
	$t_{dH}(W/R)$	WAIT/READY Delay Time from rising edge of Φ WAIT/READY HIGH, Wait Mode		150		130	ns
WAIT/READY	$t_{dRd}(W/R)$	WAIT/READY Delay Time from rising edge of RxC Data Bit, Ready Mode	10	13	10	13	Φ periods
	$t_{dTx}(W/R)$	WAIT/READY Delay Time from center of Transmit Data Bit, Ready Mode	5	9	5	9	Φ periods
	$t_{dL}(W/R)$	WAIT/READY Delay Time from rising edge of Φ WAIT/READY LOW, Ready Mode		120		120	ns

$T_A = 0°C$ to $70°C$, $V_{CC} = +5V, \pm 5\%$

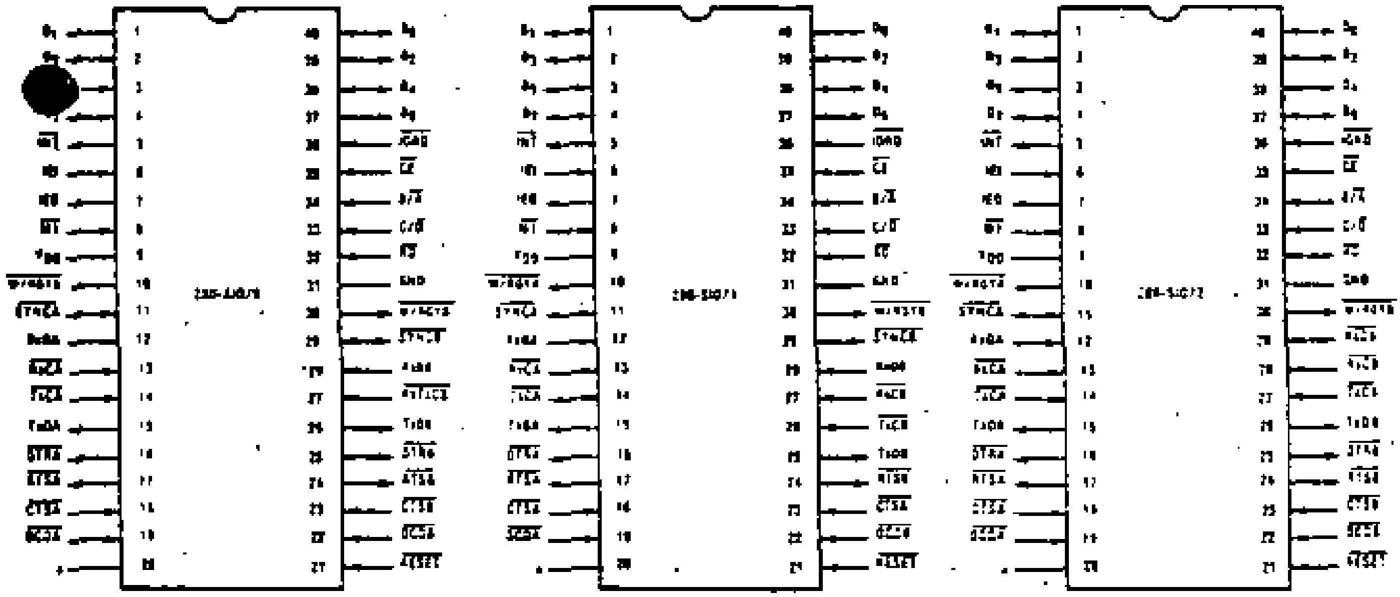
Symbol	Parameter	Min.	Max.	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	−0.3	+0.45	V	
V_{IHC}	Clock Input High Voltage	V_{CC} −0.6	+5.5	V	
V_{IL}	Input Low Voltage	−0.3	+0.8	V	
V_{IH}	Input High Voltage	+2.0	+5.5	V	
V_{OL}	Output Low Voltage		+0.4	V	$I_{OL} = 2.0$ mA
V_{OH}	Output High Voltage	+2.4		V	$I_{OH} = -250$ µA
I_{LI}	Input Leakage Current	−10	+10	µA	$0 < V_{IN} < V_{CC}$
I_Z	3-State Output/Data Bus Input Leakage Current	−10	+10	µA	$0 < V_{IN} < V_{CC}$
$I_{L(SY)}$	SYNC Pin Leakage Current	−40	+10	µA	
I_{CC}	Power Supply Current		100	mA	

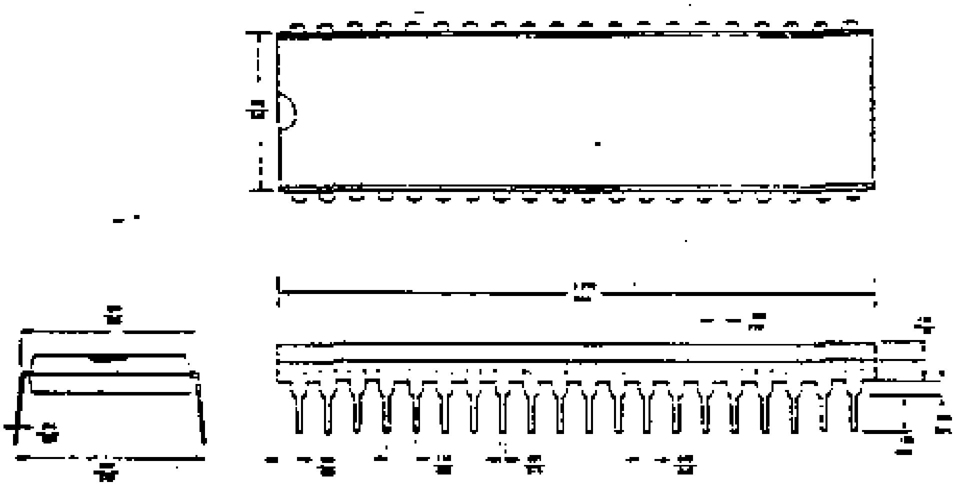
$T_A = 25°C$, $f = 1$ MHz

Symbol	Parameter	Min.	Max.	Unit	Test Condition
C	Clock Capacitance		40	pF	Unmeasured pins returned to ground
C_{IN}	Input Capacitance		5	pF	
C_{OUT}	Output Capacitance		10	pF	

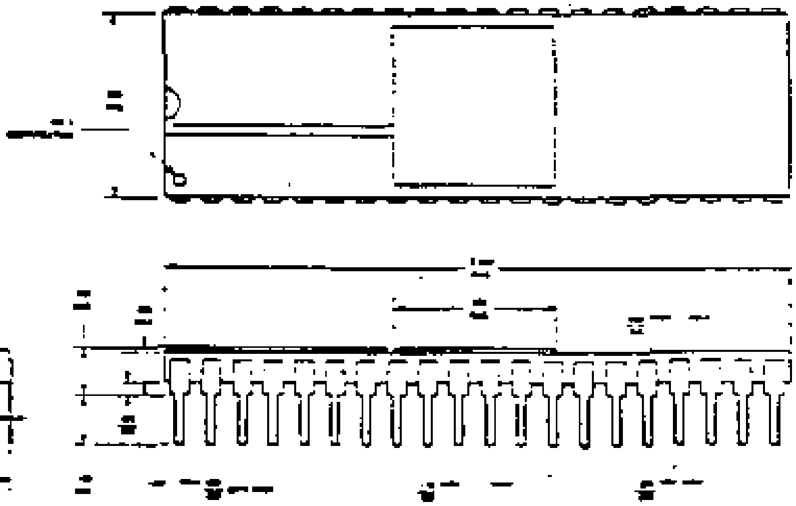


$C_L = 50$ pF. Increase delay by 10 ns for each 50 pF increase in C_L, up to 200 pF maximum.





40-Pin Plastic



40-Pin Ceramic

/0 — Type 0 Bonding
/1 — Type 1 Bonding
/2 — Type 2 Bonding

C — Ceramic Package
P — Plastic Package

S — Standard Range (5V, ±5%, 0°C to 70°C)
E — Extended Range (5V, ±5%, −40°C to 85°C)
M — Military Range (5V, ±10%, −55°C to 125°C)

Examples:

Z80-SIO/1 CS (Type 1 bonding, ceramic package, standard range)

Z80-SIO/2 PS (Type 0 bonding, plastic package, standard range)

EAST

Sales & Tech. Center
Zilog, Inc.
76 Treble Cove Road
N. Billerica, MA 01862
TEL 617 667 2179
TWX 710 347 6660

MIDATLANTIC

Tech. Center
Zilog, Inc.
P.O. Box 92
Bergenfield, NJ 07621
TEL 201 385 9158
TWX 710 991 9771

MIDWEST

Sales and Tech. Center
Zilog, Inc.
130 E. Higgins
Suite 114
Schaumburg, IL 60195
TEL 312 885 8080
TWX 910 291 1064

NORTH CENTRAL

Sales and Tech. Center
Zilog, Inc.
1303 Bethel Road
Columbus, Ohio 43220
TEL 614 457 0822
TWX 810 482 1707

SOUTHWEST REGION

Sales and Tech. Center
Zilog, Inc.
18002 Sky Park Blvd.
Suite K
Irvine, CA 92714
TEL 714 549 2891
TWX 910 595 2803

NORTHWEST

Sales & Tech. Center
Zilog, Inc.
10340 Bubb Road
Cupertino, CA 95014
TEL 408 446 4666 x261
TWX 910 338 7621

WEST GERMANY

Zilog GmbH
Zugspitzstrasse 4
D-8011 Vaterstetten
West Germany
TEL 8106 4035
TELEX 529110 zilog d.

JAPAN

Zilog, Inc. Japan
Kyoshin Bldg.
13-14 Sakuragaoka-Machi
Shibuya-Ku Tokyo 150
Japan
TEL 03-476-3010

UNITED KINGDOM

Zilog (U.K.) Limited
Nicholson House
Maidenhead SL6 1LD
Berkshire United Kingdom
TEL (0628) 36131
TELEX [illegible]

10460 Bubb Road, Cupertino, California 95014 Telephone: (408) 446-4666 TWX 910-338-7621

03-0039-03 Printed in U.S.A.

Z80®-CPU Z80A-CPU

Product Specification

MARCH 1978

The Zilog Z80 product line is a complete set of microcomputer components, development systems and support sofrware. The Z80 microcomputer component set includes all of the circuits necessary to build high-performance microcomputer systems with virtually no other logic and a minimum number of low cost standard memory elements.

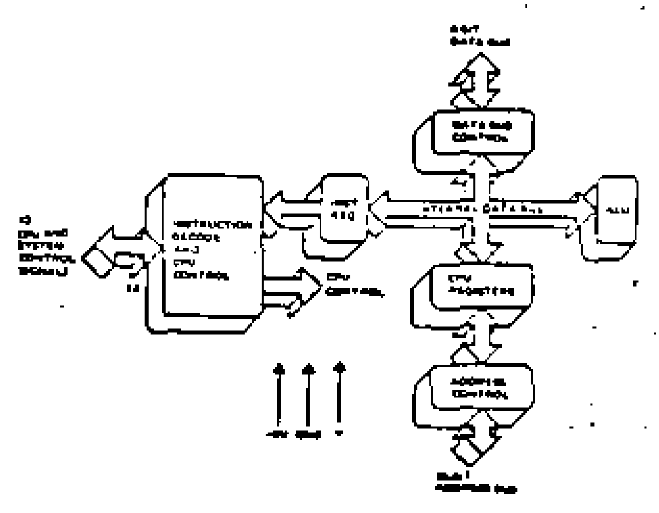
The Z80 and Z80A CPU's are third generation single chip microprocessors with unrivaled computational power. This increased computational power results in higher system through-put and more efficient memory utilization when compared to second generation microprocessors. In addition, the Z80 and Z80A CPU's are very easy to implement into a system because of their single voltage requirement plus all output signals are fully decoded and timed to control standard memory or peripheral circuits. The circuit is implemented using an N-channel, ion implanted, silicon gate MOS process.

Figure 1 is a block diagram of the CPU, Figure 2 details nternal register configuration which contains 208 bits ead/Write memory that are accessible to the programmer. The registers include two sets of six general purpose registers that may be used individually as 8-bit registers or 16-bit register pairs. There are also two sets of accumulator and flag registers. The programmer has access to either set of main or alternate registers through a group of exchange instructions. This alternate set allows foreground/background mode of operation or may be reserved for very fast Interrupt response. Each CPU also contains a 16-bit stack pointer which permits simple implementation of multiple level interrupts, unlimited subroutine nesting and simplification of many types of data handling.

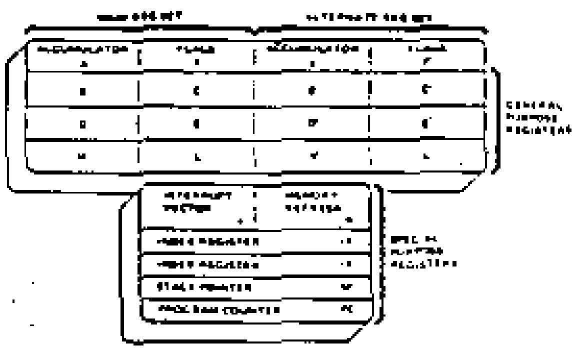
The two 16-bit index registers allow tabular data manipulation and easy implementation of relocatable code. The Refresh register provides for automatic, totally transparent refresh of external dynamic memories. The I register is used in a powerful interrupt response mode to form the upper 8 bits of a pointer to a interrupt service address table, while the interrupting device supplies the lower 8 bits of the pointer. An indirect call is then made to this service address.

FEATURES

- Single chip, N-channel Silicon Gate CPU.
- 158 instructions—includes all 78 of the 8080A instructions with total software compatibility. New instructions include 4-, 8- and 16-bit operations with more useful addressing modes such as indexed, bit and relative.
- 17 internal registers.
- Three modes of Fast interrupt response plus a non-maskable interrupt.
- Directly interfaces standard speed static or dynamic memories with virtually no external logic.
- 1.0 μs instruction execution speed.
- Single 5 VDC supply and single-phase 5 volt Clock.
- Out-performs any other single chip microcomputer in 4-, 8-, or 16-bit applications.
- All pins TTL Compatible
- Built-in dynamic RAM refresh circuitry.

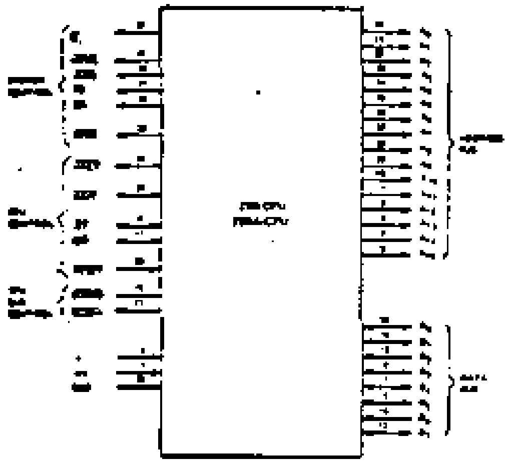


Z80, Z80A CPU BLOCK DIAGRAM



Z80, Z80A CPU REGISTERS

Z80, Z80A-CPU Pin Description



Z80, Z80A CPU PIN CONFIGURATION

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

D_0-D_7 (Data Bus) — Tri-state input/output, active high. D_0 - D_7 constitute an 8-bit bidirectional data bus. The data bus is used for data exchanges with memory and I/O devices.

$\overline{M_1}$ (Machine Cycle one) — Output, active low, $\overline{M_1}$ indicates that the current machine cycle is the OP code fetch cycle of an instruction execution.

$\overline{MREQ}$ (Memory Request) — Tri-state output, active low. The memory request signal indicates that the address bus holds a valid address for a memory read or memory write operation.

$\overline{IORQ}$ (Input/Output Request) — Tri-state output, active low. The $\overline{IORQ}$ signal indicates that the lower half of the address bus holds a valid I/O address for a I/O read or write operation. An $\overline{IORQ}$ signal is also generated when an interrupt is being acknowledged to indicate that an interrupt response vector can be placed on the data bus.

$\overline{RD}$ (Memory Read) — Tri-state output, active low. $\overline{RD}$ 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.

(Memory Write) — Tri-state output, active low. $\overline{WR}$ indicates that the CPU data bus holds valid data to be stored in the addressed memory or I/O device.

$\overline{RFSH}$ (Refresh) — Output, active low, $\overline{RFSH}$ indicates that the lower 7 bits of the address bus contain a refresh address for dynamic memories and the current $\overline{MREQ}$ signal should be used to do a refresh read to all dynamic memories.

$\overline{HALT}$ (Halt state) — Output, active low. $\overline{HALT}$ indicates that the CPU has executed a HALT software 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 NOP's to maintain memory refresh activity.

$\overline{WAIT}$ (Wait) — Input, active low. $\overline{WAIT}$ indicates to the Z-80 CPU that the addressed memory or I/O devices are not ready for a data transfer. The CPU continues to enter wait states for as long as this signal is active.

$\overline{INT}$ (Interrupt Request) — Input, active low. The Interrupt Request signal is generated by I/O devices. A request will be honored at the end of the current instruction if the internal software controlled interrupt enable flip-flop (IFF) is enabled.

$\overline{NMI}$ (Non Maskable Interrupt) — Input, active low. The non-maskable interrupt request line has a higher priority than $\overline{INT}$ and is always recognized at the end of the current instruction, independent of the status of the interrupt enable flip-flop. $\overline{NMI}$ automatically forces the Z-80 CPU to restart to location 0066_H.

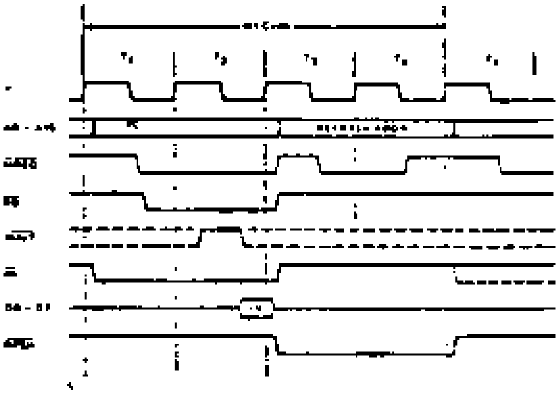
$\overline{RESET}$ — Input, active low. $\overline{RESET}$ initializes the CPU as follows: reset interrupt enable flip-flop, clear PC and registers I and R and set interrupt to 5080A mode. During reset time, the address and data bus go to a high impedance state and all control output signals go to the inactive state.

$\overline{BUSRQ}$ (Bus Request) — Input, active low. The bus request signal has a higher priority than $\overline{NMI}$ and is always recognized at the end of the current machine cycle and is used to request the CPU address bus, data bus and tri-state output control signals to go to a high impedance state so that other devices can control these busses.

$\overline{BUSAK}$ (Bus Acknowledge) — Output, active low. Bus acknowledge is used to indicate to the requesting device that the CPU address bus, data bus and tri-state control bus signals have been set to their high impedance state and the external device can now control these signals.

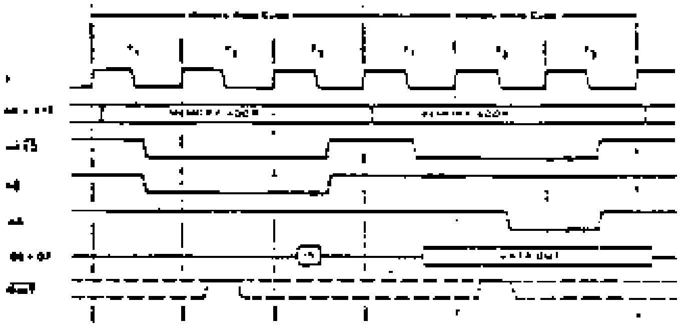
INSTRUCTION OP CODE FETCH

The program counter content (PC) is placed on the address bus immediately at the start of the cycle. One half clock time later $\overline{MREQ}$ goes active. The falling edge of $\overline{MREQ}$ can be used directly as a chip enable to dynamic memories. $\overline{RD}$ when active indicates that the memory data should be enabled onto the CPU data bus. The CPU samples data with the rising edge of the clock state T_3. Clock states T_3 and T_4 of a fetch cycle are used to refresh dynamic memories while the CPU is internally decoding and executing the instruction. The refresh control signal $\overline{RFSH}$ indicates that a refresh read of all dynamic memories should be accomplished.



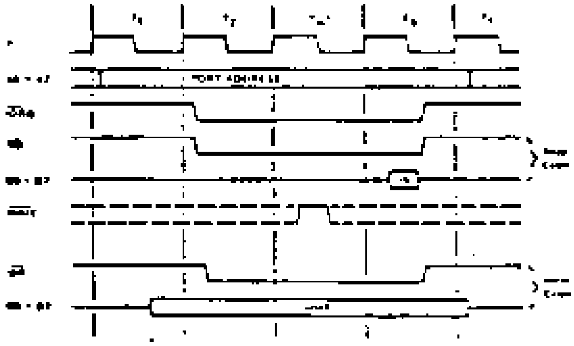
MEMORY READ OR WRITE CYCLES

Illustrated here is the timing of memory read or write cycles other than an OP code fetch (M_1 cycle). The $\overline{MREQ}$ and $\overline{RD}$ signals are used exactly as in the fetch cycle. In the case of a memory write cycle, the $\overline{MREQ}$ also becomes active when the address bus is stable so that it can be used directly as a chip enable for dynamic memories. The $\overline{WR}$ line is active when data on the data bus is stable so that it can be used directly as a R/W pulse to virtually any type of semiconductor memory.



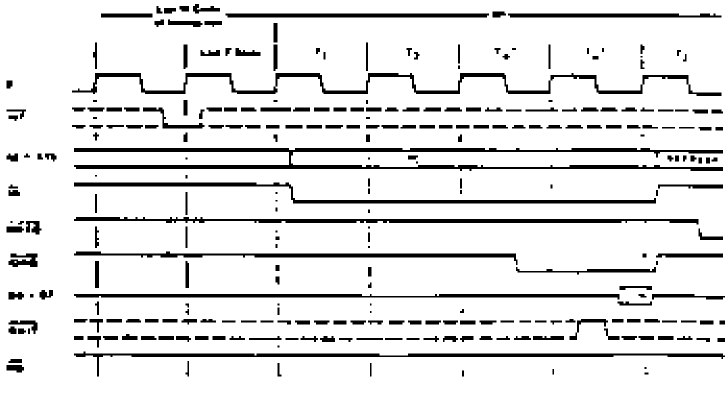
INPUT OR OUTPUT CYCLES

Illustrated here is the timing for an I/O read or I/O write operation. Notice that during I/O operations a single wait state is automatically inserted (Tw*). The reason for this is that during I/O operations this extra state allows sufficient time for an I/O port to decode its address and activate the $\overline{WAIT}$ line if a wait is required.



INTERRUPT REQUEST/ACKNOWLEDGE CYCLE

The interrupt signal is sampled by the CPU with the rising edge of the last clock at the end of any instruction. When an interrupt is accepted, a special M_1 cycle is generated. During this M_1 cycle, the $\overline{IORQ}$ signal becomes active (instead of $\overline{MREQ}$) to indicate that the interrupting device can place an 8-bit vector on the data bus. Two wait states (Tw*) are automatically added to this cycle so that a ripple priority interrupt scheme, such as the one used in the Z80 peripheral controllers, can be easily implemented.



Z80, Z80A Instruction Set

The following is a summary of the Z80, Z80A instruction set showing the assembly language mnemonic and the symbolic operation performed by the instruction. A more detailed listing appears in the Z80-CPU technical manual, and assembly language programming manual. The instructions are divided into the following categories:

8-bit loads	Miscellaneous Group
16-bit loads	Rotates and Shifts
Exchanges	Bit Set, Reset and Test
Memory Block Moves	Input and Output
Memory Block Searches	Jumps
8-bit arithmetic and logic	Calls
16-bit arithmetic	Restarts
General purpose Accumulator & Flag Operations	Returns

In the table the following terminology is used.

b ≡ a bit number in any 8-bit register or memory location

cc ≡ flag condition code

NZ	≡	non zero
Z	≡	zero
NC	≡	non carry
C	≡	carry
PO	≡	Parity odd or no over flow
PE	≡	Parity even or over flow
P	≡	Positive
M	≡	Negative (minus)

d ≡ any 8-bit destination register or memory location

dd ≡ any 16-bit destination register or memory location

e ≡ 8-bit signed 2's complement displacement used in relative jumps and indexed addressing

L ≡ 8 special call locations in page zero. In decimal notation these are 0, 8, 16, 24, 32, 40, 48 and 56

n ≡ any 8-bit binary number

nn ≡ any 16-bit binary number

r ≡ any 8-bit general purpose register (A, B, C, D, E, H, or L)

s ≡ any 8-bit source register or memory location

s_b ≡ a bit in a specific 8-bit register or memory location

ss ≡ any 16-bit source register or memory location

subscript "L" ≡ the low order 8 bits of a 16-bit register

subscript "H" ≡ the high order 8 bits of a 16-bit register

( ) ≡ the contents within the ( ) are to be used as a pointer to a memory location or I/O port number

8-bit registers are A, B, C, D, E, H, L, I and R

16-bit register pairs are AF, BC, DE and HL

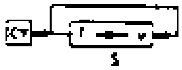
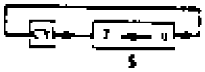
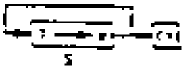
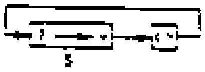
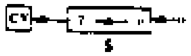
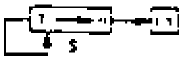
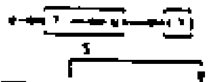
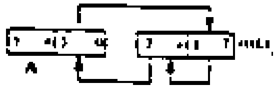
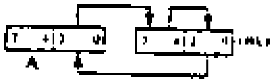
16-bit registers are SP, PC, IX and IY

Addressing Modes implemented include combinations of the following:

Immediate	Indexed
Immediate extended	Register
Modified Page Zero	Implied
Relative	Register Indirect
Extended	Bit

	Mnemonic	Symbolic Operation	Comments
8 BIT LOADS	LD r, s	r ← s	s ≡ r, n, (HL), (IX+e), (IY+e)
	LD d, r	d ← r	d ≡ (HL), r (IX+e), (IY+e)
	LD d, n	d ← n	d ≡ (HL), (IX+e), (IY+e)
	LD A, s	A ← s	s ≡ (BC), (DE), (nn), I, R
	LD d, A	d ← A	d ≡ (BC), (DE), (nn), I, R
16 BIT LOADS	LD dd, nn	dd ← nn	dd ≡ BC, DE, HL, SP, IX, IY
	LD dd, (nn)	dd ← (nn)	dd ≡ BC, DE, HL, SP, IX, IY
	LD (nn), ss	(nn) ← ss	ss ≡ BC, DE, HL, SP, IX, IY
	LD SP, ss	SP ← ss	ss ≡ HL, IX, IY
	PUSH ss	$(SP-1) \leftarrow ss_H; (SP-2) \leftarrow ss_L$	ss ≡ BC, DE, HL, AF, IX, IY
	POP dd	$dd_L \leftarrow (SP); dd_H \leftarrow (SP+1)$	dd ≡ BC, DE, HL, AF, IX, IY
EXCHANGES	EX DE, HL	DE ↔ HL	
	EX AF, AF'	AF ↔ AF'	
	EXX	$\begin{pmatrix}BC\\DE\\HL\end{pmatrix} \leftrightarrow \begin{pmatrix}BC'\\DE'\\HL'\end{pmatrix}$	
	EX (SP), ss	$(SP) \leftrightarrow ss_L, (SP+1) \leftrightarrow ss_H$	ss ≡ HL, IX, IY

	Mnemonic	Symbolic Operation	Comments
MEMORY BLOCK MOVES	LDI	(DE) ← (HL), DE ← DE+1 HL ← HL+1, BC ← BC-1	
	LDIR	(DE) ← (HL), DE ← DE+1 HL ← HL+1, BC ← BC-1 Repeat until BC = 0	
	LDD	(DE) ← (HL), DE ← DE+1 HL ← HL-1, BC ← BC-1	
	LDDR	(DE) ← (HL), DE ← DE+1 HL ← HL-1, BC ← BC-1 Repeat until BC = 0	
MEMORY BLOCK SEARCHES	CPI	A−(HL), HL ← HL+1 BC ← BC-1	
	CPIR	A−(HL), HL ← HL+1 BC ← BC-1, Repeat until BC = 0 or A = (HL)	A−(HL) sets the flags only A is not affected
	CPD	A−(HL), HL ← HL-1 BC ← BC-1	
	CPDR	A−(HL), HL ← HL-1 BC ← BC-1, Repeat until BC = 0 or A = (HL)	
8 BIT ALU	ADD s	A ← A + s	
	ADC s	A ← A + s + CY	CY is the carry flag
	SUB s	A ← A − s	
	SBC s	A ← A − s − CY	s ≡ r, n, (HL) (IX+e), (IY+e)
	AND s	A ← A ∧ s	
	OR s	A ← A ∨ s	
	XOR s	A ← A ⊕ s	

	Mnemonic	Symbolic Operation	Comments
	CP s	A − s	s = r, n (HL) (IX+e), (IY+e)
	'C d	d ← d + 1	d = r, (HL) (IX+e), (IY+e)
	DEC d	d ← d − 1	
16-BIT ARITHMETIC	ADD HL, ss	HL ← HL + ss	ss ≡ BC, DE HL, SP
	ADC HL, ss	HL ← HL + ss + CY	
	SBC HL, ss	HL ← HL − ss − CY	
	ADD IX, ss	IX ← IX + ss	ss ≡ BC, DE, IX, SP
	ADD IY, ss	IY ← IY + ss	ss ≡ BC, DE, IY, SP
	INC dd	dd ← dd + 1	dd ≡ BC, DE, HL, SP, IX, IY
	DEC dd	dd ← dd − 1	dd ≡ BC, DE, HL, SP, IX, IY
GP ACC. & FLAG	DAA	Converts A contents into packed BCD following add or subtract.	Operands must be in packed BCD format
	CPL	$A \leftarrow \overline{A}$	
	NEG	A ← 00 − A	
	CCF	$CY \leftarrow \overline{CY}$	
	SCF	CY ← 1	
MISCELLANEOUS	'P	No operation	
	.LT	Halt CPU	
	DI	Disable Interrupts	
	EI	Enable Interrupts	
	IM 0	Set interrupt mode 0	8080A mode
	IM 1	Set interrupt mode 1	Call to 0038_H
	IM 2	Set interrupt mode 2	Indirect Call
ROTATES AND SHIFTS	RLC s		s ≡ r, (HL) (IX+e), (IY+e)
	RL s		
	RRC s		
	RR s		
	SLA s		
	SRA s		
	SRL s		
	RLD		
	RRD		

	Mnemonic	Symbolic Operation	Comments
BIT S, R, & T	BIT b, s	$Z \leftarrow \overline{s_b}$	Z is zero flag s ≡ r, (HL) (IX+e), (IY+e)
	SET b, s	$s_b \leftarrow 1$	
	RES b, s	$s_b \leftarrow 0$	
INPUT AND OUTPUT	IN A, (n)	A ← (n)	
	IN r, (C)	r ← (C)	Set flags
	INI	(HL) ← (C), HL ← HL + 1 B ← B − 1	
	INIR	(HL) ← (C), HL ← HL + 1 B ← B − 1 Repeat until B = 0	
	IND	(HL) ← (C), HL ← HL − 1 B ← B − 1	
	INDR	(HL) ← (C), HL ← HL − 1 B ← B − 1 Repeat until B = 0	
	OUT(n), A	(n) ← A	
	OUT(C), r	(C) ← r	
	OUTI	(C) ← (HL), HL ← HL + 1 B ← B − 1	
	OTIR	(C) ← (HL), HL ← HL + 1 B ← B − 1 Repeat until B = 0	
	OUTD	(C) ← (HL), HL ← HL − 1 B ← B − 1	
	OTDR	(C) ← (HL), HL ← HL − 1 B ← B − 1 Repeat until B = 0	
JUMPS	JP nn	PC ← nn	cc { NZ PO Z PE NC P C M
	JP cc, nn	If condition cc is true PC ← nn, else continue	
	JR e	PC ← PC + e	
	JR kk, e	If condition kk is true PC ← PC + e, else continue	kk { NZ NC Z C
	JP (ss)	PC ← ss	ss = HL, IX, IY
	DJNZ e	B ← B − 1, if B = 0 continue, else PC ← PC + e	
CALLS	CALL nn	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$, PC ← nn	cc { NZ PO Z PE NC P C M
	CALL cc, nn	If condition cc is false continue, else same as CALL nn	
RESTARTS	RST L	$(SP-1) \leftarrow PC_H$ $(SP-2) \leftarrow PC_L$, $PC_H \leftarrow 0$ $PC_L \leftarrow L$	
RETURNS	RET	$PC_L \leftarrow (SP)$, $PC_H \leftarrow (SP+1)$	
	RET cc	If condition cc is false continue, else same as RET	cc { NZ PO Z PE NC P C M
	RETI	Return from interrupt, same as RET	
	RETN	Return from non-maskable interrupt	

$T_A = 0°C$ to $70°C$, $V_{cc} = +5V \pm 5\%$, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
Φ	t_c	Clock Period	.4	[12]	μsec	
	$t_{w(\Phi H)}$	Clock Pulse Width, Clock High	180	[E]	nsec	
	$t_{w(\Phi L)}$	Clock Pulse Width, Clock Low	180	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
A_{0-15}	$t_{D(AD)}$	Address Output Delay		145	nsec	$C_L = 50pF$
	$t_{F(AD)}$	Delay to Float		110	nsec	
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable from $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ or $\overline{MREQ}$	[3]		nsec	
	t_{caf}	Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	[4]		nsec	
D_{0-7}	$t_{D(D)}$	Data Output Delay		230	nsec	$C_L = 50pF$
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	50		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During M2 to M5	60		nsec	
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{WR}$	[7]			
	t_H	Any Hold Time for Setup Time	0		nsec	
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low		100	nsec	$C_L = 50pF$
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		100	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		100	nsec	
	$t_{w(\overline{MRL})}$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	
	$t_{w(\overline{MRH})}$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		90	nsec	$C_L = 50pF$
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		110	nsec	
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		100	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		110	nsec	
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		100	nsec	$C_L = 50pF$
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		130	nsec	
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		100	nsec	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		110	nsec	
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		80	nsec	$C_L = 50pF$
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ Low		90	nsec	
	$t_{DH\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ High		100	nsec	
	$t_{w(\overline{WRL})}$	Pulse Width, $\overline{WR}$ Low	[10]		nsec	
$\overline{M1}$	$t_{DL(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ Low		130	nsec	$C_L = 50pF$
	$t_{DH(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ High		130	nsec	
$\overline{RFSH}$	$t_{DL(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low		180	nsec	$C_L = 50pF$
	$t_{DH(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ High		150	nsec	
$\overline{WAIT}$	$t_{s(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{HALT}$	$t_{D(HT)}$	$\overline{HALT}$ Delay Time From Falling Edge of Clock		300	nsec	$C_L = 50pF$
$\overline{INT}$	$t_{s(IT)}$	$\overline{INT}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{NMI}$	$t_{w(\overline{NML})}$	Pulse Width, $\overline{NMI}$ Low	80		nsec	
$\overline{BUSRQ}$	$t_{s(BQ)}$	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{BUSAK}$	$t_{DL(BA)}$	$\overline{BUSAK}$ Delay From Rising Edge of Clock, $\overline{BUSAK}$ Low		120	nsec	$C_L = 50pF$
	$t_{DH(BA)}$	$\overline{BUSAK}$ Delay From Falling Edge of Clock, $\overline{BUSAK}$ High		110	nsec	
$\overline{RESET}$	$t_{s(RS)}$	$\overline{RESET}$ Setup Time to Rising Edge of Clock	90		nsec	
	$t_{F(C)}$	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		100	nsec	
	t_{mr}	$\overline{M1}$ Stable Prior to $\overline{IORQ}$ (Interrupt Ack.)	[11]		nsec	

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 75$

[2] $t_{aci} = t_c - 40$

[3] $t_{ca} = t_{w(\Phi L)} + t_r - 40$

[4] $t_{caf} = t_{w(\Phi L)} + t_r - 60$

[5] $t_{dcm} = t_c - 210$

[6] $t_{dci} = t_{w(\Phi L)} + t_r - 210$

[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 40$

[8] $t_{w(\overline{MRL})} = t_c - 40$

[9] $t_{w(\overline{MRH})} = t_{w(\Phi H)} + t_f - 30$

[10] $t_{w(\overline{WRL})} = t_c - 40$

[11] $t_{mr} = 2t_c + t_{w(\Phi H)} + t_f - 80$

NOTES

A. Data should be enabled onto the CPU data bus when $\overline{RD}$ is active. During interrupt acknowledge data should be enabled when $\overline{M1}$ and $\overline{IORQ}$ are both active.

B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.

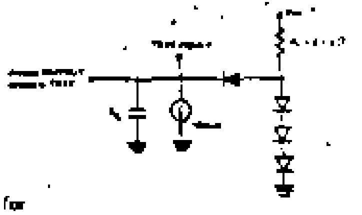
C. The $\overline{RESET}$ signal must be active for a minimum of 3 clock cycles.

D. Output Delay vs. Loaded Capacitance

$T_A = 70°C$ $V_{cc} = +5V \pm 5\%$

Add 10nsec delay for each 50pf increase in load up to a maximum of 200pf for the data bus & 100pf for address & control lines

E. Although static by design, testing guarantees $t_{w(\Phi H)}$ of 200 μsec maximum

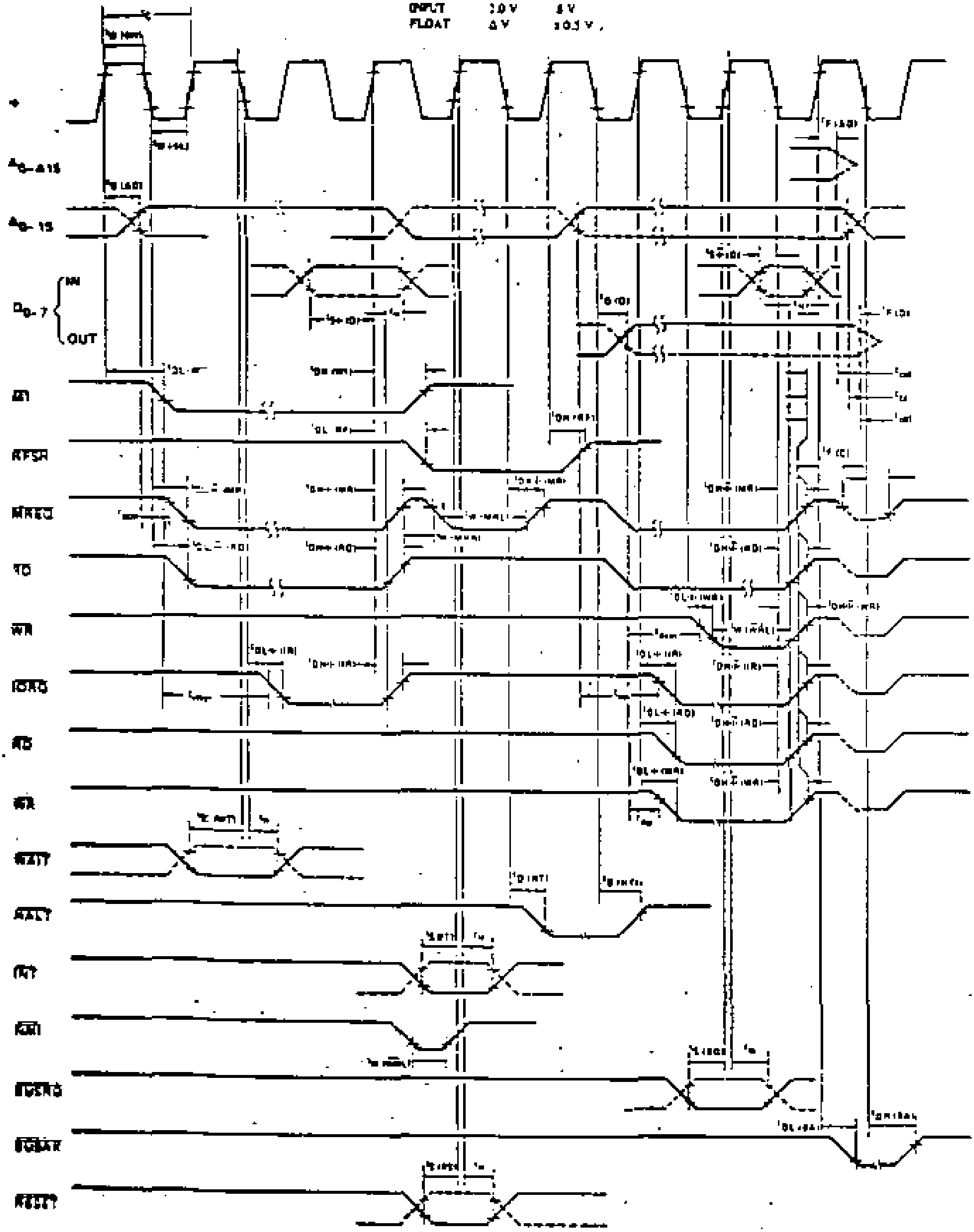


Load circuit for Output

A.C. Timing Diagram

Timing measurements are made at the following voltages, unless otherwise specified:

	"1"	"0"
CLOCK	V_{cc} -.6V	.45V
OUTPUT	2.0 V	.8 V
INPUT	2.0 V	.8 V
FLOAT	ΔV	±0.5 V



Absolute Maximum Ratings

Temperature Under Bias	Specified operating range.
Storage Temperature	-65°C to +150°C
Voltage On Any Pin with Respect to Ground	-0.3V to +7V
Power Dissipation	1.5W

*Comment

Stresses above those listed under "Absolute Maximum Rating" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other condition above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Note: For Z80-CPU all AC and DC characteristics remain the same for the military grade parts except I_{cc}.

I_{cc} = 200 mA

Z80-CPU D.C. Characteristics

T_A = 0°C to 70°C, V_{cc} = 5V ± 5% unless otherwise specified

Symbol	Parameter	Min	Typ.	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{cc} -.6		V_{cc}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{cc}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250µA
I_{CC}	Power Supply Current			150	mA	
I_{LI}	Input Leakage Current			10	µA	V_{IN}=0 to V_{cc}
I_{LOH}	Tri-State Output Leakage Current in Float			10	µA	V_{OUT}=2.4 to V_{cc}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	µA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	µA	$0 \leq V_{IN} \leq V_{cc}$

Capacitance

T_A = 25°C, f = 1 MHz,
unmeasured pins returned to ground

Symbol	Parameter	Max	Unit
C_ϕ	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

Z80-CPU Ordering Information

C – Ceramic
P – Plastic
S – Standard 5V ±5% 0° to 70°C
E – Extended 5V ±5% -40° to 85°C
M – Military 5V ±10% -55° to 125°C

Z80A-CPU D.C. Characteristics

T_A = 0°C to 70°C, V_{cc} = 5V ± 5% unless otherwise specified

Symbol	Parameter	Min	Typ	Max	Unit	Test Condition
V_{ILC}	Clock Input Low Voltage	-0.3		0.45	V	
V_{IHC}	Clock Input High Voltage	V_{cc} -.6		V_{cc}+.3	V	
V_{IL}	Input Low Voltage	-0.3		0.8	V	
V_{IH}	Input High Voltage	2.0		V_{cc}	V	
V_{OL}	Output Low Voltage			0.4	V	I_{OL}=1.8mA
V_{OH}	Output High Voltage	2.4			V	I_{OH} = -250µA
I_{CC}	Power Supply Current		90	200	mA	
I_{LI}	Input Leakage Current			10	µA	V_{IN}=0 to V_{cc}
I_{LOH}	Tri-State Output Leakage Current in Float			10	µA	V_{OUT}=2.4 to V_{cc}
I_{LOL}	Tri-State Output Leakage Current in Float			-10	µA	V_{OUT}=0.4V
I_{LD}	Data Bus Leakage Current in Input Mode			±10	µA	$0 \leq V_{IN} \leq V_{cc}$

Capacitance

T_A = 25°C, f = 1 MHz,
unmeasured pins returned to ground

Symbol	Parameter	Max	Unit
C_ϕ	Clock Capacitance	35	pF
C_{IN}	Input Capacitance	5	pF
C_{OUT}	Output Capacitance	10	pF

Z80A-CPU Ordering Information

C – Ceramic
P – Plastic
S – Standard 5V ±5% 0° to 70°C

T_A = 0°C to 70°C, Vcc = +5V ± 5%, Unless Otherwise Noted.

Signal	Symbol	Parameter	Min	Max	Unit	Test Condition
Φ	t_c	Clock Period	.25	[12]	µsec	
	$t_w(\Phi H)$	Clock Pulse Width, Clock High	110	[E]	nsec	
	$t_w(\Phi L)$	Clock Pulse Width, Clock Low	110	2000	nsec	
	$t_{r,f}$	Clock Rise and Fall Time		30	nsec	
A_{0-15}	$t_{D(AD)}$	Address Output Delay		110	nsec	C_L = 50pF
	$t_{F(AD)}$	Delay to Float		90	nsec	
	t_{acm}	Address Stable Prior to $\overline{MREQ}$ (Memory Cycle)	[1]		nsec	
	t_{aci}	Address Stable Prior to $\overline{IORQ}$, $\overline{RD}$ or $\overline{WR}$ (I/O Cycle)	[2]		nsec	
	t_{ca}	Address Stable from $\overline{RD}$, $\overline{WR}$, $\overline{IORQ}$ or $\overline{MREQ}$	[3]		nsec	
	t_{caf}	Address Stable From $\overline{RD}$ or $\overline{WR}$ During Float	[4]		nsec	
D_{0-7}	$t_{D(D)}$	Data Output Delay		150	nsec	C_L = 50pF
	$t_{F(D)}$	Delay to Float During Write Cycle		90	nsec	
	$t_{S\Phi(D)}$	Data Setup Time to Rising Edge of Clock During M1 Cycle	35		nsec	
	$t_{S\overline{\Phi}(D)}$	Data Setup Time to Falling Edge of Clock During M2 to M5	50		nsec	
	t_{dcm}	Data Stable Prior to $\overline{WR}$ (Memory Cycle)	[5]		nsec	
	t_{dci}	Data Stable Prior to $\overline{WR}$ (I/O Cycle)	[6]		nsec	
	t_{cdf}	Data Stable From $\overline{WR}$	[7]			
	t_H	Any Hold Time for Setup Time		0	nsec	
$\overline{MREQ}$	$t_{DL\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ Low		85	nsec	C_L = 50pF
	$t_{DH\Phi(MR)}$	$\overline{MREQ}$ Delay From Rising Edge of Clock, $\overline{MREQ}$ High		85	nsec	
	$t_{DH\overline{\Phi}(MR)}$	$\overline{MREQ}$ Delay From Falling Edge of Clock, $\overline{MREQ}$ High		85	nsec	
	$t_w(\overline{MRL})$	Pulse Width, $\overline{MREQ}$ Low	[8]		nsec	
	$t_w(\overline{MRH})$	Pulse Width, $\overline{MREQ}$ High	[9]		nsec	
$\overline{IORQ}$	$t_{DL\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ Low		75	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ Low		85	nsec	
	$t_{DH\Phi(IR)}$	$\overline{IORQ}$ Delay From Rising Edge of Clock, $\overline{IORQ}$ High		85	nsec	
	$t_{DH\overline{\Phi}(IR)}$	$\overline{IORQ}$ Delay From Falling Edge of Clock, $\overline{IORQ}$ High		85	nsec	
$\overline{RD}$	$t_{DL\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ Low		85	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ Low		95	nsec	
	$t_{DH\Phi(RD)}$	$\overline{RD}$ Delay From Rising Edge of Clock, $\overline{RD}$ High		85	nsec	
	$t_{DH\overline{\Phi}(RD)}$	$\overline{RD}$ Delay From Falling Edge of Clock, $\overline{RD}$ High		85	nsec	
$\overline{WR}$	$t_{DL\Phi(WR)}$	$\overline{WR}$ Delay From Rising Edge of Clock, $\overline{WR}$ Low		65	nsec	C_L = 50pF
	$t_{DL\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ Low		80	nsec	
	$t_{DH\overline{\Phi}(WR)}$	$\overline{WR}$ Delay From Falling Edge of Clock, $\overline{WR}$ High		80	nsec	
	$t_w(\overline{WRL})$	Pulse Width, $\overline{WR}$ Low	[10]		nsec	
$\overline{M1}$	$t_{DL(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ Low		100	nsec	C_L = 50pF
	$t_{DH(M1)}$	$\overline{M1}$ Delay From Rising Edge of Clock, $\overline{M1}$ High		100	nsec	
$\overline{RFSH}$	$t_{DL(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ Low		130	nsec	C_L = 50pF
	$t_{DH(RF)}$	$\overline{RFSH}$ Delay From Rising Edge of Clock, $\overline{RFSH}$ High		120	nsec	
$\overline{WAIT}$	$t_{S(WT)}$	$\overline{WAIT}$ Setup Time to Falling Edge of Clock	70		nsec	
$\overline{HALT}$	$t_{D(HT)}$	$\overline{HALT}$ Delay Time From Falling Edge of Clock		300	nsec	C_L = 50pF
$\overline{INT}$	$t_{S(IT)}$	$\overline{INT}$ Setup Time to Rising Edge of Clock	80		nsec	
$\overline{NMI}$	$t_w(\overline{NML})$	Pulse Width, $\overline{NMI}$ Low	80		nsec	
$\overline{BUSRQ}$	$t_{S(BQ)}$	$\overline{BUSRQ}$ Setup Time to Rising Edge of Clock	50		nsec	
$\overline{BUSAK}$	$t_{DL(BA)}$	$\overline{BUSAK}$ Delay From Rising Edge of Clock, $\overline{BUSAK}$ Low		100	nsec	C_L = 50pF
	$t_{DH(BA)}$	$\overline{BUSAK}$ Delay From Falling Edge of Clock, $\overline{BUSAK}$ High		100	nsec	
$\overline{RESET}$	$t_{S(RS)}$	$\overline{RESET}$ Setup Time to Rising Edge of Clock	60		nsec	
	$t_{F(C)}$	Delay to Float ($\overline{MREQ}$, $\overline{IORQ}$, $\overline{RD}$ and $\overline{WR}$)		80	nsec	
	t_{mr}	$\overline{M1}$ Stable Prior to $\overline{IORQ}$ (Interrupt Ack.)	[11]		nsec	

[12] $t_c = t_{w(\Phi H)} + t_{w(\Phi L)} + t_r + t_f$

[1] $t_{acm} = t_{w(\Phi H)} + t_f - 65$

[2] $t_{aci} = t_c - 70$

[3] $t_{ca} = t_{w(\Phi L)} + t_r - 50$

[4] $t_{caf} = t_{w(\Phi L)} + t_r - 45$

[5] $t_{dcm} = t_c - 170$

[6] $t_{dci} = t_{w(\Phi L)} + t_r - 170$

[7] $t_{cdf} = t_{w(\Phi L)} + t_r - 70$

[8] $t_{w}(\overline{MRL}) = t_c - 30$

[9] $t_{w}(\overline{MRH}) = t_{w(\Phi H)} + t_f - 20$

[10] $t_{w}(\overline{WRL}) = t_c - 30$

[11] $t_{mr} = 2t_c + t_{w(\Phi H)} + t_f - 65$

NOTES:

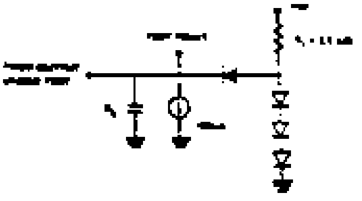
A. Data should be enabled onto the CPU data bus when $\overline{RD}$ is active. During interrupt acknowledge data should be enabled when $\overline{M1}$ and $\overline{IORQ}$ are both active.

B. All control signals are internally synchronized, so they may be totally asynchronous with respect to the clock.

C. The $\overline{RESET}$ signal must be active for a minimum of 3 clock cycles.

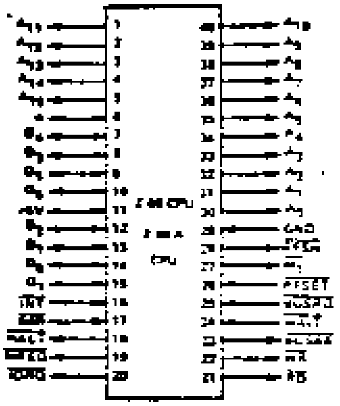
D. Output Delay vs. Loaded Capacitance
TA = 70°C Vcc = +5V ±5%
Add 10nsec delay for each 50pf increase in load up to maximum of 200pf for data bus and 100pf for address & control lines.

E. Although static by design, testing guarantees $t_{w(\Phi H)}$ of 200 µsec maximum.

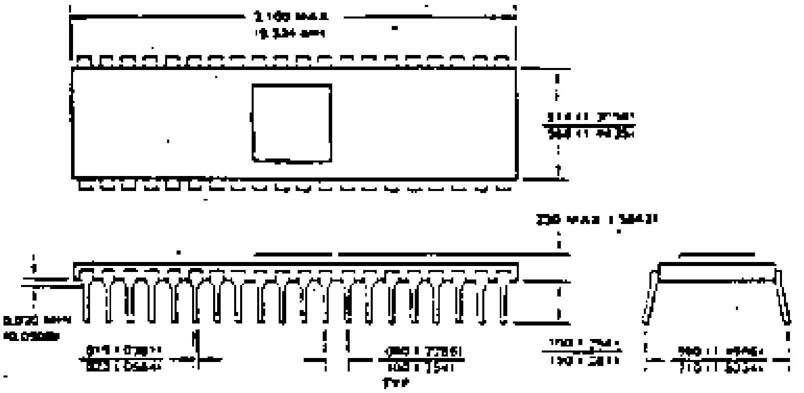


Load circuit for Output

Package Configuration



Package Outline



*Dimensions for metric system are in parentheses

EASTERN

Hallmark Electronics
4739 Commercial Drive
Huntsville, AL 35805
TEL 205 837 8700
TWX 810 726 2187

Hallmark Electronics
1302 West McNab Road
Fort Lauderdale, FL 33309
TEL 305 971 9280
TWX 510 956 9720

[illegible]k Electronics
[illegible] Lake Ellenor Drive
Orlando, FL 32809
TEL 305 855 4020
TWX 810 850 0183

Hallmark Electronics
3335 Amberton Drive
Baltimore MD 21227
TEL 301 796 9300
TWX 710 862 1942

Hallmark Electronics
1208 Front Street
Building K
Raleigh, NC 27609
TEL 919 832 4465
TWX 510 928 1831

Hallmark Electronics
Pike Industrial Park
Huntington Valley, PA 19006
TEL 215 355 7300
TWX 510 667 1750

Summit
916 Main Street
Buffalo, NY 14202
TEL 716 884 3450

Wilshire Electronics
2554 State Street
Hamden, CT 06517
TEL 203 281 1166
TWX 800 922 1734

Wilshire Electronics
1855 New Highway
Farmingdale, LI, NY 11735
TEL 516 293 5775
TWX 212 895 8707

Wilshire Electronics
One Wilshire Road
Burlington, MA 01803
TEL 617 272 8200
TWX 710 332 6359

Wilshire Electronics
1111 Paulison Avenue
Clifton, NJ 07015
TEL 201 340 1900
TWX 710 989 7052

MIDWESTERN

Hallmark Electronics
180 Crossen Avenue
Elk Grove Village, IL 60076
TEL 312 437 8800
TWX 910 223 3645

Hallmark Electronics
11870 West 91st Street
Congleton Industrial Park
Shawnee Mission, KS 66214
TEL 913 888 4747
TWX 910 749 6620

Hallmark Electronics
9201 Penn Avenue South
Suite 10
Bloomington, MN 55435
TEL 612 884 9056
TWX 910 576 3187

Hallmark Electronics
13789 Rider Trail
Earth City, MO 63045
TEL 314 291 5350
TWX 910 760 0671

Hallmark Electronics
6969 Worthington Galena Road
Worthington, OH 43085
TEL 614 846 1882

Hallmark Electronics
4846 S. 83rd E. Avenue
Tulsa, OK 74145
TEL 918 835 8458
TWX 910 845 2290

Hallmark Electronics
3100—A Industrial Terrace
Austin, TX 78758
TEL 512 837 2841
TWX 910 874 2031

Hallmark Electronics
9333 Forest Lane
Dallas, TX 75231
TEL 214 234 7300
TWX 910 867 4721

Hallmark Electronics
8000 Westglen
Houston, TX 77063
TEL 713 781 6100
TWX 910 881 2711

Hallmark Electronics
237 South Curtis
West Allis, WI 53214
TEL 414 476 1270
TWX 910 262 3186

RM Electronics
4860 South Division
Kentwood, MI 49508
TEL 616 531 9300
TWX 810 273 8779

RM Electronics
47 Chestnut Lane
Westmont, Illinois 60559
TEL 312 323 9670

MOUNTAIN

Century Electronics
121 Elizabeth, NE
Albuquerque, NM 87123
TEL 505 292 2700
TWX 910 989 0625

WESTERN

Intermark Electronics
1802 E. Carnegie Avenue
Santa Ana, CA 92705
TEL 714 540 1322
TWX 910 595 1583

Intermark Electronics
4040 Sorrento Valley Blvd.
San Diego, CA 92121
TEL 714 279 5200
714 453 9005
TWX 910 335 1515

Intermark Electronics
1020 Stewart Drive
Sunnyvale, CA 94086
TEL 408 738 1111
TWX 910 339 9312

R.V. Weatherford Co.
6921 San Fernando Road
Glendale, CA 91201
TEL 213 849 3451
TWX 910 498 2223

R.V. Weatherford Co
1550 Babbitt Avenue
Anaheim, CA 92805
TEL 714 634 9600
TWX 910 593 1334

R.V. Weatherford Co.
3095 East Third Street
Pomona, CA 91765
TEL 714 623 1261
TWX 910 581 3811

R.V. Weatherford Co.
3240 Hillview Avenue
Stanford Industrial Park
Palo Alto, CA 94304
TEL 415 493 5373

R.V. Weatherford Co.
3311 W. Earll Drive
Phoenix, AZ 85017
TEL 602 272 7144
TWX 910 951 0636

Sterling Electronics
5608 6th Avenue South
Seattle, WA 98108
TEL 206 762 9100
TLX 32-9652

Western Microtechnology
977 Benecia Avenue
Sunnyvale, CA 94086
TEL 408 737 1660

CANADA

Future Electronics
5647 Ferrier Street
Montreal, Quebec,
CANADA H4P 2K5
TEL 514 735 5775
TWX 610 421 3251

EASTERN REGION

Zilog, Inc.
76 Treble Cove Road
No. Billerica, MA 01862
TEL 617 667 2179
TWX 710 347 6660

MIDATLANTIC REGION

Zilog, Inc.
P.O. Box 92
Bergenfield, NJ 07621
TEL 201 385 9158
TWX 710 991 9771

MIDWESTERN REGION

Zilog, Inc.
1701 Woodfield Place
Suite 417
Schaumburg, IL 60195
TEL 312 885 8080
TWX 910 291 1064

SOUTHWESTERN REGION

Zilog, Inc.
17982 Sky Park Circle
Suite C
Irvine, CA 92714
TEL 714 549 2891
TWX 910 595 2803

EUROPEAN HDQTS

Zilog (UK) Ltd.
Nicholson House
Maidenhead, Berkshire
England
TEL (0628) 36131/2/3
TWX 848 609

10460 Bubb Road, Cupertino, California 95014   Telephone: (408) 446-4666   TWX: 910-338-7621

03-0027-02

MSE-B-64

Printed in U.S.A.

APPENDIX C

16-BIT MICROPROCESSOR FAMILY

THIS PAGE INTENTIONALLY LEFT BLANK.

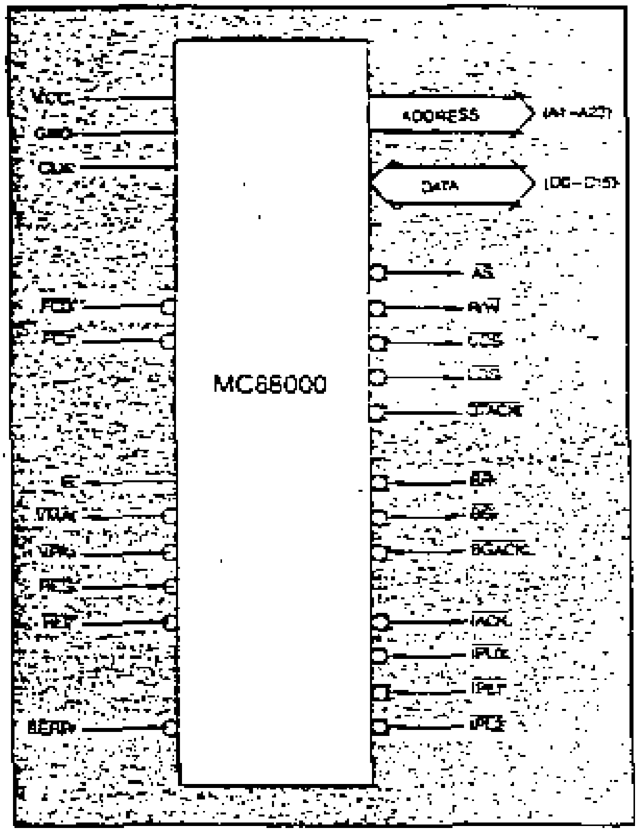
16-BIT MICROPROCESSOR FAMILY EXAMPLES

- MC - 68000 FAMILY
- Z - 8000 CPU
- Z - 8010 MMU
- Z - 8034 UPC
- Z - 8036 CIO
- Z - 8030 SCC

THIS PAGE INTENTIONALLY LEFT BLANK.

INTRODUCING THE MC68000 . . .

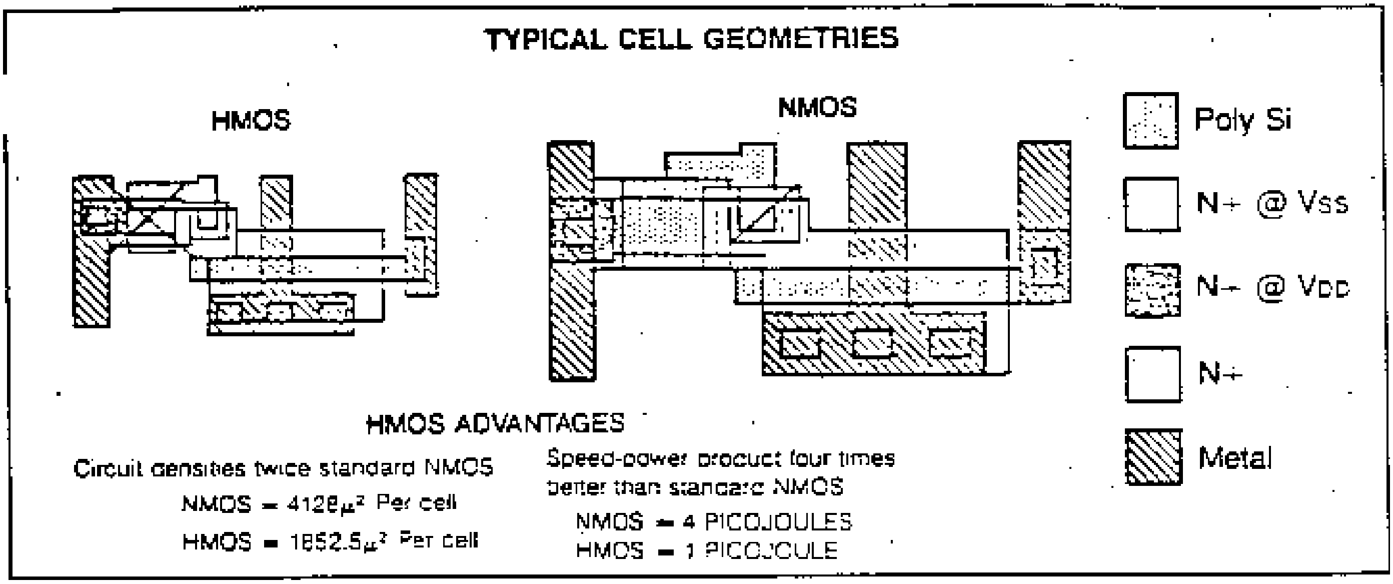
MOTOROLA'S ADVANCED COMPUTER SYSTEM ON SILICON



The MC68,000 microprocessor is housed in a 64-pin package that allows the use of separate (non-multiplexed) address and data buses. This large package provides optimum flexibility while at the same time maximizing bus through-put.

PIN IDENTIFICATION & DEFINITIONS

A1–A23	Address Leads	23-bit address bus: capable of addressing 16,777,216 bytes in conjunction with UDS and LDS.
D0–D15	Data Leads	16-bit data bus: transfers 8 or 16 bits of information.
AS	Address Strobe	Indicates valid address & provides a bus lock for indivisible operations.
R/W	Read/Write	Defines bus operation as Read or Write and controls external bus buffers.
UDS, LDS	Data Strobes	Identifies the byte(s) to be operated on according to R/W and AS.
DTACK	Data Transfer Acknowledge	Allows the bus cycle to synchronize with slow devices or memories.
BR	Bus Request	Input to the Processor from a device requesting the bus.
BG	Bus Grant	Output from the processor granting bus arbitration.
BGACK	Bus Grant Acknowledge	Confirmation signal from BG indicating a valid selection from the arbitration process.
IACK	Interrupt Acknowledge	Identifies that the bus is performing an interrupt service cycle
IPL0, IPL1, IPL2	Interrupt Priority Level	Provides the priority level of the interrupting function to the processor.
FC0, FC1	Function Code	Provides external devices with information about the current bus cycle
CLK	Clock	Master TTL input clock to the processor.
RES	Reset	Provides reset (initialization) signal to the processor and peripheral devices.
HLT	Halt	Stops the processor and allows single stepping.
BERR	Bus Error	Provides termination of a bus cycle if no response or an invalid response is received.
E	Enable	Enable clock for M6800 systems. Identifies addressed area as a 6800 compatible area.
VPA	Valid Peripheral Address	
VMA	Valid Memory Address	Indicates to 6800 family devices that a valid address is on the bus.
V_{cc}	+5 Volts	—
GND	Ground (2 pins)	—



HMOS ADVANTAGES

Circuit densities twice standard NMOS
NMOS = 4128μ^2 Per cell
HMOS = 1852.5μ^2 Per cell

Speed-power product four times better than standard NMOS
NMOS = 4 PICOJOULES
HMOS = 1 PICOJOULE

Figure 1: Comparison of HMOS and NMOS Technologies
HMOS Technology used for the MC68000 results in significant improvements to Circuit Densities and Speed-Power Products

Advances in semiconductor technology have provided the capability to put on a single silicon chip, a microprocessor at least an order of magnitude higher in performance and circuit complexity than has been previously available. The MC68000 is the first of a family of such VLSI microprocessors from Motorola. It combines state-of-the-art technology and advanced circuit design techniques with computer sciences to achieve an architecturally advanced 16-bit microprocessor containing over 68000 active devices on a silicon chip. This high density of active elements coupled with an order of magnitude increase in performance over the original MC6800 is the direct result of significant advances in semiconductor technology. Advances such as dry PLASMA etching, projection printing, and HMOS (High density short channel MOS) circuit design techniques (Figure 1) have provided a sound technological base that has allowed Motorola's system engineers, computer scientists and marketing engineers a large degree of innovative freedom. The goals of applying this innovative freedom to microprocessors are to make the microprocessor easy to use, more reliable and more flexible for applications, while maximizing performance.

The resources available to the MC68000 user consist of the following:

- 32-bit data and address registers
- 16 mega-byte direct addressing range
- 61 powerful instruction types
- operations on six main data types
- memory mapped I/O
- 14 addressing modes

Particular emphasis has been given to the architecture to make it orthogonal (regular) with respect to the registers, instructions (including all addressing modes), and data types. Orthogonality makes the architecture easy to learn and program, and, in the process, reduces both the time required to write programs and the space required to store programs. The net result is a great reduction in the cost and risk of developing software.

High systems throughput (up to an aggregate of two million instruction and data word transfers per second) is achieved even with readily available standard product memories with comparatively slow access times. The design flexibility of the data bus allows the mixing of slow and fast memories or peripherals with the processor, automatically optimizing the transfer rate on every access to keep the system operating at peak efficiency.

The hardware design of the CPU was heavily influenced by advances made in software technology. High level language compilers as well as code produced from high level languages must run efficiently on the new generation 16-bit and 32-bit microprocessors. The MC68000 supports high level languages with its consistent architecture, multiple registers and stacks, large addressing range and high level language oriented instructions (LINK, UNLINK, CHK, etc.). Also, operating systems for controlling the software operating environment of the MC68000 MPU are supported by privileged instructions, memory management, a powerful vectored multi-level interrupt and trap structure, and specific instructions (EXG, LDM, STM, TRAP, etc.).

The processor also provides both hardware and software interlocks for multiprocessor systems. The CPU chip contains bus arbitration logic for a shared bus and shared memory environment (shared with other MC68000 processors, DMA devices, etc.). Multiprocessor systems are also supported with software instructions (TEST and SET, TEST and RESET, etc.). The MC68000 offers the maximum flexibility for microprocessor based multiprocessor systems.

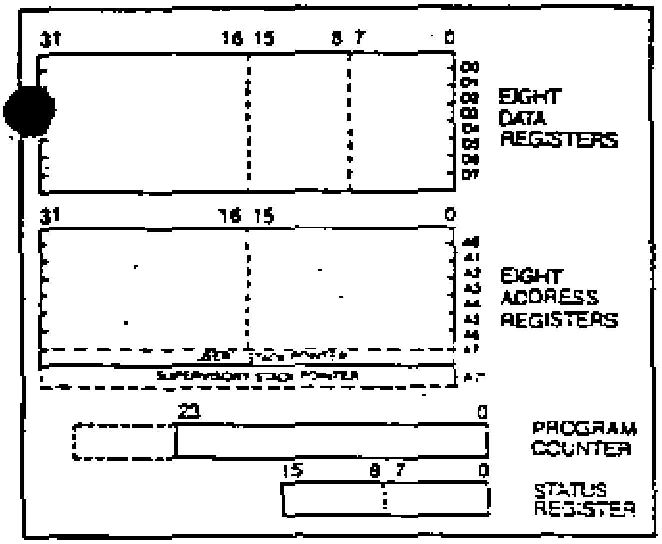


Figure 2: MC68000 Programming Model

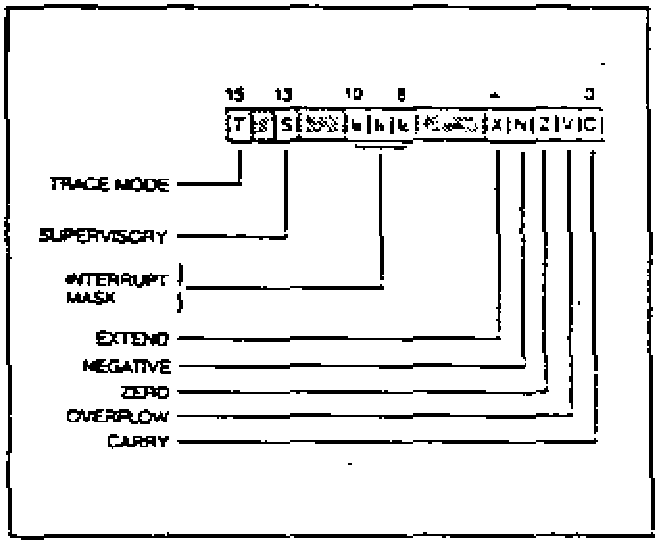


Figure 3: MC68000 Status Register

THE MC68000 CPU

Advanced architecture processors must not only offer efficient solutions to large complex problems but must be able to handle the small, simple problems with proportional efficiency. The CPU has been designed to offer the maximum in performance and versatility to solve simple and complex problems efficiently.

The MC68000 offers sixteen 32-bit registers in addition to the 24-bit program counter and 16-bit status register (Figure 2). The first eight registers (D0–D7) are used as data registers for byte (8-bit), word (16-bit) and long word (32-bit) operations. The second set of eight registers (A0–A7) may be used as software Stack Pointers and Base Address Registers. In addition, the second set of eight registers may be used for word and long word data operations. All of the sixteen registers may be used as Index Registers.

The 24-bit Program Counter provides a memory addressing range of more than 16 mega-bytes (actually 16,777,216 bytes). This large range of addressing capability, coupled with a Memory Management Unit, allows large, modular programs to be developed and operated without resorting to cumbersome and time consuming software bookkeeping and paging techniques.

The Status Register (Figure 3) contains the Interrupt Level Mask (8 levels available) as well as the Condition Code: Overflow (V), Zero (Z), Negative (N), Carry (C), and Extend (X). Additional status bits indicate that the processor is in a TRACE (T) mode or in a SUPERVISORY (S) state. Ample space remains in the Status Register for future extensions of the M68000 family.

Six basic data types are supported. These data types are:

- Bits
- BCD digits
- ASCII characters
- Bytes (8-bits)
- Words (16-bits)
- Long words (32-bits)

In addition operations on other data types such as memory addresses, status word data, etc. are provided for in the instruction set.

DEFINITIONS

EA = Effective Address
Ax = Address Register
Dx = Data Register
Rx = Address or Data Register Used as Index Register
SR = Status Register
PC = Program Counter
D_8 = Eight-Bit Offset
D_{16} = Sixteen-Bit Offset
N = 1 for Byte, 2 for Word and 4 for Long Word
( ) = Contents of
← = Replaces

TABLE I: MC68000 DATA ADDRESSING MODES

REGISTER DIRECT ADDRESSING		REGISTER INDIRECT ADDRESSING	
Data Register Direct	EA = D_x	Register Indirect	EA = (Ax)
Address Register Direct	EA = A_x	Post-increment Register Indirect	EA = (Ax), Ax ← Ax + N
Status Register Direct	EA = SR	Pre-decrement Register Indirect	Ax ← Ax − N, EA = (Ax)
ABSOLUTE DATA ADDRESSING		Register Indirect with Offset	EA = (Ax) + D_{16}
A. Absolute Short	EA = (Next Word)	Indexed Register Indirect with Offset	EA = (Ax) + (Rx) + D_8
B. Absolute Long	EA = (Next two Words)		
PROGRAM COUNTER RELATIVE ADDRESSING		IMMEDIATE DATA ADDRESSING	
Relative with Offset	EA = (PC) + D_{16}	Immediate	DATA = Next Word(s)
Relative with Index & Offset	EA = (PC) + (Rx) + D_8	Quick Immediate	INHERENT DATA

The 14 flexible addressing modes, shown in Table I, include five basic types:

- Register Direct • Immediate
- Register Indirect
- Absolute • Program Counter Relative

Included in the addressing modes is the capability to do Post-incrementing, Pre-decrementing, Offsetting and Indexing.

THE INSTRUCTION SET

The MC68000 instruction set is rich and full as evidenced by the 61 distinct types shown in Table II. Special emphasis during the design has been given to the instruction set's support of structured high level languages that facilitate ease of programming. Each instruction, with few exceptions, operates on bytes, words, and long words and most instructions can use any of the 14 addressing modes. Combining instruction types, data types, and addressing modes, over 1000 useful instructions are provided. These instructions include signed and unsigned multiply and divide, "quick" arithmetic operations, BCD arithmetic and extended operations (through traps). The processor offers the most comprehensive and flexible instruction set of any microprocessor of any class, available today. Additionally, it's highly orthogonal, proprietary microcoded structure provides a sound stable base for the future.

REDUCED SOFTWARE COST AND RISK

Advances in VLSI semiconductor technology have resulted in a significant reduction in the cost of computer hardware in recent years. The MC68000 microprocessor, for example, provides in a single integrated circuit package computing power that just a decade ago would have been three or four orders of magnitude more expensive. Software costs during this same period of time have, as a percentage of total system cost, increased significantly. This has been due primarily to inflation and the labor intensive nature of programming. Without significant architectural advances in computers, this trend can do nothing but continue. One of Motorola's major goals in developing this new microprocessor has been to reduce the costs of software. Many innovative features have been incorporated to make programming easier, faster and more reliable.

An Orthogonal 16-BIT MPU — The highly orthogonal or regular structure of the MC68000 microprocessor greatly simplifies the effort required to write programs in Assembly Language as well as in High Level Languages. Operations on integer data in registers and memory are independent of the data itself. Separate special instructions that operate on byte (8-bit), word (16-bit) and long-word (32-bit) integers are not necessary. The programmer merely has to remember one mnemonic for each type of operation and then specify data size, source addressing mode and destination addressing mode. This has helped keep the total number of instruction mnemonics for the M68000 to an easily remembered, yet complete, 61 types, eleven fewer than on Motorola's MC6800.

The dual operand nature of many of the instructions significantly increases the flexibility and power of this new Motorola microprocessor. Consistency again is maintained since *all* data registers and memory locations may be either a source or destination for most operations on integer data.

TABLE II: MC68000 INSTRUCTION SET SUMMARY

MNEMONIC	DESCRIPTION
ABCD	Add Decimal with Extend
ADD	Add
ADDX	Add with Extend
AND	Logical And
ASL	Arithmetic Shift Left
ASR	Arithmetic Shift Right
BCC	Branch Conditionally
BCHG	Bit Test and Change
BCLR	Bit Test and Clear
BRA	Branch Always
BSET	Bit Test and Set
BSR	Branch to Subroutine
BTST	Bit Test
CHK	Check Register Against Bounds
CLR	Clear Operand
CMP	Arithmetic Compare
DCNT	Decrement and Branch Non-Zero
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive Or
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LDM	Load Multiple Registers
LDQ	Load Register Quick
LEA	Load Effective Address
LINK	Link Stack
LSL	Logical Shift Left
LSR	Logical Shift Right
MOVE	Move
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Two's Complement
NEGX	Two's Complement with Extend
NOP	No Operation
NOT	One's Complement
OR	Logical Or
PACK	Pack ASCII to BCD
PEA	Push Effective Address
RESET	Reset External Devices
ROTL	Rotate Left without Extend
ROTR	Rotate Right without Extend
ROTXL	Rotate Left with Extend
ROTXR	Rotate Right with Extend
RTR	Return and Restore
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
SCC	Set Conditional
STM	Store Multiple Registers
STOP	Stop
SUB	Subtract
SUBX	Subtract with Extend
SWAP	Swap Data Register Halves
TAS	Test and Set Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test
UNLK	Unlink Stack
UNPK	Unpack BCD to ASCII

The addressing modes have been kept simple without sacrificing efficiency. All fourteen addressing modes operate consistently and are independent of the instruction operation itself. Additionally, all address registers may be used for the Direct, Register Indirect and Indexed addressing modes. (Immediate, Program Counter Relative and Absolute addressing by definition do not use address registers). For increased flexibility, any data register — as well as any address register — may be used as an Index Register. Address register consistency is maintained for stacking operations since any of the eight address registers may be utilized as User Program Stack pointers with the Register Indirect Post-increment/Predecrement addressing modes. Register A7, however, is a special register that, in addition to its normal addressing capability, functions as the System Stack Pointer when stacking the Program Counter and Status Register for subroutine calls, traps and interrupts: while in the supervisory mode.

Structured Modular Programming — The art of programming microprocessors has evolved rapidly in the past few years. Numerous advanced techniques have been developed to allow easier, more consistent and reliable generation of software. In general, these techniques require that the programmer be more disciplined in observing a defined programming structure such as modular programming. Modular programming allows a required function or process to be broken down in short modules or subroutines that are concisely defined and easily programmed and tested. Such a technique is greatly simplified by the availability of advanced macro assemblers and block structured High Level Languages such as PASCAL. Such concepts are virtually useless, however, unless parameters are easily transferred between and within software modules that operate on a reentrant and recursive basis. (To be reentrant a routine must be usable by interrupt and non-interrupt driven programs without the loss of data. A recursive routine is one that may call or use itself). The MC68000 microprocessor provides the necessary architectural features to allow efficient reentrant modular programming. The "LINK" and "UNLINK" instructions reduce subroutine call overhead in two complementary instructions by allowing the manipulation of linked lists of data areas on the stack. The "STM" (Store Multiple Registers) and "LDM" (Load Multiple Registers) instructions also reduce subroutine call programming overhead. These allow the loading or storing, via an effective address, multiple registers that are specified by the programmer. Sixteen software trap vectors are provided with the "TRAP" instruction and are useful in operating system call routines or user generated "macro routines." Other instructions that support modern structured programming techniques are PEA (Push Effective Address), LEA (Load Effective Address), RTR (Return to Restore) as well as the normal JSR, BSR and RTS.

Of course, the powerful vectored priority interrupt structure of the microprocessor allows straightforward generation of reentrant modular Input/Output routines. Eight maskable levels of priority with 192 vector locations provide maximum flexibility for I/O control. (A total of 256 vector locations are available for interrupts, hardware traps, and software traps.)

Improved Software Testability — One of the major tasks the system programmer encounters when writing software for microcomputers is the detection and correction of errors, or "debugging." The time taken to "debug" software nearly always exceeds the time it takes to write the software. In practice, the old 20/80 rule often applies: "The last 20% of the job requires 80% of the effort." The microprocessor incorporates several features that reduce the chance for errors. These features, such as Orthogonality and the Structured Modular Programming capability, have already been discussed.

Of major importance to the systems programmer are features that have been incorporated specifically to detect the occurrence of programming errors or "bugs." Several hardware traps, provided to indicate abnormal internal conditions of the MC68000 processor, detect the following error conditions:

Word access with an odd address
Illegal instructions
Unimplemented instructions
Illegal addressing mode
Illegal Memory access (bus error)
Overflow on divide (divide by zero)
Overflow condition code (separate instruction TRAPV)

Additionally, the sixteen software TRAP instructions may be utilized by the programmer to provide applications oriented error detection or correction routines.

An additional error detection tool is the CHK (Check Register Against Bounds) instruction used for array bound checking by verifying that 0 ≤ (REG) < LIMIT. A trap occurs if the register contents are negative or greater than the limit.

Finally, the MC68000 includes a facility that allows instruction-by-instruction tracing of a program being debugged. This TRACE MODE results in a trap being made to a tracing routine after each instruction execution. The TRACE MODE is available to the programmer when the microprocessor is in the SUPERVISORY state as well as the USER state, but may only be entered while in the supervisory state. The SUPERVISORY/USER states provide an additional degree of error protection for the microprocessor by providing memory protection of selected areas of memory when an external memory management device is used.

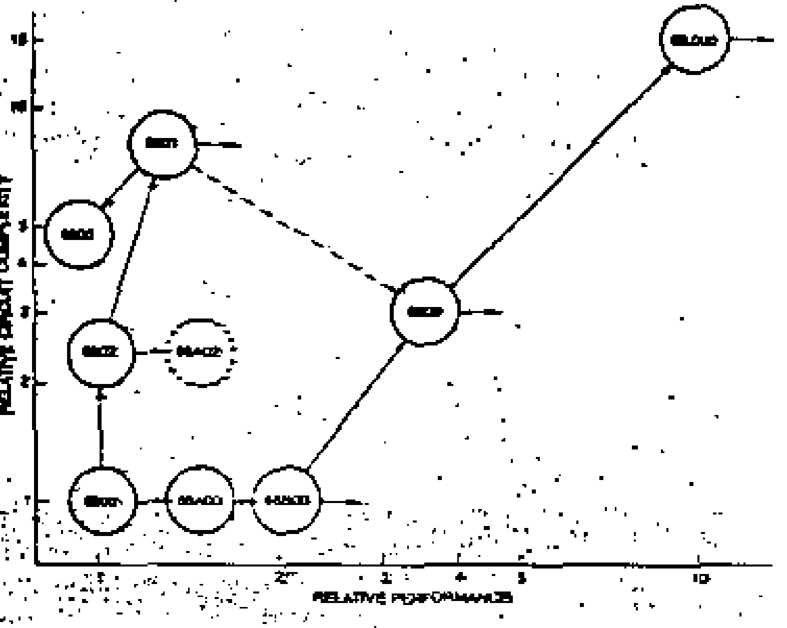


Figure 4: Motorola's Microprocessor Evolution

FUTURE FLEXIBILITY

Microprocessor VLSI circuit technology is advancing at an ever increasing rate. For example, the Motorola MC6800 — originally introduced in 1974 — has evolved into a number of more advanced products. This evolution has been along two paths: increased functionality, with the MC6802 and MC6801 computers, and increased performance with MC68A00, MC68B00 and MC6809 microprocessors. (Figure 4). The sound, well planned, architectural base provided by the original MC6800 made it possible to develop these improved products while taking full advantage of the major speed and density enhancements to NMOS VLSI. This was accomplished while maintaining an unprecedented degree of compatibility and consistency with the original MC6800 MPU.

Similarly, a major consideration in the development of the MC68000 microprocessor has been to provide a good, solid, but flexible, base for future extensibility. Several architectural concepts have been incorporated that will allow this advanced product to be enhanced as semiconductor technological advances are made. For example, the highly orthogonal structure of the CPU allows operations on 8-bit, 16-bit and 32-bit integers without the need for concatenation of registers or multiplexing of internal data buses. This regular structure of the CPU lends itself to a more consistent, reliable design that can be easily expanded.

The MC68000 incorporates a proprietary multi-level micro-programmed structure that allows significant versatility in the implementation of instructions. In fact, more than one-eighth of the instruction op-code map has been set aside specifically for implementation of future instructions. In the interim, user implementation of instructions not currently in the instruction set is possible through the use of the TRAP instruction, as well as the hardware trap structure.

MEMORY MANAGEMENT OF LARGE ADDRESSING SPACE

The ever-decreasing costs of semiconductor memories in combination with the use of high level languages and sophisticated disc operating systems allow Motorola's new generation of high performance microprocessors to be used in complex, memory intensive applications. In order to meet the needs of such applications, the MC68000 is capable of directly addressing more than 16 mega-bytes of memory. This large address space is directly accessed and managed very efficiently on a word or byte basis since operand size is specified by the instruction. The use of Upper Data Strobe (UDS) and Lower Data Strobe (LDS) signals allows easy access to high order bytes, low order bytes, or words.

Several additional useful features are provided that allow the programmer to efficiently manage memory usage. Powerful memory addressing modes such as Register Indirect, Indexed, Short and Long Absolute, and Program Counter Relative allow well-ordered access to specific memory locations. These addressing modes allow easy address calculations (Register Indirect and Indexed), direct access to memory location (Short and Long Absolute) and position independent or relocatable coding (Program Counter Relative). Of course, the Pre-decrement/Post-increment Register Indirect Addressing modes also allow efficient management of data in memory by permitting the programmer to generate as many as eight concurrent stacks or queues. Another feature allows the programmer to manage the use of nory is the CHK (Check Register Against Bounds) Instruction. This instruction permits the software implementation of a basic memory protection/management structure.

Still another significant feature provided in the MC68000 microprocessor is the distinction between a USER and a SUPERVISOR mode. The SUPERVISOR mode permits certain protected operations within the processor system. Of particular interest is that an external Memory Management Controller may be used when the processor is in the USER mode to manage the large address space for the programmer. The controller's memory management operations are transparent to the programmer when in the USER mode and can be changed or updated only in the SUPERVISOR mode. The Memory Management Controller provides both management of a variable number of variable size segments (Memory Segmentation) and dynamic management of multi-task memory relocation and protection. The Memory Management Controller regulates access to storage segments that are dedicated to read only data, read/write data, program code and protected data/code.

REDUCED CODE DENSITY AND IMPROVED SPEED

With the advent of low cost, very high density VLSI RAMS and ROMS, it might incorrectly be assumed that the number of bytes of code needed to execute a given program is no longer important. Code density, however, is very critical, since microprocessor speed is highly dependent upon the number of executed instruction words. During the early development of Motorola's MC68000 microprocessor, extensive studies were made of the use of instructions and sequences of instructions in many microprocessor applications. These studies identified not only statically frequent instructions but also dynamically frequent instructions. (The dynamic frequency of instructions is a measure of how often an instruction is executed while static frequency is a measure of how often it occurs in a program listing or is encountered by an assembler). The major contributer to the increased efficiency, as a result of the studies, is the highly regular or orthogonal structure of the architecture. The consistency of the architecture, instruction set, and addressing modes significantly reduces the number of instructions needed to accomplish a given task. Additionally, many instructions have been included to specifically improve code density and speed. For example, single word Add and Subtract instructions using Quick Immediate addressing allow fast, small value arithmetic operations on data registers and memory. A Load Quick Immediate (LDQ) provides the ability to load a small (8-bit) signed word into any register in a single word operation. In order to improve the speed of loop operations, a single word instruction for Decrement Count by One and Branch if non-zero (DCNT) is included. Of course, the TRAP, Store Multiple Registers (STM), Load Multiple Registers (LDM), Link Stack (LINK), Unlink Stack (UNLK) and Check Limit (CHK) instructions significantly reduce code requirements for subroutines, operating system calls and stacking operations.

Other instructions that help reduce coding requirements and improve performance of arithmetic operations are Signed and Unsigned Multiply (MULS and MULU), Signed and Unsigned Divide (DIVS and DIVU), BCD Arithmetic (ABCD, SBCD, PACK and UNPK) as well as the standard binary integer operations. In order to improve the efficiency of moving or transferring data, a powerful MOVE data instruction has been incorporated that allows the transfer of bytes, words and long words and operates in all data addressing modes. Thus: register-to-register, register-to-memory, memory-to-register and memory-to-memory transfers are permitted.

In addition to the powerful instructions that provide a substantial improvement in processor through-put, numerous architectural features significantly reduce the execution times for all instructions. The separate (non-multiplexed) address and data buses, instruction pre-fetch pipeline and 32-bit internal registers are major contributors to the processor's unequaled performance. As an example of the performance capability of the MC68000 Table III and the accompanying graphs in figures 5 and 6 summarize the execution times for a number of common instructions. For comparison purposes, similar information is provided for Zilog's Z-8000 microprocessor. It is interesting to note that the MC68000 has significantly faster execution times.

TABLE III — EXECUTION TIMES FOR *MOVB R, SRC* INSTRUCTION FOR VARIOUS ADDRESSING MODES

Source Addressing	Motorola MC68000	Zilog Z-8000
Register	0.5us	0.75us
Indirect Register	1.0	1.75
Absolute Addressing (Direct)	1.5	2.25
Indexed Addressing	1.5	2.50
Immediate	1.0	1.00

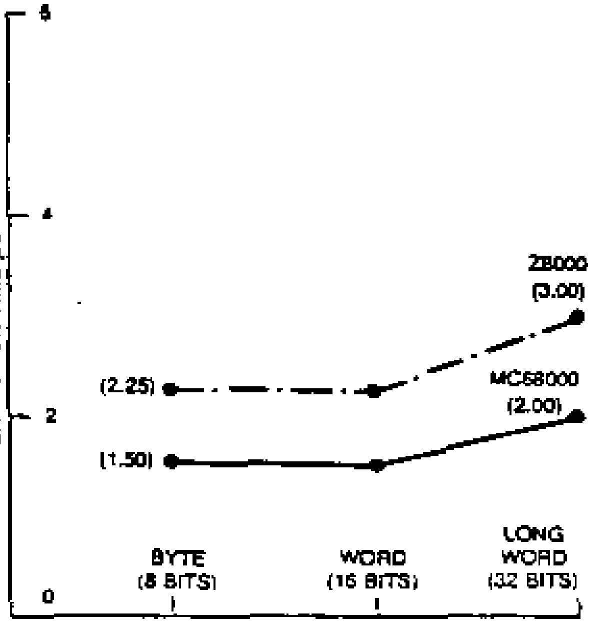


FIGURE 5: Execution Time for the *Add Data Element to a register from a short Absolute Address* Instruction.

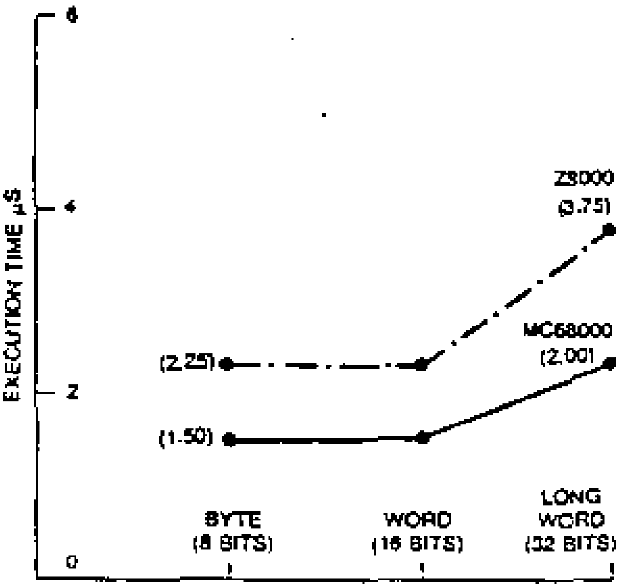


FIGURE 6: Execution Time for the *move a data element from memory to a register from short Absolute Address* Instruction.

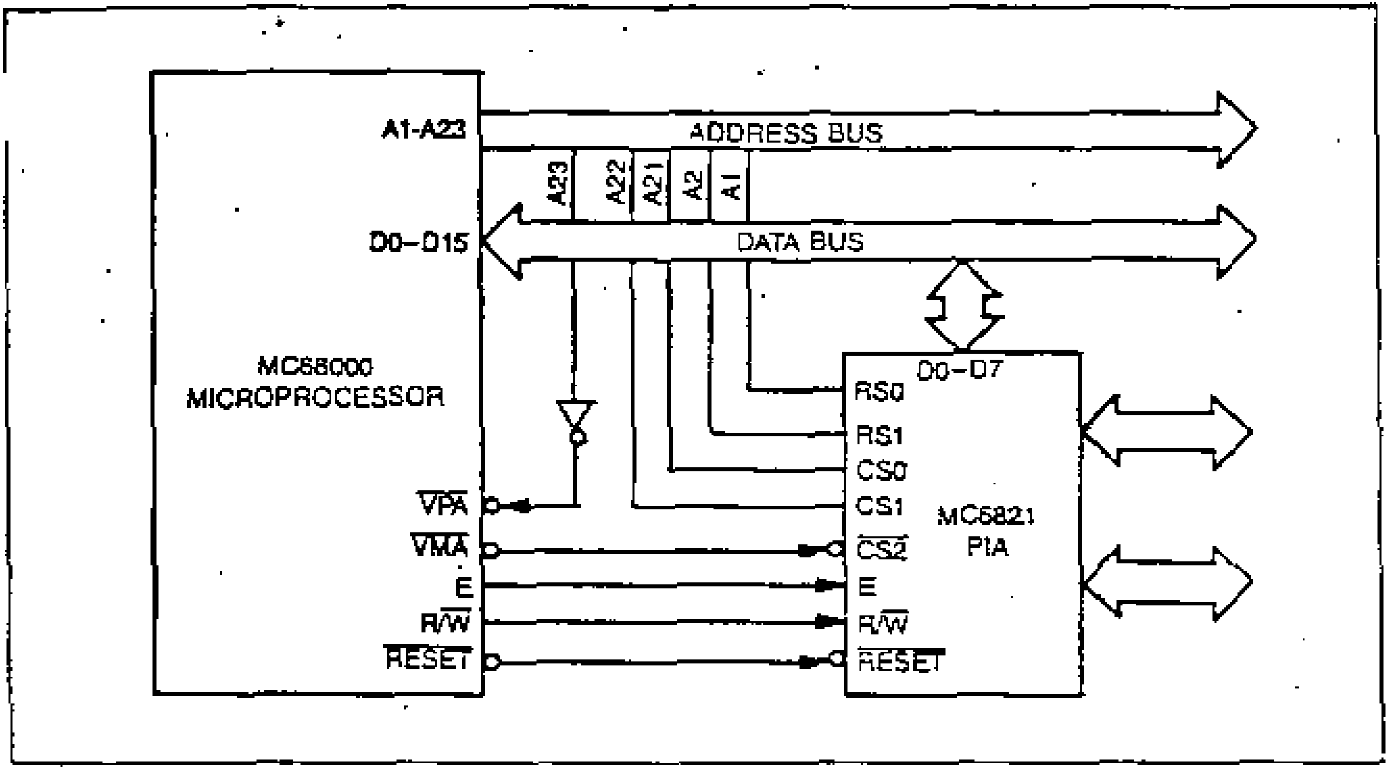


Figure 7: Example of MC68000 Interface Connections for MC6821 Peripheral Interface Adapter

SOFTWARE SUPPORT AND MC6800 COMPATIBILITY

The system designers and programmers using the MC68000 in an application have available a complete, compatible system of hardware and software. The microprocessor is supported by a full range of software development tools including disc operating systems, debug aids, assemblers, and high level languages. In addition, a translator will allow the present M6800 Family user to convert existing programs to run on the MC68000 with a minimum of programmer intervention.

The careful planning of this new microprocessor provides a superset of the MC6800 instruction set enhanced by the addition of more and larger registers, powerful orthogonal structure and many flexible addressing modes. This allows efficient translation of existing MC6800 programs, which can then be further optimized by taking full advantage of the versatile and powerful features of the MC68000.

This careful planning of similarities between the MC68000 and the MC6800 does not stop at software compatibility (by translation) but also extends to peripheral controller interfacing. Motorola's extensive line of intelligent M6800 family peripherals (including the MC6854 Advanced Data Link Controller and the MC68488 General Purpose Interface Adapter) can be directly and easily interfaced to the MC68000. Three signal lines: Enable (E), Valid Memory Address (VMA), and Valid Peripheral Address (VPA) are provided to simplify the interface to Motorola's standard MC6800 peripherals as shown in Figure 7. Interface to the new MC6801E (Single Chip Programmable Controller) is also possible, allowing user implementation of specialized input/output functions. In addition, the MC68000 is supported by unique peripheral controllers expected of an advanced architecture microprocessor, including a DMA Controller and a Memory Management Unit.

The MC68000 is not just a component. By a unique blend of VLSI design, software engineering and careful planning, the MC68000 it is Motorola's Advanced Computer System on Silicon.

MOTOROLA Semiconductor Products Inc.
P.O. BOX 20912 • PHOENIX, ARIZONA 85036 • A SUBSIDIARY OF MOTOROLA INC.

Z8001/Z8002 CPU Central Processing Unit

77

Zilog

Product Brief

August 1979

Features

- Regular, easy-to-use architecture.
- Instruction set more powerful than many minicomputers.
- Directly addresses 8M bytes.
- Eight user-selectable addressing modes.
- Seven data types that range from bits to 32-bit long words and word strings.
- System and normal operating modes; separate code, data and stack spaces.
- Sophisticated interrupt structure.
- Resource-sharing capabilities for multiprocessing systems.
- Multi-programming and compiler support.
- Memory management and protection provided by Z8010 Memory Management Unit.
- 32-bit operations, including signed multiply and divide.
- Z-Bus compatible.

Description

The Z8000 is an advanced high-end 16-bit microprocessor that spans a wide variety of applications ranging from simple stand-alone computers to complex parallel-processing systems. Essentially a monolithic minicomputer central processing unit, the Z8000 CPU is characterized by an instruction set more powerful than many minicomputers; resources abundant in registers, data types, addressing modes and addressing range; and a regular architecture that enhances throughput by avoiding critical bottlenecks such as implied or dedicated registers.

CPU resources include sixteen 16-bit general-purpose registers, seven data types that range from bits to 32-bit long words and word strings, and eight user-selectable addressing modes. The 110 distinct instruction types can be combined with the various data types and addressing modes to form a powerful set of 414 instructions. Moreover, the instruction set exhibits a high degree of regularity: most instructions can use any of the five main addressing modes and can operate on byte, word and long-word data types.

The CPU can operate in either system or normal modes. The distinction between these two modes permits privileged operations, thereby improving operating system organization and implementation. Multiprogramming is supported by the "atomic" Test and Set instruction; multiprocessing by a combination of instruction and hardware features; and compilers by multiple stacks, special instructions and addressing modes.

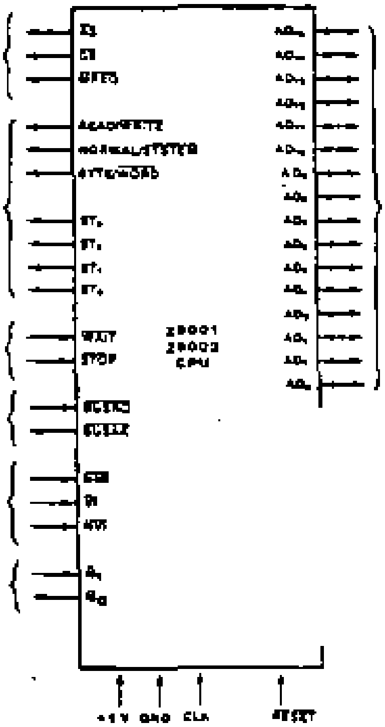


Figure 1. Pin Functions

MSE C-13

Description (Continued)

The Z8000 CPU is offered in two versions: the Z8001 48-pin segmented CPU and the Z8002 40-pin non-segmented CPU. The main difference between the two is in addressing range. The Z8001 can directly address 8M bytes of memory; the Z8002 directly addresses 64K bytes. The two operating modes—system and normal—and the distinction between code, data and stack spaces within each mode allows memory extension up to 48M bytes for the Z8001 and 384K bytes for the Z8002.

To meet the requirements of complex, memory-intensive applications, a companion memory-management device is offered for the Z8001. The Z8010 Memory Management Unit manages the large address space by providing features such as segment relocation and memory protection. The Z8001 can be used with or without the Z8010. If used by itself, the Z8001 still provides an 8M byte direct addressing range, extendable to 48M bytes.

Register Organization. The Z8000 CPU is a register-oriented machine that has sixteen 16-bit general-purpose registers. The Z8002 CPU has one stack pointer register, and the Z8001 has two.

Stacks. The Z8001 and Z8002 can use stacks located anywhere in memory. Two implied stack pointers are available: the system stack pointer and the normal stack pointer.

Refresh. The Z8000 CPU contains a counter that can be used to refresh dynamic memory automatically.

Program Status Registers. This group of status registers contains the program counter, flags, and control bits. These are automatically saved when an interrupt or trap occurs, and a new status group is loaded.

Interrupt and Trap Structure. The CPU supports three types of interrupts: vectored and nonvectored maskable, and nonmaskable. There are four traps: system call, unimplemented instruction, privileged instruction, and segmentation trap.

Data Types. Z8000 instructions can operate on bits, BCD digits (4 bits), bytes (8 bits), words (16 bits), long words (32 bits), byte strings and word strings up to 64K bytes long.

Segmentation and Memory Management. The Z8001 can directly access 8M bytes of address space, using a segmented addressing scheme, implemented via the Z8010 MMU Memory Management Unit. The 8M bytes of Z8001 address space is divided into 128 relocatable segments of up to 64K bytes each. The addresses entered into instructions and output by the CPU in executing them are *logical* addresses. The MMU translates these logical addresses into addresses in physical memory. This process—relocation—is transparent to the user software.

Addressing Modes. Eight addressing modes are provided in the instruction set: Register (R), Immediate (IM), Indirect Register (IR), Direct Address (DA), Indexed (X), Relative Address (RA), Base Address (BA), and Base Indexed (BX).

Input/Output. A set of I/O instructions performs 8-bit or 16-bit transfers between CPU and I/O devices.

Multimicroprocessor Support. A pair of CPU pins is used in conjunction with certain instructions to coordinate multiple microprocessors.

Instruction Set. The Z8000 has in its repertoire the nine categories of instructions following:

Load and exchange

Arithmetic

Logic

Program control

Bit manipulation

Rotate and shift

Block transfer and string manipulation

Input/output

CPU control

Status Lines. Seven pins of the Z8000 are dedicated to the issuance of status information. Three are the function select lines Read/Write, Normal/System, and Byte/Word. The other four lines (ST_0-ST_3) issue codes denoting type of operation (program or I/O reference, data or stack memory request, or internal operation), acknowledging external requests (segment trap or interrupt), and initiating memory refresh cycles.

71

Z8010 MMU Memory Management Unit

Zilog

Product Brief

Preliminary

August 1979

Features

Dynamic segment relocation makes software addresses independent of physical memory addresses.

Sophisticated access validation protects memory areas from unauthorized or unintentional access.

MMU architecture supports multiprogramming systems.

Sixty-four variable-sized segments from 256 to 64K bytes can be managed within a total physical address space of 16M bytes; all 64 segments are randomly accessible.

Multiple MMUs can support several translation tables for each of the six Z8001 address spaces.

Description

Declining memory costs coupled with the increasing power of microprocessors has accelerated the use of high-level languages, sophisticated operating systems, complex programs and large data bases in microcomputer systems. The Z8001 microprocessor CPU supports these trends with an eight megabyte direct address space as well as a rich and powerful instruction set. The Z8010 Memory Management Unit (MMU) provides flexible and efficient support for this large address space by offering dynamic segment relocation as well as numerous memory-protection features.

The primary memory of a computer is one of its major resources. As such, the management of this resource becomes a major concern as demands on it increase. These demands arise from multiple users (or multiple tasks within a dedicated application), the need to increase system integrity by limiting access

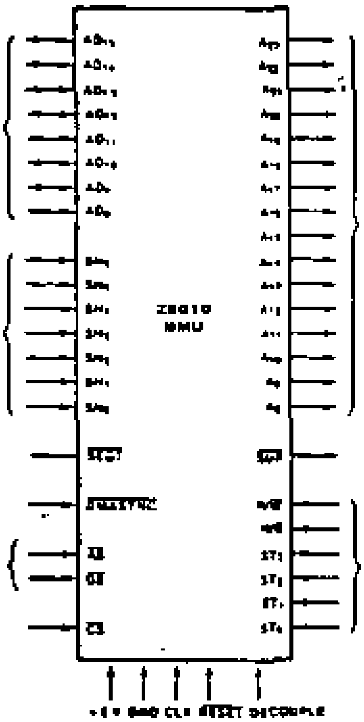


Figure 1. Pin Functions

MSE C-15

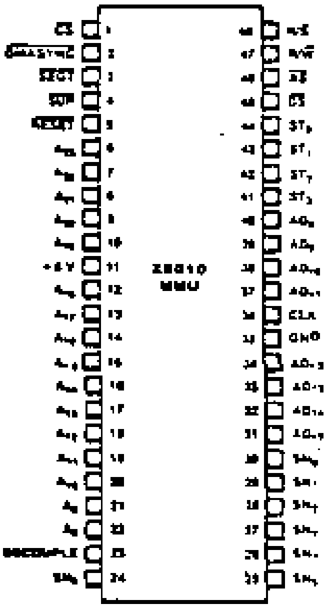


Figure 2. Pin Assignments

C8048-0051 C8048 0033

Description (Continued)

to various portions of the memory, and from the need to structure large, complex programs and systems.

Multiple tasks (or users) of a system that can reside anywhere in memory are called *relocatable*. Generally, systems in which all tasks are relocatable offer far greater flexibility in responding to changing system environments. Another aspect of multiple-task environments is sharing; separate tasks can execute the same program on different data, or several tasks may execute different programs using the same data.

Unfortunately, a problem that arises in multiple-task systems is that of system integrity. Tasks must be protected from unwanted interactions with other tasks; user tasks must be prohibited from performing operating system functions; and user tasks must also be protected from themselves so they cannot overflow the areas allotted to them.

In addition to these considerations, support for the design and implementation of large, complex programs and systems is itself an important consideration. Modern trends are toward the partitioning of a complex task into small, simple, self-contained subtasks that have well-defined interfaces. Because these subtasks interact with each other, communication between them must be carefully controlled. Memory-management systems can offer effective solutions for implementing large systems modularly designed.

The Z8010 Memory Management Unit supports multiple-process and large modular software systems with dynamic segment relocation. Futhermore, it enhances system integrity with a powerful set of memory protection features.

Relocation. Dynamic segment relocation makes user software addresses independent of the physical memory addresses, thereby freeing the user from specifying where information is actually located in the physical memory and providing a flexible, efficient method for supporting multi-programming systems.

The Z-MMU uses a translation table to transform the 23-bit logical addresses from the Z8001 CPU into 24-bit addresses for the physical memory. Memory segments are variable in size from 256 bytes to 64K, in increments of 256 bytes. Pairs of Z-MMUs support the 128 segment numbers available for the various Z8001 CPU address spaces. Within an address space, any number of Z-MMUs can be used to accommodate multiple translation tables for system and normal operating modes, or to support more sophisticated memory-management systems.

System Integrity. Z-MMU memory-protection features safeguard memory areas from unauthorized or unintended access by associating special access restrictions with each segment. A segment is assigned a "personality" consisting of several attributes when it is initially entered into the Z-MMU. When a memory reference is made, these attributes are checked against the status information supplied by the Z8001 CPU. If a mismatch occurs, a trap is generated and the CPU is interrupted. The CPU can then check the status registers of the MMU to determine the cause and take appropriate action to correct the problem.

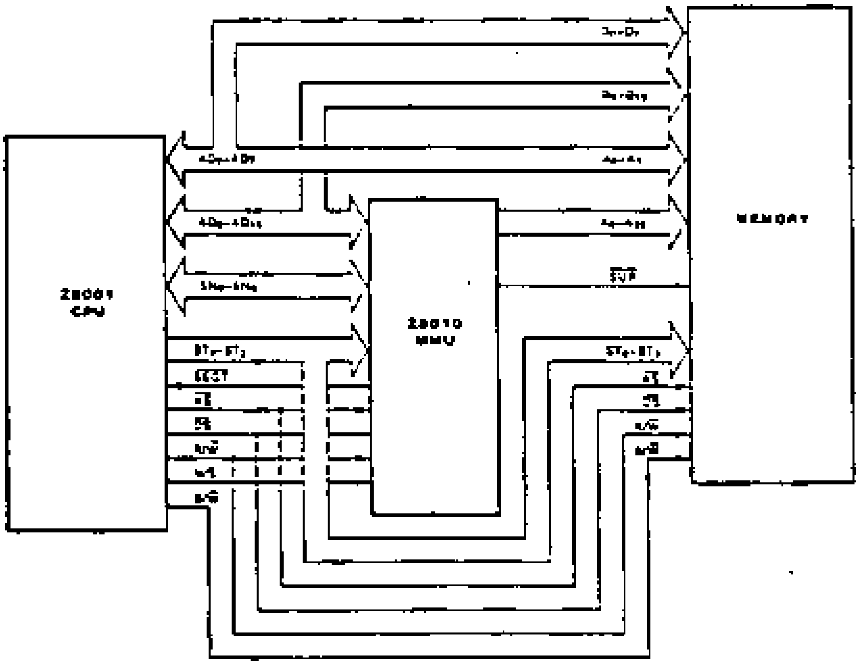


Figure 3. The MMU in a Z8000 System

Z8034 UPC Universal Peripheral Controller

Zilog

Product Brief

Preliminary

August 1979

Features

- Complete slave microcomputer, for distributed-processing Z-bus use.
- Unmatched power of Z8 architecture, instruction set.
- Three programmable I/O ports, two with 2-wire handshake, or any combination of data and control lines.
- Six levels of priority interrupts to Z-UPC.
- Two programmable 8-bit counter/timers with 6-bit prescalers.
- 256 byte register file, accessible by both master CPU and Z-UPC, as allocated by Z-UPC program.
- 2K bytes of on-chip program ROM for efficiency, versatility.

Description

The Z-UPC Universal Peripheral Controller is a distributed microcomputer that performs the three basic interfacing functions needed to interface a CPU with peripherals: device control by ROM-resident internal software, data manipulation, such as reformatting or arithmetic, and data buffering in internal registers.

The Z-UPC is similar to the Z8 microcomputer and uses the Z8 instruction set. Under program control, its three 8-line I/O ports can be tailored to the needs of its user. Permanently configured as a single-chip controller with 2K bytes of internal ROM, the Z-UPC executes instructions in 2.2 µs average using a 4-MHz clock source. Its register file contains 256 bytes, of which 234 are general-purpose registers, 19 are status and control registers, and three are port registers.

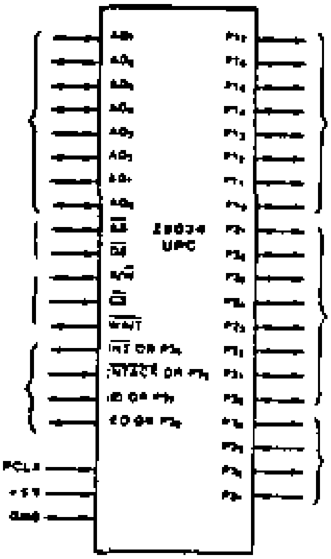


Figure 1. Pin Functions

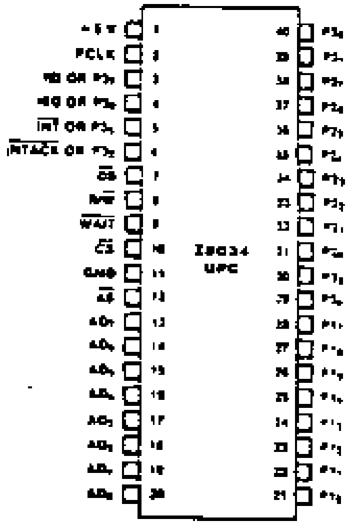


Figure 2. Pin Assignments

Description (Continued)

The Z-UPC Universal Peripheral Controller is an intelligent device that generates all the control signals peripheral devices need. Because it does off-line arithmetic, translates data before transmitting, and buffers data, the Z-UPC unburdens the master CPU, thereby increasing the overall speed and efficiency of the system in which it resides.

Based upon the Z8 microcomputer architecture, the Z-UPC offers fast execution time, efficient use of memory, and sophisticated interrupt, I/O, and bit manipulation. Its powerful and extensive instruction types, combined with its efficient internal register addressing scheme, not only speeds program execution, but also efficiently packs program into the on-chip ROM.

A unique characteristic of the Z-UPC is its register file, which contains I/O port and control registers that can be accessed both by the Z-UPC program and by its associated master CPU. This results in byte efficiency, programming efficiency, and address space efficiency because Z-UPC instructions can operate directly on I/O data without moving it to and from an accumulator. It also allows the Z-UPC user to allocate as data buffer between the CPU and the peripheral all register space not in use as accumulators, address pointers, index registers, or stack. Registers not used as buffer are protected against CPU access. The register file is divided into 16 groups of 16 working registers each. A register pointer uses fast, short-format instructions to access any one of these groups quickly, resulting in fast and easy task switching. Two-way communication between the master CPU and the register file is facilitated by another pointer that positions 16 interface registers anywhere within the register file. These registers are accessed directly by both the master CPU and the slave Z-UPC. Four more registers, similarly accessed, convey control and status information.

All of the Z-bus's daisy-chained priority interrupt system can be implemented in the Z-UPC under software control, or the Z-UPC can be programmed to function in a polled environment. In all, the Z-UPC has 24 pins that can be dedicated to I/O functions. Grouped logically into three 8-line ports, they can be programmed in many combinations of inputs, outputs, and bidirectional lines, with or without handshake and with push-pull or open-drain outputs.

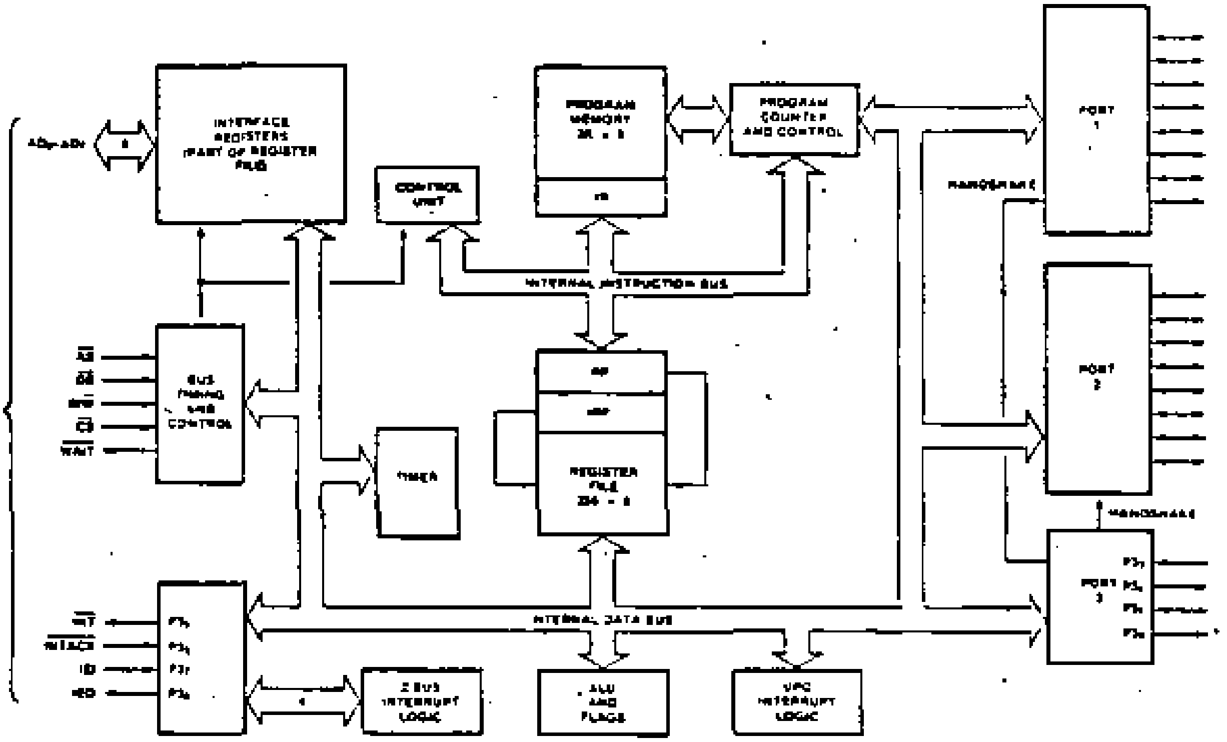


Figure 3. Functional Block Diagram

53

Z8036 CIO Counter/Timer and Parallel I/O Unit

Zilog

Product Brief

Preliminary

August 1979

Features

Two independent 8-bit double-buffered bidirectional I/O ports plus a special-purpose 4-bit I/O port.

Four handshake modes including IEEE-488.

Wait/Request line for high speed data transfer.

Three independent 16-bit counters.

All registers read/write and directly addressable.

Flexible pattern recognition logic, programmable as 16-input interrupt controller.

Description

The Z8036 CIO Counter/timer and Parallel I/O element is a general purpose peripheral circuit that satisfies most counter/timer and parallel I/O needs encountered in system designs. This versatile device contains three I/O ports and three counter/timers. Many programmable options tailor its configuration to specific applications. The use of the device is simplified by making all internal registers (command, status, and data) readable and (except for status bits) writable. Also, each register is given its own unique address so it can be accessed directly—no special sequential operations are required. The Z-CIO is directly Z-bus compatible.

Either 8-bit I/O port can be a handshake, byte port or a bit port. In the bit mode, data direction is programmable bit by bit. In the handshake mode, the ports can be input, output, or bidirectional, and they may be linked to form a 16-bit port. The four handshake modes include IEEE-488, interlocked (for interfacing to a Z-UPC, Z-FIO or another Z-CIO), strobed and pulsed. The pulsed mode connects one counter/timer with the handshake logic for interfacing a mechanical device such as a printer. The 4-bit port provides handshake controls, special controls (Wait/Request) or general-purpose I/O.

The counter/timer section contains three 16-bit counters, two of which can be software-configured as a 32-bit counter/timer. Up to

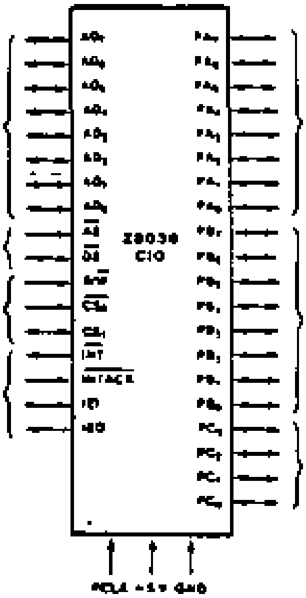


Figure 1. Pin Functions

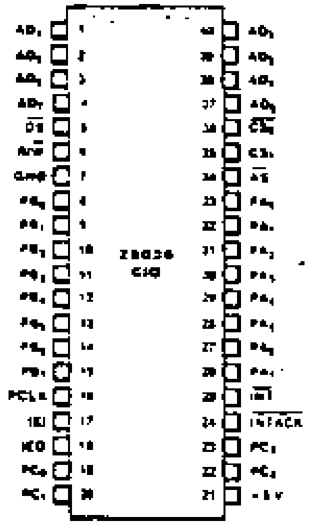
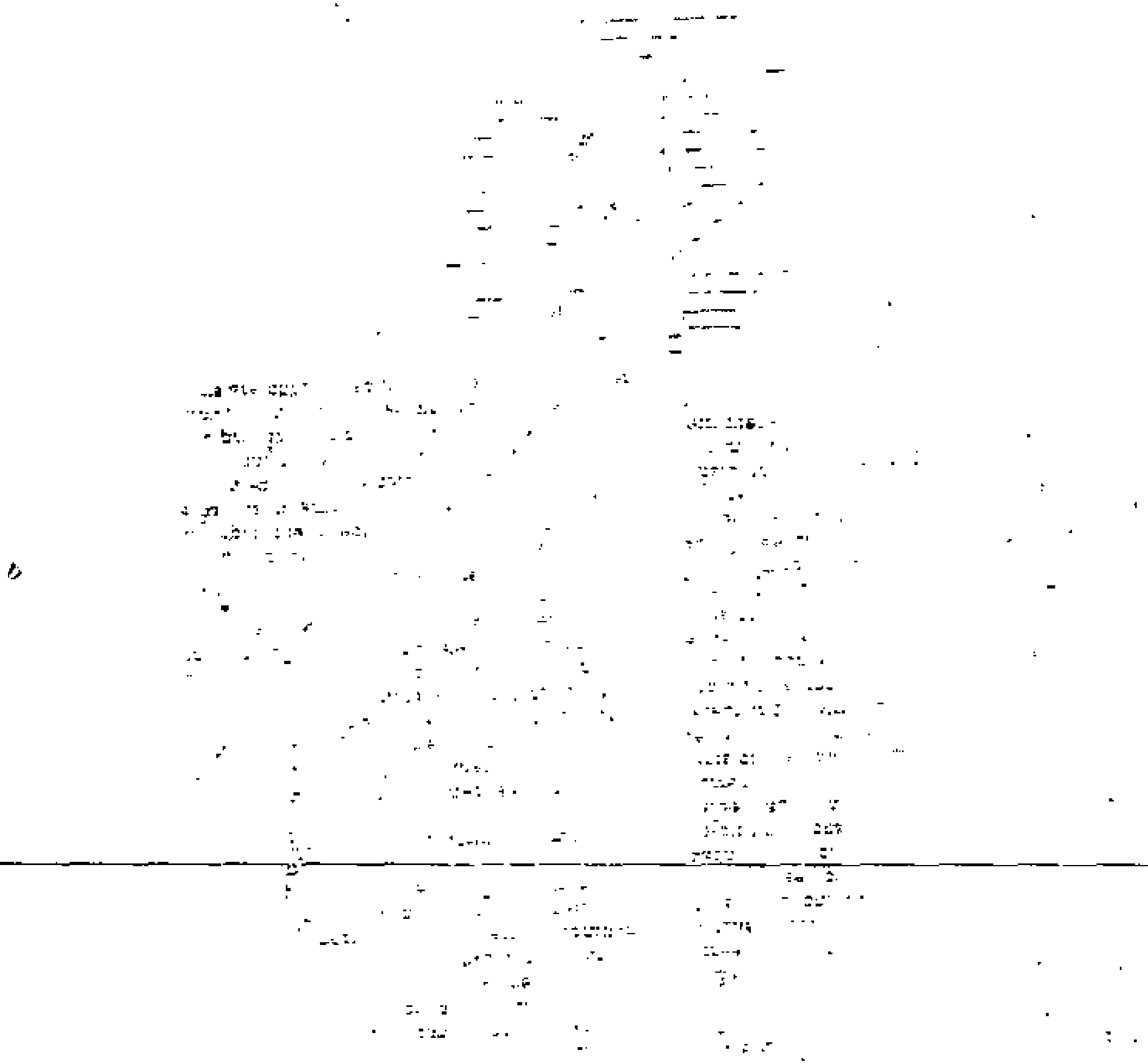


Figure 2. Pin Assignments

MSE C-19

C8050-0025 C8053-0026



b

Description (Continued)

four I/O lines for each counter are available for direct external control and status information. All counters have a programmable output duty cycle, continuous or single-cycle operation, and the counting process can be programmed to be either retriggered or nonretriggered.

Figure 3 shows how the Z-CIO is used. The two general purpose 8-bit ports are similar. They can be programmed as handshake driven, double-buffered ports (input, output, or bidirectional) or as control ports in which the direction of each bit is individually programmable. Port B can also be specified to provide external access for two of the counter/timers. Each port includes pattern recognition logic allowing interrupt generation when a specified pattern is detected. The pattern recognition logic can be programmed so that the port functions like a priority interrupt controller.

To control these capabilities, each port contains 13 registers. Three of these, the input, output, and buffer registers, are data path registers. Two others, the mode specification and handshake specification registers, define the mode of the port and specify what handshake to use, if any. The reference pattern for the pattern recognition logic is defined in three registers, the pattern polarity, pattern transition, and pattern mask registers. The detailed characteristics of each bit path (for example, the direction of data flow, or whether a path is inverting or noninverting) are programmed using the data path polarity, data direction, and special I/O control registers. The primary control and status bits are grouped in a single register so that after the ports are configured initially, only this register need be accessed often. One register contains the interrupt vector associated with each port. To facilitate initialization, the port logic is designed so that if a capability of the port is not required the registers associated with that capability are ignored and need not be programmed.

The function of port C depends upon the roles of ports A and B. Port C provides handshake lines for the other two when required. Any bits of port C not so used can be used as I/O lines or as external access to the third counter/timer.

Besides the data input and output registers, three registers are needed. These specify the details of each bit path: data path polarity, data direction, and special I/O control.

The three counter/timers are all identical. Each is composed of a 16-bit down-counter, a 16-bit time constant register (which holds the value loaded into the down-counter), a 16-bit current count register (used to read the contents of the down-counter), and two 8-bit registers for control and status (the mode select and control registers). All three share a common vector register.

Each counter/timer can be programmed as either counter or timer. Up to four port I/O lines can be designated as external access lines for it. The lines are: Counter Input, Gate Input, Trigger Input, and Counter/Timer Output. Three different counter/timer output duty cycles are available: pulse, one-shot, or square wave. The operation of the counter/timer can be specified to be either single cycle or continuous. The counting sequence may be retriggered or nonretriggered, under program control.

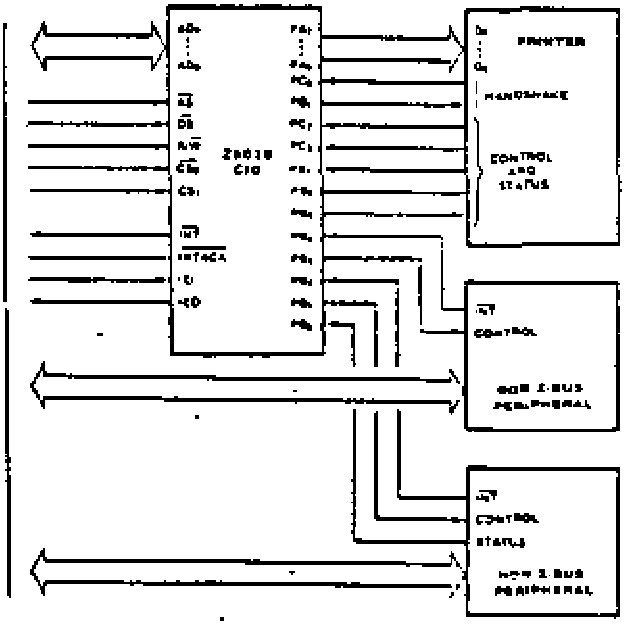


Figure 3. Functional Block Diagram

65

Z8030 SCC Serial Communications Controller

Zilog

Product Brief

Preliminary

August 1979

Features

- Two independent, 0 to 1 Megabit-per-second, full-duplex channels, each with its own quartz oscillator, baud-rate generator, and digital phase-locked loop for clock recovery.
- Multi-protocol operation under program control.
- Asynchronous mode with 5 to 8 bits and 1, 1½, or 2 stop bits per character; programmable clock factor; break detection and generation; parity, overrun, and framing error detection.
- Local loopback and auto-echo modes.
- Bisynchronous mode with internal or external character synchronization on one or two sync characters and CRC generation and checking with CRC-16 or CRC-CCITT preset to either 1s or 0s.
- SDLC/HDLC mode with comprehensive frame-level control, automatic zero insertion and deletion, I-field residue handling, abort generation and detection, CRC generation and checking, and loop mode operation.
- Programmable for NRZ, NRZI, or FM coding.

Description

The Z-SCC Serial Communication Controller is a dual-channel, multi-protocol data communication peripheral for Z-bus use. It is software-configured to satisfy a wide variety of serial communication applications. Its basic function is serial-to-parallel and parallel-to-serial conversion. However, the Z-SCC also contains a repertoire of new, sophisticated internal functions that minimize the need for external random logic on the circuit card.

The Z-SCC handles asynchronous formats, synchronous byte-oriented protocols such as IBM Bisync, and synchronous bit-oriented protocols such as HDLC and IBM SDLC. This versatile device also supports virtually any other serial data transfer application (cassette or diskette interface, for example).

The device can generate and check CRC

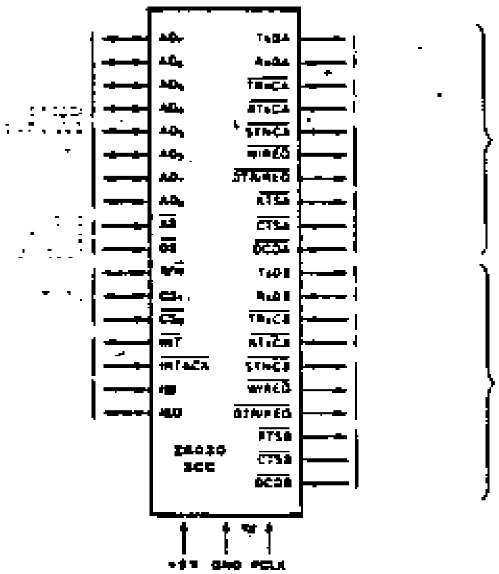


Figure 1. Pin Functions

MSE C-21

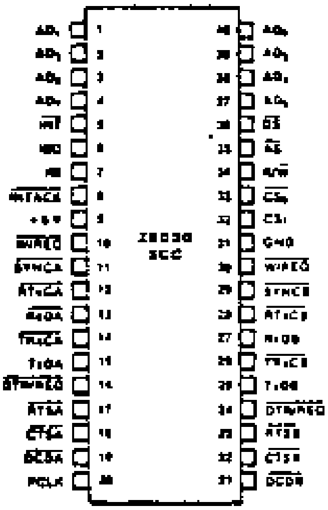


Figure 2. Pin Assignments

C6051-0039 C8051-0041

Description (Continued)

codes in any synchronous mode and can be programmed to check data integrity in various modes. It also has facilities for modem controls in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

As is standard among Zilog peripheral components, the Z-bus daisy-chain interrupt heirarchy is supported.

The Z-SCC contains the necessary multiplexed address/data bus interface with strobe and chip select lines to function as a Z-bus peripheral. It includes internal control and interrupt logic, two full-duplex channels and two baud-rate generators. Associated with each channel are several read and write registers for mode control as well as the logic necessary to interface to modems or other external devices.

The read and write register group for each channel includes eight control registers, two sync-character registers, and four status registers. Each baud rate generator has two read/write registers for holding the time constant that determines baud rate. Associated with the interrupt logic is a write register for interrupt vector and three read registers: vector with status, vector without status, and interrupt pending status.

The logic for both channels provides formatting, synchronization and validation for data transferred to and from the channel interface. The modem control inputs are monitored by the control logic under program control. All of the modem control signals are general purpose in nature and optionally can be used for functions other than modem control.

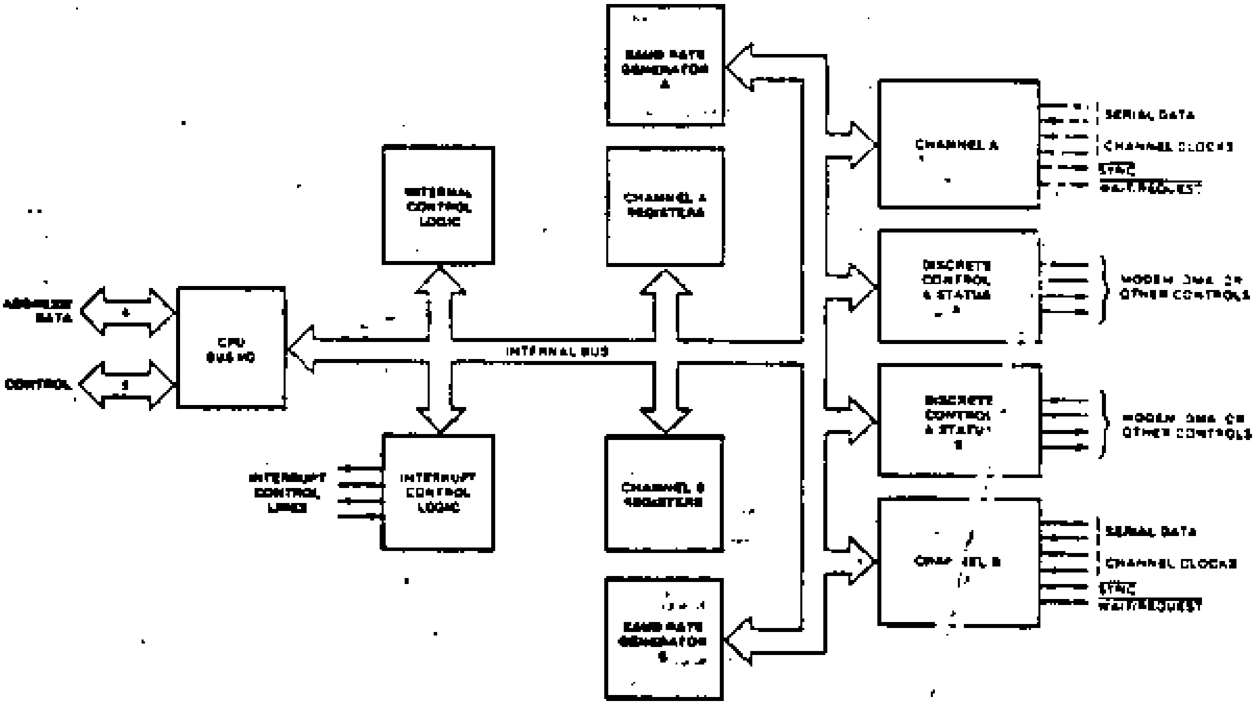


Figure 1. Functional Block Diagram

Typical Applications

Figure 4 shows how a Z-SCC can be connected with channel A programmed for the Synchronous Data Link Control (SDLC) Loop mode, functioning as a secondary station. If NRZI or FM coding is used, no clock lines are required because the clock can be recovered from the received data, using the Z-SCC's on-chip digital phase locked loop (DPLL). Another Z-SCC (not shown), programmed for the SDLC mode, would be the controlling station, polling the loop for traffic. The figure shows a typical, asynchronous serial port being serviced by channel B of the Z-SCC. It could just as well support another synchronous data link, or even a high-speed link, transferring data via a DMA controller.

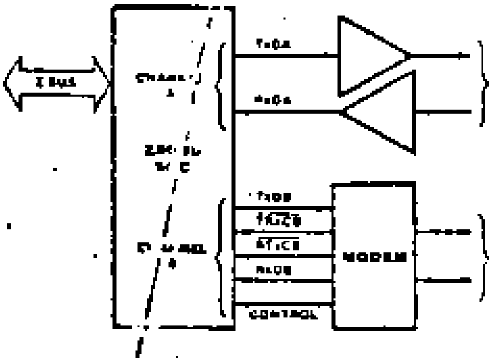


Figure 4. Loop Secondary Station and Serial Port