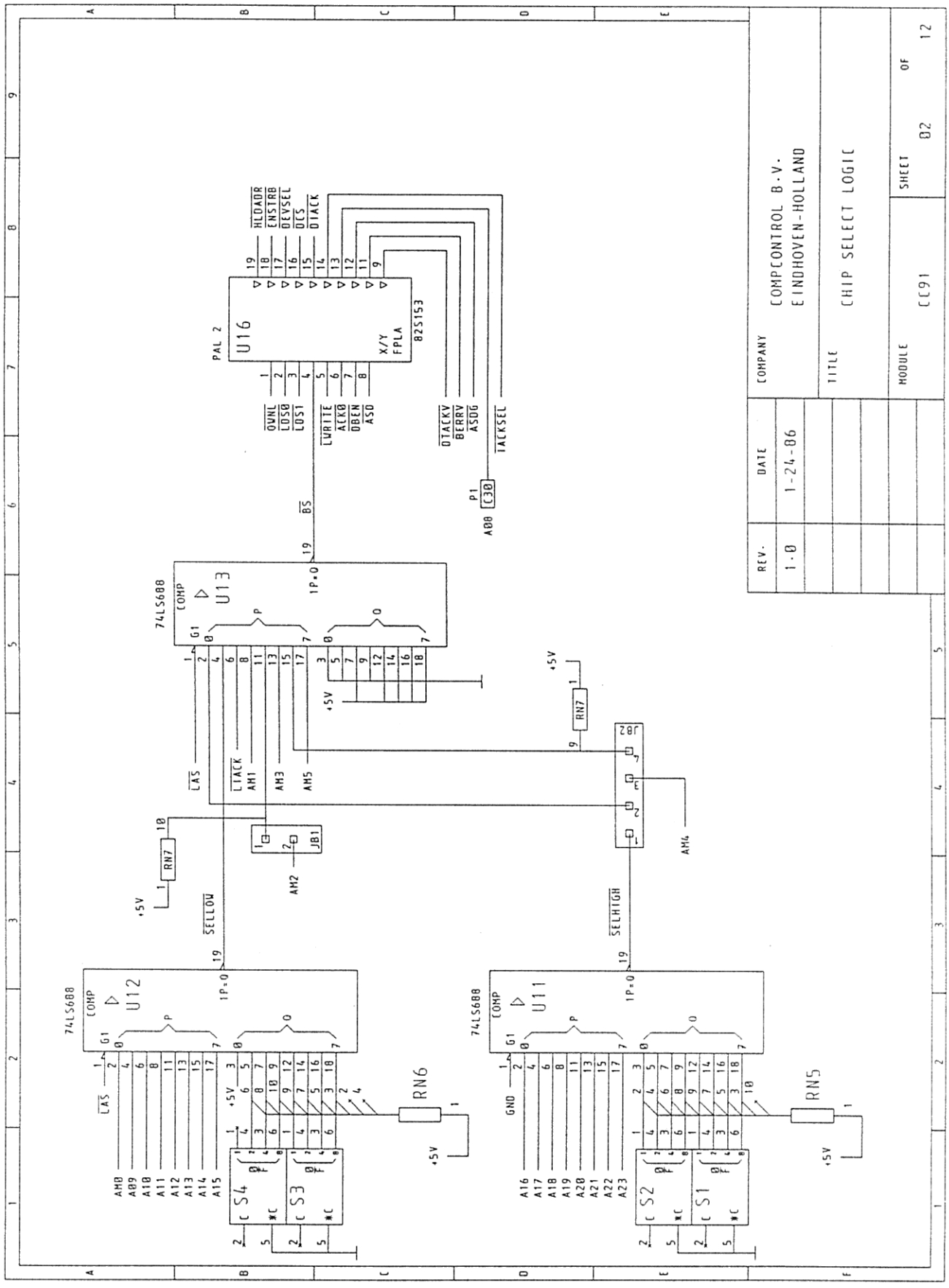
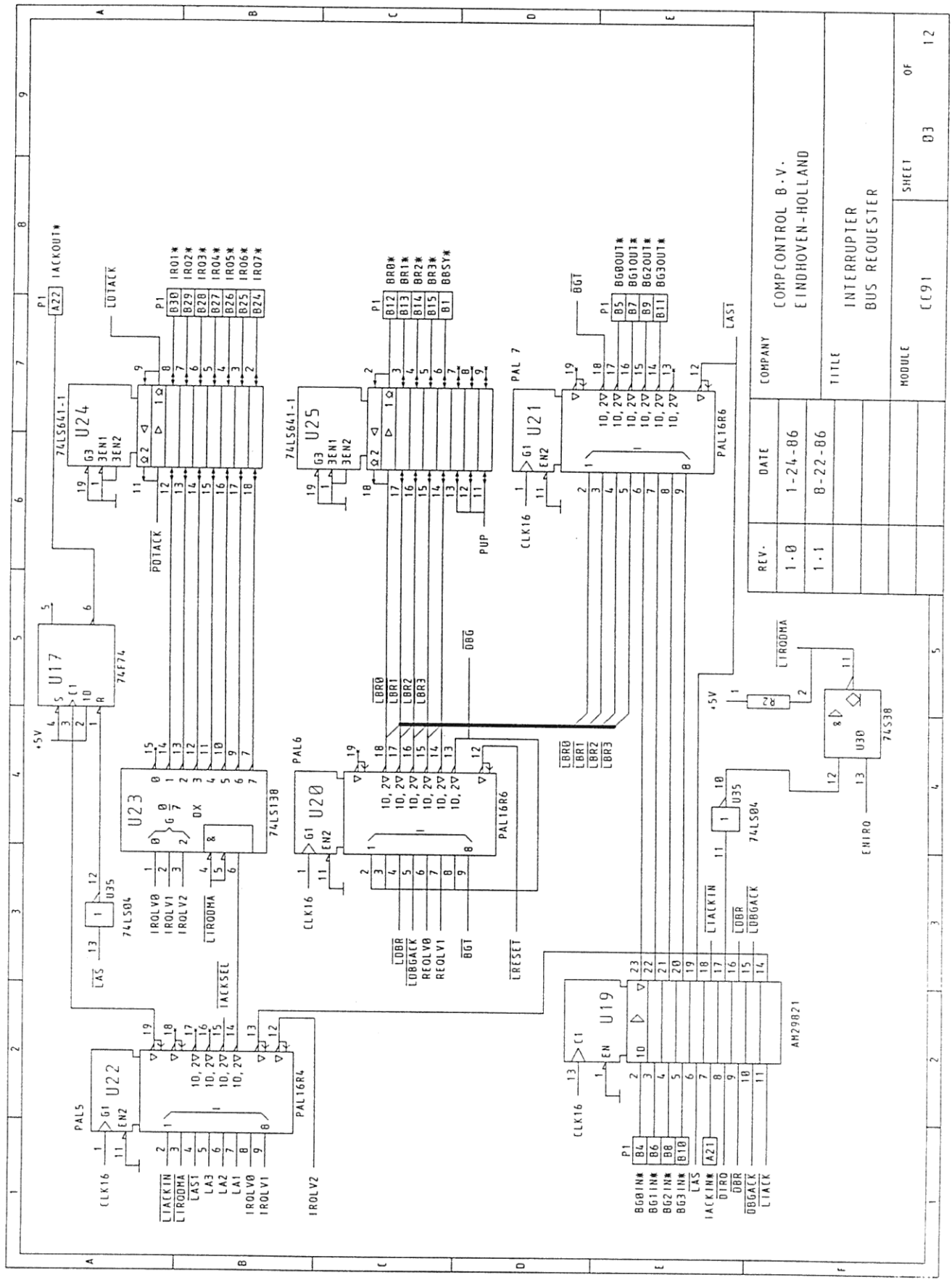
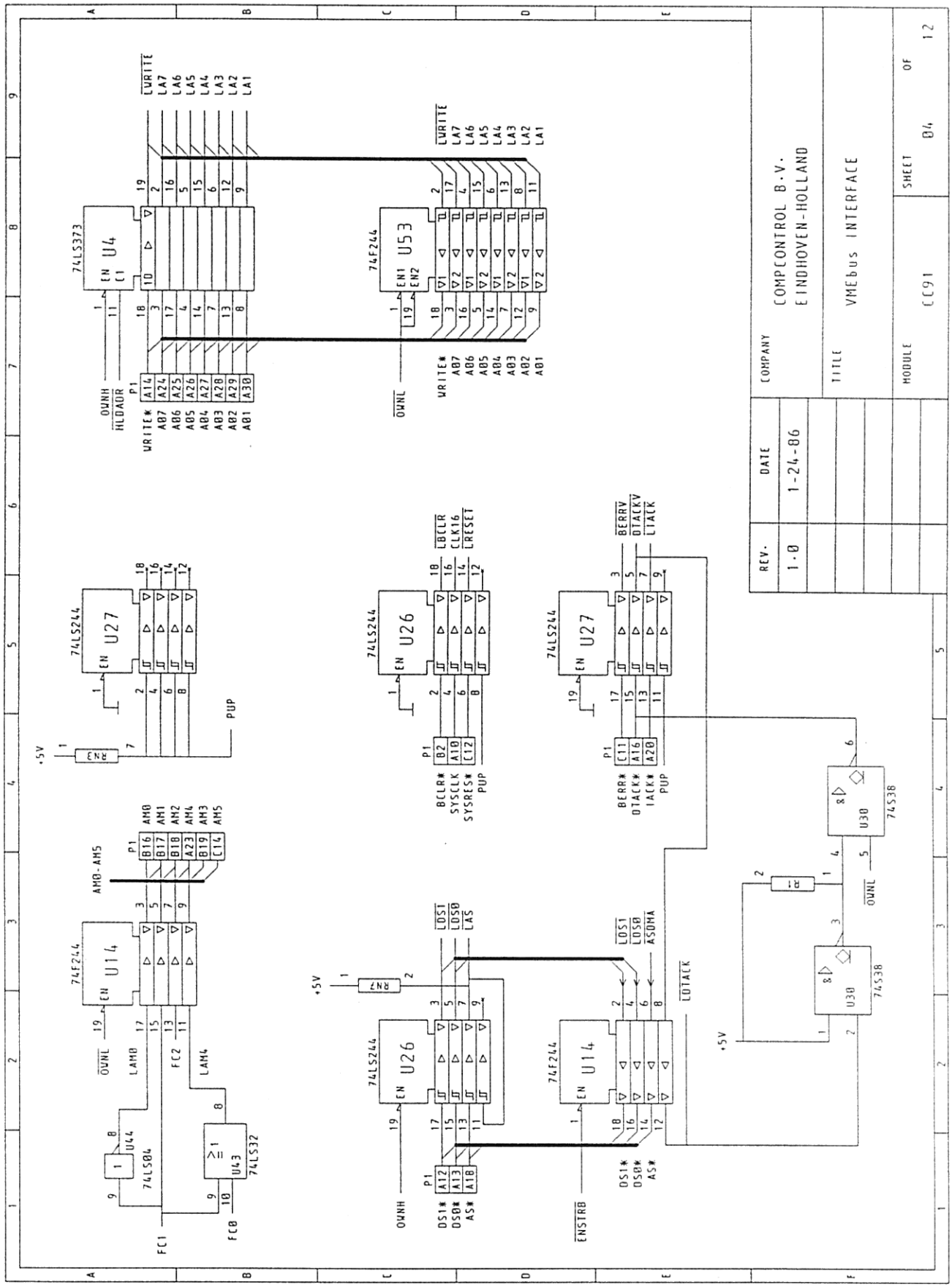
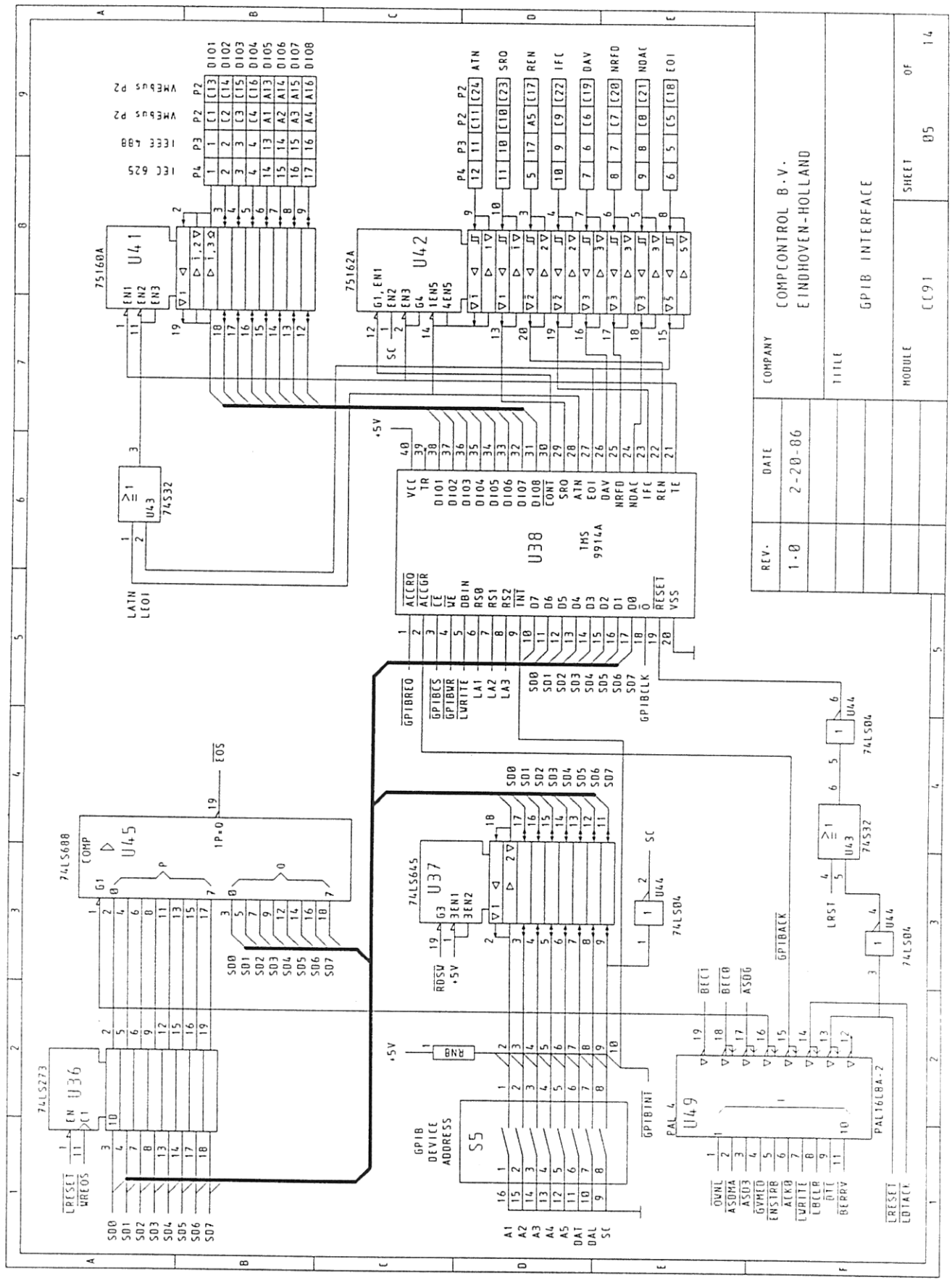
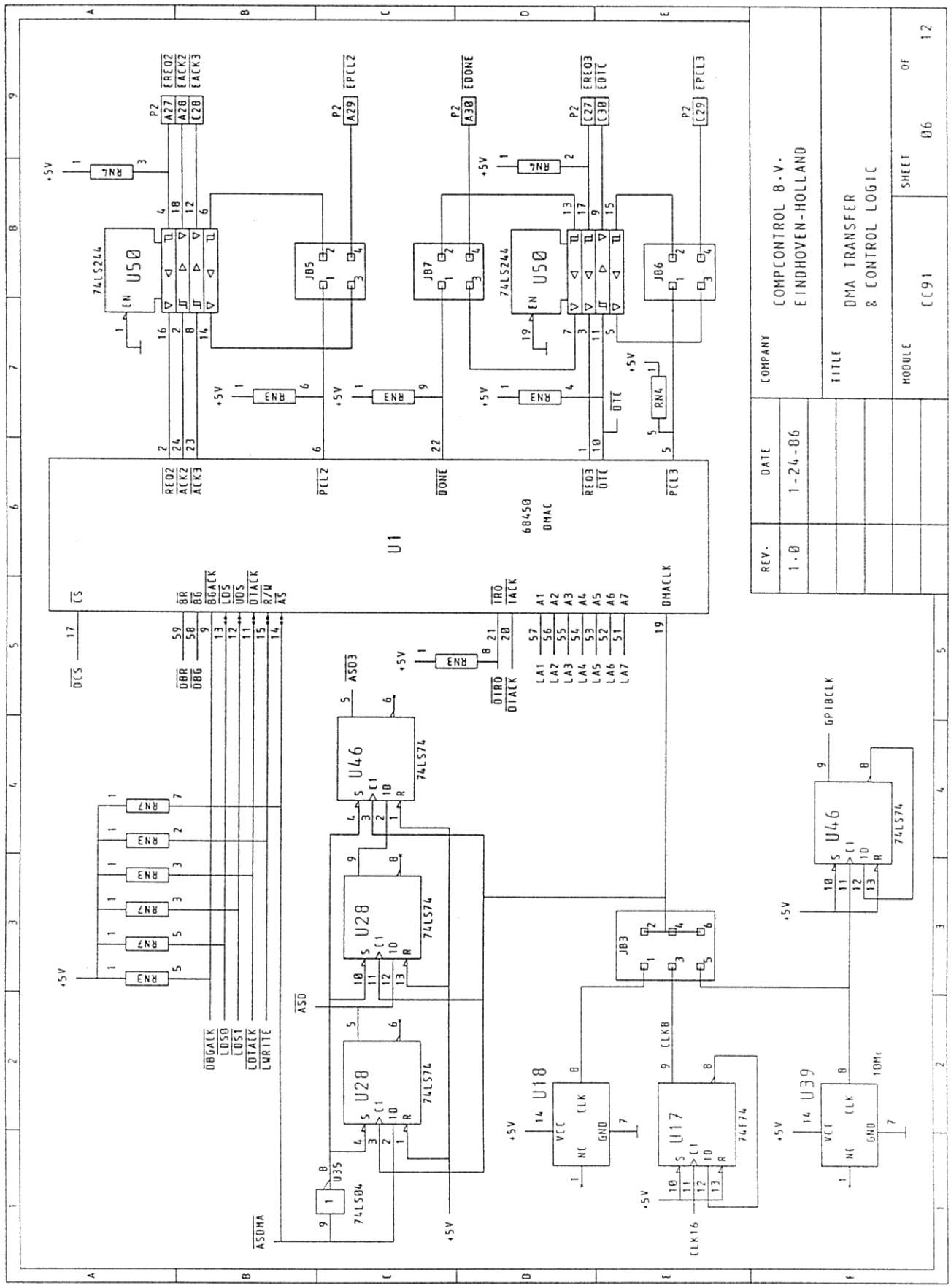
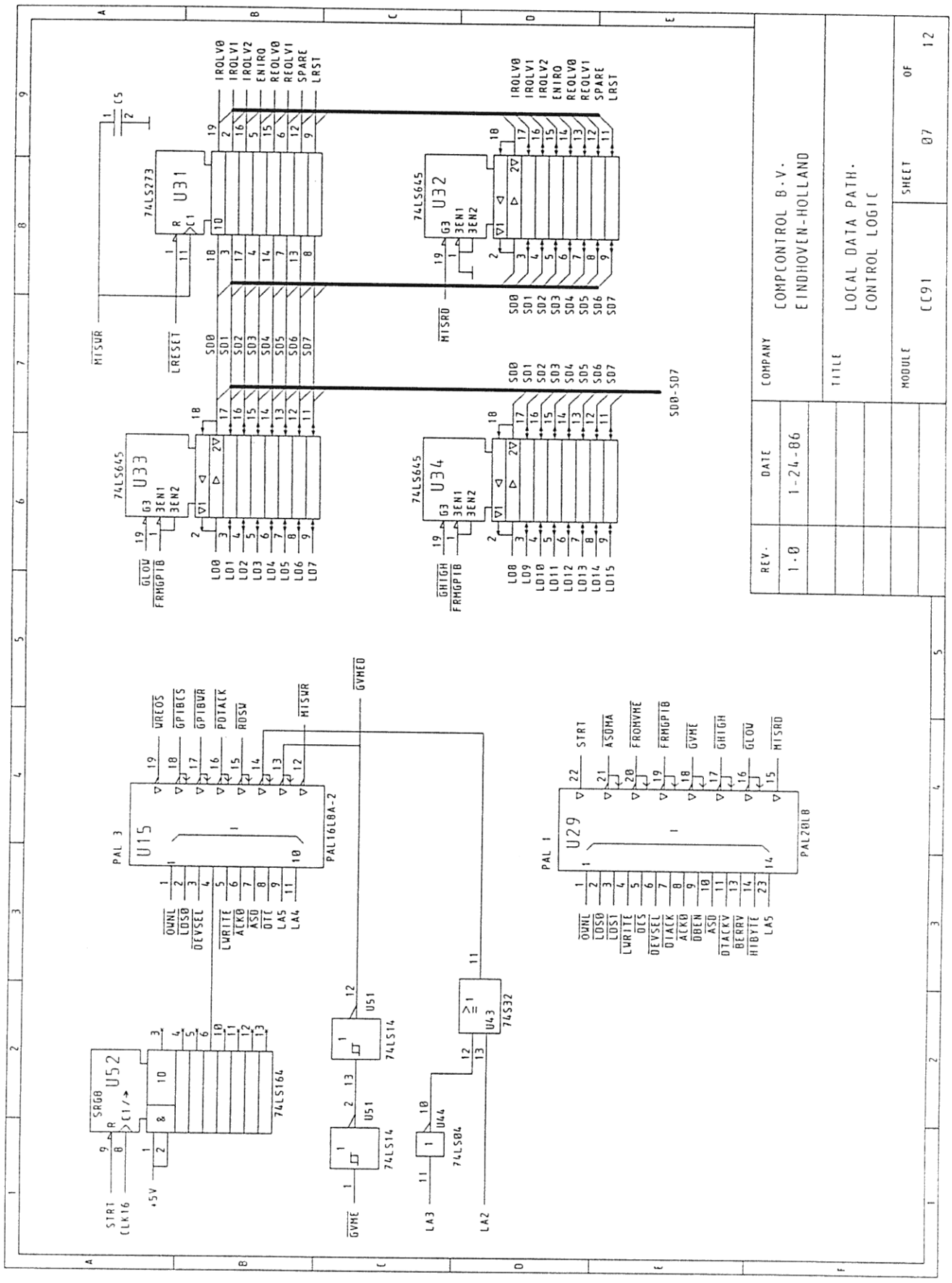

COMPANY
COMPCONTROL B.V.
EINDHOVEN-HOLLAND
TITLE
CHIP SELECT LOGIC
MODULE
CC91
SHEET
02
OF
12
DATE
1-24-86
REV.
1.0
PAL 2
U16
X/Y
FPLA
82S153
HLDADR
ENSTRB
DEVSEL
DTCS
DTACK
OVNL
LDS0
LDS1
LWRITE
ACK0
DBEN
AS0
DTACKV
BERRV
ASDG
TACKSEL
BS
P1
A00
C30
74LS688
COMP
U13
U12
U11
G1
1P=Q
LAS
LTACK
AM1
AM3
AM5
AM2
AM4
JB1
JB2
RN7
+5V
SELLOW
SELHIGH
GND
AM8
AM9
A10
A11
A12
A13
A14
A15
A16
A17
A18
A19
A20
A21
A22
A23
RN6
RN5
S4
S3
S2
S1


COMPCONTROL B.V.
EINDHOVEN-HOLLAND
COMPANY
INTERRUPTER
BUS REQUESTER
TITLE
CC91
MODULE
SHEET 03 OF 12
DATE
1-24-86
8-22-86
REV.
1.0
1.1
U24
74LS641-1
U25
74LS641-1
U21
PAL 7
PAL16R6
U17
74F74
U23
74LS138
U20
PAL6
PAL16R6
U22
PAL5
PAL16R4
U19
AM29821
U35
74LS04
U30
74S38
IACKOUT*
LOTACK
POTACK
IRQ1* IRQ2* IRQ3* IRQ4* IRQ5* IRQ6* IRQ7*
BR0* BR1* BR2* BR3* BBSY*
BGT
BG0OUT* BG1OUT* BG2OUT* BG3OUT*
LAS1
PUP
CLK16
+5V
LBR0 LBR1 LBR2 LBR3
OBG
IROLV0 IROLV1 IROLV2
LIRQOMA
LAS
IACKSEL
LRESET
BGT
LOBR
LOBGACK
REQLV0
REQLV1
LIACKIN
ENIRQ
BG0IN* BG1IN* BG2IN* BG3IN*
IACKIN*
DIRQ
OBR
OBGACK
LIACK
LIACKIN
LIRQOMA
LAS1
LA3
LA2
LA1
IROLV0
IROLV1
IROLV2

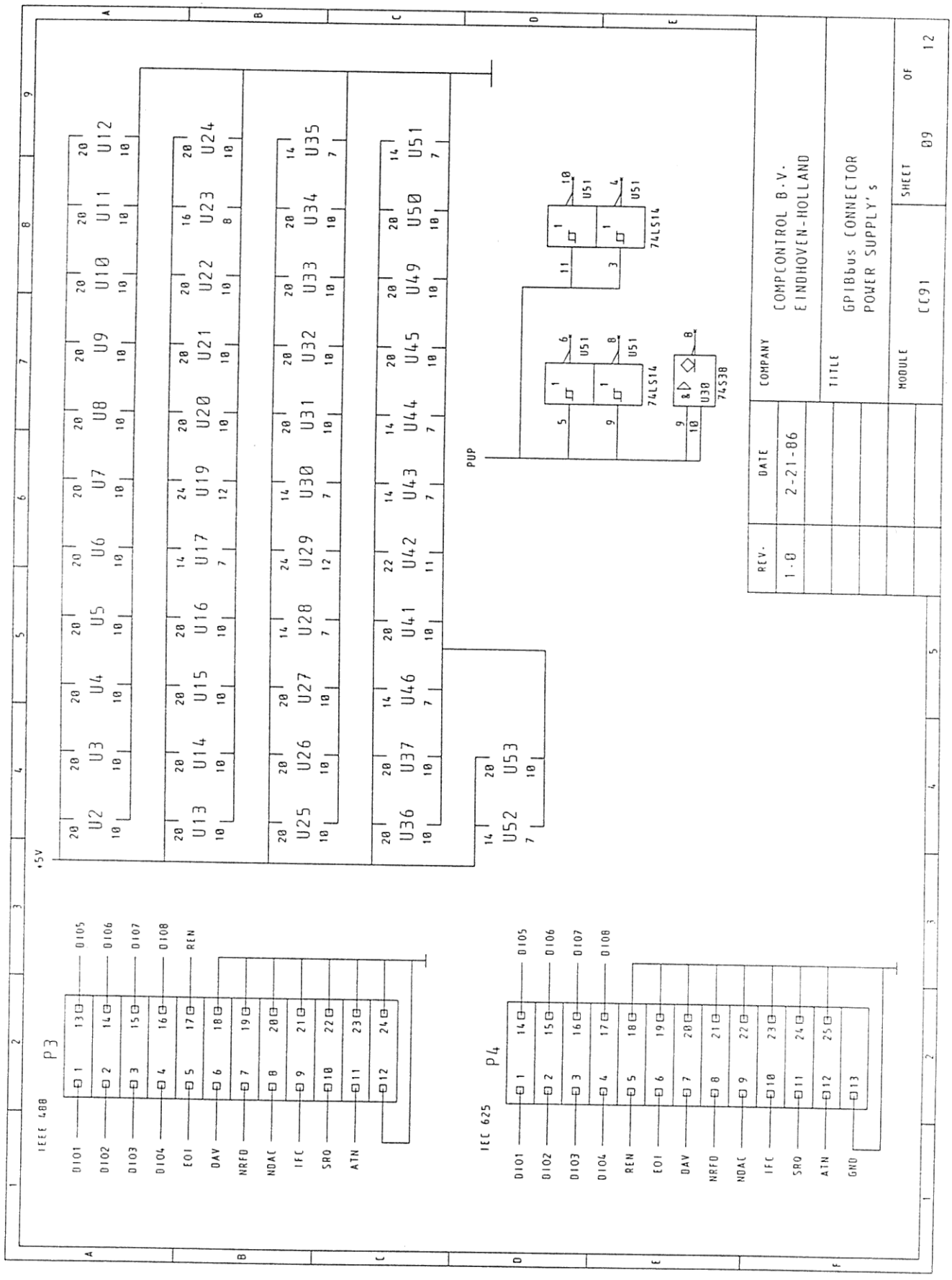

COMPCONTROL B.V.
EINDHOVEN-HOLLAND
COMPANY
VMEbus INTERFACE
TITLE
CC91
MODULE
SHEET 04 OF 12
REV. 1.0
DATE 1-24-86
74LS373 U4
74F244 U53
74LS244 U27
74LS244 U26
74F244 U14
74LS04 U44
74LS32 U43
74S38 U38
LWRITE
LA7 LA6 LA5 LA4 LA3 LA2 LA1
WRITE* A07 A06 A05 A04 A03 A02 A01
OWNH
HLOADR
OWNL
BERRV
DTACKV
LTACK
LBCLR
CLK16
LRESET
BCLR* SYSCLK SYSRES* PUP
BERR* DTACK* IACK* PUP
AM0-AM5
LDS1
LDS0
LAS
ASOMA
LDTACK
DS1* DS0* AS*
ENSTRB
LAM0
LAM4
FC1
FC2
FC0
+5V
RN3
RN7
R1
PUP

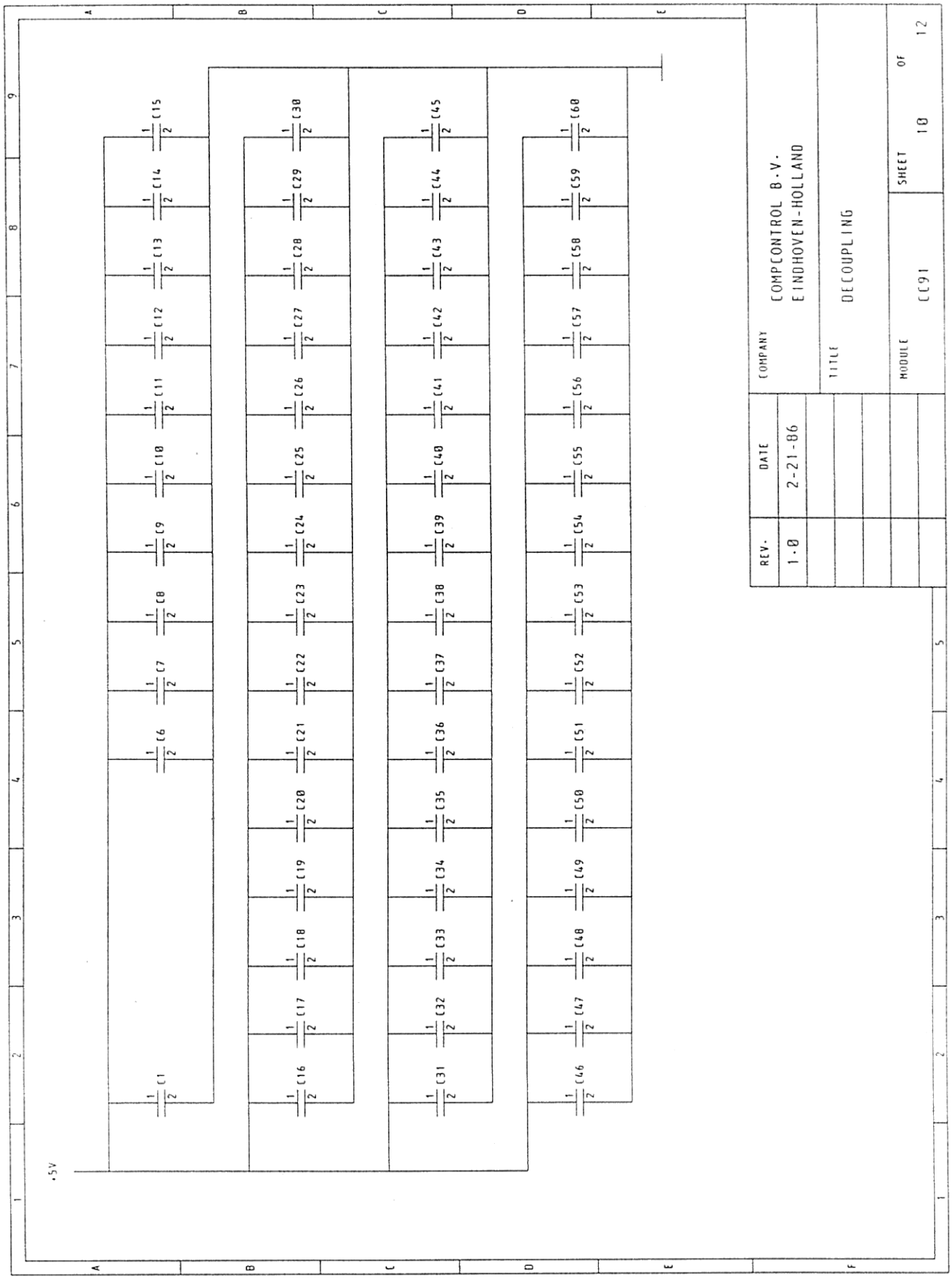

COMPCONTROL B.V.
EINDHOVEN-HOLLAND
COMPANY
GPIB INTERFACE
TITLE
CC91
MODULE
SHEET 05 OF 14
DATE
2-20-86
REV.
1.0
U38
TMS 9914A
U41
75160A
U42
75162A
U43
74S32
U44
74LS04
U45
74LS688
COMP
U36
74LS273
U37
74LS645
U49
PAL 4
PAL16L8A-2
S5
GPIB DEVICE ADDRESS
VMEbus P2
IEEE 488
IEC 625
LATN
LEOI
GPIBREQ
GPIBCLK
GPIBINT
GPIBACK
LRST
LRESET
WREOS
RDSW
RNB
EOS
SC

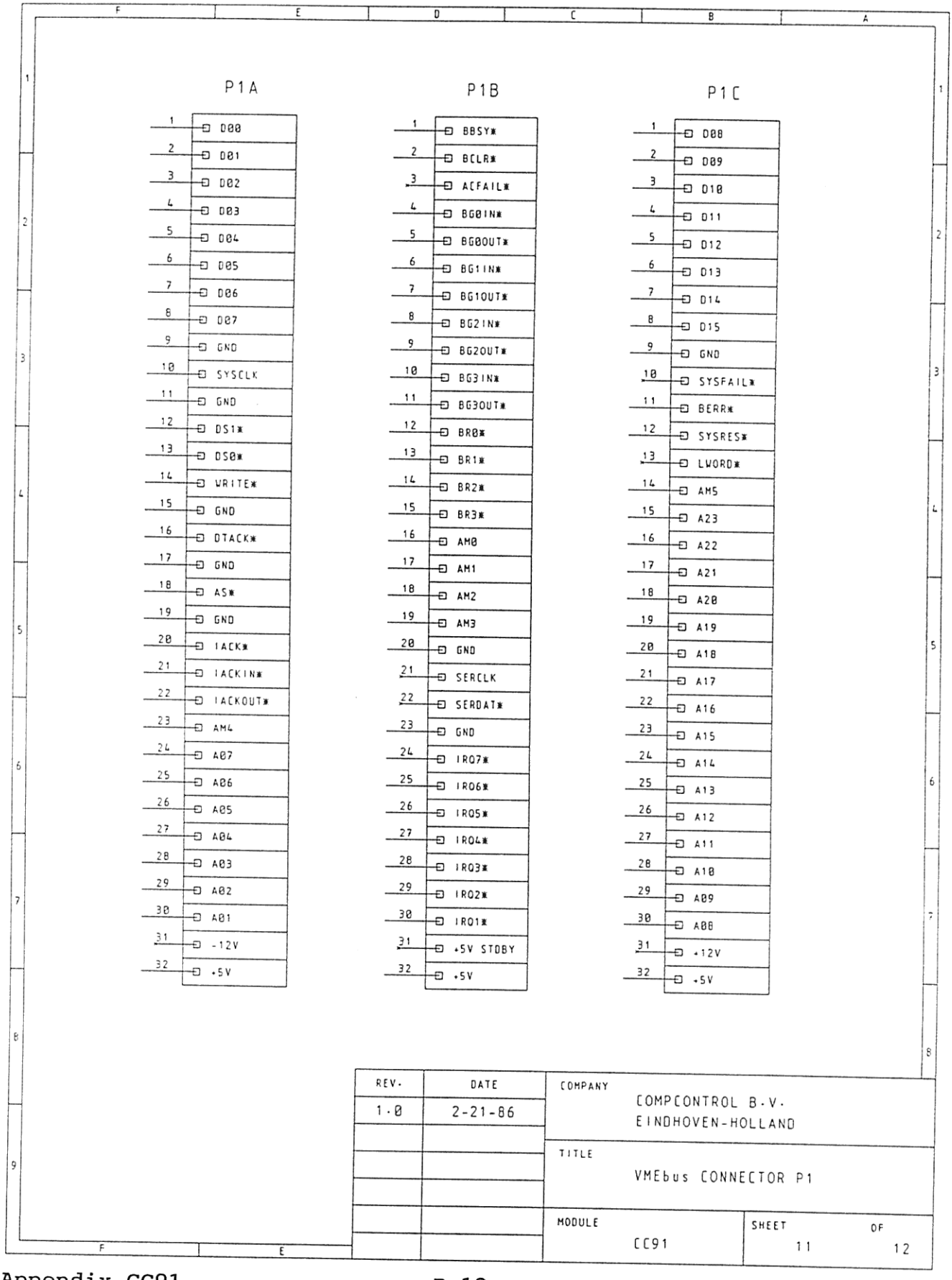

COMPCONTROL B.V.
EINDHOVEN-HOLLAND
COMPANY
DATE
1-24-86
REV.
1.0
TITLE
DMA TRANSFER
& CONTROL LOGIC
MODULE
CC91
SHEET
06
OF
12
U1
68450
DMAC
U50
74LS244
U46
74LS74
U28
74LS74
U35
74LS04
U18
U17
74F74
U39
10M
JB3
JB5
JB6
JB7
RN3
RN4
RN7
P2
GPIBCLK
CLK8
CLK16
DMACLK
+5V

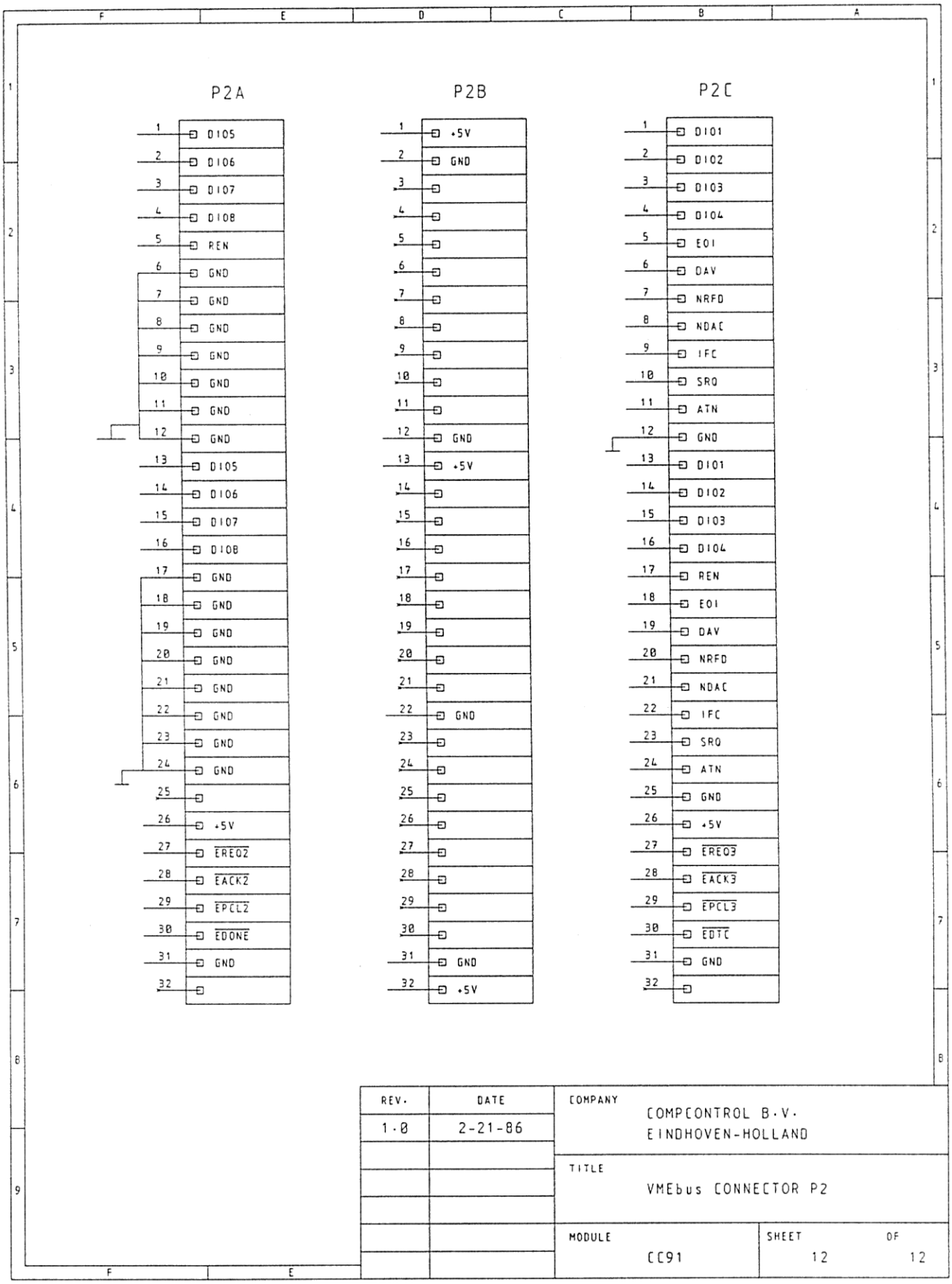

COMPCONTROL B.V.
EINDHOVEN-HOLLAND
COMPANY
DATE
1-24-86
REV.
1.0
TITLE
LOCAL DATA PATH.
CONTROL LOGIC
MODULE
CC91
SHEET
07
OF
12
U31
74LS273
U32
74LS645
U33
74LS645
U34
74LS645
U15
PAL 3
PAL16L8A-2
U29
PAL 1
PAL20L8
U52
SR68
74LS164
U51
74LS14
U44
74LS04
U43
74LS32
SD0-SD7
IROLV0
IROLV1
IROLV2
ENIRQ
REQLV0
REQLV1
SPARE
LRST
MISWR
LRESET
MISRD
GLOW
FRMGPIB
GHIGH
LD0
LD1
LD2
LD3
LD4
LD5
LD6
LD7
LD8
LD9
LD10
LD11
LD12
LD13
LD14
LD15
WREOS
GPIBCS
GPIBWR
PDTACK
RDSW
GVMEO
OWNL
LOS0
DEVSEL
LWRITE
ACK0
ASD
DTC
LA5
LA4
STRT
ASDMA
FROMVME
GVME
HTBYTE
BERRV
DTACKV
OCS
LOS1
LA3
LA2
CLK16
+5V


U1
68450
RN4
RN7
RN7
RN3
+5V
+5V
FC0
FC1
FC2
COMPCONTROL B.V.
EINDHOVEN-HOLLAND
COMPANY
TITLE
68450 DMAC
MODULE
CC91
SHEET
08
OF
12
DATE
2-21-86
REV.
1.0


COMPCONTROL B.V.
EINDHOVEN-HOLLAND
COMPANY
GPIBbus CONNECTOR
POWER SUPPLY's
TITLE
CC91
MODULE
SHEET 09 OF 12
DATE
2-21-86
REV.
1.0
+5V
PUP
74LS14
74S38
IEEE 488
P3
IEC 625
P4
DIO1
DIO2
DIO3
DIO4
DIO5
DIO6
DIO7
DIO8
EOI
DAV
NRFD
NDAC
IFC
SRQ
ATN
REN
GND

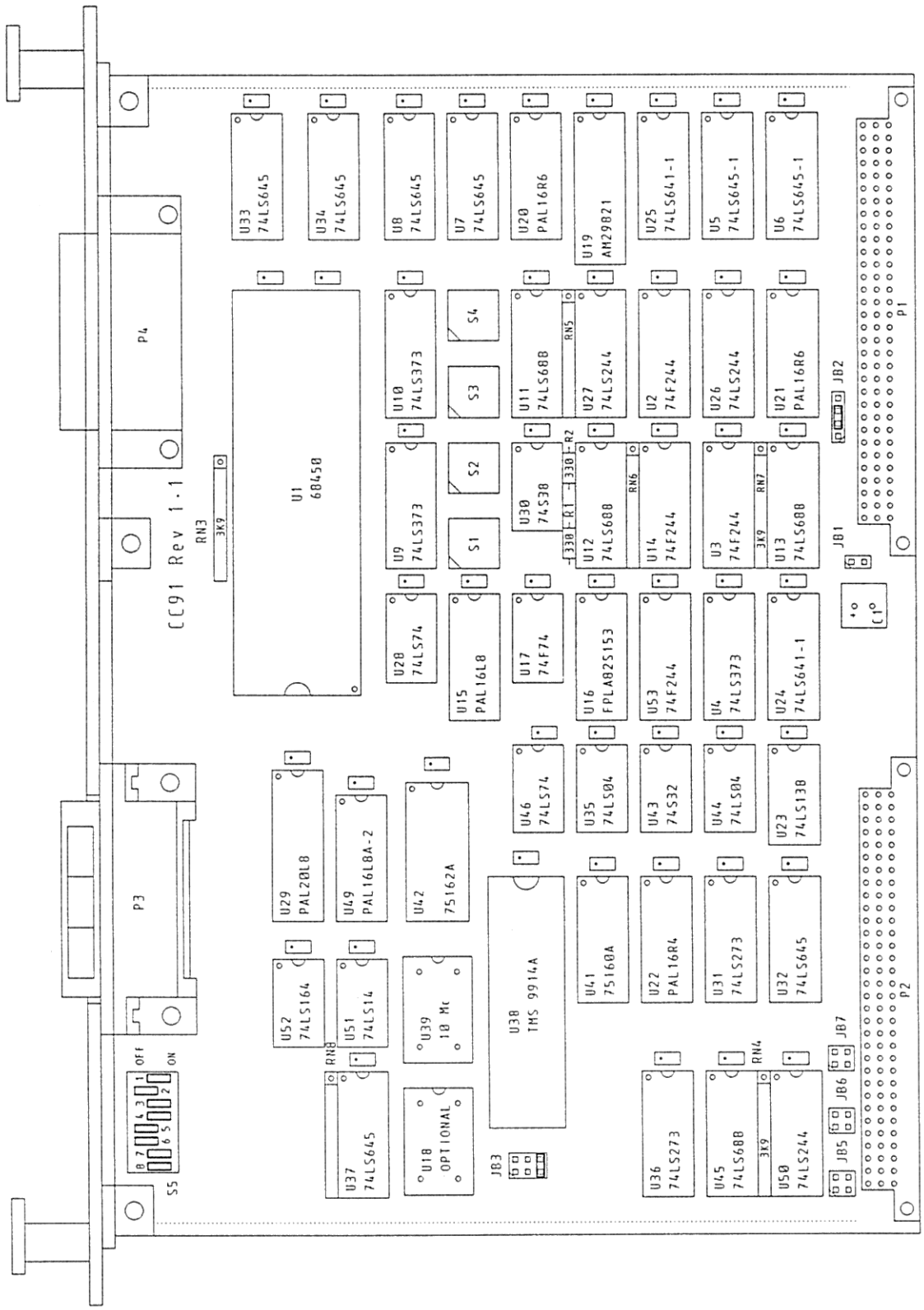

+5V
C1
C6
C7
C8
C9
C10
C11
C12
C13
C14
C15
C16
C17
C18
C19
C20
C21
C22
C23
C24
C25
C26
C27
C28
C29
C30
C31
C32
C33
C34
C35
C36
C37
C38
C39
C40
C41
C42
C43
C44
C45
C46
C47
C48
C49
C50
C51
C52
C53
C54
C55
C56
C57
C58
C59
C60
COMPCONTROL B.V.
EINDHOVEN-HOLLAND
DECOUPLING
CC91
REV.
1.0
DATE
2-21-86
SHEET 10 OF 12


P1A
1 D00
2 D01
3 D02
4 D03
5 D04
6 D05
7 D06
8 D07
9 GND
10 SYSCLK
11 GND
12 DS1*
13 DS0*
14 WRITE*
15 GND
16 DTACK*
17 GND
18 AS*
19 GND
20 IACK*
21 IACKIN*
22 IACKOUT*
23 AM4
24 A07
25 A06
26 A05
27 A04
28 A03
29 A02
30 A01
31 -12V
32 +5V
P1B
1 BBSY*
2 BCLR*
3 ACFAIL*
4 BG0IN*
5 BG0OUT*
6 BG1IN*
7 BG1OUT*
8 BG2IN*
9 BG2OUT*
10 BG3IN*
11 BG3OUT*
12 BR0*
13 BR1*
14 BR2*
15 BR3*
16 AM0
17 AM1
18 AM2
19 AM3
20 GND
21 SERCLK
22 SERDAT*
23 GND
24 IRQ7*
25 IRQ6*
26 IRQ5*
27 IRQ4*
28 IRQ3*
29 IRQ2*
30 IRQ1*
31 +5V STDBY
32 +5V
P1C
1 D08
2 D09
3 D10
4 D11
5 D12
6 D13
7 D14
8 D15
9 GND
10 SYSFAIL*
11 BERR*
12 SYSRES*
13 LWORD*
14 AM5
15 A23
16 A22
17 A21
18 A20
19 A19
20 A18
21 A17
22 A16
23 A15
24 A14
25 A13
26 A12
27 A11
28 A10
29 A09
30 A08
31 +12V
32 +5V
REV.
1.0
DATE
2-21-86
COMPANY
COMPCONTROL B.V.
EINDHOVEN-HOLLAND
TITLE
VMEbus CONNECTOR P1
MODULE
CC91
SHEET
11
OF
12


P2A
1 D105
2 D106
3 D107
4 D108
5 REN
6 GND
7 GND
8 GND
9 GND
10 GND
11 GND
12 GND
13 D105
14 D106
15 D107
16 D108
17 GND
18 GND
19 GND
20 GND
21 GND
22 GND
23 GND
24 GND
25
26 +5V
27 EREQ2
28 EACK2
29 EPCL2
30 EDONE
31 GND
32
P2B
1 +5V
2 GND
3
4
5
6
7
8
9
10
11
12 GND
13 +5V
14
15
16
17
18
19
20
21
22 GND
23
24
25
26
27
28
29
30
31 GND
32 +5V
P2C
1 D101
2 D102
3 D103
4 D104
5 EOI
6 DAV
7 NRFD
8 NDAC
9 IFC
10 SRQ
11 ATN
12 GND
13 D101
14 D102
15 D103
16 D104
17 REN
18 EOI
19 DAV
20 NRFD
21 NDAC
22 IFC
23 SRQ
24 ATN
25 GND
26 +5V
27 EREQ3
28 EACK3
29 EPCL3
30 EDTC
31 GND
32
REV.
1.0
DATE
2-21-86
COMPANY
COMPCONTROL B.V.
EINDHOVEN-HOLLAND
TITLE
VMEbus CONNECTOR P2
MODULE
CC91
SHEET
12
OF
12

APPENDIX C

COMPONENT LAYOUT



APPENDIX D

LIST OF COMPONENTS

INTEGRATED CIRCUITS

U1	68450
U2,U3	74S244, AS244, F244
U4	74LS373
U5,U6	74LS645-1
U7,U8	74ALS645A
U9,U10	74LS373
U11-U13	74LS688, AMD2521
U14	74S244, AS244, F244
U15	PAL16L8A-2
U16	IFL82S153
U17	74F74
U18	XTAL Oscillator (optional)
U19	AM29821, 74AS821
U20,U21	PAL16R6A-2
U22	PAL16R4A-2
U23	74LS138
U24,U25	74LS641-1
U26,U27	74LS244, ALS244
U28	74LS74
U29	PAL20L8A-2
U30	74S38
U31	74LS273
U32-U34	74LS645
U35	74LS04
U36	74LS273
U37	74LS645
U38	TMS9914A
U39	XTAL Oscillator (10 Mc)
U41	75LS160A
U42	75LS162A
U43	74S32
U44	74LS04
U45	74LS688
U46	74LS74
U49	PAL16L8A-2
U50	74LS244, ALS244
U51	74LS14
U52	74LS164
U53	74F244

RESISTOR NETWORKS

RN3-RN8	3.9 K 10-Pin SIP

MISCELLANEOUS

R1,R2	Resistor 330 Ohm
C1	Elco 47uF 16V
C5	Capacitor 220 pF
O6-O60	Decoupling Capacitor 0.1 uF
S1-S4	Binary Coded Hex Switch
S5	Dil Switch
P1	96-Pin DIN41612 Connector
P2	96-Pin DIN41612 Connector
P3	24-Pin IEEE-488 Connector
P4	25-Pin IEC-625 Connector
JB1	2-Pin Jumper Block (1 X 2)
JB2	4-Pin Jumper Block (1 X 4)
JB3	6-Pin Jumper Block (2 X 3)
JB5	4-Pin Jumper Block (2 X 2)
JB6	4-Pin Jumper Block (2 X 2)
JB7	4-Pin Jumper Block (2 X 2)

NOTE: Some parts may have been replaced by their equivalent types.

APPENDIX E

CONNECTOR PIN ASSIGNMENTS

Pin Assignments P1 CC91

PIN NUMBER	ROW A SIGNAL MNEMONIC	ROW B SIGNAL MNEMONIC	ROW C SIGNAL MNEMONIC
1	D00	BBSY*	D08
2	D01	BCLR*	D09
3	D02		D10
4	D03	BG0IN*	D11
5	D04	BG0OUT*	D12
6	D05	BG1IN*	D13
7	D06	BG1OUT*	D14
8	D07	BG2IN*	D15
9	GND	BG2OUT*	GND
10	SYSCLK	BG3IN*	
11	GND	BG3OUT*	BERR*
12	DS1*	BR0*	SYSRESET*
13	DS0*	BR1*	
14	WRITE*	BR2*	AM5
15	GND	BR3*	A23
16	DTACK*	AM0	A22
17	GND	AM1	A21
18	AS*	AM2	A20
19	GND	AM3	A19
20	IACK*	GND	A18
21	IACKIN*		A17
22	IACKOUT*		A16
23	AM4	GND	A15
24	A07	IRQ7*	A14
25	A06	IRQ6*	A13
26	A05	IRQ5*	A12
27	A04	IRQ4*	A11
28	A03	IRQ3*	A10
29	A02	IRQ2*	A09
30	A01	IRQ1*	A08
31			
32	+5 Volts	+5 Volts	+5 Volts

Pin Assignments P2 CC91

PIN NUMBER	ROW A SIGNAL MNEMONIC	ROW B SIGNAL MNEMONIC	ROW C SIGNAL MNEMONIC
1	DIO5	+5 Volts	DIO1
2	DIO6	GND	DIO2
3	DIO7		DIO3
4	DIO8		DIO4
5	REN		EOI
6	GND		DAV
7	GND		NRFD
8	GND		NDAC
9	GND		IFC
10	GND		SRQ
11	GND		ATN
12	GND	GND	GND
13	DIO5	+5 Volts	DIO1
14	DIO6		DIO2
15	DIO7		DIO3
16	DIO8		DIO4
17	GND		REN
18	GND		EOI
19	GND		DAV
20	GND		NRFD
21	GND		NDAC
22	GND	GND	IFC
23	GND		SRQ
24	GND		ATN
25			GND
26	+5 Volts		+5 volts
27	EREQ2*		EREQ3*
28	EACK2*		EACK3*
29	EPCL2*		EPCL3*
30	EDONE*		EDTC*
31	GND	GND	GND
32		+5 Volts	

Pin Assignments P3 CC91
IEEE-488 Connector

PIN NUMBER	SIGNAL MNEMONIC	PIN NUMBER	SIGNAL MNEMONIC
1	DIO1	13	DIO5
2	DIO2	14	DIO6
3	DIO3	15	DIO7
4	DIO4	16	DIO8
5	EOI	17	REN
6	DAV	18	GND
7	NRFD	19	GND
8	NDAC	20	GND
9	IFC	21	GND
10	SRQ	22	GND
11	ATN	23	GND
12	GND	24	GND

Pin Assignments P4 CC91
IEC-625 Connector

PIN NUMBER	SIGNAL MNEMONIC	PIN NUMBER	SIGNAL MNEMONIC
1	DIO1	14	DIO5
2	DIO2	15	DIO6
3	DIO3	16	DIO7
4	DIO4	17	DIO8
5	REN	18	GND
6	EOI	19	GND
7	DAV	20	GND
8	NRFD	21	GND
9	NDAC	22	GND
10	IFC	23	GND
11	SRQ	24	GND
12	ATN	25	GND
13	GND		

APPENDIX F

MEMORY MAP

All addresses are offset from the hardware installed base address of the CC91 module.

$000 $0FF	DMA CONTROLLER REGISTERS	
$100 $11F	CONTROL REGISTER	one 8-bit register, 16 times duplicated at odd addresses.
$120 $12F	EOS REGISTER	one 8-bit register, 8 times duplicated at odd addresses.
$130 $13F	GPIB CONTROLLER REGISTERS	eight 8-bit registers at odd addresses.
$140 $15F	CONTROL REGISTER	
$160 $16F	EOS REGISTER	
$170 $17F	GPIB CONTROLLER REGISTERS	
$180 $19F	CONTROL REGISTER	
$1A0 $1AF	EOS REGISTER	
$1B0 $1BF	GPIB CONTROLLER REGISTERS	
$1C0 $1DF	CONTROL REGISTER	
$1E0 $1EF	EOS REGISTER	
$1F0 $1FF	GPIB CONTROLLER REGISTERS	

APPENDIX G

DMA CONTROLLER REGISTERS

$00	CHANNEL 0	$01
$3E		$3F
$40	CHANNEL 1	$41
$7E		$7F
$80	CHANNEL 2	$81
$BE		$BF
$C0	CHANNEL 3	$C1
$FE	GCR	$FF

Note: The General Control Register only resides at address $FF

The following table shows the register arrangement of channel 0. Each register can be accessed by byte or word. However when the STR bit in the CCR is set, only byte access is possible.

	high	low	
$00	CSR 0	CER 0	$01
$02			$03
$04	DCR 0	OCR 0	$05
$06	SCR 0	CCR 0	$07
$08			$09
$0A	MTC 0		$0B

Table continued on next page

DMAC Register Table continued

$0C	MAR 0 (H)	$0D
$0E	MAR 0 (L)	$0F
$10		$11
$12		$13
$14	DAR 0 (H)	$15
$16	DAR 0 (L)	$17
$18		$19
$1A	BTC 0	$1B
$1C	BAR 0 (H)	$1D
$1E	BAR 0 (L)	$1F
$20		$21
$22		$23
$24	NIV 0	$25
$26	EIV 0	$27
$28	MFC 0	$29
$2A		$2B
$2C	CPR 0	$2D
$2E		$2F
$30	DFC 0	$31
$32		$33
$34		$35
$36		$37
$38	BFC 0	$39
$3A		$3B
$3C		$3D
$3E		$3F

DMA CONTROLLER REGISTER DEFINITIONS

CSR	Channel Status Register
CER	Channel Error Register
DCR	Device Control Register
OCR	Operation Control Register
SCR	Sequence Control Register
CCR	Channel Control Register
NIV	Normal Interrupt Vector
EIV	Error Interrupt vector
CPR	Channel Priority Register
MFC	Memory Function Codes
DFC	Device Function Codes
BFC	Base Function Codes
MTC	Memory Transfer Counter
BTC	Base Transfer Counter
MAR	Memory Address Register
DAR	Device Address Register
BAR	Base Address Register
GCR	General Control Register

APPENDIX H

GPIB CONTROLLER REGISTERS

All addresses are offset from the base address of the CC91 module.

READ REGISTERS

$130		Int Status 0	$131
		Int Status 1	$133
		Addr Status	$135
		Bus Status	$137
		Addr Switch	$139
			$13B
		Cmd Pass Thru	$13D
		Data In	$13F

WRITE REGISTERS

$130		Int Mask 0	$131
		Int Mask 1	$133
			$135
		Auxiliary Cmd	$137
		Address	$139
		Serial Poll	$13B
		Parallel Poll	$13D
		Data Out	$13F

APPENDIX I

CONTROL REGISTER

```
                CONTROL REGISTER (bit)

                7  6  5  4  3  2  1  0
                -----------------------
$101           |  |  |  |  |  |  |  |  |
                -----------------------
                |  |  |  |  |  |  |  |
                |  |  |  |  |  |  |  ---   IRQLV0
                |  |  |  |  |  |  ------   IRQLV1    IRQ LEVEL
                |  |  |  |  |  ---------   IRQLV2
                |  |  |  |  ------------   ENIRQ     IRQ ENABLE
                |  |  |  ---------------   REQLV0    REQUEST LEVEL
                |  |  ------------------   REQLV1
                |  ---------------------   spare
                ------------------------   LRST      GPIB CONTR RESET
```

APPENDIX J

CC91 PROGRAM EXAMPLE

*** PROGRAM TO BE SUPPLIED ***

APPENDIX K

DATA SHEET 68450 DMAC

Hitachi Microcomputer Device Technical Information
September, 1982

Revision of HD68450 Data Sheet

Content

Page

1 Replace the mnemonic for pin 21, "$\overline{IREQ}$" with "$\overline{IRQ}$".

2 Revise the value of "Power Dissipation".

Before:	Revised:
1.0W typ., 1.75W max.	1.4W typ., 2.0W max. at Ta = 25 degrees C and V_{CC} = 5.0V

3 Revise the following items of "AC Electrical Specification".
No. 94 and 95 are newly added.

No.	Item	Symbol	Revised Value (ns)					
			4MHz		6MHz		8MHz	
			min	max	min	max	min	max
16	$\overline{DS}$ In "High" to $\overline{DDIR}$ "High" Impedance	t DSHDRZ		160		140		120
17	$\overline{DS}$ In "High" to $\overline{DBEN}$ "High" Impedance	t DSHDBZ		160		140		120
18	Clock "High" to Data Out Valid (MPU Read)	t CHDVM		240		170		130
21	$\overline{DS}$ In "High" to $\overline{DTACK}$ "High"	t DSHDTH		140		130		110
34	Clock "High" to Address/FC Valid	t CHAV		160		140		120
35	Clock "High" to Address/FC/ Data "High" Impedance	t CHAZx		140		120		100
94	Address In to $\overline{AS}$ In "Low"	t AIASL	0		0		0	
95	$\overline{AS}$, $\overline{DS}$ In "High" to Address Invalid	t SIHAIV	0		0		0	

5 Ignore indices "91" and "92" in "Figure 3, AC Electrical Waveforms-MPU Read/Write".

5 Replace index "25" with "23" in "Figure 3, AC Electrical Waveforms-MPU Read/Write".

14 Delete the following sentence from the paragraph titled "$\overline{PCL}$ AS AN ABORT INPUT":
When this function has been programmed, the $\overline{PCL}$ line is only active after the channel has been started.

35 Add the following comment to "Figure 37, Bus Exception Flow Diagram":
When the DMAC is in "IDLE MODE WAITING FOR $\overline{BEC}$ CLEAR", (BER or RTY is asserted in this state) the MPU cannot access the DMAC registers. BER or RTY must be negated before the MPU accesses the DMAC. The MPU can access the DMAC registers while HLT or RRT is asserted.

40 Add the following sentence after the 6th line from the top of the right column (ending with"... and is restarted."):
In order to detect an external abort of the above kind, the user is advised to examine the PCT bit of the channel status register (CSR) along with the ERR bit. If the PCT bit is set and ERR bit is reset, then he can conclude that an external abort has occured in the DONE cycle and take an appropreate action.

40 Replace the content of the section titled "MULTIPLE ERRORS" with the following sentence:
The DMAC detects and services multiple errors, however the content of CER (channel error register) retains the first error that the channel has encountered regardless of what type of errors occur after the first one.

41 The following change has to be made on "Figure 43 An Example of Connection with Peripheral Devices in 6800 Mode":
The AND gate input to $\overline{OE}$ of 74LS373 should be changed to a NAND gate.
Change "$A_0 \sim A_{23}$" to "$A_1 \sim A_{23}$"
Change "$D_0 \sim D_7$" to "$D_0 \sim D_7$"

42 Replace HD 68450 with HD 68000
MPU MPU
(Lower left block of page)

HITACHI

HITACHI-EUROPE MÜNCHEN, HEADQUARTER
Hitachi Electronic Components Europe GmbH
Hans-Pinsel-Straße 3
8013 Haar b. München
☎ (089) 4614-0
Telex: 5-22593
Telefax: (089) 463151

HITACHI-EUROPE DÜSSELDORF
Hitachi Electronic Components Europe GmbH
Königsallee 6
4000 Düsseldorf 1
☎ (0211) 84995
Telex: 8-584536
Telefax: (0211) 324612

HITACHI-EUROPE STUTTGART
Hitachi Electronic Components Europe GmbH
Fabrikstraße 17
7024 Filderstadt
☎ (0711) 772011
Telex: 7-255267

HITACHI-EUROPE PARIS
Hitachi Electronic Components Europe GmbH
Bureau de Représentation en France
95-101, Rue Charles-Michels
F-93200 Saint Denis
☎ 01-8216015
Télex: 611387
Téléfax: 01-2436997

HITACHI-EUROPE MILANO
Hitachi Electronic Componer
Europe GmbH
Via B. Davanzati, 27
I-20158 Milano
☎ 02-3763024
Telex: 320343
Telefax: 02-683730

HITACHI-UNITED KINGDOM LONDON, HEADQUARTER
Hitachi Electronic Components (UK) Ltd.
Hitec House,
221-225 Station-Road
Harrow, Middlesex, HA1 2XL
☎ 01-8611414
Telex: 936293
Telefax: 01-8636646

HITACHI-UNITED KINGDOM STOCKHOLM
Hitachi Electronic Components (UK) Ltd.
Box 1602, 16311 Spanga
Hankadalsgatan 10 Kista
☎ 0046-87510035
Telex: 14106 (Hitecst S)
Telefax: 08-7515073

Ä-12-82-06.UP450 Printed in West-Germany

HITACHI MICROCOMPUTER SYSTEM
HD68450-4, HD68450-6, HD68450-8
DMAC (Direct Memory Access Controller)

HITACHI

DIRECT MEMORY ACCESS CONTROLLER

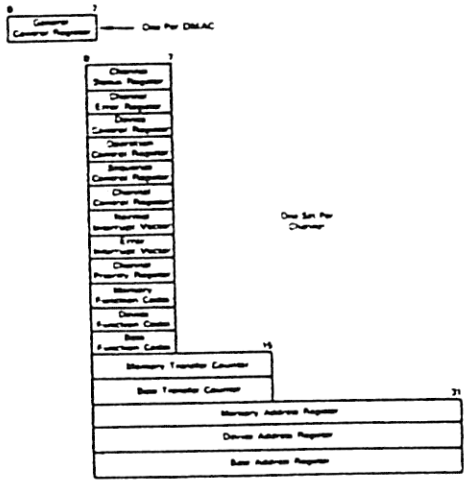
Microprocessor implemented systems are becoming increasingly complex, particularly with the advent of high-performance 16-bit MPU devices with large memory addressing capability. In order to maintain high throughput, large blocks of data must be moved within these systems in a quick, efficient manner with minimum intervention by the MPU itself.

The HD68450 Direct Memory Access Controller (DMAC) is designed specifically to complement the performance and architectural capabilities of the HD68000 MPU by providing the following features:

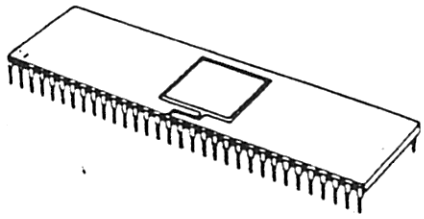
- HMCS68000 Bus Compatible
- 4 independent DMA Channels
- Memory-to-Memory, Memory-to-Device, Device-to-Memory Transfers
- MMU Compatible
- Array-Chained and Linked-Array-Chained Operations
- On-Chip Registers that allow Complete Software Control by the System MPU
- Interface Lines that Provide for Requesting, Acknowledging, and Incidental Control of the Peripheral Devices
- Transfers to/from HMCS68000 or HMCS6800 Peripherals
- Variable System Bus Bandwidth Utilization
- Programmable Channel Prioritization
- 2 Vectored interrupts for each Channel
- Auto-Request and External-Request Transfer Modes
- Up to 4 Megabytes/Second Transfer Rates
- +5 Volt Operation

The DMAC functions by transferring a series of operands (data) between memory and device; operand sizes can be byte, word, or long word. A block is a sequence of operations; the number of operands in a block is determined by a transfer count. A single-channel operation may involve the transfer of several blocks of data between memory and device.

■ DMAC ACCESSIBLE REGISTERS

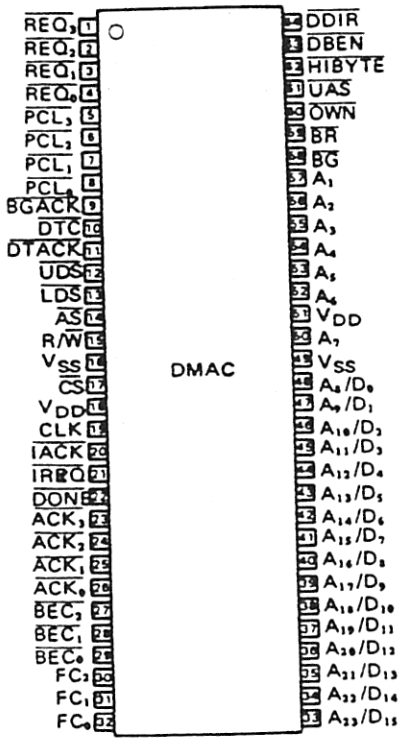


HD68450-4, HD68450-6, HD68450-8,



(DC-64)

■ PIN ARRANGEMENT



(Top View)

■ ABSOLUTE MAXIMUM RATINGS

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

* With respect to V_{SS} (SYSTEM GND)

(NOTE) Permanent LSI damage may occur if maximum ratings are exceeded. Normal operation should be under recommended operating conditions. If these conditions are exceeded, it could affect reliability of LSI.

■ RECOMMENDED OPERATING CONDITIONS

Item	Symbol	min	typ	max	Unit
Supply Voltage	V_{CC}*	4.75	5.0	5.25	V
Input Voltage	V_{IH}*	2.0	–	V_{CC}	V
	V_{IL}*	-0.3	–	0.8	V
Operating Temperature	T_{opr}	0	25	70	°C

* With respect to V_{SS} (SYSTEM GND)

■ ELECTRICAL CHARACTERISTICS

● DC CHARACTERISTICS (V_{CC} = 5V ±5%, V_{SS} = 0V, Ta = 0 ~ +70°C, unless otherwise noted.)

Item		Symbol	Test Condition	min	typ	max	Unit
Input "High" Voltage		V_{IH}		2.0	–	V_{CC}	V
Input "Low" Voltage		V_{IL}		V_{SS}-0.3	–	0.8	V
Input Leakage Current	$\overline{CS}$, $\overline{IACK}$, $\overline{BG}$, CLK, $\overline{BEC_0}$ ~ $\overline{BEC_2}$, $\overline{REQ_0}$ ~ $\overline{REQ_3}$	I_{in}		–	–	10	μA
Three-State (Off State) Input Current	A_1 ~ A_7, D_0 ~ D_{15}/A_8 ~ A_{23}, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{UAS}$, $\overline{DTACK}$, $\overline{BGACK}$, $\overline{OWN}$, $\overline{DTC}$, $\overline{HIBYTE}$, $\overline{DDIR}$, $\overline{DBEN}$, FC_0 ~ FC_2	I_{TSI}		–	–	10	μA
Open Drain (Off State) Input Current	$\overline{IREQ}$, $\overline{DONE}$	I_{ODI}		–	–	20	μA
Output "High" Voltage	A_1 ~ A_7, D_0 ~ D_{15}/A_8 ~ A_{23}, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{UAS}$, $\overline{DTACK}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{OWN}$, $\overline{DTC}$, $\overline{HIBYTE}$, $\overline{DDIR}$, $\overline{DBEN}$, $\overline{ACK_0}$ ~ $\overline{ACK_3}$, $\overline{PCL_0}$ ~ $\overline{PCL_3}$, FC_0 ~ FC_2	V_{OH}	I_{OH} = -400 μA	2.4	–	–	V
Output "Low" Voltage	A_1 ~ A_7, FC_0 ~ FC_2	V_{OL}	I_{OL} = 3.2 mA	–	–	0.5	V
	D_0 ~ D_{15}/A_8 ~ A_{23}, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, R/$\overline{W}$, $\overline{DTACK}$, $\overline{BR}$, $\overline{OWN}$, $\overline{DTC}$, $\overline{HIBYTE}$, $\overline{DDIR}$, $\overline{DBEN}$, $\overline{ACK_0}$ ~ $\overline{ACK_3}$, $\overline{UAS}$, $\overline{PCL_0}$ ~ $\overline{PCL_3}$, $\overline{BGACK}$,	V_{OL}	I_{OL} = 5.3 mA	–	–	0.5	
	$\overline{IREQ}$, $\overline{DONE}$	V_{OL}	I_{OL} = 8.9 mA	–	–	0.5	
Power Dissination		P_D	f = 8 MHz	–	1.0	1.75	W
Capacitance		C_{in}	V_{in} = 0V, Ta = 25°C, f = 1 MHz	–	–	15	pF

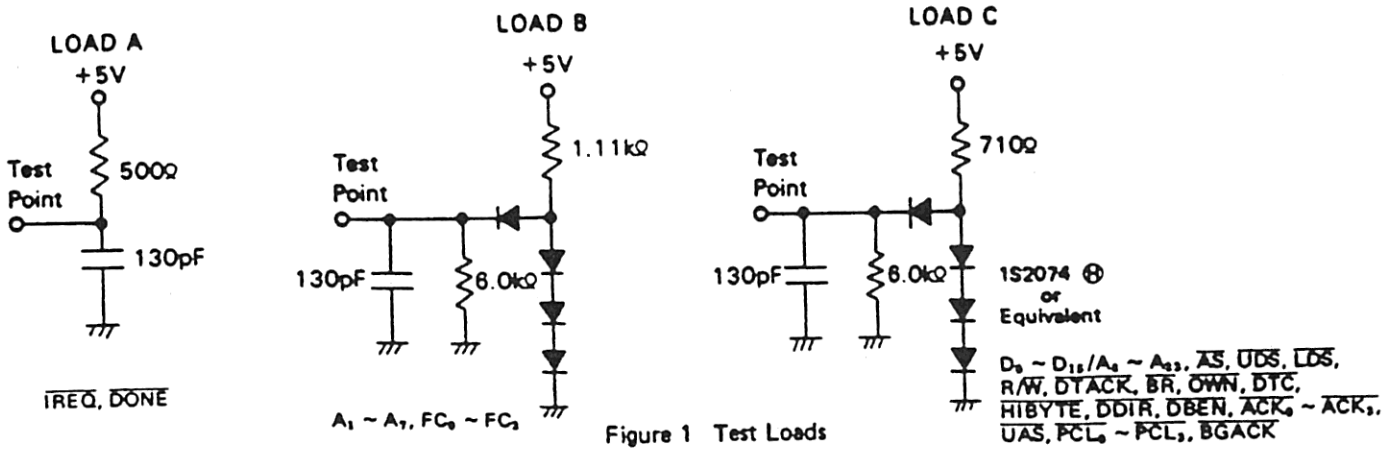


Figure 1 Test Loads

● **AC ELECTRICAL SPECIFICATION (V_{CC} = 5.0V ±5%, V_{SS} = 0V, T_a = 0 ~ +70°C)**

No.	Item	Symbol	Test Condition	4 MHz		6 MHz		8 MHz		Unit
				min	max	min	max	min	max	
	Frequency of Operation	f		2	4	2	6	2	8	MHz
1	Clock Period	t_{cyc}		250	500	167	500	125	500	ns
2	Clock Width "Low"	t_{CL}		115	250	75	250	55	250	ns
3	Clock Width "High"	t_{CH}		115	250	75	250	55	250	ns
4	Clock Fall Time	t_{Cf}		–	10	–	10	–	10	ns
5	Clock Rise Time	t_{Cr}		–	10	–	10	–	10	ns
6	Asynchronous Input Setup Time	t_{ASI}		30	–	25	–	20	–	ns
7	Data In to $\overline{DS}$ In "Low"	t_{DIDSL}		0	–	0	–	0	–	ns
8	Data In to Clock "Low" (Setup Time)	t_{DICL}		30	–	25	–	15	–	ns
9	$\overline{DS}$ In "High" to Data In Valid	t_{DSHDI}		0	–	0	–	0	–	ns
10	Clock "High" to $\overline{DDIR}$ "High" Impedance Off	t_{CHDRZO}		–	120	–	100	–	80	ns
11	Clock "High" to $\overline{DBEN}$ "High" Impedance Off	t_{CHDBZO}		–	120	–	100	–	80	ns
12	Clock "High" to $\overline{DDIR}$ "Low" (MPU Write)	t_{CHDRLM}		–	90	–	80	–	70	ns
13	Clock "High" to $\overline{DDIR}$ "High" (MPU Write)	t_{CHDRHM}		–	90	–	80	–	70	ns
14	Clock "Low" to $\overline{DBEN}$ "Low (MPU Cycle)	t_{CLDBLM}		–	90	–	80	–	70	ns
15	Clock "Low" to $\overline{DBEN}$ "High" (MPU Cycle)	t_{CLDBHM}		–	90	–	80	–	70	ns
16	$\overline{DS}$ In "High" to $\overline{DDIR}$ "High" Impedance	t_{DSHDRZ}		–	120	–	100	–	80	ns
17	$\overline{DS}$ In "High" to $\overline{DBEN}$ "High" Impedance	t_{DSHDBZ}		–	120	–	100	–	80	ns
18	Clock "High" to Data Out Valid (MPU Read)	t_{CHDVM}		–	140	–	120	–	100	ns
19	$\overline{DS}$ In "High" to Data "High" Impedance	t_{DSHDZ}	Fig. 2 ~ 6	–	160	–	140	–	120	ns
20	Clock "Low" to $\overline{DTACK}$ "Low"	t_{CLDTL}		–	90	–	80	–	70	ns
21	$\overline{DS}$ In "High" to $\overline{DTACK}$ "High"	t_{DSHDTH}		–	90	–	80	–	70	ns
22	$\overline{DS}$ In "High" to $\overline{DTACK}$ "High" Impedance	t_{DSHDTZ}		–	220	–	200	–	180	ns
23	$\overline{DTACK}$ "Low" to $\overline{DS}$ In "High"	t_{DTLDSH}		0	–	0	–	0	–	ns
24	$\overline{REQ}$ Width "Low"	t_{REQL}		2.0	–	2.0	–	2.0	–	Clk. Per.
25	Clock "High" to $\overline{BR}$ "Low"	t_{CHBRL}		–	90	–	80	–	70	ns
26	Clock "High" to $\overline{BR}$ "High"	t_{CHBRH}		–	90	–	80	–	70	ns
27	$\overline{BR}$ "Low" to $\overline{BG}$ "Low"	t_{BRLBGL}		0	–	0	–	0	–	ns
28	$\overline{BR}$ "Low" to MPU Cycle End ($\overline{AS}$ In "High")	t_{BRLASH}		0	–	0	–	0	–	ns
29	MPU Cycle End ($\overline{AS}$ In "High") to $\overline{BGACK}$ "Low"	t_{ASHBL}		4.5	5.5	4.5	5.5	4.5	5.5	Clk. Per.
30	Clock "High" to $\overline{BGACK}$ "Low"	t_{CHBL}		–	90	–	80	–	70	ns
31	Clock "High" to $\overline{BGACK}$ "High"	t_{CHBH}		–	90	–	80	–	70	ns
32	$\overline{REQ}$ "Low" to $\overline{BGACK}$ "Low"	t_{REQLBL}		12.0	–	12.0	–	12.0	–	Clk. Per.
33	Clock "Low" to $\overline{BGACK}$ "High" Impedance	t_{CLBZ}		–	120	–	100	–	80	ns
34	Clock "High" to Address/FC Valid	t_{CHAV}		–	140	–	120	–	120	ns
35	Clock "High" to Address/FC/Data "High" Impedance	t_{CHAZx}		–	120	–	100	–	80	ns
36	Clock "High" to Address/FC/ Data Invalid	t_{CHAZn}		0	–	0	–	0	–	ns
37	Clock "Low" to Address "High" Impedance	t_{CLAZ}		–	120	–	100	–	80	ns
38	$\overline{UAS}$ "High" to Address Invalid	t_{UHAI}		50	–	40	–	30	–	ns

(to be continued)

No.	Item	Symbol	Test Condition	4 MHz		6 MHz		8 MHz		Unit
				min	max	min	max	min	max	
39	$\overline{AS}$, $\overline{DS}$ "High" to Address/FC/Data Invalid	t_{SHAZ}		50	–	40	–	30	–	ns
40	Address/FC Valid to $\overline{AS}$, $\overline{DS}$ "Low" (Read)	t_{AVSL}		50	–	40	–	30	–	ns
41	Clock "High" to $\overline{UAS}$ "Low"	t_{CHUL}		–	90	–	80	–	70	ns
42	Clock "High" to $\overline{UAS}$ "High"	t_{CHUH}		–	90	–	80	–	70	ns
43	Clock "Low" to $\overline{UAS}$ "High" Impedance	t_{CLUZ}		–	120	–	100	–	80	ns
44	Clock "High" to $\overline{AS}$, $\overline{DS}$ "Low"	t_{CHSL}		–	80	–	70	–	60	ns
45	Clock "Low" to $\overline{AS}$, $\overline{DS}$ "High"	t_{CLSH}		–	90	–	80	–	70	ns
46	Clock "Low" to $\overline{AS}$, $\overline{DS}$ "High" Impedance	t_{CLSZ}		–	120	–	100	–	80	ns
47	Clock "Low" to $\overline{DS}$ "Low" (Write)	t_{CLDSL}		–	80	–	70	–	60	ns
48	$\overline{AS}$ Width "Low"	t_{ASL}		545	–	350	–	255	–	ns
49	$\overline{DS}$ Width "Low"	t_{DSL}		420	–	265	–	190	–	ns
50	$\overline{DS}$ "High" to R/$\overline{W}$ "High"	t_{SHRH}		60	–	50	–	40	–	ns
51	Clock "High" to R/$\overline{W}$ "Low"	t_{CHRL}		–	90	–	80	–	70	ns
52	Clock "High" to R/$\overline{W}$ "High"	t_{CHRH}		–	90	–	80	–	70	ns
53	Clock "Low" to R/$\overline{W}$ "High" Impedance	t_{CLRZ}		–	120	–	100	–	80	ns
54	Address/FC Valid to R/$\overline{W}$ "Low"	t_{AVRL}		110	–	50	–	25	–	ns
55	R/$\overline{W}$ "Low" to $\overline{DS}$ "Low" (Write)	t_{RLSL}		285	–	170	–	120	–	ns
56	Clock "Low" to $\overline{OWN}$ "Low"	t_{CLOL}		–	90	–	80	–	70	ns
57	Clock "Low" to $\overline{OWN}$ "High"	t_{CLOH}		–	90	–	80	–	70	ns
58	Clock "High" to $\overline{OWN}$ "High" Impedance	t_{CHOZ}		–	120	–	100	–	80	ns
59	Clock "High to $\overline{DDIR}$ "Low" (Read)	t_{CHDRL}		–	90	–	80	–	70	ns
60	Clock "High" to $\overline{DDIR}$ "High"	t_{CHDRH}		–	90	–	80	–	70	ns
61	Clock "Low" to $\overline{DDIR}$ "High" Impedance	t_{CLDRZ}		–	120	–	100	–	80	ns
62	Clock "Low" to $\overline{DBEN}$ "Low"	t_{CLDBL}		–	90	–	80	–	70	ns
63	Clock "Low" to $\overline{DBEN}$ "High"	t_{CLDBH}		–	90	–	80	–	70	ns
64	Clock "Low" to $\overline{DBEN}$ "High" Impedance	t_{CLDBZ}	Fig. 2 ~ 6	–	120	–	100	–	80	ns
65	Clock "High" to $\overline{HIBYTE}$ "Low"	t_{CHHIL}		–	90	–	80	–	70	ns
66	Clock "High" to $\overline{HIBYTE}$ "High"	t_{CHHIH}		–	90	–	80	–	70	ns
67	Clock "Low" to $\overline{HIBYTE}$ "Low"	t_{CLHIL}		–	90	–	80	–	70	ns
68	Clock "Low" to $\overline{HIBYTE}$ "High" Impedance	t_{CLHIZ}		–	120	–	100	–	80	ns
69	Clock "High" to $\overline{ACK}$ "Low"	t_{CHACL}		–	90	–	80	–	70	ns
70	Clock "High" to $\overline{ACK}$ "High"	t_{CHACH}		–	90	–	80	–	70	ns
71	Clock "Low" to $\overline{ACK}$ "Low"	t_{CLACL}		–	90	–	80	–	70	ns
75	Clock "High" to $\overline{DTC}$ "Low"	t_{CHDTL}		–	90	–	80	–	70	ns
76	Clock "High" to $\overline{DTC}$ "High"	t_{CHDTH}		–	90	–	80	–	70	ns
77	Clock "Low" to $\overline{DTC}$ "High" Impedance	t_{CLDTZ}		–	120	–	100	–	80	ns
78	$\overline{DTC}$ Width "Low"	t_{DTCL}		1.0	–	1.0	–	1.0	–	Clk. Per.
81	Clock "Low" to $\overline{PCL}$ "Low" (1/8 Clock)	t_{CLPL}		–	90	–	80	–	70	ns
82	Clock "Low" to $\overline{PCL}$ "High" (1/8 Clock)	t_{CLPH}		–	90	–	80	–	70	ns
83	$\overline{PCL}$ Width "Low" (1/8 Clock)	t_{PCLL}		4.0	–	4.0	–	4.0	–	Clk. Per.
84	$\overline{DTACK}$ "Low" to Data In (Setup Time)	t_{DALDI}		–	180	–	120	–	90	ns
85	Data In to Clock "Low" (Setup Time)	t_{DICL}		30	–	25	–	15	–	ns
86	$\overline{DS}$ "High" to Data Invalid (Hold Time)	t_{SHDI}		0	–	0	–	0	–	ns
87	$\overline{DS}$ "High" to $\overline{DTACK}$ "High"	t_{SHDAH}		0	240	0	160	0	120	ns
88	$\overline{BEC}$ "Low" to $\overline{DTACK}$ "Low"	t_{BECDAL}		50	–	50	–	50	–	ns
89	$\overline{BEC}$ Width "Low"	t_{BECL}		2.0	–	2.0	–	2.0	–	Clk. Per.
90	$\overline{OWN}$ "Low" to $\overline{UAS}$ "Low"			50	–	40	–	30	–	ns
91	$\overline{DDIR}$ "Low" to $\overline{DBEN}$ "Low"			50	–	40	–	30	–	ns
92	$\overline{DBEN}$ "High" to $\overline{DDIR}$ "High"			50	–	40	–	30	–	ns
93	$\overline{DTACK}$ Width "High"	t_{DTH}		10	–	10	–	10	–	ns

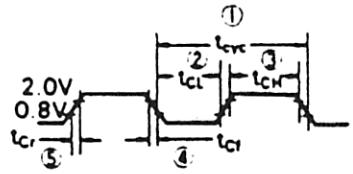


Figure 2 Input Clock Waveform

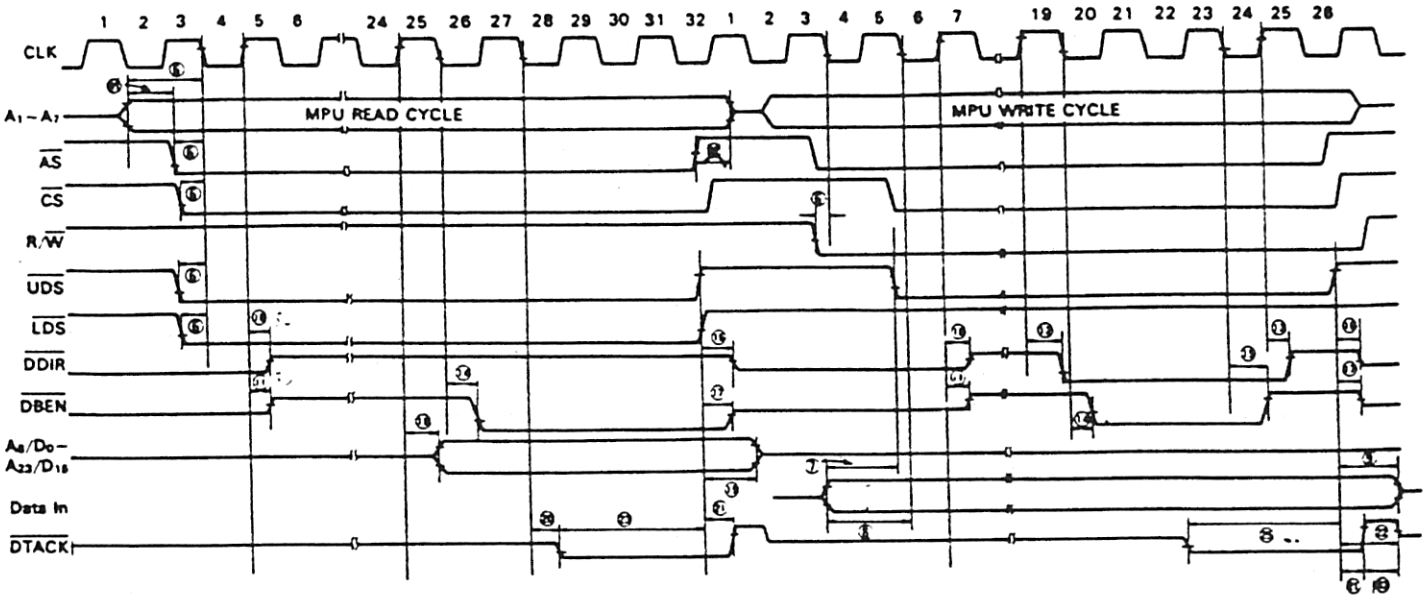
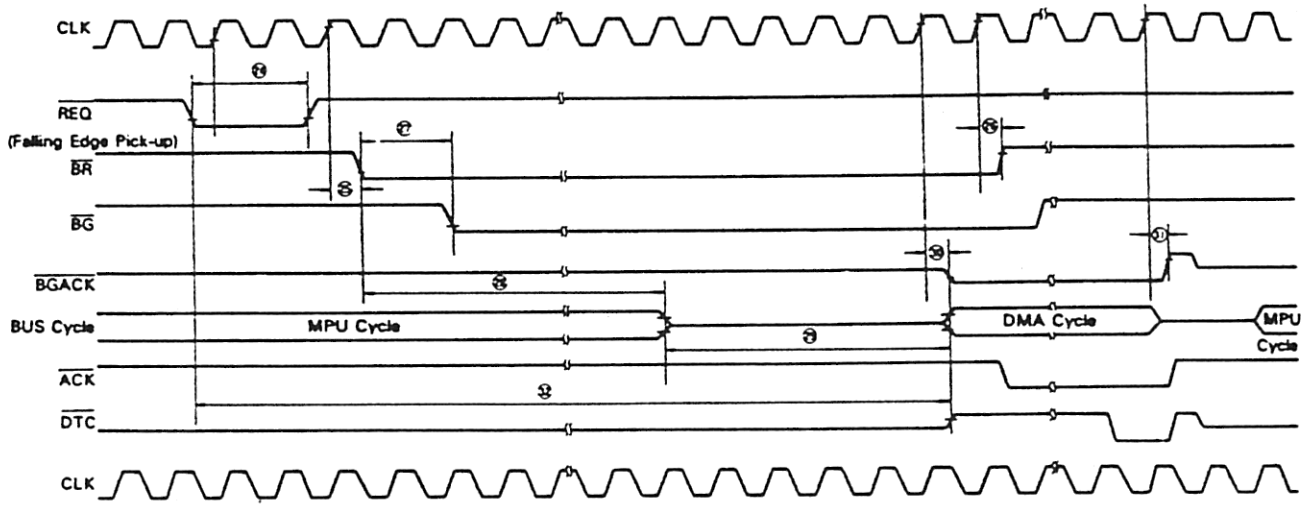


Figure 3 AC Electrical Waveforms – MPU Read/Write



(NOTES) 1) Setup time for the asynchronous inputs $\overline{BG}$, $\overline{BGACK}$, $\overline{BEC_0}$ ~ $\overline{BEC_2}$, $\overline{CS}$, $\overline{IACK}$, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, and R/$\overline{W}$ guarantees their recognition at the next falling edge of the clock. Setup time for $\overline{REQ_0}$ ~ $\overline{REQ_3}$, $\overline{PCL_0}$ ~ $\overline{PCL_3}$, $\overline{DTACK}$, and $\overline{DONE}$ guarantees their recognition at the next rising edge of the clock.

2) Timing measurements for Input pins are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 volts. Timing measurements for Output pins are referenced to and from a low voltage of 0.5 volts and a high voltage of 2.4 volts.

3) These waveforms should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Refer to other functional descriptions and their related diagrams for device operation.

Figure 4 AC Electrical Waveforms – Bus Arbitration

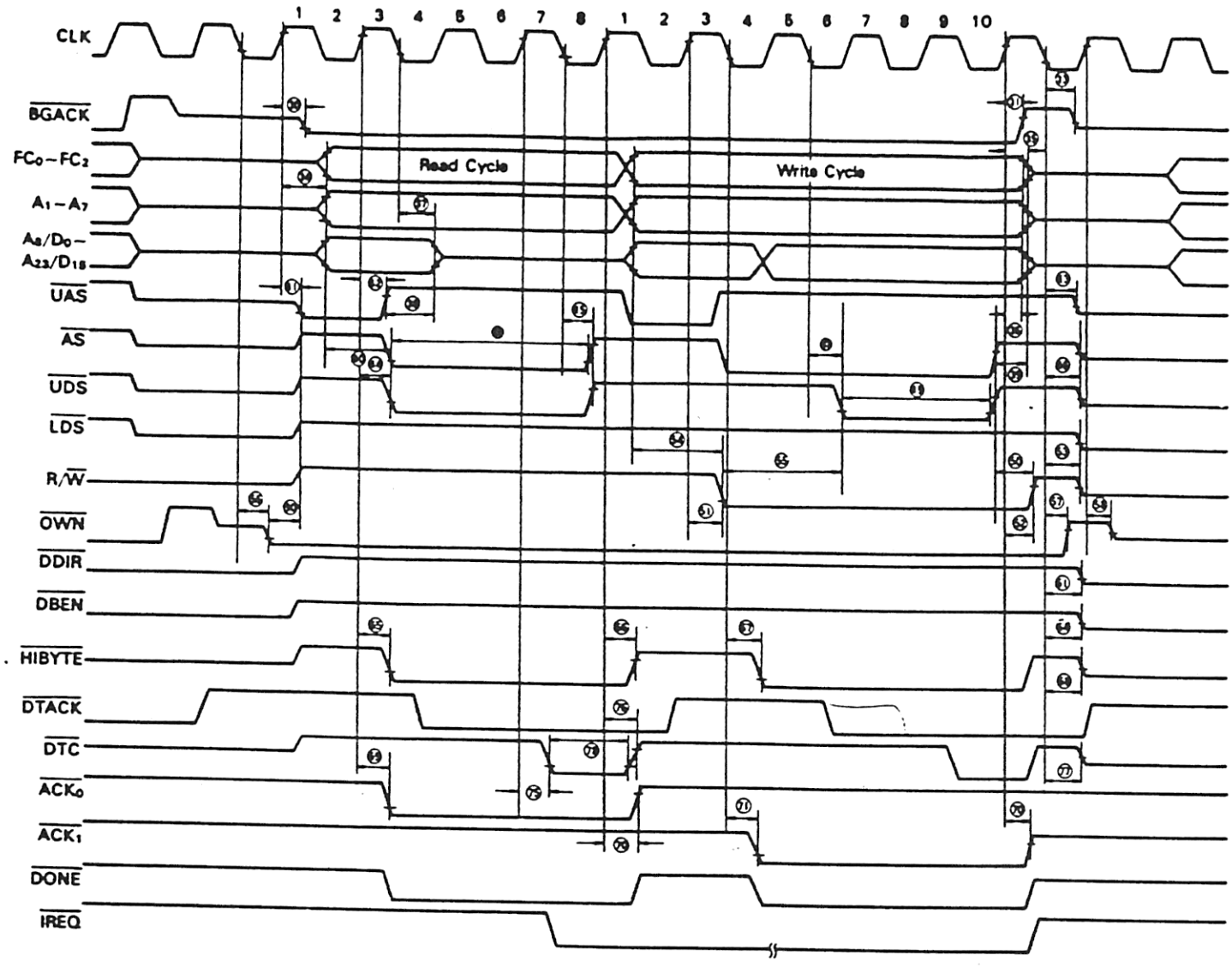


Figure 5 AC Electrical Waveforms – DMA Read/Write (Single Cycle)

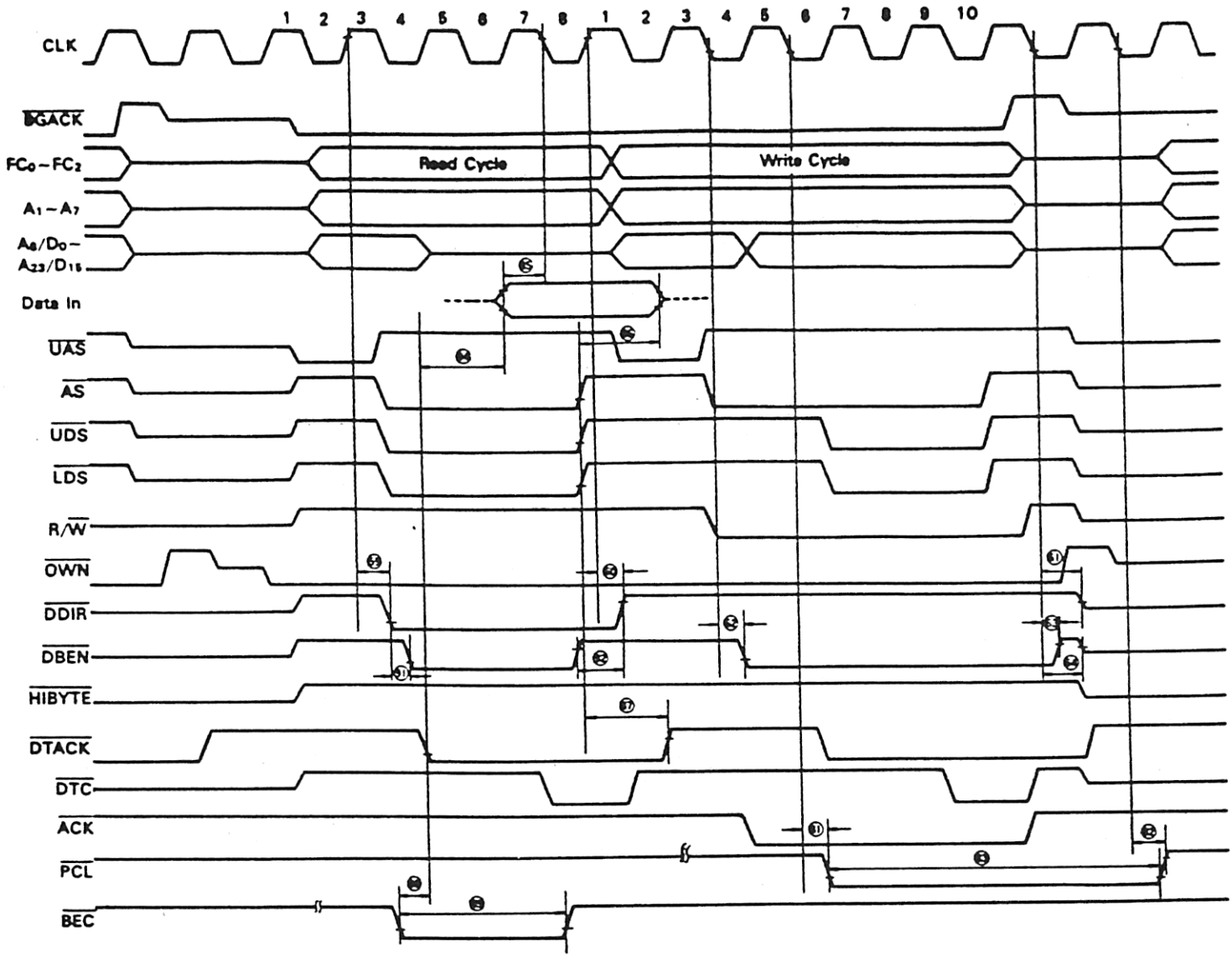


Figure 6 Electrical Waveforms – DMA Read/Write (Dual Cycle)

■ GENERAL DESCRIPTION

This document defines the HD68450, a four channel DMA Controller. The operation of each channel is independent of the other channels. The controller supports single-address or dual-address transfers. The controller supports unchained, array chained, or link chained operations. The device interface includes lines for requesting, acknowledging, and providing incidental control for the device.

The DMAC functions by transferring a series of operands between memory and device; operand sizes can be byte, word, or long word. A block is a sequence of operands; the number of operands in a block is determined by the transfer count. A single channel operation may involve the transfer of several blocks of data between memory and device.

NOTE:

Throughout the specification, signals are discussed using the terms active and inactive or asserted or negated independent of whether the signal is active in the logic one state or the logic zero state.

■ SIGNAL DESCRIPTION

The following section identifies the signals used in connecting to the HMCS68000 bus and peripherals using the DMA controller. Each signal has a basic definition of its use, a detailed description of the operation of each signal is contained in subsequent sections. Specific timing information is also contained in subsequent sections.

In the following definitions, "MPU mode" refers to the state when the DMAC is chip selected. The term "DMA mode" refers to the state when the DMAC assumes ownership of the bus. The DMAC is in the "IDLE mode" at all other times.

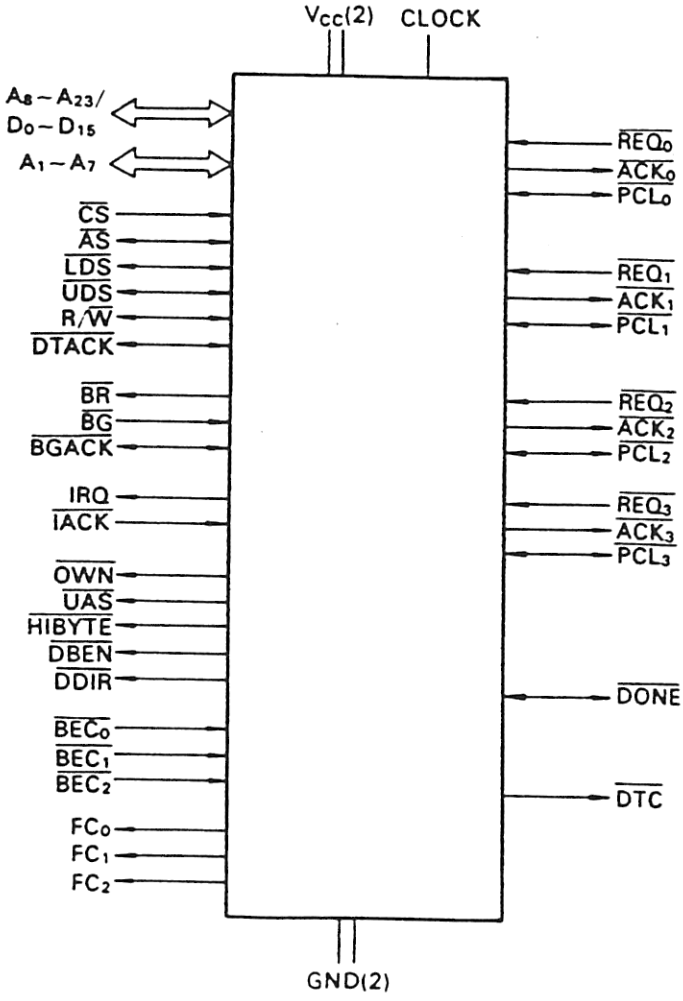


Figure 7 Input and Output Signals

• ADDRESS DATA BUS (A_8/D_0 through A_{23}/D_{15})

Input/Output — Three-statable
Active high

These lines are time multiplexed for data and address leads. The lines $\overline{DDIR}$, $\overline{DBEN}$, $\overline{UAS}$ and $\overline{OWN}$ are used to control the demultiplexing of the data/address lines with external gating. This is explained in a later section.

The bi-directional data lines ($D_0 \sim D_{15}$) are used to transfer data between the MPU, DMAC, memory and peripheral devices. Address lines are outputs to address memory and peripheral devices.

• ADDRESS BUS (A_1 through A_7)

Input/Output — Three-statable
Active high

In the MPU mode, the low order seven address lines specify which of the internal registers is accessed. The address map for these registers is shown in Table 1. During a DMA bus cycle, $A_1 \sim A_7$ are outputs containing the low order address bits of the location being accessed.

• FUNCTION CODES (FC_0 through FC_2)

Output — Three-statable
Active high

These output signals provide the function codes during DMA bus cycles. They are three-stated while in MPU mode or IDLE mode.

• CLOCK (CLK)

Input

This is the HMCS68000 system clock and must not be gated off at any time. Transferring to or from the DMAC registers, sampling of channel request lines, and gating of all control lines are done internally in conjunction with the CLK input.

• CHIP SELECT ($\overline{CS}$)

Input
Active low

This input signal is used to select the DMAC for programmed transfers to and from the DMAC. The DMAC is deselected when the $\overline{CS}$ input is inactive. If the $\overline{CS}$ input is asserted during a bus cycle which is generated by the DMAC, the DMAC internally terminates the bus cycle, signals an address error, but does not perform an operation. This protects DMAC registers during bus cycles which are generated by itself. However, bus cycles

generated by any other bus masters, including other DMACs, may address and change the DMAC's internal registers and, consequently, the operation of the DMAC.

- **ADDRESS STROBE ($\overline{AS}$)**

Input/Output — Three-statable
Active low

In the MPU or IDLE modes, this signal is monitored by the DMAC if it is requesting, and has been granted, permission to become bus master. In the DMA mode, this signal is an output indicating that the DMAC has placed a valid address on the bus.

- **UPPER ADDRESS STROBE ($\overline{UAS}$)**

Output — Three-statable
Active low

This line is an output to latch the upper address bits on the multiplexed data/address lines. Further explanation is given in later sections and diagrams. It is three-stated during the MPU mode and the IDLE mode.

- **OWN ($\overline{OWN}$)**

Output — Three-statable
Active low

This line is asserted by the DMAC during DMA mode. It is used to control the output of the transparent latch used to latch the address lines. This line may also be used to control the direction of bi-directional buffers when the loads on $\overline{AS}$, $\overline{LDS}$, $\overline{UDS}$, R/$\overline{W}$ and other signals exceed the drive of the DMAC pins. It is three-stated during the MPU mode and the IDLE mode.

- **DATA DIRECTION ($\overline{DDIR}$)**

Output — Three-statable

This line controls the direction of data through a bidirectional buffer on the data bus. It is three-stated during the IDLE mode.

- **DATA BUS ENABLE ($\overline{DBEN}$)**

Output — Three-statable
Active low

This line controls the output of bidirectional buffers on the multiplexed data/address bus. It is three-stated during the IDLE mode.

- **HIGH BYTE ($\overline{HIBYTE}$)**

Output — Three-statable
Active low

This line is used when the operand size is byte in the implicit addressing operation. It is asserted when data is present on the upper eight bits of the data bus. It is three-stated during the MPU mode and the IDLE mode.

- **READ/WRITE (R/$\overline{W}$)**

Input/Output — Three-statable
Active low

Read/Write (R/$\overline{W}$) is an input in the MPU mode and an output during the DMA mode. In the MPU mode, it is used to control the direction of data flow through the DMAC's input/output data bus interface. In the DMA mode, R/$\overline{W}$ is an output to memory and I/O controllers. It is held three-stated during IDLE mode.

- **UPPER DATA STROBE ($\overline{UDS}$)**

Input/Output — Three-statable
Active low

- **LOWER DATA STROBE ($\overline{LDS}$)**

Input/Output — Three-statable
Active low

These lines are extensions of the address lines indicating which byte or bytes of data (LSB, MSB) of the addressed word are being addressed

- **DATA TRANSFER ACKNOWLEDGE ($\overline{DTACK}$)**

Input/Output — Three-statable
Active low

In the MPU mode $\overline{DTACK}$ is an output indicating that the DMAC has completed the requested data transfer (read or write).

In the DMA mode, the DMAC monitors $\overline{DTACK}$ to determine when a data transfer has completed. In the event that a preemptory bus exception occurs prior to or concurrent with $\overline{DTACK}$, the $\overline{DTACK}$ response is ignored and the bus exception honored. In the IDLE mode, this signal is held in three-state.

- **BUS EXCEPTION CONTROLS ($\overline{BEC_0}$ through $\overline{BEC_2}$)**

Input
Active low

These lines provide an encoded signal indicating some exceptional bus condition. See Page 35 for details on bus exceptions.

- **BUS REQUEST ($\overline{BR}$)**

Output
Active low

The Bus Request ($\overline{BR}$) output is generated by the DMAC to request ownership of the bus.

- **BUS GRANT ($\overline{BG}$)**

Input
Active low

The Bus Grant ($\overline{BG}$) input indicates to the DMAC that it is to be the next bus master. This signal is originated by the MPU and propagated via a daisy chain or other arbitration mechanism. The DMAC cannot assume ownership until both

$\overline{AS}$ and $\overline{BGACK}$ become inactive. Once the DMAC acquires the bus, it does not continue to monitor the $\overline{BG}$ input.

- **BUS GRANT ACKNOWLEDGE ($\overline{BGACK}$)**

Input/Output — Three-statable
Active low

Bus Grant Acknowledge ($\overline{BGACK}$) is a bidirectional control line. As an output, it is generated by the DMAC to indicate that it is the bus master.

As an input, $\overline{BGACK}$ is monitored by the DMAC in order to determine whether or not the current bus master is a DMA device or not. $\overline{BGACK}$ must be inactive before the DMAC may assume ownership of the bus.

- **INTERRUPT REQUEST ($\overline{IRQ}$)**

Output — Open drain
Active low

Interrupt Request ($\overline{IRQ}$) is used to interrupt the MPU.

- **INTERRUPT ACKNOWLEDGE ($\overline{IACK}$)**

Input
Active low

Interrupt acknowledge ($\overline{IACK}$) is an input to the DMAC indicating that the current bus cycle is an interrupt acknowledge cycle. By the MPU, the DMAC responds with the contents of the normal or exception interrupt vector register of the highest priority channel requesting an interrupt. $\overline{IACK}$ is not serviced if the DMAC has not generated $\overline{IRQ}$.

- **CHANNEL REQUEST ($\overline{REQ_0}$ through $\overline{REQ_3}$)**

Input
Falling edge or active low

The four $\overline{REQx}$ inputs ($\overline{REQ_0}$ ~$\overline{REQ_3}$) are falling edge sensitive inputs when the request mode is cycle steal. The $\overline{REQx}$ inputs are low level sensitive when the request mode is burst.

- **CHANNEL ACKNOWLEDGE ($\overline{ACK_0}$ through $\overline{ACK_3}$)**

Output — Non-three-statable
Active low

The four $\overline{ACKx}$ lines ($\overline{ACK_0}$ ~ $\overline{ACK_3}$) indicate to a requesting peripheral device that the bus has been acquired and that the requested bus cycle is beginning. The $\overline{ACKx}$ line may be used as part of the enable circuit for bus interface to the peripheral.

- **PERIPHERAL CONTROL ($\overline{PCL_0}$ through $\overline{PCL_3}$)**

Input/Output — Three-statable
Active low

The four $\overline{PCLx}$ lines ($\overline{PCL_0}$ ~ $\overline{PCL_3}$) are multi-purpose lines which may be individually programmed to be a START output, an Enable Clock, READY, ABORT, STATUS, or INTERRUPT input.

- **DONE ($\overline{DONE}$)**

Input/Output — Open drain
Active low

The one $\overline{DONE}$ output is normally high and goes low concurrent with $\overline{ACKx}$ if that channel's operation is completed as a result of that transfer. As an input, it allows the device to indicate a normal termination of the operation.

- **DEVICE TRANSFER COMPLETE ($\overline{DTC}$)**

Output — Three-statable
Active low

The single device transfer complete output is normally high and goes low to signal to the device that the data transfer is complete. On a write to memory operation, it indicates that the data has been successfully stored. On a read from memory operation, it indicates that the data is present at the device and should be latched.

■ INTERNAL ORGANIZATION

The DMAC has four largely independent DMA channels. Each channel has its own set of channel registers. These registers define and control the activity of the DMAC in processing a channel operation.

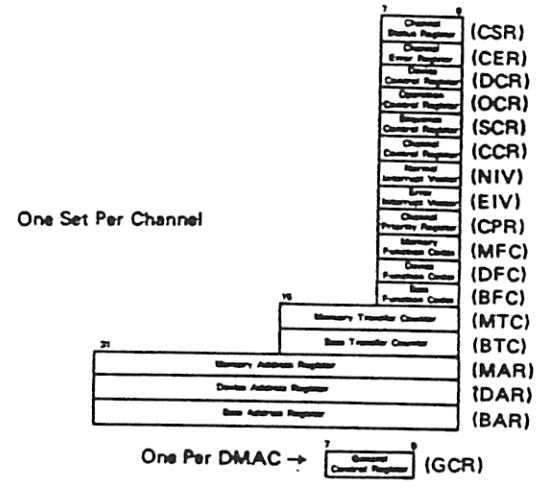


Figure 8 Internal Registers

- **REGISTER ORGANIZATION**

The internal accessible register organization is represented in Table 1. Address space not used within the address map is reserved for future expansion. A read from a reserved location in the map results in a read from the "null register". The null register returns all ones for data and results in a normal bus cycle. A write to one of these locations results in a normal bus cycle but no write occurs. Unused bits of a defined register read as zeros.

Table 1 DMAC Register Addressing Assignments

Register	Address Bits 7	6	5	4	3	2	1	0	Mode
Channel Status Register	c	c	0	0	0	0	0	0	R/$\overline{W}$*
Channel Error Register	c	c	0	0	0	0	0	1	R
Device Control Register	c	c	0	0	0	1	0	0	R/$\overline{W}$
Operation Control Register	c	c	0	0	0	1	0	1	R/$\overline{W}$
Sequence Control Register	c	c	0	0	0	1	1	0	R/$\overline{W}$
Channel Control Register	c	c	0	0	0	1	1	1	R/$\overline{W}$
Memory Transfer Counter	c	c	0	0	1	0	1	b	R/$\overline{W}$
Memory Address Register	c	c	0	0	1	1	s	s	R/$\overline{W}$
Device Address Register	c	c	0	1	0	1	s	s	R/$\overline{W}$
Base Transfer Counter	c	c	0	1	1	0	1	b	R/$\overline{W}$
Base Address Register	c	c	0	1	1	1	s	s	R/$\overline{W}$
Normal Interrupt Vector	c	c	1	0	0	1	0	1	R/$\overline{W}$
Error Interrupt Vector	c	c	1	0	0	1	1	1	R/$\overline{W}$
Channel Priority Register	c	c	1	0	1	1	0	1	R/$\overline{W}$
Memory Function Codes	c	c	1	0	1	0	0	1	R/$\overline{W}$
Device Function Codes	c	c	1	1	0	0	0	1	R/$\overline{W}$
Base Function Codes	c	c	1	1	1	0	0	1	R/$\overline{W}$
General Control Register	1	1	1	1	1	1	1	1	R/$\overline{W}$

cc: 00 – Channel #0, 01 – Channel #1
10 – Channel #2, 11 – Channel #3
ss: 00 – High-order, 01 – Upper middle,
10 – Lower middle, 11 – Low-order
b: 0 – High-order, 1 – Low-order
* See Channel Status Register on Page 31.

• DEVICE CONTROL REGISTER (DCR)

The DCR is a device oriented control register. The XRM bits specifies whether the channel is in burst or cycle steal request mode. The DTYP bits define what type of device is on the channel. If the DTYP bits are programmed to be a HMCS6800 device the PCL definition is ignored and the $\overline{PCL}$ line is an Enable clock input. If the DTYP bits are programmed to be a device with $\overline{READY}$, the PCL definition is ignored and the $\overline{PCL}$ line is a ready input (active low). The DPS bit defines what port size the device has. The PCL bits define the function of the $\overline{PCL}$ line. When the content of the DTYP bits implies HMCS6800 compatible device, or Device with $\overline{ACK}$ and $\overline{READY}$, the content of the PCL bits is disregarded. The XRM bits are ignored if an auto request mode in the OCR is selected.

7 6	5 4	3	2	1 0
XRM	DTYP	DPS	0	PCL

XRM External Request Mode
- 00 Burst Transfer Mode
- 01 (undefined, reserved)
- 10 Cycle Steal Mode without Hold
- 11 Cycle Steal Mode with Hold

DTYP Device Type
- 00 HMCS68000 compatible device, explicitly addressed
- 01 HMCS6800 compatible device, explicitly addressed
- 10 Device with $\overline{ACK}$, implicitly addressed
- 11 Device with $\overline{ACK}$ and $\overline{READY}$, implicitly address

DPS Device Port Size
- 0 8 Bit Port
- 1 16 Bit Port

PCL Peripheral Control Line
- 00 Status Input (can be read by reading CSR)
- 01 Status Input with Interrupt
- 10 Start Pulse, Negative 1/8 CLK
- 11 Abort Input

0 Bit 2 Not Used

• OPERATION CONTROL REGISTER (OCR)

The OCR is an operation oriented register. The DIR bit defines the direction of the transfer, to or from memory. The SIZE bits define the size of the operand and how the transfer count and address registers are to be handled. The CHAIN bits tells the DMAC if any or what type of chaining is to be performed. The REQG bits define how requests for transfers are generated. Chaining and requests are discussed in a later section.

7	6	5 4	3 2	1 0
DIR	0	SIZE	CHAIN	REQG

DIR Direction
- 0 Transfer from memory to device
- 1 Transfer from device to memory

0 Bit 6 Unused

SIZE Operation Size
- 00 Byte
- 01 Word
- 10 Long word
- 11 (undefined, reserved)

CHAIN Chaining Operation
- 00 Chain operation is disabled
- 01 (undefined, reserved)
- 10 Array chaining
- 11 Linked chaining

REQG DMA Request Generation Method
- 00 Auto-request at rate limited by General Control Register (GCR)
- 01 Auto-request at maximum rate
- 10 $\overline{REQ}$ line initiates an operand transfer
- 11 Auto-request the first operand, external request for subsequent operands

• SEQUENCE CONTROL REGISTER (SCR)

The SCR is used to define the sequencing of memory device addresses.

7	6	5	4	3 2	1 0
0	0	0	0	MAC	DAC

0 Bits 7, 6, 5, 4 Not Used

MAC Memory Address Count
- 00 Memory address register does not count
- 01 Memory address register counts up
- 10 Memory address register counts down
- 11 (undefined, reserved)

DAC Device Address Register Count
- 00 Device address register does not count
- 01 Device address register counts up
- 10 Device address register counts down
- 11 (undefined, reserved)

- **CHANNEL CONTROL REGISTER (CCR)**

The CCR is used to start or terminate the operation of a channel. The register also determines if an interrupt is to be generated at the termination of an operation (normal or error termination). Setting the STR bit causes immediate activation of the channel; the channel will be ready to accept requests immediately. The STR and CNT bits of the register may not be reset by a write to the register. The software abort bit (SAB) may be used to terminate the operation. Setting the SAB bit will reset STR and CNT. Setting the HLT bit will halt the channel, and resetting the HLT bit will resume the operation.

7	6	5	4	3	2	1	0
STR	CNT	HLT	SAB	INT	0	0	0

STR Start Operation
- 0 No operation is pending
- 1 Start operation

CNT Continue Operation
- 0 No continuation is pending
- 1 Continue operation

HLT Halt Operation
- 0 Operation not halted
- 1 Operation halted

SAB Software Abort
- 0 Channel operation not aborted
- 1 Abort channel operation

INT Interrupt Enable
- 0 No interrupts enabled
- 1 Interrupts enabled

0 Bits 2, 1, 0 Not Used

- **CHANNEL STATUS REGISTER (CSR)**

The CSR is a register containing the status of the channel.

7	6	5	4	3	2	1	0
COC	BTC	NDT	ERR	ACT	0	PCT	PCS

COC Channel Operation Complete
- 0 Channel operation incomplete
- 1 Channel operation complete

BTC Block Transfer Complete
- 0 Block transfer incomplete
- 1 Block transfer complete

NDT Normal Device Termination
- 0 No normal device termination
- 1 Device terminated operation normally

ERR Error Bit
- 0 No errors
- 1 Error as coded in CER

ACT Channel Active
- 0 Channel not active
- 1 Channel active

PCT $\overline{PCL}$ Transition
- 0 No $\overline{PCL}$ transition occurred
- 1 $\overline{PCL}$ transition occurred

PCS The State of the $\overline{PCL}$ Input Line
- 0 $\overline{PCL}$ low
- 1 $\overline{PCL}$ high

0 Bit 2 Not Used

- **CHANNEL ERROR REGISTER (CER)**

The CER is an error condition status register. The ERR bit of CSR indicates if there is an error or not. Bits 0–4 indicate what type of error occurred.

7	6	5	4	3	2	1	0
0	0	0	ERROR CODE				

0 Bits 7, 6, 5 Not Used

ERROR CODE
- 00000 No error
- 00001 Configuration error
- 00010 Operation timing error
- 00011 (undefined, reserved)
- 001rr Address error
- 010rr Bus error
- 011rr Count error
- 10000 External abort
- 10001 Software abort

rr register or counter code
- 01 Memory address or memory counter
- 10 Device address
- 11 Base address or base counter

- **CHANNEL PRIORITY REGISTER (CPR)**

The CPR is used to define the priority level for each channel.

7	6	5	4	3	2	1	0
0	0	0	0	0	0	CP	

CP Channel Priority
These two bits determine the priority (0–3) of the channel.

0 Bits 7, 6, 5, 4, 3, 2 Not Used

- **GENERAL CONTROL REGISTER (GCR)**

The GCR is used to define what portion of the bus cycles is available to the DMAC for limited rate auto-request generation.

7	6	5	4	3	2	1	0
0	0	0	0	BT		BR	

BT Burst Time
The number of DMA clock cycles per burst that the DMAC allows in the auto-request at a limited rate of transfer is controlled by these two bits. The number is 2** (BT+4) (two to the BT+4 power).

BR Bandwidth Ratio
The amount of the bus bandwidth utilized by the auto-request at a limited rate transfer is controlled by these two bits. The ratio is 2** (BR+1) (two to the BR+1 power).

0 Bits 7, 6, 5, 4 Not Used

- **ADDRESS REGISTERS**

Three 32-bit registers are utilized to implement the Memory Address Register, Device Address Register, and the Base Address Register. Due to packaging limitations, only the least significant twenty-four bits are connected to the address output pins.

- **FUNCTION CODE REGISTERS**

Each address register has a function code register associated with it. The function code registers are three bits wide and are loaded via the lowest three bits of the data bus.

- **TRANSFER COUNT REGISTERS**

Two sixteen bit counters per channel are provided to implement the Memory Transfer Counter and the Base Transfer Counter.

- **INTERRUPT VECTOR REGISTERS**

Each channel has an interrupt vector register and an error interrupt vector register, each consisting of eight bits.

■ OPERATION DESCRIPTION

- **GENERAL DESCRIPTION**

A DMAC channel operation proceeds in three principal phases. During the initialization phase, the MPU configures the channel control registers, sets initial addresses, and starts the channel. During the transfer phase, the DMAC accepts requests for data operand transfers, and provides addressing and bus controls for the transfers. The termination phase occurs after the operation is completed when the DMAC reports the status of the operation. Refer to Figure 9 through Figure 11 for MPU/DMAC communication timings.

This section describes DMAC operation. A brief description of the device/DMAC communication is given first. Next, the transfer phase is covered, including how the DMAC recognizes requests and how the DMAC arranges for data transfer. Following this, the initialization phase is described. The termination phase is covered, introducing chaining, error signaling, and bus exceptions. A description of the channel priority scheme rounds out the section.

The MPU reads and writes the DMAC internal registers to control the operations. Figure 9 indicates the timing diagram when the MPU reads the contents of the DMAC internal register. the MPU outputs $A_1 \sim A_{23}$, $\overline{AS}$, R/$\overline{W}$, $\overline{UDS}$, and $\overline{LDS}$ and accesses the internal register. The DMAC outputs data on the bus, the buffer control signals ($\overline{DDIR}$ and $\overline{DBEN}$), and $\overline{DTACK}$. Read cycle consists of sixteen clocks. Figure 10 shows the timing of the MPU writes to the DMAC. Write cycle consists of thirteen clocks.

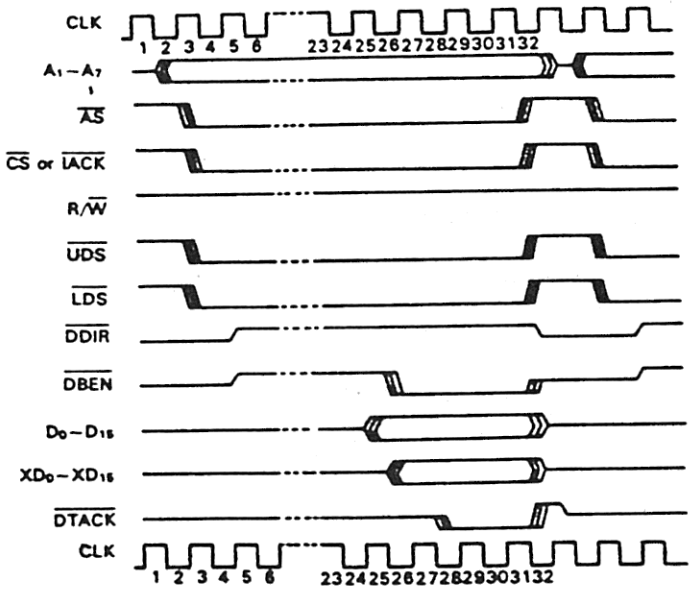


Figure 9 MPU Read from DMAC – Word

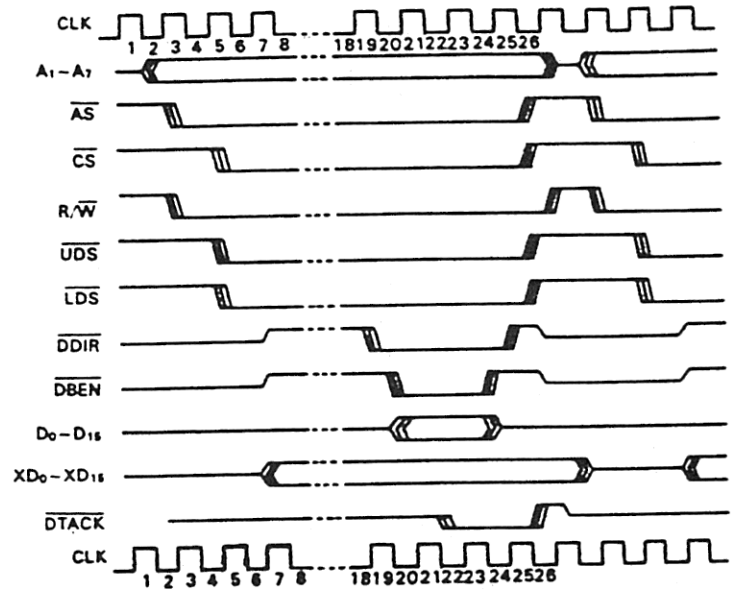
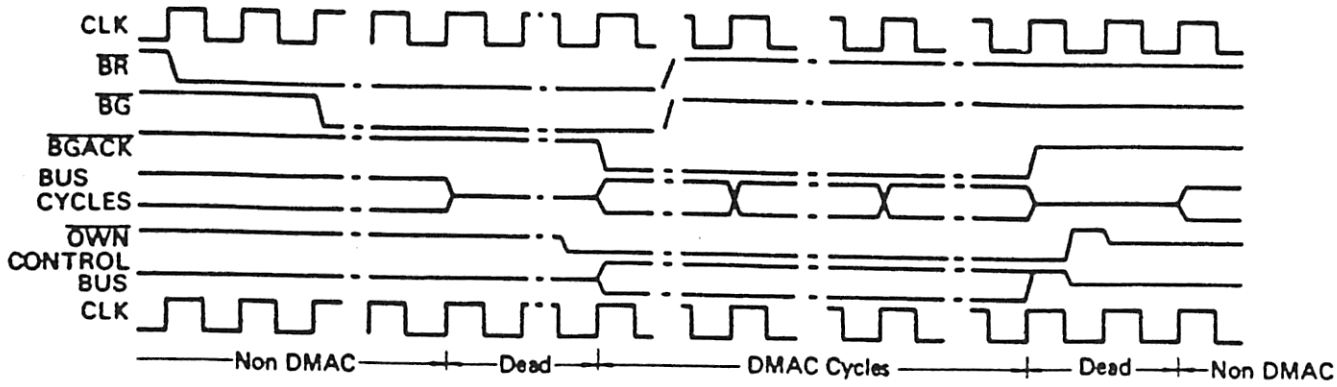


Figure 10 MPU Write to DMAC – Word

(NOTES)
1) The $\overline{CS}$ equation includes data strobe.
2) $D_0 \sim D_{15}$ represent the multiplexed address/data pins which are used for data only during MPU mode.
3) $XD_0 \sim XD_{15}$ is the external or 68000 system data bus, i.e. on the system side of the data buffers.
4) Cycle lengths reflect the response of the current HD68000 MPU.
5) In the MPU read from DMAC mode, the DMAC will not give $\overline{DTACK}$ until the data is guaranteed valid on the system data bus for one half clock.
6) During the MPU read, the DMAC must remove signals within one clock after $\overline{AS}$ is negated.
7) The DMAC will negate $\overline{DTACK}$ within one clock after $\overline{AS}$ is negated.
8) During the MPU write to DMAC, the $\overline{DDIR}$ line will be driven low to direct the data buffers toward to DMAC before the buffers are enabled.
9) The DMAC will latch the data (MPU write to DMAC) before asserting $\overline{DTACK}$. Once the data is latched the DMAC will negate $\overline{DBEN}$ and $\overline{DDIR}$ in the proper order.
10) $\overline{CS}$ will be removed within one clock after $\overline{AS}$ is negated.
11) Note that $\overline{DDIR}$ and $\overline{DBEN}$ must drive out of tristate when $\overline{CS}$ is detected and then must be re-tristated at the end of the cycle.
12) The clock reference shown in this diagram is the CPU clock.

Figure 11 indicates the DMAC bus arbitration timing. The DMAC asserts $\overline{BR}$ to request the bus mastership. The MPU issues $\overline{BG}$ to grant the ownership in the next bus cycle. After the end of the current cycle, the DMAC starts its own bus cycle accompanied with the dead cycles.



(NOTES) 1) Note the timing of the OWN signal. It will drive active one half clock prior to the start of the first DMAC cycle. It will drive inactive one half clock after the end of the last DMAC cycle. At this same time, all other control signals will tristate. One half clock after this, the OWN signal will tristate.
2) CONTROL BUS refers to the control pins such as DBEN, AS, ACK, etc. on the DMAC.
3) BR signal will be negated one clock after BGACK signal is asserted.

Figure 11 DMAC Bus Arbitration Timing

• DEVICE/DMAC COMMUNICATION

Communication between peripheral devices and the DMAC is accommodated by five signal lines. Each channel has a request (REQ), an acknowledge (ACK), and a peripheral control line (PCL). The last two lines, the DONE and DTC lines, are shared among the four channels.

(1) REQUEST (REQ)

A channel can make a request for service by asserting the individual channel request line.

(2) ACKNOWLEDGE (ACK)

Each channel has an acknowledge line which is activated during transfers to or from the device. This line is used to implicitly address the device which is transferring the data. It may also be used to control the buffering circuits between the device and the HMCS68000 bus.

(3) PERIPHERAL CONTROL LINE (PCL)

Each channel has a peripheral control line. The function of this line is quite flexible, and is determined by the programmed state of DCR.

The DTYP bits of the DCR define what type of device is on the channel. If the DTYP bits are programmed to be a HMCS6800 device, the PCL definition is ignored and the PCL line is an Enable clock input. If the DTYP bits are programmed to be a device with READY, the PCL definition is ignored and the PCL line is a ready input.

The PCL line is active at all times when the PCL line is programmed as a Status input, Interrupt input, a Ready input, or an Enable input. When programmed to be an Abort input it is only active after the channel has been started.

PCL AS A STATUS INPUT

The PCL line may be programmed as a status input. The status level can be determined by reading the PCS bit in the CSR. If a negative transition occurs and remains stable for two DMAC clock cycles on the PCL line, the PCT bit of the CSR is set. This bit is cleared by resetting the DMAC or writing to the PCT bit of the Channel Status Register.

PCL AS AN INTERRUPT

The PCL line may also be programmed to generate an interrupt on a negative transition. This enables an interrupt which is requested if the PCT bit of the CSR is set.

PCL AS A STARTING CLOCK PULSE

The PCL line may be programmed to output a single pulse. The duration of the active low pulse is eight clock cycles, and starts when the channel is activated.

PCL AS AN ABORT INPUT

The PCL line may be programmed to be an negative transition abort input which terminates an operation by signaling the abort error. When this function has been programmed, the PCL line is only active after the channel has been started. The negative transition must remain stable in a low level for a minimum of two DMAC clock cycles.

PCL AS AN ENABLE INPUT

If the DTYP bits are programmed to be a HMCS6800 device, the PCL definition is ignored and the PCL line is an Enable clock input. The Enable clock downtime must be as long as five clock cycles, and must be high for a minimum of three DMAC clock cycles, but need not be synchronous with the clock.

PCL AS A READY INPUT

If the DTYP bits are programmed to be a device with READY, the PCL definition is ignored and the PCL line is a READY input. The READY is an active low input.

(4) DONE

DONE is an active low signal which is asserted when the memory transfer count is exhausted, and there are no more links to pick up in a chaining operation or the continue bit is not set. It is asserted and negated coincident with the acknowledge signal of the last operand part.

The DMAC also monitors the state of the $\overline{DONE}$ line while acknowledging a device. If the device asserts $\overline{DONE}$, the DMAC will terminate the operation after the transfer of the current operand. The DMAC terminates the operation by clearning the ACT bit of the CSR, and setting the COC and NDT bits of the CSR. If both the DMAC and the device asserts $\overline{DONE}$, the device termination is not recognized, but the channel operation does terminate.

(5) DATA TRANSFER COMPLETE

$\overline{DTC}$ is an active low signal which is asserted when the actual data transfer is accomplished. If data is being transferred from the device, $\overline{DTC}$ is asserted to indicate that the data is valid at the device, and should be latched. If a preemptory bus exception terminates the bus cycle, $\overline{DTC}$ is not asserted. $\overline{DTC}$ is an active signal whenever the DMAC is a bus master. It is asserted for both memory and peripheral DMAC initiated transfers.

- **REQUESTS**

Requests activate the DMAC to transfer an operand. The REQG bits of the OCR determine the manner in which requests are generated. Requests may be externally generated by circuitry in the device, or internally generated by the auto-request mechanism. Usually a single operation uses only one method of request generation, but an operation can auto-request the first transfer and then wait for the device to request further transfers.

(1) AUTO-REQUEST TRANSFERS

The auto-request mechanism provides generation of requests within the DMAC. These requests can be generated at either of two rates: maximum-rate, so that the channel always has a request pending, or limited-rate. The limited rate auto-request feature functions by monitoring the bus utilization.

AUTO-REQUEST BUS UTILIZATION

The DMAC monitors bus utilization to control the limited-rate auto-request (LRAR) feature. This monitoring is also used to determine when an external request device has paused.

The DMAC divides time into equal length sample intervals by counting clock cycles. The end of one sample interval marks the beginning of the next. During a sample interval, the DMAC notes bus and channel activity. At the end of the interval, decisions are made which affect channel operations during the next sample interval, as shown in Figure 12.

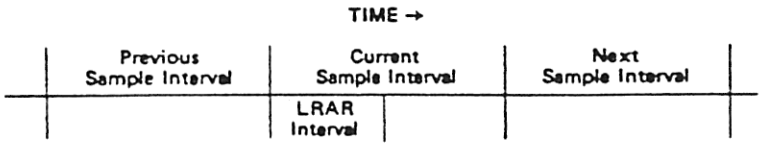


Figure 12 DMAC Sample Intervals

Based on the DMA activity during a sample interval, the DMAC allows limited-rate auto-requests for some initial portion of the next sample interval. The length of the sample interval, and the portion of the sample interval during which limited rate auto-requests can be made are controlled by the BT and BR parameters in the GCR. The length in clock cycles of the subinterval during which the DMAC allows limited-rate auto-requests is controlled by the BT. The number is 2** (BT+4). For example, if BT equals TWO and the DMA utilization of the bus was low during the previous sample interval, then the DMAC generates as many auto-request transfers as is possible during the first 64 clock cycles of the current sample interval.

The ratio of the length of the sample interval to the length of the limited-rate auto-request interval is controlled by the BR bits. This same parameter is used to determine the level of DMA bus utilization during the sample interval. If the fraction of DMA clock cycles during a sample interval exceeds the programmed utilization level, the DMAC will not allow limited-rate auto-requests during the next sample interval. Either ratio is 2** (BR+1) (2 raised to the BR+1 power). For example, if BR equals THREE, then at most one out of 16 clock cycles during a sample interval can be a DMA cycle, and still the DMAC would allow limited-rate auto-requests during the next sample interval. The DMAC monitors $\overline{BGACK}$ during each clock cycle to determine whether or not that clock cycle is used by a DMA device. If the $\overline{BGACK}$ input is active, the DMAC assumes that that clock cycle is for a DMA device. If it is inactive, the DMAC assumes that it is not a DMA cycle.

The sample interval length is not a direct parameter, but is equal to 2** (BT+BR+5) clock cycles. Thus the sample interval can vary from 32 to 2048 clock cycles.

AUTO-REQUEST

If the REQG bits in the OCR indicate auto-request at the maximum rate, the DMAC acquires the bus after the operation is started and transfers data until channel termination. The DMAC does not relinquish the bus until termination. If a request is made by another channel of equal or higher priority, the DMAC services that channel and then resumes the auto-request sequence.

If the REQG bits indicate auto-request at a limited rate, the channel generates requests only during the limited rate auto-request interval and then only when the bus utilization was below the required threshould during the previous sample interval. As a consequence, if an auto-request at maximum rate transfer is started, no limited rate auto-requests are generated before the termination of the maximum rate auto-request operation.

The $\overline{ACK}$, $\overline{PCL}$ and $\overline{DTC}$ lines are held inactive during an auto-request operation if the device type is HMCS68000 compatible. Consequently, any channel may be used for the auto-request function in addition to its normal application without disturbing any peripheral devices connected to that channel.

Refer to Figure 13 for more specific timing diagrams.

(2) EXTERNAL REQUESTS

If the REQG bits of the OCR indicate that the $\overline{REQ}$ line generates requests, the transfer requests are generated externally. The request line associated with each channel allows the device to externally generate requests for DMA transfers. When the device wants an operand transferred, it makes a request by asserting the request line. The external request mode is determined by the XRM bits of the DCR, which allows both burst and cycle steal request modes. The burst request mode allows a channel to request the transfer of multiple operands using consecutive bus cycles. The cycle steal request mode allows a channel to request the transfer of a single operand.

BURST REQUEST RECOGNITION

In the burst request mode, the $\overline{REQ}$ line is an active low input. The device requests an operand transfer by asserting $\overline{REQ}$. The DMAC services the request by arbitrating for the HMCS68000 bus, obtaining the bus, and notifying the peripheral by asserting the acknowledge line. If the request line is active

when the DMAC asserts acknowledge, and remains active at least until the DMAC asserts device transfer complete, the DMAC recognizes a valid request for another operand, which will be transferred during the next bus cycle if the channel has priority. If the request line is negated before the DMAC asserts device complete, the DMAC determines there is no valid request for an operand transfer, and no transfers are generated for that channel. Channels of the same or higher priority within the same DMA Controller may have DMA operand transfer requests serviced during this mode.

If the request is negated before the first transfer cycle has started, the cycle will terminate with the DMAC returning the bus.

Refer to Figure 14 for more specific timing diagrams.

CYCLE STEAL REQUEST RECOGNITION

In the cycle steal request mode, the device requests an operand transfer by generating a falling edge on the $\overline{REQ}$ line. The DMAC services a request by arbitrating for the bus, obtaining the bus and notifying the peripheral by asserting the acknowledge line.

After an request edge has been asserted it must remain at the assertion level at least two clock cycles. The request line must be inactive at least one clock cycle before a request is made. If another request from the channel is received before the first operand part of a former request is acknowledged, the second request is not recognized.

After the DMAC completes the transfer, it may service another channel, relinquish the bus, or hold the bus and wait for another request. If there are pending requests from other channels, one of the requesting channels is serviced. If there are no requests, the XRM bits determine whether the DMAC will relinquish the bus, or retain ownership.

If the XRM bits specify cycle steal with hold, the DMAC will retain ownership. The bus is not given up for arbitration until the channel operation terminates or until the device pauses. The device is determined to have paused if it does not make any requests during the next full sample interval. The sample interval counter is free running and is not reset or modified by this mode of operation. The sample interval counter is the same counter that is used for Limited Rate Auto Request and is programmed via the GCR.

If the XRM bits specify cycle steal without hold, the DMAC will relinquish the bus. If the device generates a request before DMAC asserts $\overline{DTC}$ for the last operand part, the DMAC will retain ownership of the bus, and that request will be serviced before the DMAC relinquishes the bus.

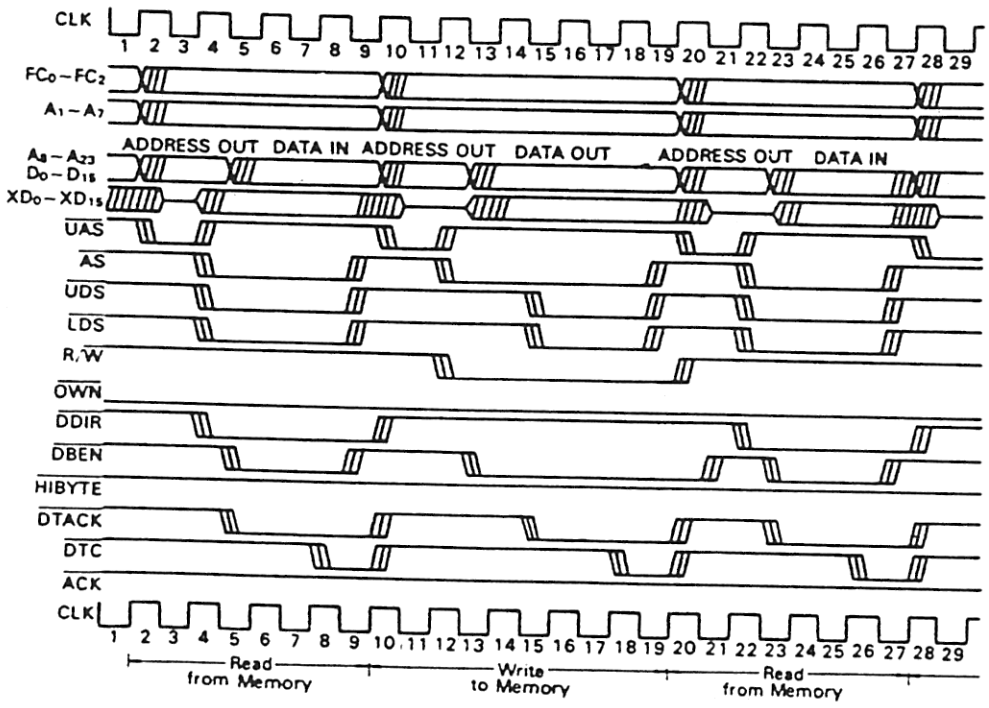
Refer to Figure 15 and Figure 16 for more specific timing diagrams.

REQUEST RECOGNITION IN DUAL-ADDRESS TRANSFERS

In a following section dual-address transfers are defined. Dual address transfer is an exception to the request recognition rules in the previous paragraphs. Refer to the Explicitly Addressed Device section for information.

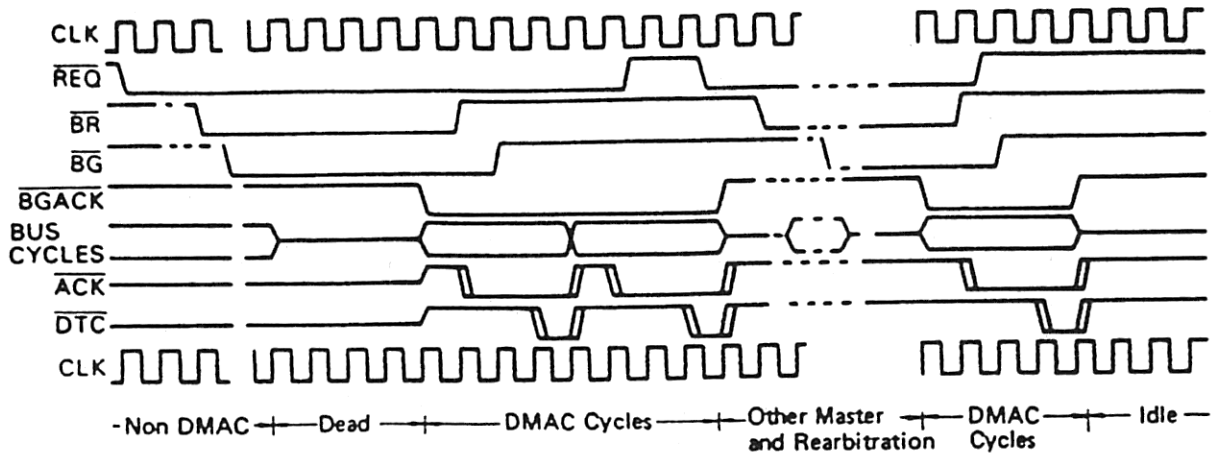
(3) MIXED REQUEST GENERATION

A single channel can mix the two request generation methods. By appropriately programming the REQG bits of the OCR, when the channel is started, the DMAC auto-requests the first transfer. Subsequent requests are then generated externally by the device. The $\overline{ACK}$ and $\overline{PCL}$ lines perform their normal functions in this operation.



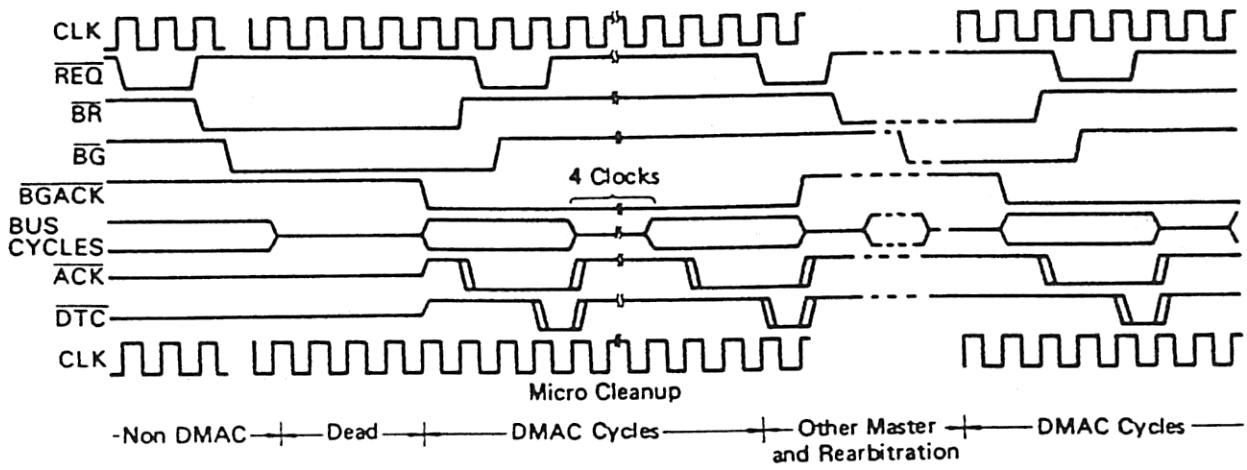
(NOTE) 1) Note that $\overline{ACK}$, $\overline{DONE}$, $\overline{DTC}$, and $\overline{HIBYTE}$ are always inactive in this mode. For comments on the other signals, see notes on the dual addressing mode with 8 bit device as source.

Figure 13 DMAC Auto Request Read – Write – Read Cycles



(NOTE) 1) Note that in the diagrams showing request timing it is assumed that only one channel is active.

Figure 14 DMAC Burst Mode Request Timing



(NOTES) 1) In this mode the device must re-assert $\overline{REQ}$ one clock before the assertion edge of $\overline{DTC}$ of the last bus cycle or lose the bus. The $\overline{REQ}$ signal is edge triggered.
2) The time labeled "micro cleanup" is the time it takes for the internal sequencer to start another bus cycle if no other channel has requests pending.

Figure 15 DMAC Cycle Steal Mode Request Timing

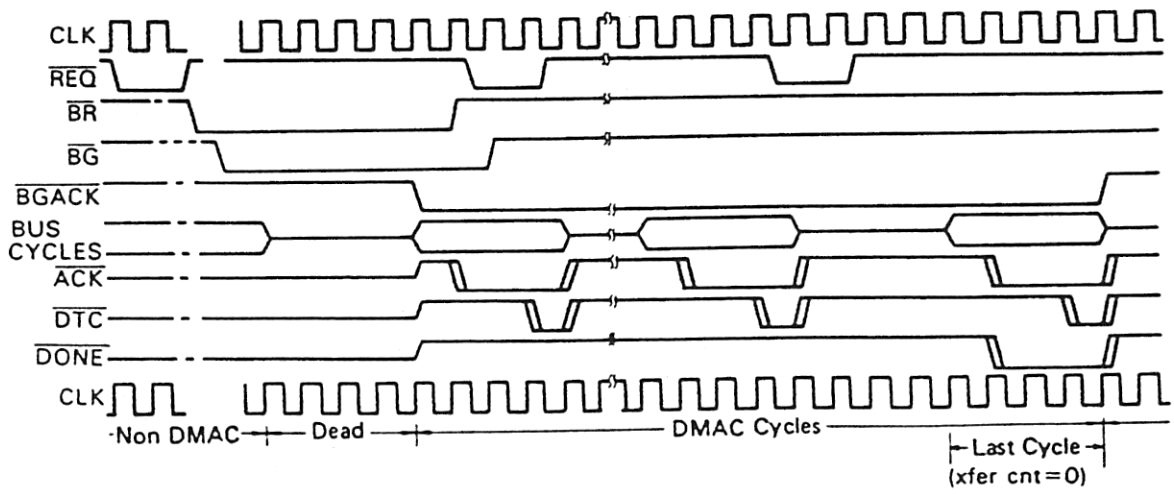


Figure 16 DMAC Cycle Steal-Hold Mode Request Timing

• DATA TRANSFERS

(1) DEVICE PROTOCOLS

All DMAC data transfers are assumed to be between memory and another device. The word "memory" means a 16-bit HMCS68000 bus compatible device. By programming the DCR, the characteristics of the device may be assigned. Each channel can communicate using any of the following protocols.

DTYP Device Type

00	HMCS68000 compatible device	Dual Addressing
01	HMCS6800 compatible device	Dual Addressing
10	Device with $\overline{ACK}$	Single Addressing
11	Device with $\overline{ACK}$ and $\overline{READY}$	Single Addressing

DUAL ADDRESSING

HMCS68000 and HMCS6800 compatible devices may be explicitly addressed. This means that before the peripheral transfers data, a data register within the device must be addressed. Because the address bus is used to address the peripheral, the data cannot be directly transferred to/from the memory because the memory also requires addressing. Instead, the data is transferred from the source to the DMAC and held in an internal DMAC holding register. A second bus transfer between the DMAC and the destination is then required to complete the operation. Because both the source and destination of the transfer are explicitly addressed, this protocol is also called dual-addressed.

Request Recognition in Dual-Address Transfers

The request recognition protocols defined in a previous section apply to dual-address operations. Requests are recognized during the transfer to/from the DMAC holding register and the peripheral as described in the request protocol section. This protocol requires the request to be asserted before the signal $\overline{DTC}$ is asserted, to have request recognition for the next cycle.

During the portion of the operation when the operand or operand part is transferred between the DMAC holding register and memory, requests are also recognized. During the transfer between memory and the holding register, $\overline{DTC}$ is not asserted, so it may not be used as reference point for request recognition during this portion of the operation. However, requests will be recognized if they are asserted prior to the portion of the cycle where $\overline{DTC}$ would have been asserted. This point is one half clock cycle before the upper and lower data strobes are negated.

HMCS68000 Compatible Device Transfers

In this operation, when a request is received, the bus is obtained and the transfer completed using the HMCS68000 bus protocol as shown in Figures 17 and 18. Refer to Figures 19 through 22 for timing information.

HMCS6800 Compatible Device Transfers

When a channel is programmed to perform HMCS6800 compatible transfers, the $\overline{PCL}$ line for that channel is defined as an Enable clock input. The DMAC performs data transfers between itself and the device using the HMCS6800 bus protocol, with the $\overline{ACK}$ output providing the valid memory address signal. This operation is necessary since the HMCS6800 bus is synchronous and the HMCS68000 bus is asynchronous. Figure 23 illustrates this protocol. This operation provides DMAC compatibility with existing HMCS6800 and other synchronous devices. Refer to Figures 24 and 25 for timing information. Figure 44 illustrates a sample circuit diagram of a two-6800 device system.

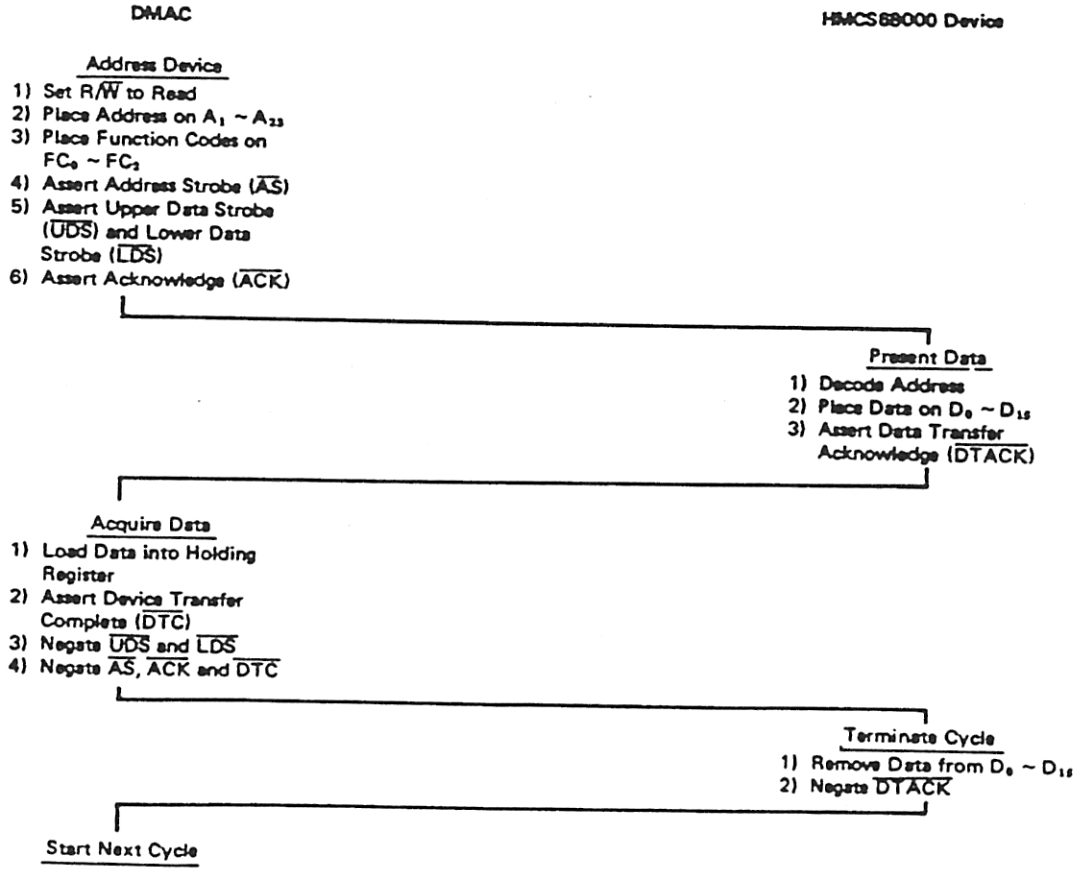


Figure 17 Word Read Cycle Flowchart HMCS68000 Type Device

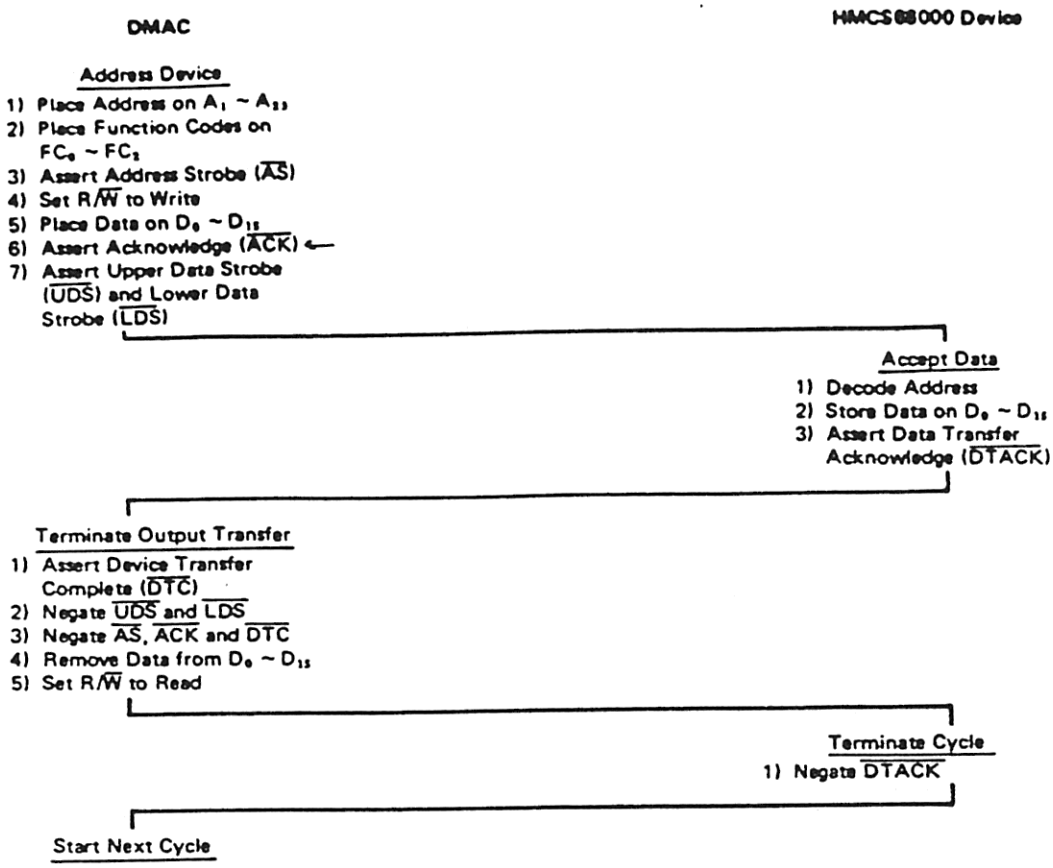
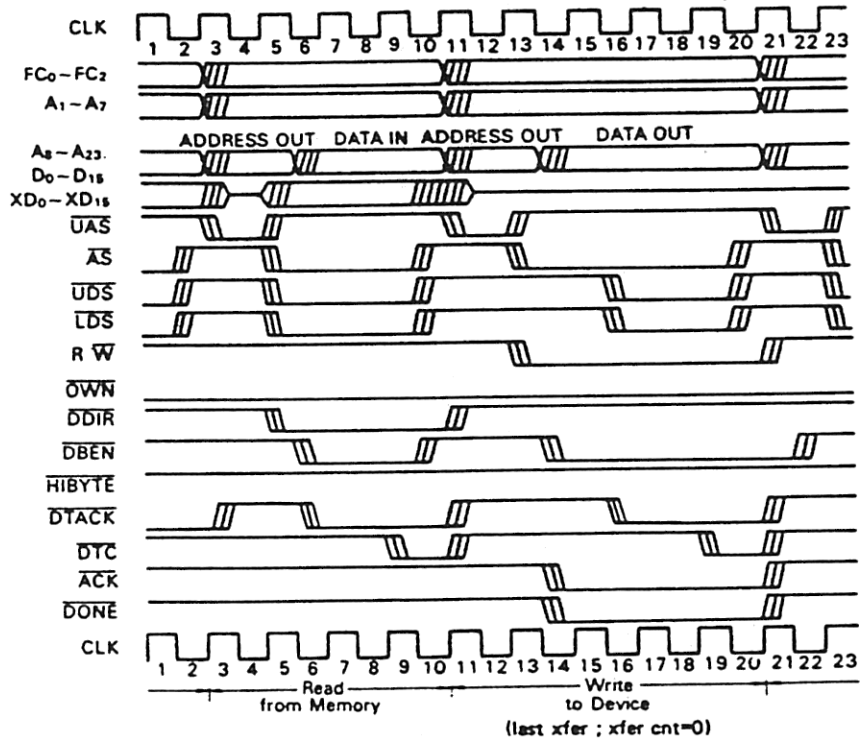
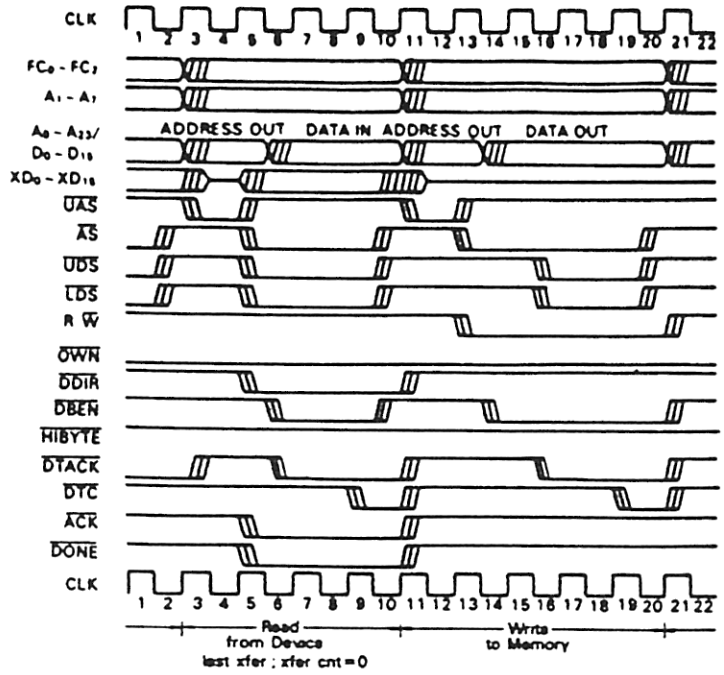


Figure 18 Word Write Cycle Flowchart HMCS68000 Type Device



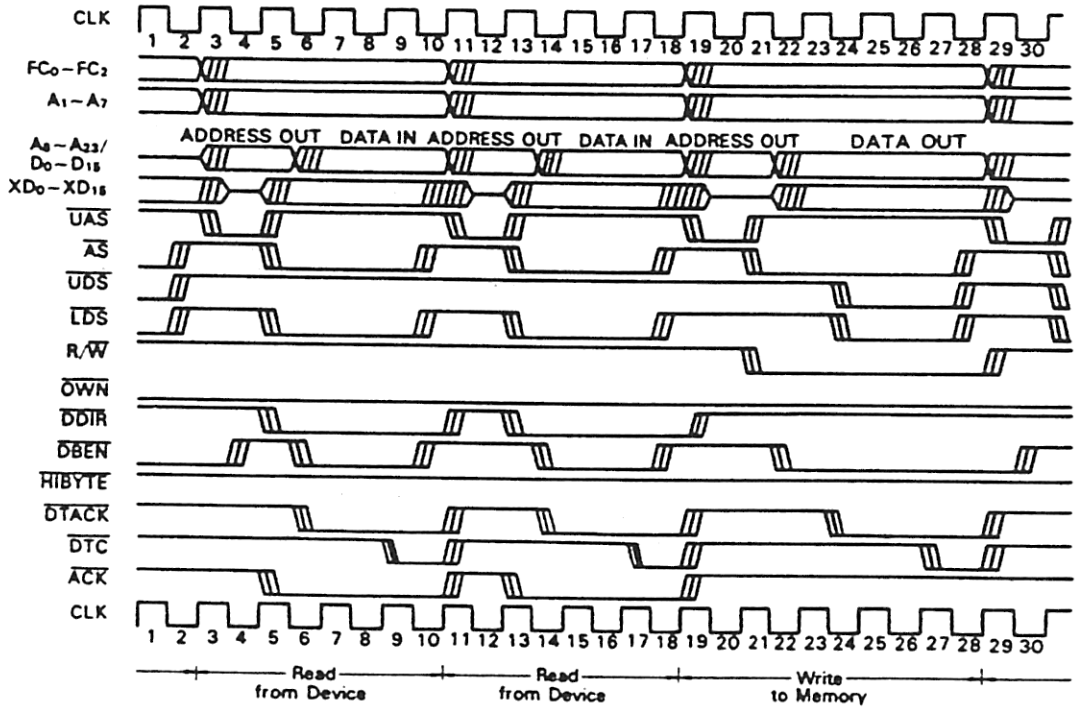
(NOTE) 1) This mode is identical to the dual address with 8 bit device as destination except that the device has a 16 bit port; i.e. both data strobes are asserted and data is on both halves of the bus. See the notes on the dual addressing mode with 8 bit device as source.

Figure 19 Dual Addressing Mode with 16 Bit Device as Destination (Read-Write Cycles)



(NOTE) 1) This mode is identical to the dual address with 8 bit device as source except that the device has a 16 bit port; i.e. both data strobes are asserted and data is on both halves of the bus. See the notes on the dual addressing mode with 8 bit device as source.

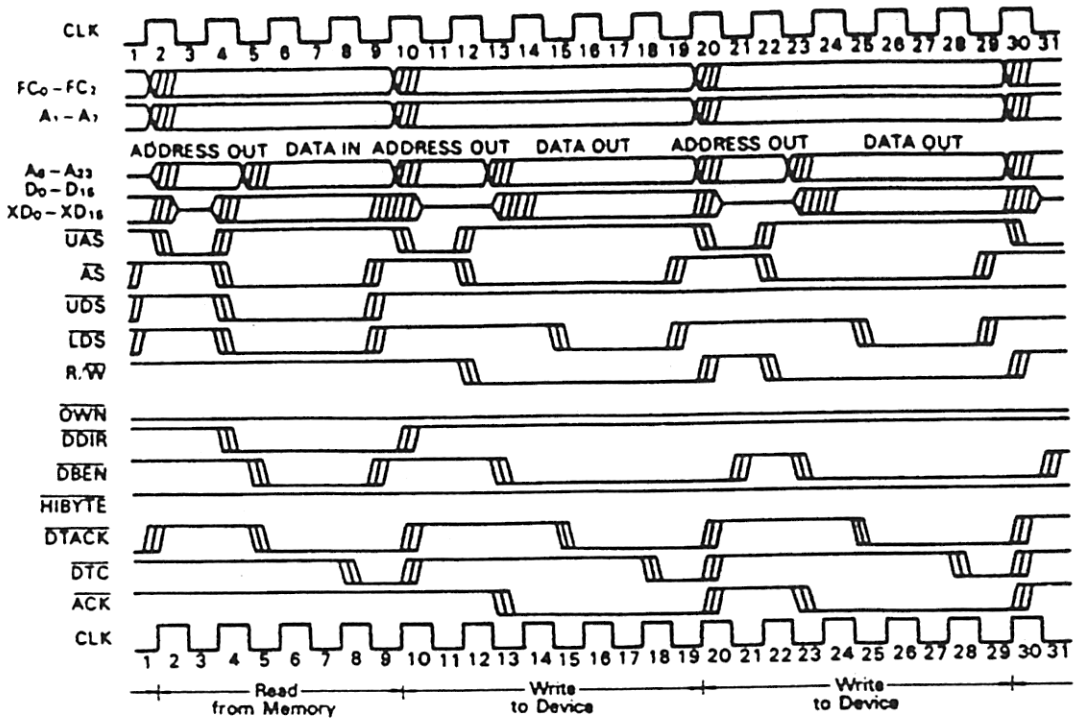
Figure 20 Dual Addressing Mode with 16 Bit Device as Source (Read-Write Cycles)



(NOTES) 1) In this mode the device is the source and memory is the destination.
2) In the dual addressing mode HIBYTE is not used since the DMAC will put the data on the correct half of the data bus.
3) The ACK timing is similar to the single addressing mode ACK. The Read from Device ACK is asserted one clock into the cycle, and negated during the first CLK after the cycle has terminated. The Write to Device ACK is asserted one and one half clocks into the cycle (to avoid buffer conflicts) and removed the first CLK after the cycle has terminated. (See Figure 22)
4) Note that where Data Out is shown (clock 22) the data must be valid on the external data bus by the end of clock 23.

General Notes on Dual Addressing Mode
1) DBEN and DDIR are used in this mode with DBEN always changing on falling edge of CLK and DDIR always changing on rising edge of CLK.
2) Note that for consecutive reads from device or memory, DDIR need not be returned to the inactive (high) state between cycles.
3) Data In should be latched on the last CLK in the cycle (17 above).

Figure 21 Dual Addressing Mode with 8 Bit Device as Source (Read-Write Cycles)



(NOTES) 1) These cycles are similar to the dual addressing mode 8 bit device as source with the exception that the device is now the destination. See the comments that refer to the dual mode 8 bit source.
2) Shown above in the Write to Device, the data strobes are not asserted until clock Δ15 (and Δ25) to allow data setup and travel time.

Figure 22 Dual Addressing Mode with 8 Bit Device as Destination (Read-Write Cycles)

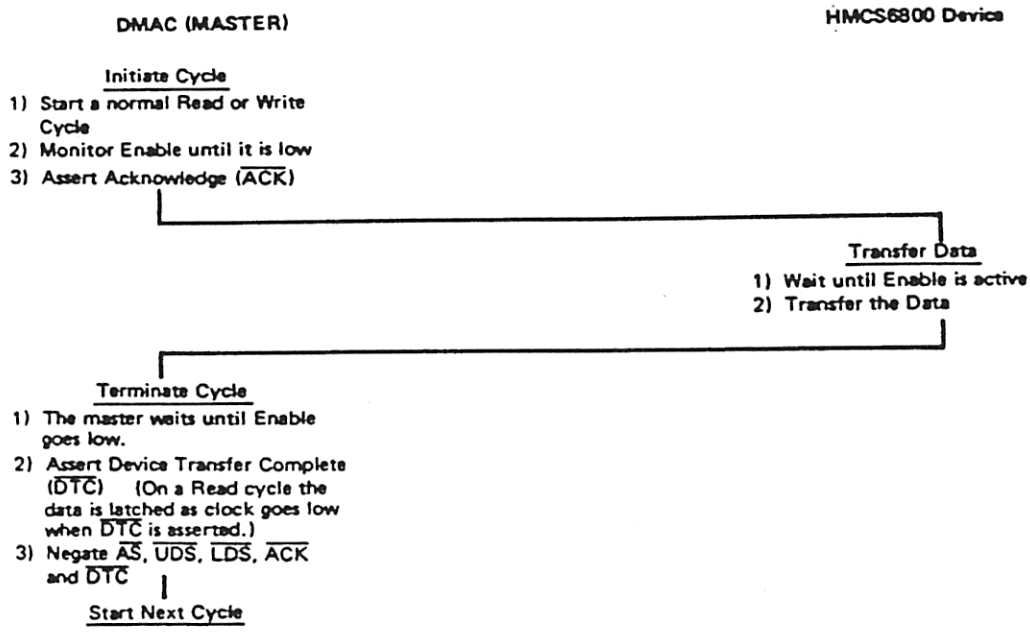
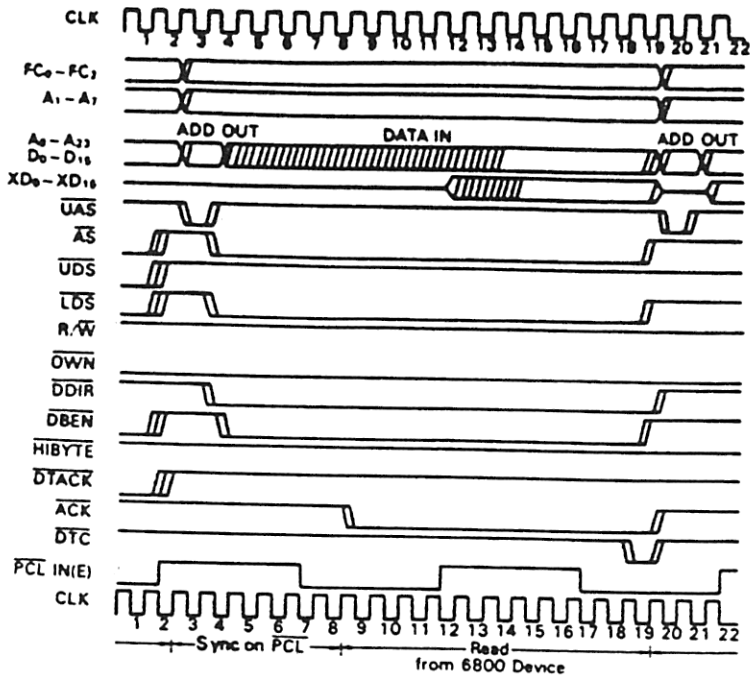


Figure 23 HMCS6800 Cycle Flowchart



(NOTES)
1) The DMAC should latch the data during clock 19.
2) The logic should allow back to back operations on successive E pulses is possible.
3) The ACK low to E high time should be at least 250 ns worst case.
4) The clock reference shown above is the DMAC clock.
5) The E clock duty cycle shown above is an example. A 40% duty cycle is acceptable (4 up, 6 down like HD68000). The E clock must be low for a minimum of 5 clock cycles.

Figure 24 6800 Compatible Dual Addressing Mode (Read Cycle)

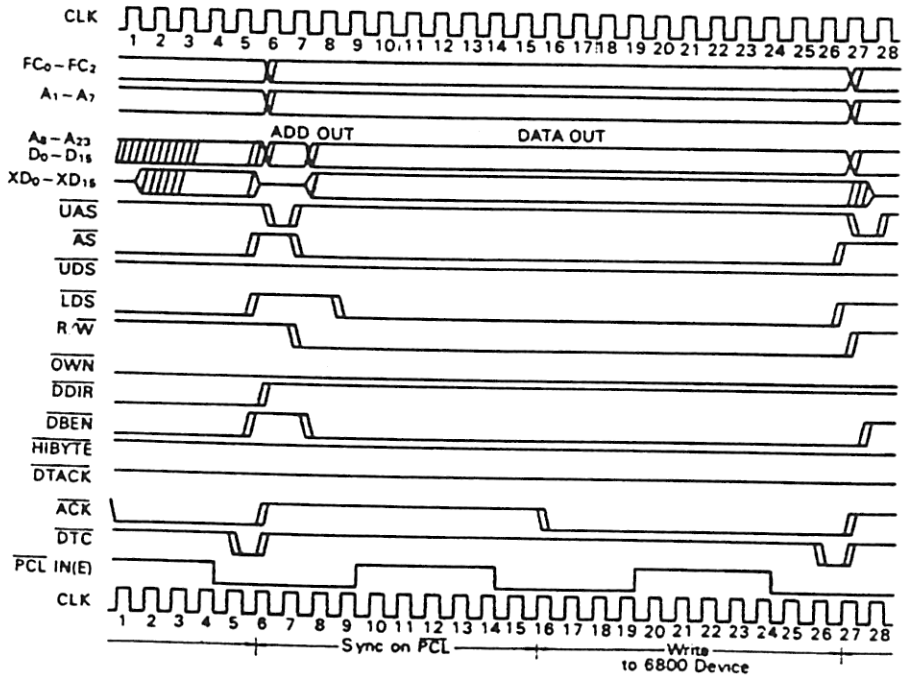


Figure 25 6800 Compatible Dual Addressing Mode (Write Cycle)

An Example of a Dual Address Transfer

This section contains an example of a dual address transfer using Table 7 of Dual-Address Sequencing. The table is reproduced here as Table 2. The transfer mode of this example is the following:

1. Device Port size = 8 bits
2. Operand size = Long Word (32 bits)
3. Memory to Device Transfer
4. Source (Memory) Counts up, Destination (Device) Counts Down
5. Memory Transfer Counter = 2

In this mode, a data transfer from the source (memory) is done according to the 6th row of Table 2, since the port size of the memory is always 16 bits. A data transfer to the destination (device) is done according to the 3rd row of Table 2. Table 3 shows the data transfer sequence.

The memory map of this example is shown in Table 4. The operand consists of BYTE A through BYTE D in memory of Table 4. Prior to the transfer, MAR and DAR are set to 00000012 and 00000108 respectively. The operand is transferred to the 8 bit port device according to the order of transfer number in Table 3.

Table 2 Dual-Address Sequencing (Table 7)

Row No.	Port Size	Operand Size	Part Size	Operand Part Addresses	Address Increment		
					+	=	-
1	8	BYTE	BYTE	A	+2	0	-2
2	8	WORD	BYTE	A, A+2	+4	0	-2
③	8	LONG	BYTE *4	A, A+2, A+4, A+6 *3 *5 *7 *8	+8	0	-8 *10
4	16	BYTE	PACK	A	+P	0	-P
5	16	WORD	WORD	A	+2	0	-2
⑥	16	LONG	WORD *2	A, A+2 *1 *6	+4 *9	0	-4

Table 3 An Example of a Data Transfer for One Operand

SRC: Source (Memory), DST Destination (Device), HR: Holding Register (DMAC Internal Reg.)

Transfer No.	Data Transfer	Address Output	Data Size on Bus	DMAC Registers after Transfer		Comment
				MAR	DAR	
0	–	–	–	00000012	00000108	Initial Register Setting
1	SRC → HR	00000012 *1	WORD *2	00000014	00000108	Higher order 16 bits of operand is fetched.
2	HR → DST	00000108 *3	BYTE *4	00000014	0000010A	Higher order 16 bits of operand is transferred.
3	HR → DST	0000010A *5	BYTE *4	00000014	0000010C *10	
4	SRC → HR	00000014 *6	WORD *2	00000016 *9	0000010C	Lower order 16 bits of operand is fetched
5	HR → DST	0000010C *7	BYTE *4	00000016	0000010E	Lower order 16 bits of operand is transferred.
6	HR → DST	0000010E *8	BYTE *4	00000016	00000110 *10	
6'	–	–	–	00000016	00000110	MAR, DAR are pointing the next operand addresses when the transfer is complete.

Mode: Port size = 8, Operand size = Long Word, Memory to Device, Source (Memory) Counts Up, Destination (Device) Counts Down

Table 4 Memory Map for the Example of the Data Transfer

ADDRESS			ADDRESS
00000010			00000011
00000012	BYTE A *1	BYTE B *1	00000013
00000014	BYTE C *6	BYTE D *6	00000015
00000016			00000017

Source (Memory)

ADDRESS			
00000106			00000107
00000108	BYTE A *3		00000109
0000010A	BYTE B *5		0000010B
0000010C	BYTE C *7		0000010D
0000010E	BYTE D *8		0000010F
00000110			00000111

Destination (Device)

SINGLE ADDRESSING MODE

Implicitly addressed devices do not require addressing of data register before data may be transferred. Transfers between memory and these devices are controlled by the request/acknowledge protocol. Such peripherals require only one bus cycle to transfer data between themselves and memory, and the DMAC internal holding register is not used. Because only the memory is addressed during a data transfer, this protocol is also called single-address.

Device with $\overline{ACK}$ Transfers

Under this protocol, the device is not explicitly addressed and communication is performed with a two signal request/acknowledge handshake. When a request is generated using the request method programmed in the control registers, the DMAC obtains the bus and responds with acknowledge. The DMAC asserts all HMCS68000 bus control signals needed for the transfer.

When the transfer is from memory to a device, data is valid when $\overline{DTACK}$ is asserted and remains valid until the data strobes are negated. The assertion of $\overline{DTC}$ from the DMAC may be used to latch the data, as the data strobes are not removed until 1/2 clock after the assertion of $\overline{DTC}$.

When the transfer is from device to memory, data must be valid on the HMCS68000 bus before the DMAC asserts the data strobes. The data strobes are asserted one clock period after $\overline{ACK}$ is asserted. Further definition of this protocol is explained in Figures 26, 27 and timing diagrams in Figures 28 and 29.

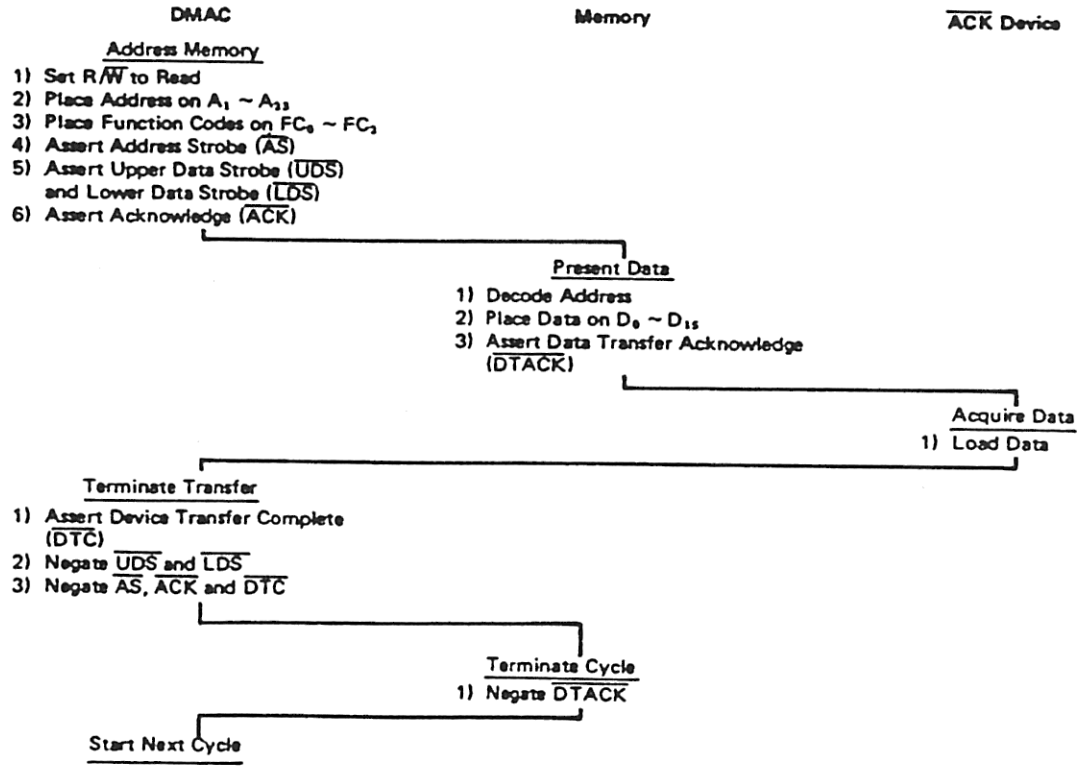


Figure 26 Word from Memory to Device with $\overline{ACK}$

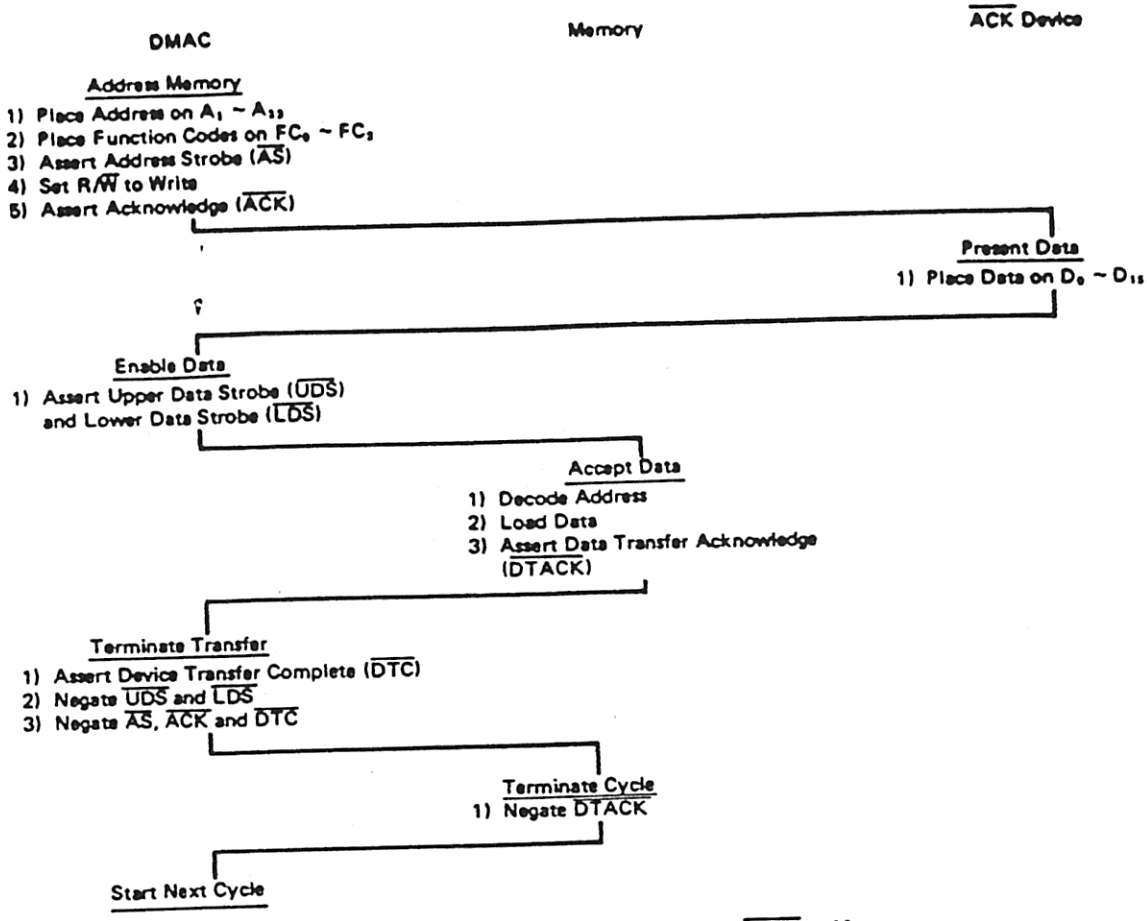
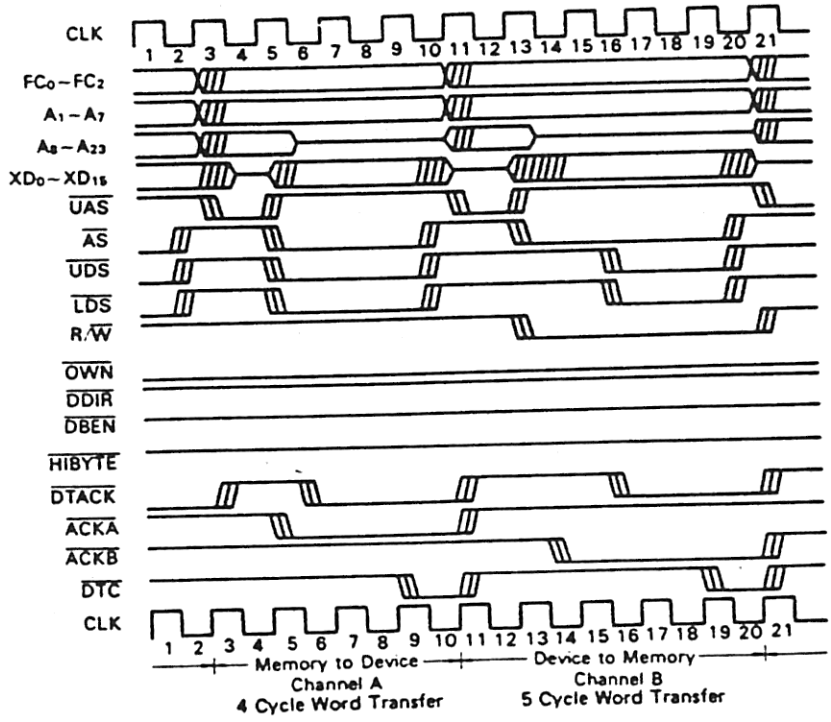
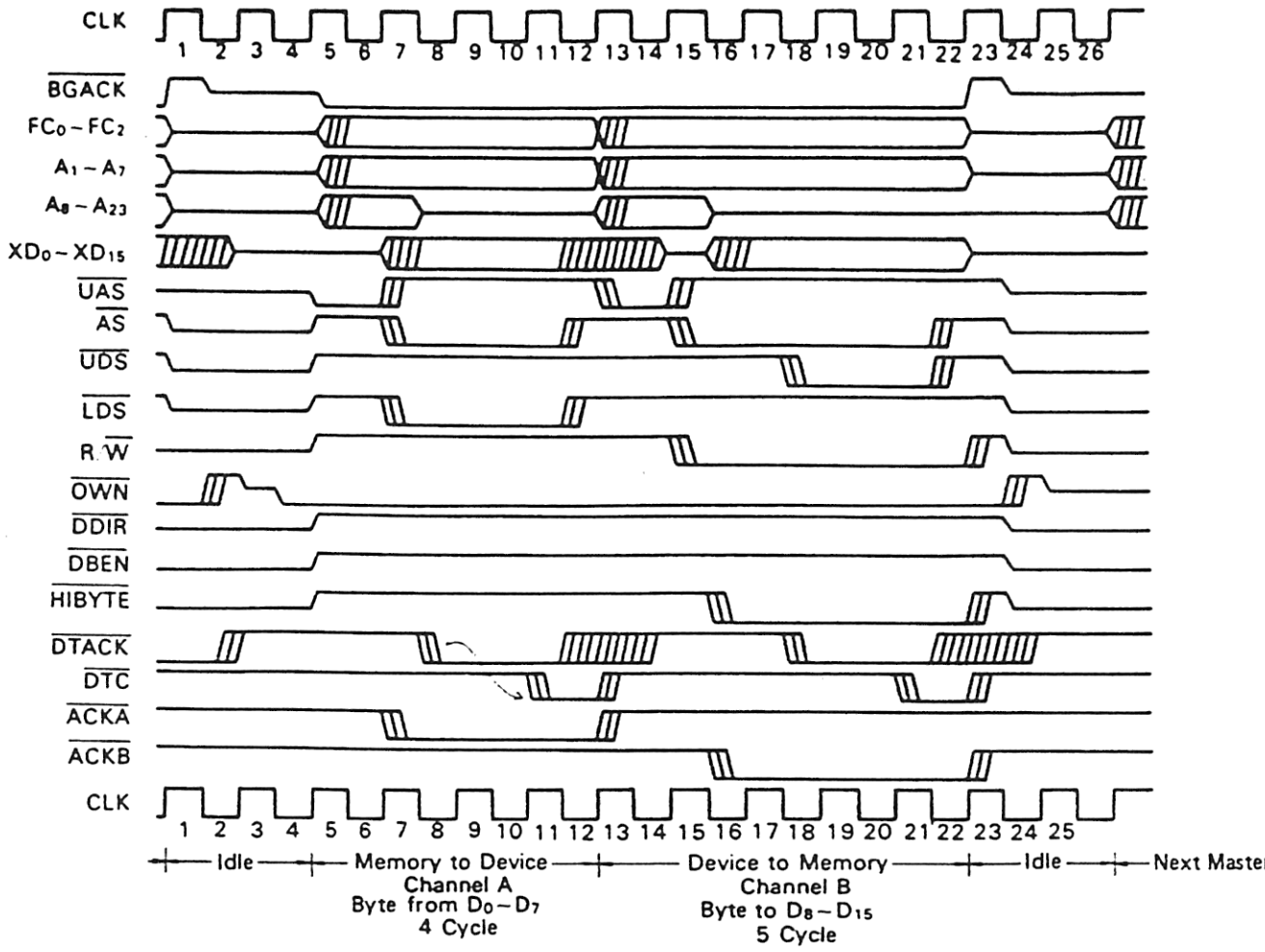


Figure 27 Word from Device with ACK to Memory



(NOTES) 1) These cycles are identical to the 8 bit transfers with the exception that all transfers are 16 bits (word) so that both data strobes are always asserted and HIBYTE is always inactive. See the comments on single addressing mode with 8 bit devices.
2) The A and B on the ACK signals are there to distinguish two different channels. The actual channel numbers could be any of 0 to 3.

Figure 28 Single Addressing Mode with 16 Bit Devices as Sources and Destinations (Read-Write Cycles)



(NOTES)
1) Any signal change shown before clock 4 is due to the previous master (MPU or other DMAC).
2) All signals except $\overline{OWN}$ tristate during the clock 2 after the end of the last DMAC cycle. $\overline{OWN}$ tristates one half clock later.
3) $\overline{DDIR}$ and $\overline{DBEN}$ remain in the inactive state during single address mode cycles.
4) A four (4) cycle transfer is the minimum cycle length for a Memory to Device transfer.
5) A five (5) cycle transfer is the minimum cycle length for a Device to Memory transfer.
6) $\overline{HIBYTE}$ (when used as fold signal) is used to gate the high bus data to the low bus during a Memory to Device transfer, and low bus data to the high bus during a Device to Memory transfer.
7) Note that if the transfer is 8 bits wide, only one data strobe (and possibly $\overline{HIBYTE}$) is asserted.
8) Address bits 8 through 23 are shown in tristate beginning clock 8 (and 16). In this mode only they can be driven similar to A_1 through A_7 if need be.

The followings are notes on the Memory to Device transfer.

9) $\overline{ACK}$ is not asserted until one clock into the cycle (clock 7) due to the fact that it may have just been negated (clock 5). It must remain negated for at least one clock.
10) $\overline{DTC}$ is asserted one clock before the end of the cycle to signal the peripheral that the data is valid and should be latched at this time. It also indicates that a successful transfer is being completed (no Berr, Retry, etc.).

The followings are notes on the Device to Memory transfer.

11) $\overline{ACK}$ cannot be asserted until one and one half clocks into the cycle (clock 16) to allow the R/$\overline{W}$ signal to settle in the write mode (clock 15). R/$\overline{W}$ cannot be asserted until one clock into the cycle (clock 15) because of buffer collisions.
12) The data strobes cannot be asserted until clock 18 to allow the data to become valid at the memory (data setup).
13) The A and B on the $\overline{ACK}$ signals are there to distinguish two different channels. The actual channel numbers could be any of 0 to 3.

Figure 29 Single Addressing Mode with 8 Bit Devices as Sources and Destinations

Device with $\overline{ACK}$ and $\overline{READY}$ Transfers

Under this protocol, the device is not explicitly addressed and communication is performed using a three signal request/acknowledge/ready handshake. The ready input to the DMAC is provided by the $\overline{PCL}$ line, the use of this protocol forces it to be an active low input. When a request is generated using the request method programmed in the control registers, the DMAC obtains the bus and asserts acknowledge to notify the device that the transfer is to take place. The DMAC asserts all HMCS68000 bus control signals needed for the transfer and holds them until the device responds with $\overline{READY}$. After $\overline{READY}$ is received the bus cycle terminates normally.

When the transfer is from memory to a device, data is valid when $\overline{DTACK}$ is asserted and is valid until the data strobes are negated. The assertion of $\overline{DTC}$ from the DMAC may be used to latch the data, as the data strobes are not removed until 1/2 clock after the assertion of $\overline{DTC}$.

When the transfer is device to memory, data must be valid on the HMCS68000 bus before the DMAC asserts the data strobes. The data strobes are held asserted until the device asserts $\overline{READY}$. Further definition of this protocol is explained in Figures 30, 31 and the timing diagrams in Figure 32.

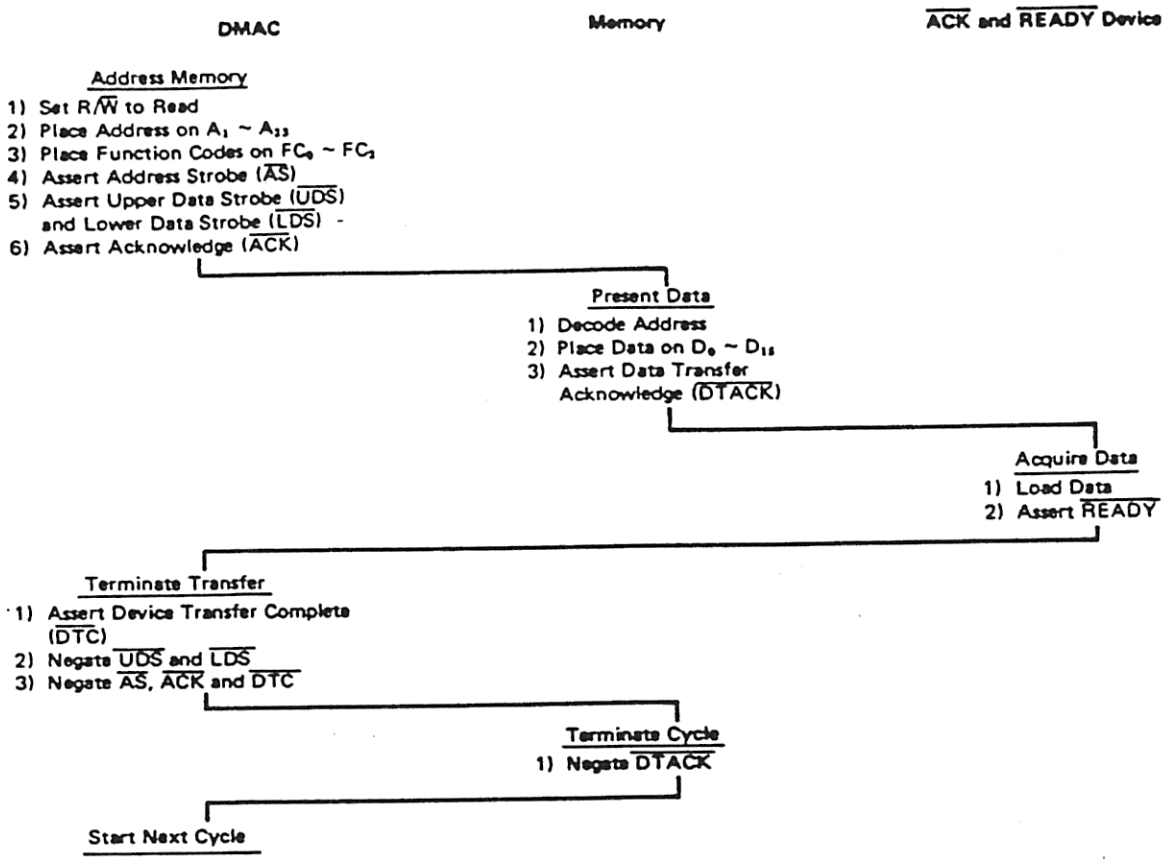


Figure 30 Word from Memory to Device with $\overline{ACK}$ and $\overline{READY}$

DMAC | **Memory** | **$\overline{ACK}$ and $\overline{READY}$ Device**

Address Memory
1) Place Address on $A_1 \sim A_{23}$
2) Place Function Codes on $FC_0 \sim FC_2$
3) Assert Address Strobe ($\overline{AS}$)
4) Set R/$\overline{W}$ to Write
5) Assert Acknowledge ($\overline{ACK}$)

Present Data
1) Place Data on $D_0 \sim D_{15}$
2) Assert $\overline{READY}$

Enable Data
1) Assert Upper Data Strobe ($\overline{UDS}$) and Lower Data Strobe ($\overline{LDS}$)

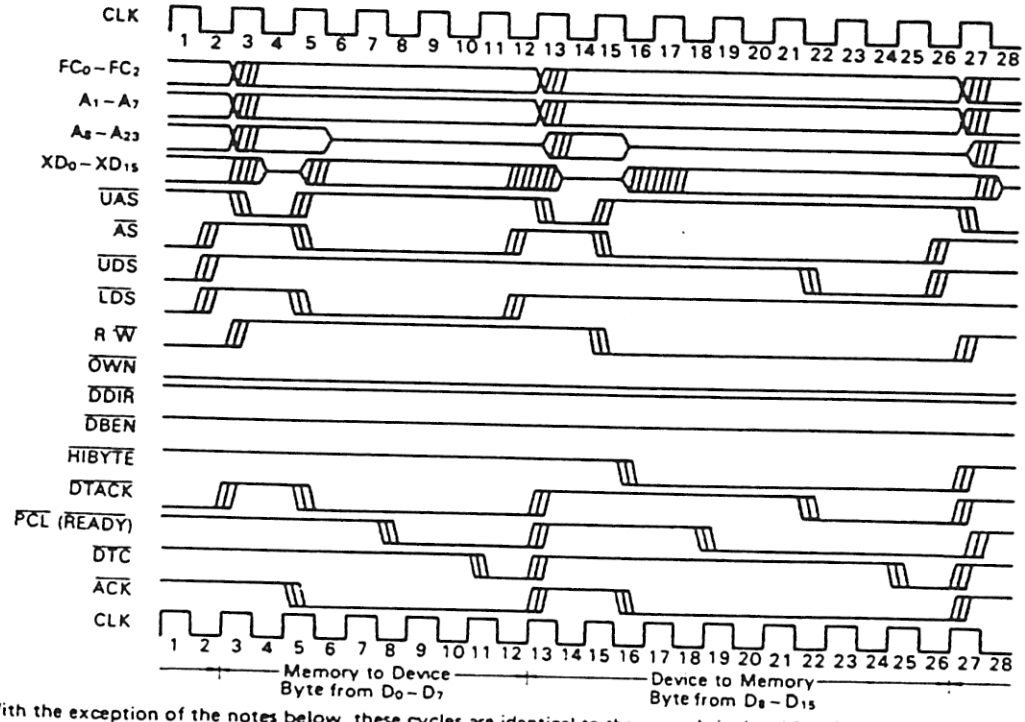
Accept Data
1) Decode Address
2) Load Data
3) Assert Data Transfer Acknowledge ($\overline{DTACK}$)

Terminate Transfer
1) Assert Device Transfer Complete ($\overline{DTC}$)
2) Negate $\overline{UDS}$ and $\overline{LDS}$
3) Negate $\overline{AS}$, $\overline{ACK}$ and $\overline{DTC}$

Terminate Cycle
1) Negate $\overline{DTACK}$

Start Next Cycle

Figure 31 Word from Device with $\overline{ACK}$ and $\overline{READY}$ to Memory



(NOTES)
1) With the exception of the notes below, these cycles are identical to the normal single addressing mode cycles. See the comments on the 8 bit single addressing mode transfer.
2) In the Memory to Device transfer only, the $\overline{READY}$ ($\overline{PCL}$) line is used as a "second $\overline{DTACK}$" i.e. both $\overline{READY}$ and $\overline{DTACK}$ are required to terminate the cycle.
3) In the Device to Memory transfer, the $\overline{READY}$ input is used to delay the assertion of the data strobes. Once $\overline{READY}$ is detected, the data strobes are asserted and $\overline{DTACK}$ is sampled to terminate the cycle. $\overline{AS}$ is asserted at the beginning of the cycle as usual.

Figure 32 Single Addressing Mode with 8 Bit Devices as Sources and Destinations with $\overline{PCL}$ Used as a $\overline{READY}$ Input (Read-Write Cycles)

(2) OPERANDS AND ADDRESSING

Three factors enter into how the actual data is handled: port size, operand size and address sequencing.

PORT SIZE

The DCR is used to program the device port size

DPS Device Port Size
- 0 8 bit port
- 1 16 bit port

The port size is the number of bits of data which the device can transfer in a single bus cycle. During a DMAC bus cycle, a 16-bit port transfers 16 bits of data on $D_0 \sim D_{15}$, while an 8-bit port transfers 8 bits of data, either on $D_0 \sim D_7$ or on $D_8 \sim D_{15}$. The memory is always assumed to have a port size of 16.

OPERAND SIZE

OCR is used to program the operand size.

SIZE Operand Size
- 00 Byte
- 01 Word
- 10 Long word
- 11 (undefined, reserved)

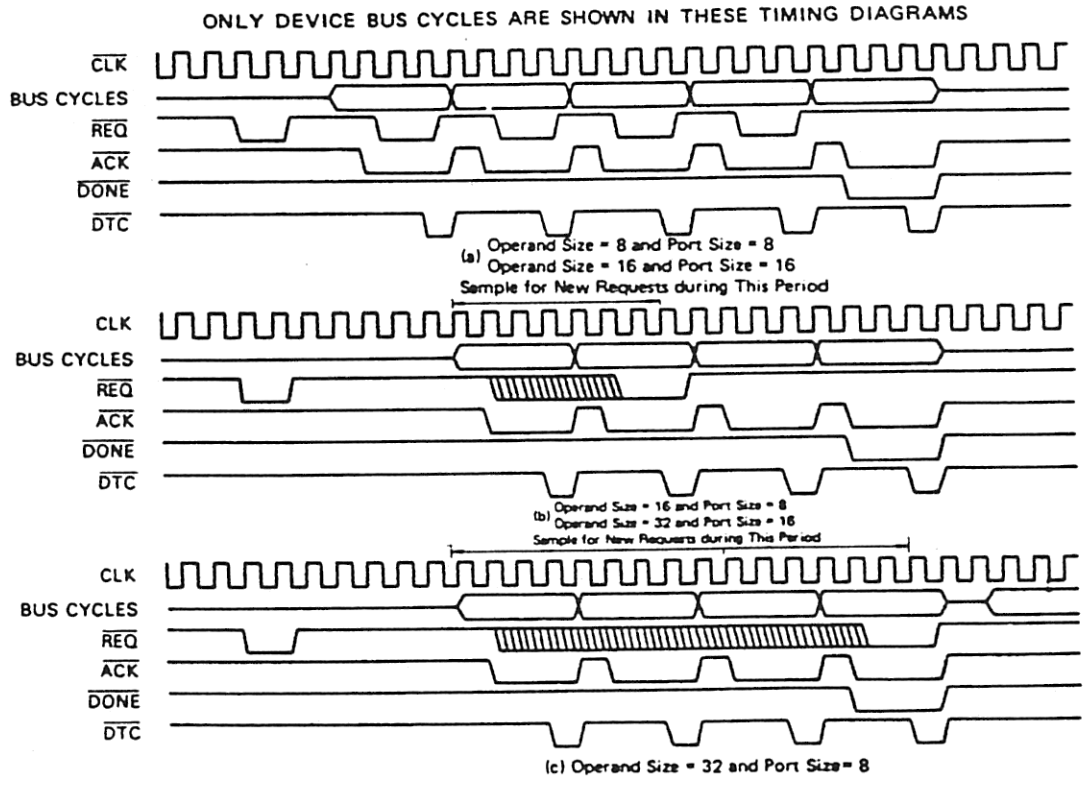
The operand size is the number of bits of data to be transferred to honor a single request. Multiple bus cycles may be required to transfer the operand through the device port. A byte operand consists of 8 bits of data, a word operand consists of 16 bits of data, a long word operand consists of 32 bits of data. The transfer counter counts the number of operands transferred.

For single-address operations, the port size and the operand size must be the same. 68000 and 6800 type devices may not use byte operands when the port size is 16 bits and the request generation method is the request pin. (REQG = 10 or 11)

Table 5 Operation Combinations

Addressing	Device Type	Port	Operand			REQG*
			Byte	Word	LW	
Dual	68000, 6800	8	Yes	Yes	Yes	00, 01, 10, 11
Dual	68000, 6800	16	Yes	Yes	Yes	00, 01
Dual	68000, 6800	16	No	Yes	Yes	10, 11
Single	with $\overline{ACK}$ or	8	Yes	No	No	00, 01, 10, 11
	$\overline{ACK}$ & $\overline{READY}$	16	No	Yes	No	00, 01, 10, 11

*Refer to Page 11.



(NOTES)
1) The above cycles reflect both dual and single addressing mode.
2) In the dual addressing case, the memory references are not shown above, only the device references. Therefore all cycles have $\overline{ACK}$ asserted.
3) Note that when the operand size exceeds the port size, the DMAC should sample the request pins (new priority check) until the $\overline{DTC}$ of the last bus cycle has occurred.

Figure 33 Request and Acknowledge Generation vs. Operand Size

ADDRESS SEQUENCING

The sequence of addresses generated depends upon the port size, operand size, whether the addresses are to count up, down, or not change and whether the transfer is explicitly or implicitly addressed. The Sequence Control Register is used to program the memory address count method and the device address count method.

MAC Memory address count
- 00 Memory address register does not count
- 01 Memory address register counts up
- 10 Memory address register counts down
- 11 (undefined, reserved)

DAC Device address register count
- 00 Device address register does not count
- 01 Device address register counts up
- 10 Device address register counts down
- 11 (undefined, reserved)

Single-Address Transfers

Single-address transfers require the device port size and the operand size to be equal. Address sequencing is determined by the port size and the sequence control register as shown in Table 6. If the operand size is byte, the memory address increment is one (1). If the operand size is word, the memory address increment is two (2). If the memory address register does not count, the memory address is unchanged after the transfer. If the memory address counts up, the increment is added to the memory address; if the memory address counts down, the increment is subtracted from the memory address. The memory address is changed after the operand is transferred.

Table 6 Single Address Sequencing

Port Size	Operand Size	Memory Address Increment		
		+	=	-
8	Byte	+1	0	-1
16	Word	+2	0	-2

Dual-Address Transfers

In dual-address operations, the operand size need no match the port size: Thus the transfer of an operand may require several transfers between device and memory. Each pair of transfers, between memory and DMAC and between DMAC and device, transfers a portion of the operand, called the operand part.

The addresses of the operand parts are in a linear increasing sequence. The step between the addresses of parts is two (2). The size of the parts is the minimum of the port size and operand size. The number of parts is the operand size divided by the port size. The address increment is added or subtracted after the operand is transferred.

If the port size is 16 bits, the operand size is byte, and the request generation method is auto request or auto request at a limited rate, the DMAC packs consecutive transfers. This means that word transfers are made from the associated address with an address increment of two (2). If the initial source address location contains a single byte, the first transfer is a byte transfer to the internal DMAC holding register, and subsequent transfers from the source are word transfers. If the initial destination location contains a single byte, the first transfer is a byte transfer from the internal DMAC holding register, and any remaining byte remains in the holding register. Likewise, if either the final source or destination location contains a single byte, only a byte transfer is done. Packing is not performed if the address does not count; each byte is transferred by a separate access to the same location.

(3) ADDRESS REGISTER OPERATION

The DMAC has three 32-bit address registers per channel: the memory address register (MAR), the device address register (DAR), and the base address register (BAR).

The MAR is used in all operations because of the assumption that all operations are between memory and a device. The MAR is sequenced as previously described. This register is either initialized before the channel operation is started, or is loaded during chaining or continue operations which are defined in a later section.

Table 7 Dual-Address Sequencing

Port Size	Operand Size	Part Size	Operand Part Address	Address Increment		
				+	=	-
8	Byte	Byte	A	+2	0	-2
8	Word	Byte	A, A+2	+4	0	-4
8	Long	Byte	A, A+2, A+4, A+6	+8	0	-8
16	Byte	Pack	A	+P	0	-P
16	Word	Word	A	+2	0	-2
16	Long	Word	A, A+2	+4	0	-4

P = 1 if packing is not done
= 2 if packing is done

The DAR is used to address devices or memory in dual-address operations. It is initiated before starting the channel operation, and is sequenced as previously described.

The BAR register is used only in chaining or continue operations. It is sequenced only in regard to chaining operations.

(4) FUNCTION CODE REGISTER OPERATIONS

There are three function code registers per channel: the memory function code register, the device function code register, and the base function code register. The function code registers correspond to the address registers and are output on $FC_0 \sim FC_2$ when the corresponding address register provides the address for a DMA bus cycle.

(5) TRANSFER COUNT REGISTER OPERATION

The DMAC has two 16-bit transfer counter registers per channel: the memory transfer counter, and the base transfer counter.

The memory transfer counter is used in all operations to count the number of operands transferred in a block. The mem-

ory transfer counter is decremented by one as each operand is transferred. This register is either initialized before the channel operation is started, or is loaded during chaining or continue operations.

Both the memory transfer counter and the base transfer counter have a terminal count of zero (0). If either register is initialized or loaded with a terminal count when the channel is configured to use that register, a count error is signaled.

- **INITIATION AND CONTROL OF CHANNEL OPERATION**

The Channel Control Register provides mechanisms for starting, continuing, halting, or aborting an operation. It also controls the enabling of interrupts from a channel.

(1) OPERATION INITIATION

To initiate the operation of a channel the STR bit of the CCR is set to start the operation. Setting the STR bit causes the immediate activation of the channel, the channel will be ready to accept requests immediately. The channel initiates the operation by cleaning the STR bit and setting the channel active bit in the CSR. Any pending requests are cleared, and the channel is then ready to receive requests for the new operation. If the channel is configured for an illegal operation, the configuration error is signaled, and no channel operation is run. The illegal operations include the selection of any of the options marked "(undefined, reserved)". If the operation is dual-address, the device address register should have been previously initialized. The channel cannot be started if any of the ACT, COC, BTC, NDT or ERR bits is set in the CSR. In this case, the channel signals the operation timing error.

If the operation is unchained, the memory address register and the memory transfer counter should have been previously initialized.

If the operation is chained, the base address register, and the base transfer counter should have been previously initialized.

(2) OPERATION CONTINUATION

The continue bit (CNT) allows multiple blocks to be transferred in unchained operations. The CNT bit is set in order to continue the current channel operation. If an attempt is made to continue a chained operation, a configuration error is signaled. The base address register and base transfer counter should have been previously initialized.

The continue bit may be set as the channel is started or while the channel is still active. The operation timing error bit is signaled if a continuation is otherwise attempted.

(3) HALT

The CCR has a halt bit which allows suspension of the operation of the channel. If this bit is set, a request may still be generated and recognized, but the DMAC does not attempt to acquire the bus or to make transfers for the halted channel. When this bit is reset, the channel resumes operation and services any request that may have been received while the channel was halted.

(4) SOFTWARE ABORT

The CCR has a software abort bit (SAB) which allows the current operation of the channel to be aborted. The writing of a one (1) into the SAB bit causes a channel abort error to be signaled. When the CCR is read, the SAB always reads as zero (0).

(5) INTERRUPT ENABLE

The CCR has an interrupt enable bit (INT) which allows the channel to request interrupts on the completion of block transfers or on the termination of channel operations. If INT is set, the channel can request interrupts. If it is clear, the channel may not request interrupts.

- **BLOCK TERMINATION**

As part of the transfer of an operand, the DMAC decrements the memory transfer counter. If this counter is decremented to the terminal count, the transfer counter is exhausted and the operand is the last operand of the block. The channel operation is complete if the operation is unchained and there is no continuation, or if the operation is chained and the chain is exhaused. The DMAC notifies the device of channel completion via the $\overline{\text{DONE}}$ output. When the transfer has been completed, the ACT bit of the CSR is cleared, and the COC bit is set.

The occurrence of a bus exception during a bus cycle being run for a channel, or the occurrence of some error in the channel terminates the block transfer and the channel operation. The bit of the CER corresponding to the error being signal is set. The ACT of the CSR is cleared and the COC and ERR bits are set.

(1) CHANNEL STATUS REGISTER

The channel status register contains the status of the channel. The register is cleared by writing a one (1) into each bit of the register to be cleared. Those bits positions which contain a zero (0) in the write data remain unaffected.

COC

The channel operation complete bit is set if the DMA transfer has completed. The COC bit is set following the termination, whether successful or not, of any DMA operation. This bit must be cleared in order to start another channel operation. This bit is cleared only by writing the channel status register or resetting the DMAC.

PCS

The peripheral status bit reflects the state of the $\overline{\text{PCL}}$ I/O line regardless of its programmed function. This bit is unaffected by write operations.

PCT

The peripheral control transition bit is set if an falling edge transition has occurred on the $\overline{\text{PCL}}$ line. This bit is cleard only by writing the channel status register or resetting the DMAC.

BTC

Block transfer complete is set when the memory transfer count is exhausted, the operation is unchained, and the continue bit is set. This bit must be cleared before another continuation is attempted, otherwise an operation timing error is signaled. This bit is cleared only by writing the channel status register or resetting the DMAC.

NDT

Normal device termination is set when the device terminates the channel operation by asserting the $\overline{\text{DONE}}$ line while the device was being acknowledged. This bit is cleared only by writing the channel status register or resetting the DMAC.

ERR

This bit is used to report the occurrence of error conditions. It is set if any errors have been signaled. This bit is cleared only by writing the channel status register or resetting the DMAC.

ACT

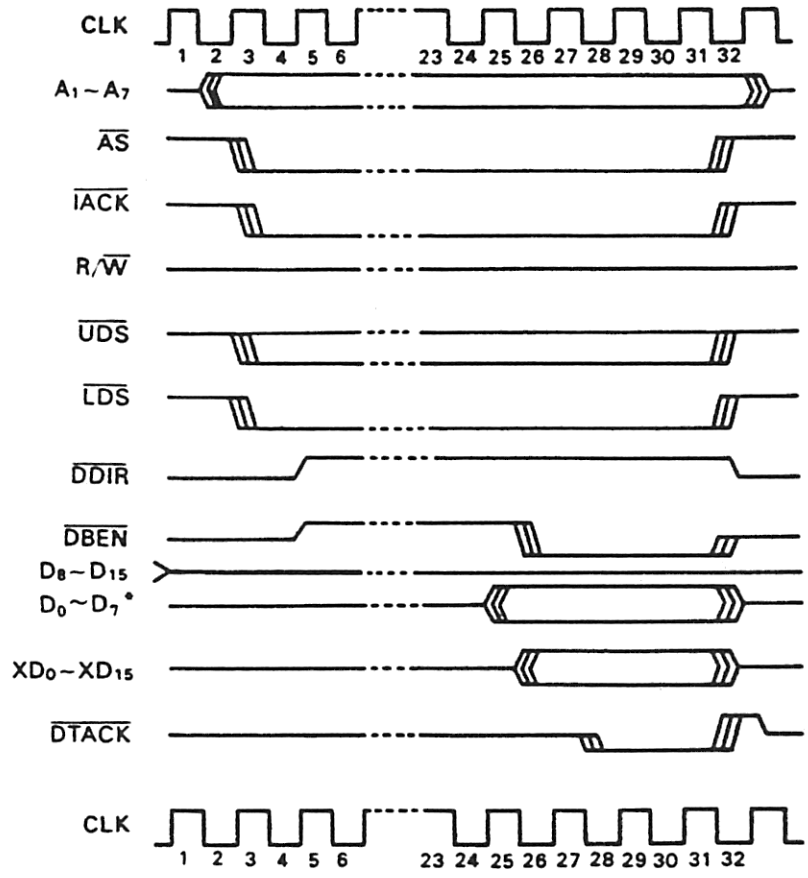
This is the channel active bit. It is asserted after the channel has been started. The bit remains set until the channel operation terminates. This bit is unaffected by write operations.

(2) INTERRUPTS

The INT bit of the CCR determines if an interrupt can be generated. The interrupt request is generated if INT is set and the bits $\overline{COC}$ or BTC are set in the CSR or the PCT bit is set and the $\overline{PCL}$ line is programmed to be an interrupt input.

If a channel has an interrupt request, the DMAC makes an interrupt request by asserting the $\overline{IRQ}$ output. If the DMAC has an interrupt request pending, and receives an $\overline{IACK}$ from the MPU the DMAC provides an interrupt vector. If multiple channels have interrupt requests pending, the determination of which channel presents its interrupt vector is made using the same priority scheme defined for channel operations.

The interrupt vector returned to the MPU comes from either the normal or the error interrupt vector register. The normal interrupt register is used unless the ERR bit of CSR is set, in which case the error interrupt vector register is used. The content of the interrupt vector register is placed on D_0 ~ D_7, and $\overline{DTACK}$ is asserted to indicate that the vector is on the data bus. If a reset bus exception occurs, all interrupt vector registers are set to $0F (binary 00001111), the value of the uninitialized interrupt vector.



* MPX A_8~A_{23}/D_8~D_{15} pins
Interrupt Vector Register is Output.

(NOTES) 1) This cycle is similar to the chip select cycle except it is triggered by $\overline{IACK}$. See the notes on the chip select cycle.
2) $\overline{IACK}$ will be negated within one clock after $\overline{AS}$ is negated.
3) The clock referenced above is the CPU clock.

Figure 34 MPU $\overline{IACK}$ Cycle to DMAC

(3) MULTIPLE BLOCK OPERATION

When the memory transfer counter is exhausted, there are further blocks to be transferred if the channel is chained and the chain is not exhausted. The DMAC provides the reinitialization of the memory address register and the memory transfer counter in these cases.

CONTINUED OPERATIONS

When the memory transfer counter is exhausted and the continue bit of the CCR is set, the DMAC performs a continuation of the channel operation. The base address, base function code, and base transfer count registers are copied into the memory address, memory function code, and memory transfer count registers. The block transfer complete (BTC) bit of the CSR is set, the continue bit is reset, and the channel begins a new block transfer. If the memory transfer counter is loaded with a terminal count, the count error is signaled.

ARRAY CHAINING

This type of chaining uses an array in memory consisting of memory addresses and transfer counts. Each entry in the array is six bytes long and, consists of four bytes of address followed by two bytes of transfer count. The beginning address of this array is in the base address register, and the number of entries in the array is in the base transfer counter. Before starting any block transfers, the DMAC fetches the entry currently pointed to by the base address register. The address information is placed in the memory address register, and the count information is placed in the memory transfer counter. As each chaining entry is fetched, the base transfer counter is decremented by one. After the chaining entry is fetched, the base address register is incremented to point the next entry. When the base transfer counter reaches a terminal count, the chain is exhausted, and the entry just fetched determines the last block of the channel operation.

The memory format for supporting the Array Chaining is shown in Figure 35. The array must start at an even address, or the entry fetch results is an address error. If a terminal count is loaded into the memory transfer counter, the count error is signaled. Since the base registers may be read by the MPU, appropriate error recovery information is available should the DMAC encounter an error anywhere in the chain.

LINKED CHAINING

This type of chaining uses a list in memory consisting of memory address, transfer counts, and link addresses. Each entry in the chain list is ten bytes long, and consists of four bytes of memory address, two bytes of transfer count and four bytes of link address. The address of the first entry in the list is in the base address register, and the base transfer counter is unused. Before starting any block transfers, the DMAC fetches the entry currently pointed to by the base address register. The address information is placed in the memory address register, the count information is placed in the memory transfer counter, and the link address replaces the current contents of the base address register. The channel then begins a new block transfer. As each chaining entry is fetched, the update base address register is examined for the terminal link which has all 32 bits equal to zero. When the new base address is the terminal address, the chain is exhausted, and the entry just fetched determines the last block of the channel operation.

The memory format for this type of chaining is shown in Figure 36. This type of chaining allows entries to be easily removed or inserted without having to reorganize data within the chain. Since the end of the chain is indicated by a terminal link, the number of entries in the array need not be specified to the DMAC. All entries in the array must start at even address, or the entry fetch results in an address error. If a terminal count is loaded into the memory transfer counter, the count error is signaled. Becaused the MPU can read all of the DMAC registers, all necessary error recovery information is available to the operating system.

Table 8 Chaining Mode Address/Count Information

Chaining Mode	Base Addr. Register	Base TC	Completed When
Array Chaining	BA of Array	No. of Entries In Array	Base Transfer Counter – 0
Linked Chaining	BA of Array	–	Pointer = 0

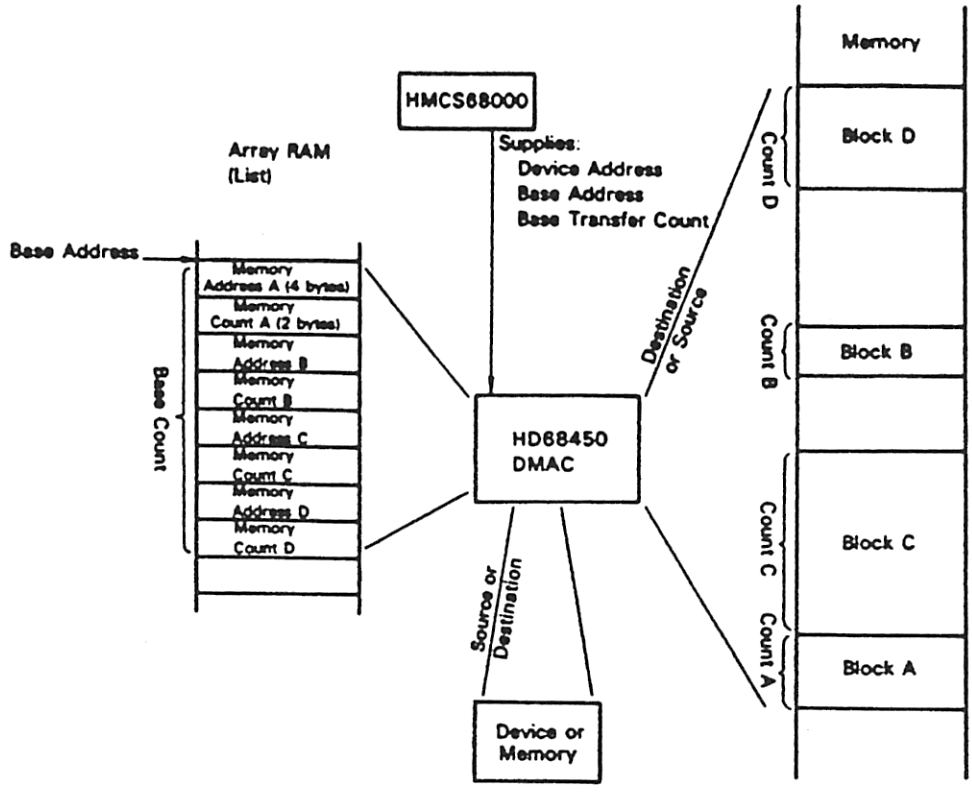


Figure 35 Array Chain Transfer

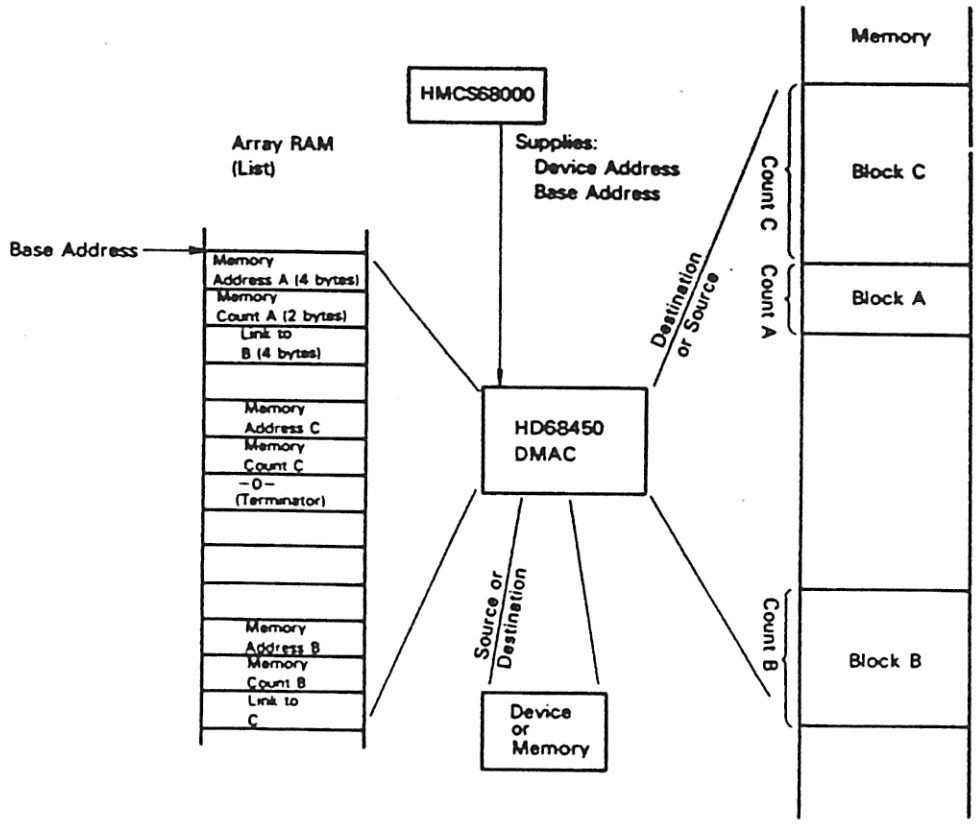


Figure 36 Linked Array Chain Transfer

(4) BUS EXCEPTION CONDITIONS

The DMAC has three lines for bus exception conditions. A priority encoder can be used to generate these signals. In order to guarantee reliable decoding, the DMAC verifies that the incoming code has been stable for two DMAC clock cycles before acting on it. The lines are encoded in the following manner (0 = active).

$\overline{BEC}$ 2 1 0

1 1 1 – No exception condition
1 1 0 – Halt
1 0 1 – Bus error
1 0 0 – Retry
0 1 1 – Relinquish bus and retry
0 1 0 – (undefined, reserved)
0 0 1 – (undefined, reserved)
0 0 0 – Reset

These signals indicate the presence of bus exceptions. All bus exceptions except halt are preemptory. The occurrence of a preemptory bus exception during a DMAC bus cycle forces the DMAC to terminate the bus cycle in an orderly manner. The preemptory bus exception must arrive prior to or in coincidence with $\overline{DTACK}$ in order to be recognized as an abnormal bus termination. Here coincident means meeting the same set up requirements for the same sampling edge of the clock. The DMAC does not generate any bus cycles if a bus exception condition exists, and thus will not honor any requests until it is removed. However, the DMAC still recognizes requests. The reserved bus exceptions are not used by the DMAC, they should not be asserted during DMAC operations, as the result may be unpredictable.

HALT

The halt exception causes the DMAC to complete the operation in progress and three-state the bus. It does not rearbitrate for the bus until this exception is removed. When halt is negated, the DMAC resumes normal operation.

Refer to Figure 38 for more specific timing diagram.

BUS ERROR

The bus error exception is generated by external circuitry to indicate the current transfer cannot be successfully completed and is to be aborted. The recognition of this exception during a DMAC bus cycle signals the internal bus error condition for the channel for which the current bus cycle is being run.

Refer to Figure 39 for more specific timing diagram.

RETRY

The retry exception causes the DMAC to terminate the present operation and retry that operation when retry is re-

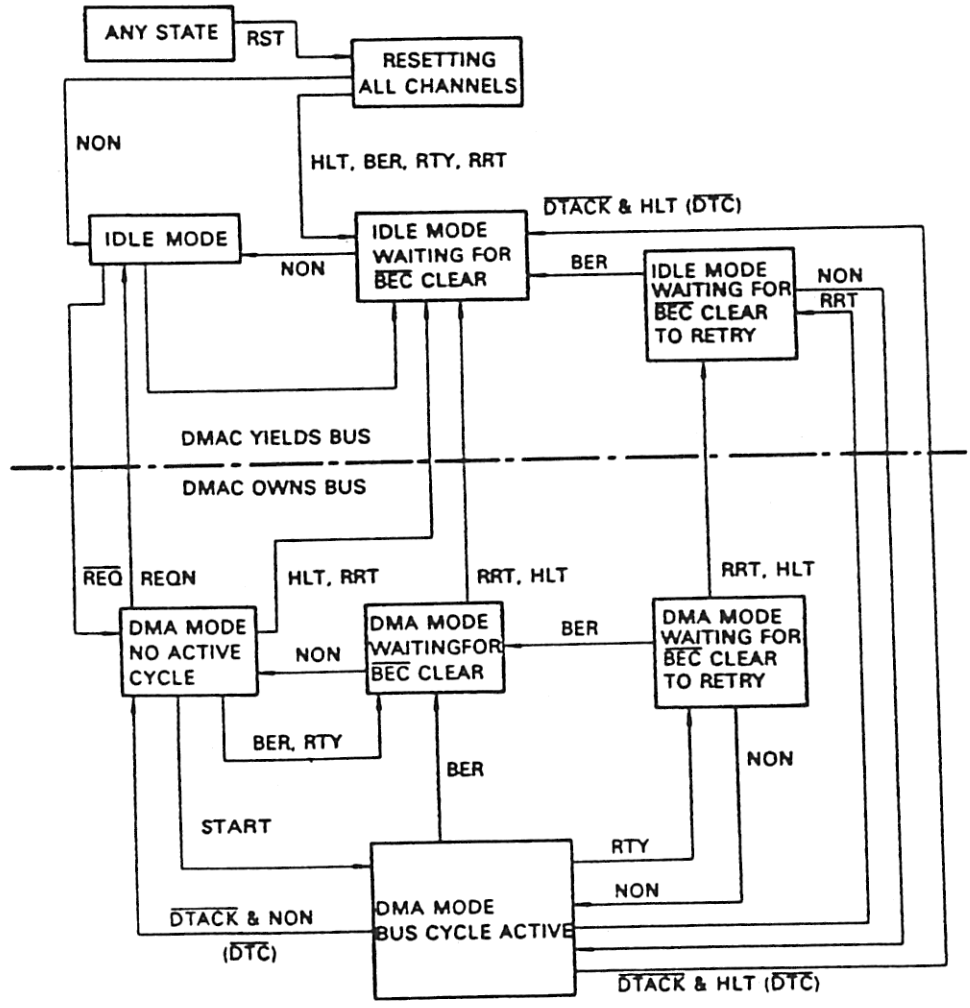


Figure 37 Bus Exception Flow Diagram

moved. The bus is not relinquished for rearbitration and the operation is reinitiated when retry is removed.

Refer to Figure 40 for more specific timing diagram.

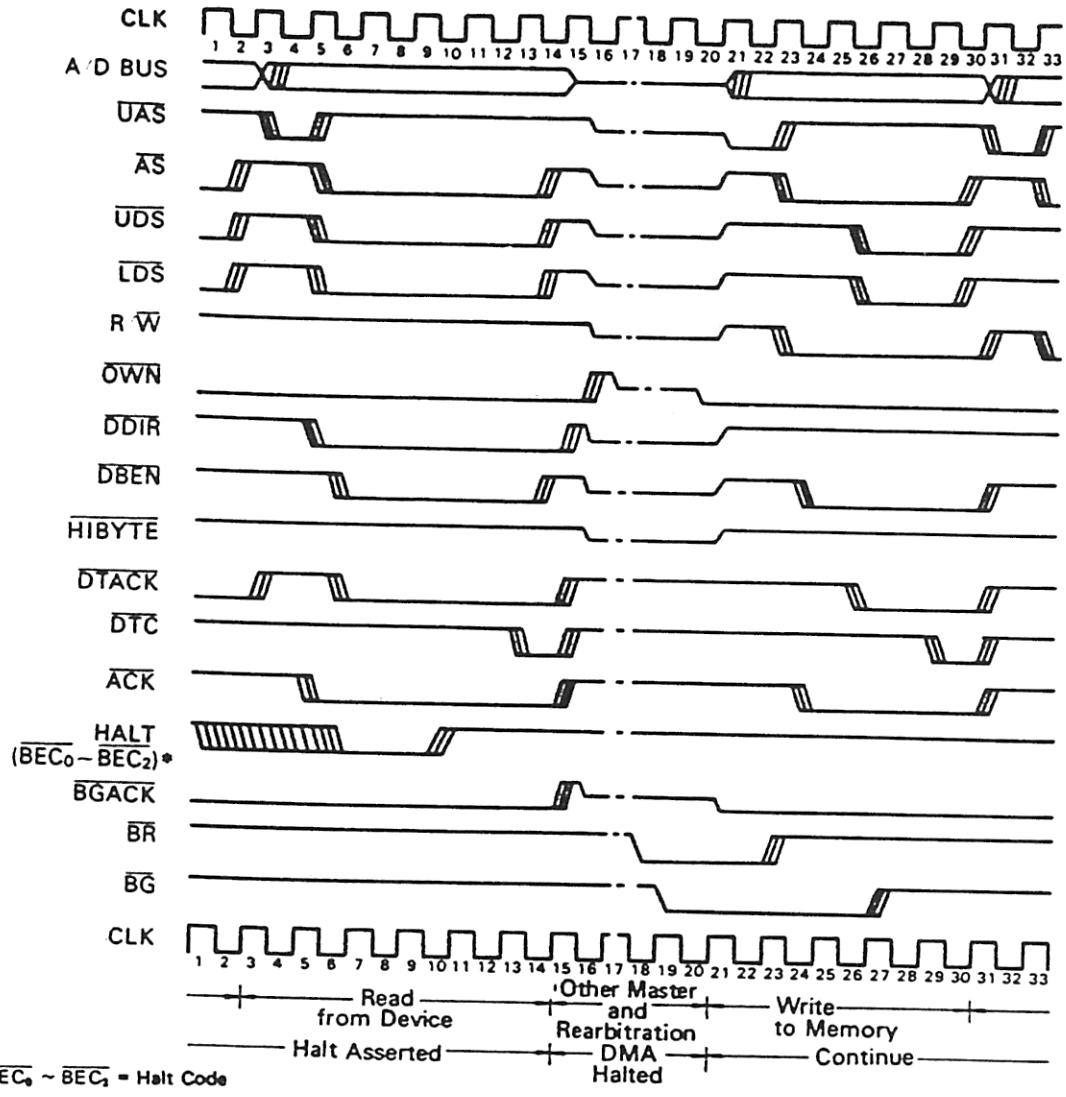
RELINQUISH AND RETRY

The relinquish and retry exception causes the DMAC to three-state all bus master controls and when the exception is removed, rearbitrate for the bus to retry the previous operation.

Refer to Figure 41 for more specific timing diagram.

RESET

The reset exception provides a means of resetting and initializing the DMAC from an external source. If the DMAC is bus master when the reset is received, the DMAC relinquishes the bus. Reset clears GCR, DCR, OCR, SCR, CCR, CSR, CPR, and CER for all channels. This resets STR, CNT, ACT and the interrupt generation bits and clears the status and error registers. The interrupt vector registers are set to $0F, the HD68000 uninitialized interrupt vector number.



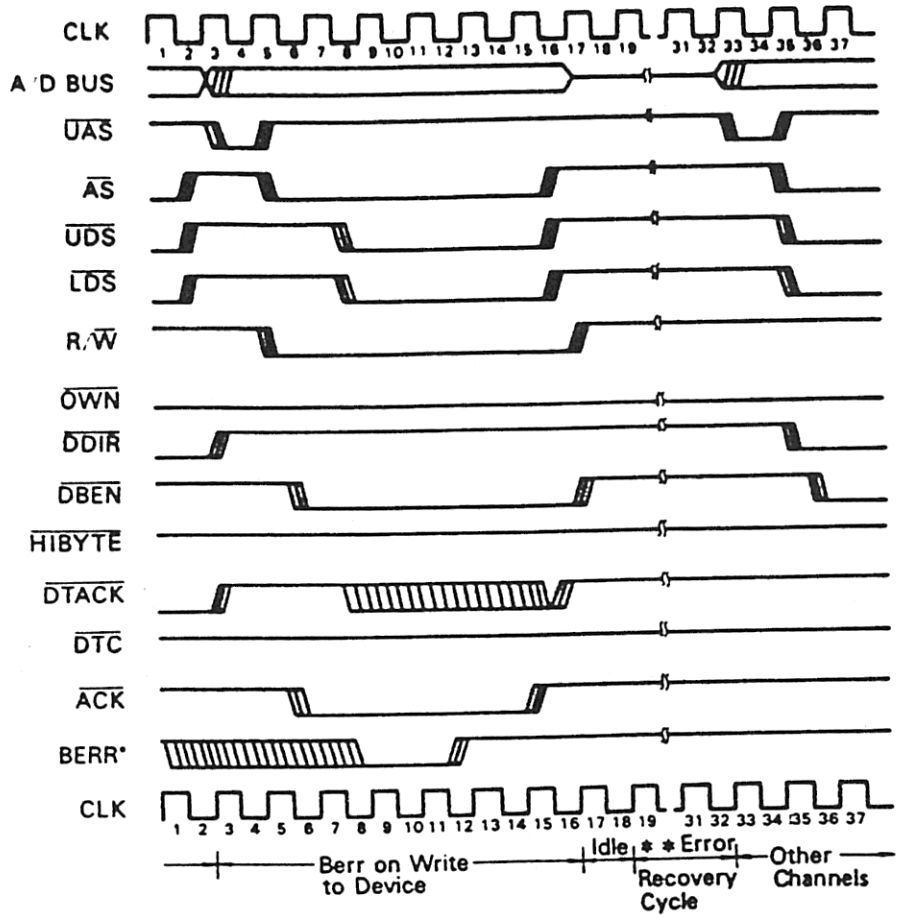
(NOTES) The following notes refer to all bus exceptions.

1) The Bus Exception will be acted upon if it is detected INTERNALLY before or at the same time as $\overline{DTACK}$.
2) The Bus Exception pins must be stable for two clocks before the DMAC will take any action. In addition, if the $\overline{BEC}$ pins are moving but are not stable, and a $\overline{DTACK}$ is also received, the DMAC will wait for the $\overline{BEC}$ pins to resolve (remain stable for two clocks) before terminating the cycle. If the $\overline{BEC}$ pins resolve to an exception code, the DMAC will act accordingly. If they resolve to the normal mode, the DMAC will terminate if a $\overline{DTACK}$ is received and will continue normal operation.
   NOTE: As long as the $\overline{BEC}$ pins are moving the DMAC will not start another cycle.
3) If possible, the DMAC should allow exceptions to be honored if they are asserted after $\overline{DTACK}$ is asserted but before the cycle has finished terminating.
4) If a cycle is not running, the Retry and Berr exceptions will be ignored except that no bus cycles are started as long as anything is detected on the $\overline{BEC}$ pins.

The following refer to the Halt exception only.

5) If halt is received during a cycle, it does not terminate the operation. $\overline{DTACK}$ is still required.
6) If halt is received when no bus cycle is running, the DMAC will simply give up the bus if it owns it.
7) The DMAC will not attempt to re-acquire the bus until HALT has been negated.

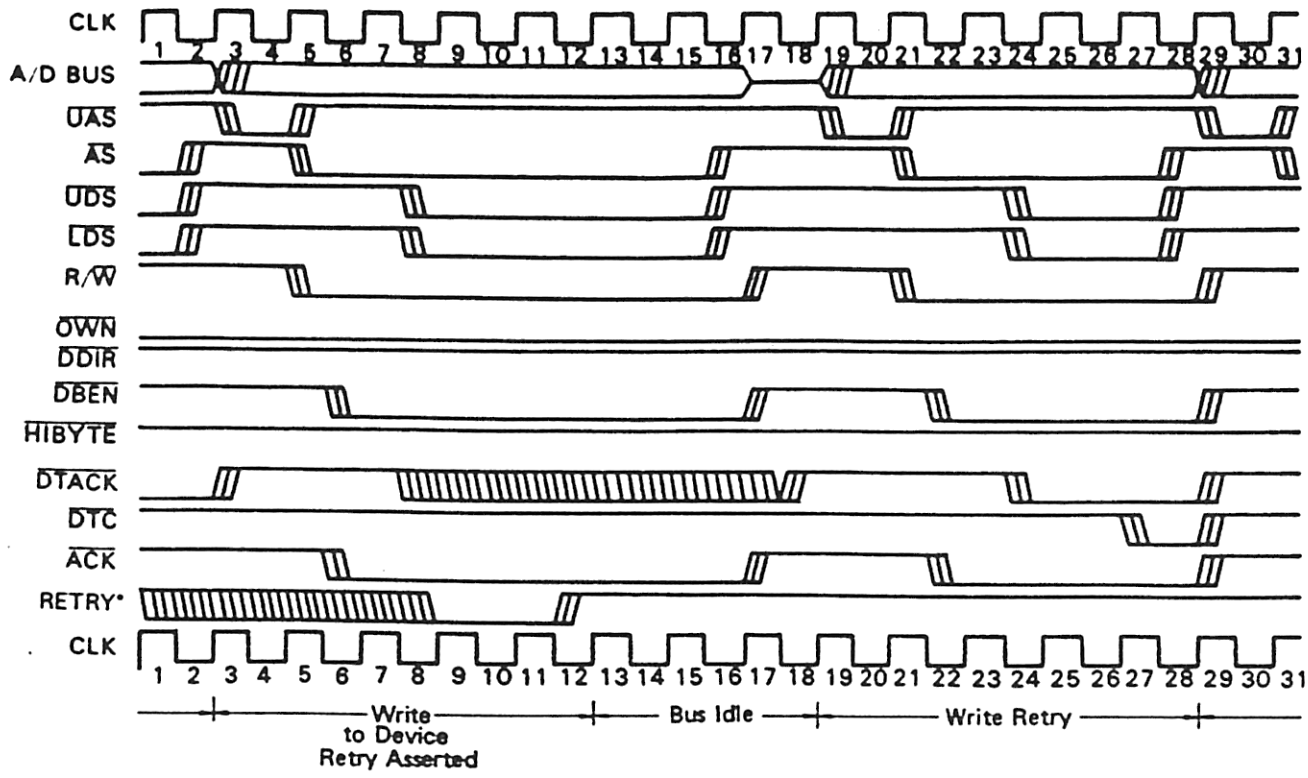
Figure 38 Halt Operation



* $\overline{BEC_4}$ ~ $\overline{BEC_3}$ = Bus Error Code
** Single and Dual Cycle Source Address Error – 24 clocks
Dual Cycle Destination Address Error – 28 clocks

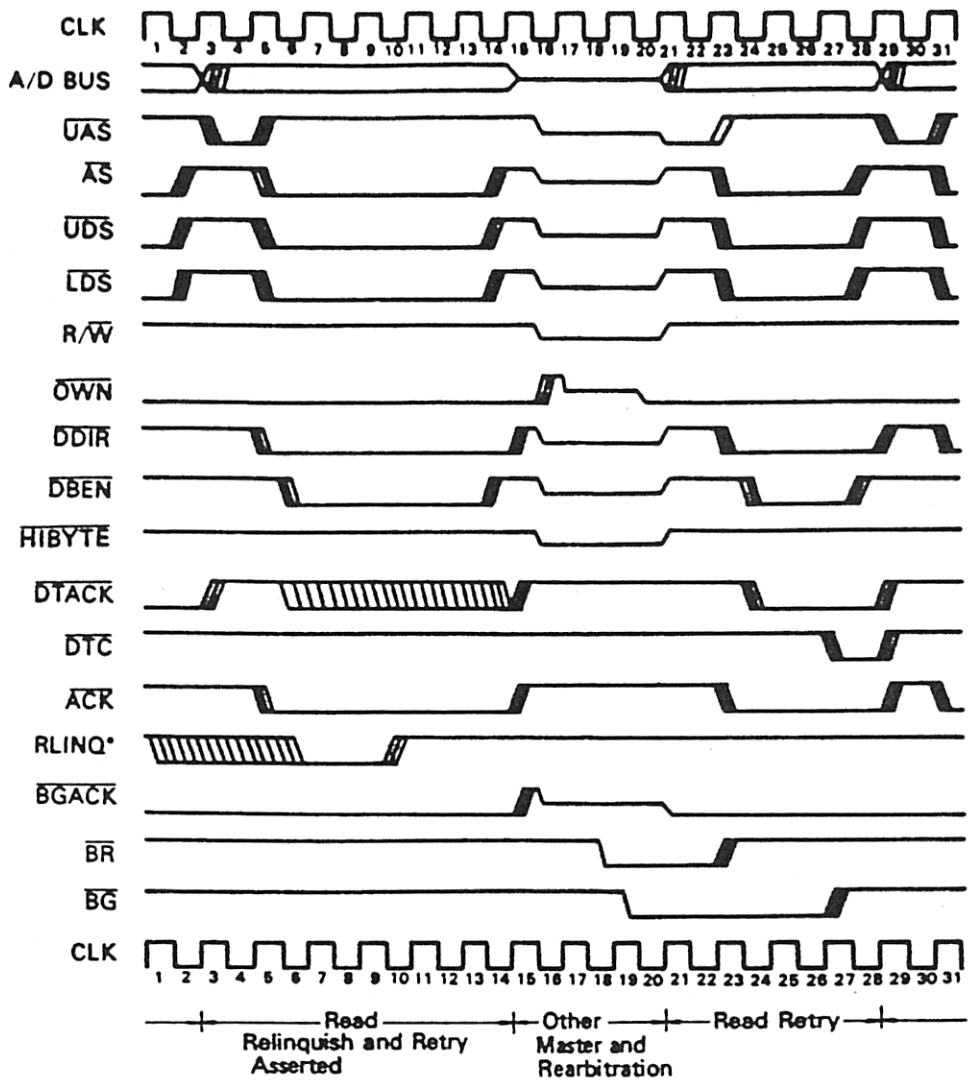
(NOTES) 1) In the case of preemptory bus exception, the bus cycle will always terminate immediately, but normally; i.e. it will sequence off as if a $\overline{DTACK}$ had been received. $\overline{DTC}$ will not be asserted.
2) In the case of a Berr, the DMAC will not terminate the cycle until the $\overline{BEC}$ pins have been stable in the Berr code for at least two clocks, even if a $\overline{DTACK}$ is also received.
3) See the bus exception comments below the Halt diagram.

Figure 39 Berr Operation



(NOTES) 1) In the case of a preemptory bus exception, the bus cycle should always terminate immediately, but normally; i.e. it should sequence off as if a $\overline{DTACK}$ had been received.
2) In the case of a Retry, the DMAC should not terminate the cycle until the $\overline{BEC}$ pins have been stable in the Retry code for at least one clock, even if a $\overline{DTACK}$ is also received.
3) See the bus exception comments below the Halt diagram.

Figure 40 Retry Operation



* $\overline{BEC_0}$ ~ $\overline{BEC_3}$ = Relinquish and Retry Code

(NOTES) 1) In the case of a preemptory bus exception, the bus cycle should always terminate immediately, but normally; i.e. it should sequence off as if a $\overline{DTACK}$ had been received.
2) In the case of a Relinquish and Retry, the DMAC should not terminate the cycle until the $\overline{BEC}$ pins have been stable in the Relinquish and Retry code for at least two clocks, even if a $\overline{DTACK}$ is also received.
3) See the bus exception comments below the Halt diagram.

Figure 41 Relinquish and Retry Operation

BEC CONTROLS

If the BEC controls are asserted to a state that is undefined/reserved, this version of the DMAC will enter a wait state and resume operation when the exception is removed.

(3) ERROR CONDITIONS

When an error is signaled on a channel, all activity on that channel is stopped. The ACT bit of the CSR is cleared, and the COC bit is set. The ERR bit of the CSR is set, and the error code indicated in the CER All pending operations are cleared, so that both the STR and CNT bits of CCR are cleared.

SOURCES OF ERRORS

Enumerated below are the error signals and their sources.

Configuration Error

A configuration error is signaled if chaining is programmed and the continue bit is also set. Configuration error is signaled if DTYP specifies a single-address transfer, and the device port size is not the same as the operand size. Configuration error is signaled if DTYP is 68000 or 6800, DPS is 16 bits, SIZE is 8 bits, and REQG is 10 or 11 (request pin). Setting an undefined configuration will signal a configuration error. The undefined configurations are: XPM = 01, MAC = 11, DAC = 11, CHAIN = 01, SIZE = 11.

Operation Timing Error

An operation timing error is signaled if an attempt is made to continue an operation without STR being simultaneously set or if the channel is not active. Operation timing error is signaled if an attempt to set STR is made with ACT, COC, BTC, NPT, or ERR asserted. Operation timing error is signaled if an attempt to write to the DCR, OCR, SCR, GCR, MAR, DAR or MTC is made with STR or ACT asserted. Operation timing error is signaled if an attempt to assert CNT is made when CHAIN is 10 or 11 (chaining modes). Operation timing error is signaled if an attempt to assert CNT is made when BTC and ACT are asserted.

Address Error

Address error is signaled if an odd address operation is attempted with word or long word operands or if $\overline{CS}$ or $\overline{IACK}$ is asserted while the DMAC is bus master. The address error is asserted after the odd address is encountered, this is consistent with the processor operation.

Bus Error

A Bus error occurred during the last bus cycle generated by the channel.

Count Error

A count error is signaled if the memory or base transfer count registers are initialized with terminal count. A count error is signaled if a terminal count is encountered during continue or chain processing.

Abort

An abort error is signaled if the $\overline{PCL}$ line was configured as an abort input and made an active transition, or if the channel operation was aborted by the SAB bit of the CCR.

Note: When the $\overline{PCL}$ line is used as an abort input, the PCT bit should be cleared prior to starting the channel. If the PCT bit is set prior to the channel being started, the DMAC will recognize this as an external abort when the channel is started.

When the transfer mode is set to dual addressing mode, the transfer direction is set to I/O device to memory, and $\overline{PCL}$ signal is set to external about input mode, the external abort for that channel will be ignored after a $\overline{DONE}$ input from the I/O device is received during the channel's I/O device-to-memory data transfer cycle. After the $\overline{DONE}$ input, the channel will accept external abort when the channel is properly reinitialized and is restarted.

-X-

ERROR RECOVERY PROCEDURES

If an error occurs during a DMA transfer, appropriate information is available to the operating system to allow a "soft failure" operation. The operating system must be able to determine how much data was transferred, where the data was transferred to, and what type of error occurred.

The information available to the operating system consists of the present values of the Memory Address, Device Address and Base Address Registers, the Memory Transfer and Base Transfer Counters, and control, status, and error registers. After the successful completion of any transfer, the memory and device address registers points to the location of the next operand to be transferred and the memory transfer counter contains the number of operands yet to be transferred. If an error occurs during a transfer, that transfer has not completed and the registers contain the values they had before the transfer was attempted. If the channel operation uses chaining, the Base Address Register points to the next chain entry to be serviced, unless the termination occurred while attempting to fetch an entry in the chain. In that case, the Base Address Register points to the entry being fetched.

MULTIPLE ERRORS X

The DMAC will log the first error encountered in the channel error resister. If an error is pending in the error register and another error is encountered the second error will not be logged. This is true in the case of an error pending and an attempt is made to set the STR bit. In this case the Operation timing error is not logged in the error register but the error is recognized internally and the channel is not started.

• CHANNEL PRIORITIES

Each channel has a priority level, determined by the contents of the Channel Priority Register (CPR). The priority of a channel is a number from 0 to 3, with 0 being the highest priority level. When multiple requests are pending at the DMAC, the channel with the highest priority receives first service. The priority of a channel is independent of the device protocol or the request mechanism for that channel. If there are several requesting channels at the highest priority level, a round-robin resolution is used, that is, as long as these channels continue to have requests, the DMAC does operand transfers in rotation.

■ APPLICATIONS INFORMATION

This section contains examples of how to interface various I/O devices to a HMCS68000/DMAC based system.

Figure 42 shows an example of how to demultiplex the address/data bus.

Figure 43 indicates the example of how to latch the data, when the DMAC has two channels which operate in 6800 mode.

Figure 44 indicates the example of inter-device connection in the HMCS68000 system.

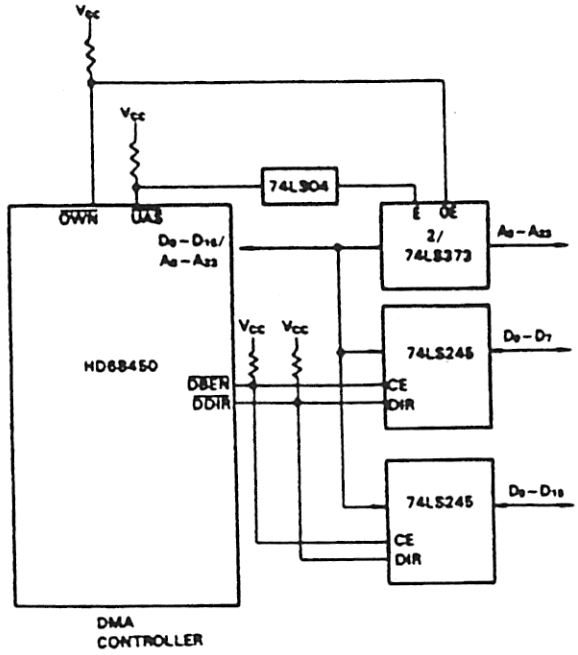


Figure 42 Required Multiplexed Data/Address Hardware for the Bus Control Logic

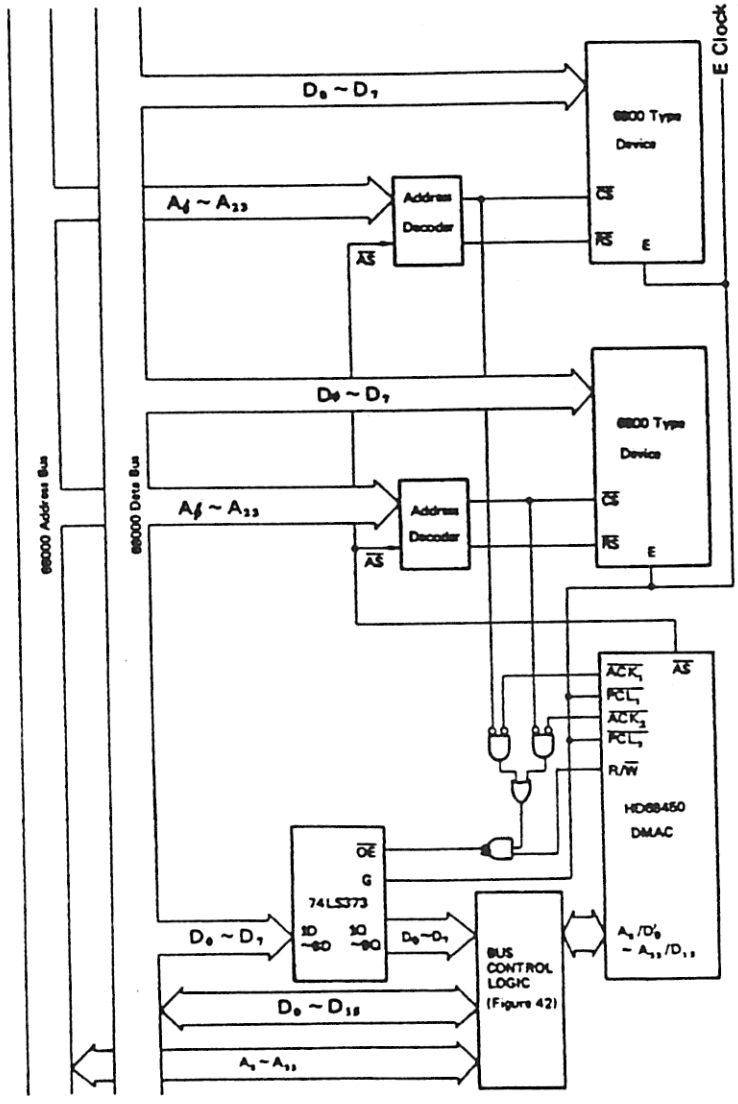


Figure 43 An Example of Connection with Peripheral Devices in 6800 Mode

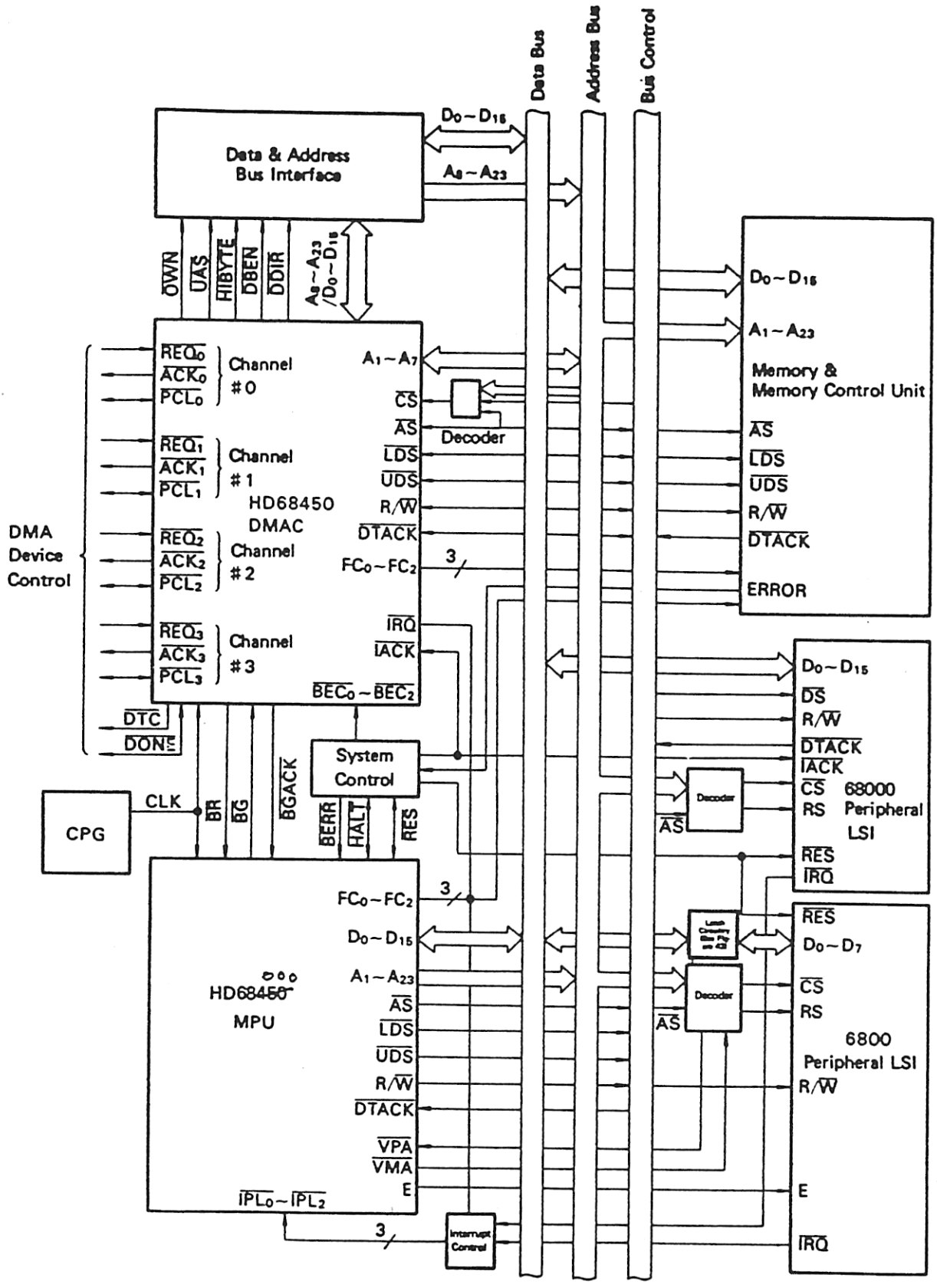


Figure 44 An Example of Inter-device Connection in the HMCS68000 System

APPENDIX L

DATA SHEET TMS9914A GPIB CONTROLLER

IMPORTANT NOTICES

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

TI cannot assume any responsibility for any circuits shown or represent that they are free from patent infringement.

TABLE OF CONTENTS

LIST OF APPENDICES

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

1.1 DESCRIPTION

The TMS9914A provides an interface between a Microprocessor System and the General Purpose Interface Bus (GPIB) specified in the IEEE-488 1975/78 standards and the IEEE-488A 1980 supplement. The device is controlled and configured through 8-bit memory mapped registers and enables all aspects of the standards to be implemented, including talker, listener and controller.

1.2 KEY FEATURES

- Handles all IEEE-488 1975/78 functions
- Compatible with IEEE-488A 1980 supplement
- Talker and listener function (T,TE,L,LE)
- Automatic source and acceptor handshakes (SH,AH)
- Controller with pass control
- System Controller capabilities
- Device trigger and device clear capabilities (DT,DC)
- Optional automatically cleared 'request service bit'
- Parallel and serial poll facilities (PP)
- Remote/local function with local lockout (RL)
- Single or dual primary addressing
- Secondary address capabilities
- Direct interface to SN75160/161/162 bus transceivers with no additional logic
- Compatible with most microprocessors
- Direct memory access facilities
- Memory-mapped microprocessor interface

1.3 RELATIONSHIP TO THE TMS9914

The TMS9914A is compatible with the TMS9914 and may replace it in any application without software alterations. New features are included on the TMS9914A which increase the flexibility of the device. These features are disabled at power-up and must be programmed by the user as needed.

TMS9914A VS. TMS9914:

1) Byte Output interrupt modification (see Section 2.1.1 and Appendix B)

2) Reduced bus settling time T1 (see Section 2.1.6 and Section 3.3)

3) Modification to the Service Request Function (see Section 3.5)

4) Addition of a second request service (rsv) bit which is automatically cleared (see Section 2.1.6 and Section 3.5)

1.4 INTRODUCTION TO THE IEEE-488 1975/78 INTERFACE BUS

The GPIB is designed to allow up to 15 instruments within a localized area to communicate with each other over a common bus. Each device has a unique address, read from external switches at power-on, to which it responds. Information is transmitted in byte serial bit parallel format and may consist of either device-dependent data or interface messages, commonly referred to as data or commands, respectively.

Device data may be sent by any one device (the talker) and received by a number of other devices (listeners). Instructions, such as select range, select function, or measurement data for processing or printout, may be sent in this way.

One of the devices on the bus, designated the Controller in charge (Controller), may send interface control messages. Devices can be assigned to the bus as listeners or talkers by sending their unique talk or listen addresses and may be switched between remote and local control.

The bus itself consists of a 24-wire shielded cable. Eight lines carry data; 8 are control lines; 8 are signal and system grounds. A diagram showing the IEEE bus configuration is given in Appendix A.

Three of the management lines operate as a three-line handshake between talker (or controller) and listeners. No new data is sent until each device addressed to listen has received the last byte and is ready for the next. This method of asynchronous communication ensures that the data rate is suited to the slowest active listener, as well as ensuring compatibility over a wide range of devices.

The remainder of this manual assumes working familiarity with the IEEE-488 1975/78 standards. Terminology and abbreviations defined within these standards are freely used throughout. The IEEE convention of lower case for local messages and upper case for remote messages (received via the interface) is used in all acronyms.

1.5 TYPICAL APPLICATIONS

The TMS9914A is used when an intelligent instrument is required to communicate with an IEEE-488 General Purpose Interface Bus (GPIB). It performs the interface function between the microprocessor and bus and relieves the processor of the task of maintaining the IEEE protocol. By utilizing the interrupt capabilities of the device, the bus does not have to be continually polled, and fast responses to changes in the interface configuration can be achieved.

A block diagram showing the TMS9914A in a typical application is given in Figure 1-1.

The GPIB input/output pins are connected to the IEEE-488 bus via bus transceivers. The direction of data flow is controlled by the TE and $\overline{\text{CONT}}$ outputs generated on the TMS9914A. The SN75160, 75161 and 75162 (see Appendix C) are designed specifically for use with a GPIB interface. The TE and $\overline{\text{CONT}}$ signals are routed within the devices so that the buffers on particular lines are controlled as required by the TMS9914A. Other buffers may be used, but they may require a small amount of external logic, particularly around the EOI line buffer.

Communication between the microprocessor and TMS9914A is carried out via memory-mapped registers. There are 13 registers within the TMS9914A, 6 of which are read and 7 write. These registers both pass control data to and get status information from the device.

The three least significant address lines from the MPU are connected to the register select lines RS0, RS1, and RS2 and determine the particular register selected. The high order address lines are decoded by external logic to cause the $\overline{\text{CE}}$ input to the TMS9914A to be pulled low when any one of eight consecutive addresses are selected. Thus the internal registers appear to be situated at eight consecutive locations within the MPU address space. Reading or writing to these locations transfers information between the TMS9914A and the microprocessor. Note that reading and writing to the same location will not access the same register within the TMS9914A since they are either read-only or write-only registers. For example, a read operation with RS2·RS0 = 011 gives the current status of the GPIB interface control lines, whereas a write to this location loads the auxiliary command register.

Each device on the bus interface is given a 5-bit address enabling it to be addressed as a talker or listener. This address is set on an external DIP switch (usually at the rear of an instrument) before power-on.

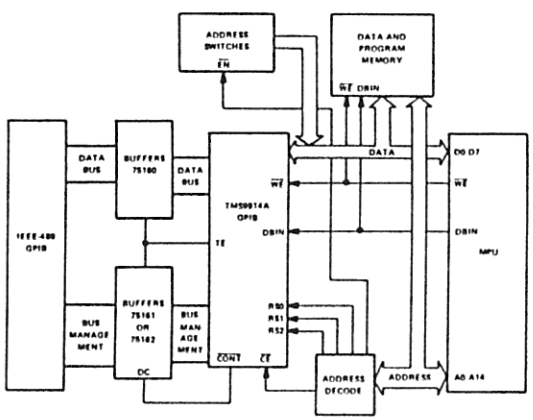


FIGURE 1-1 – TYPICAL TMS9914A APPLICATION

Typical TMS9914A configuration utilizes registers 100 or 101 as an address switch register (see Table 2-1). This register may consist of a DIP switch which drives the data lines via tristate buffers when one of these addresses is read. This allows the host MPU to read a device address which is manually set and write this address into the address register of the TMS9914A for device identification on the bus. The TMS9914A responds by causing a My Address (MA) interrupt and entering the required addressed state when this address is detected on the GPIB data lines.

2. ARCHITECTURE

The block diagram of the internal architecture of the TMS9914A is given in Figure 2-1. As previously stated, there are 13 MPU accessible registers of which 6 are read and 7 are write. These registers handle all communication between the IEEE-488 1975/78 bus and microprocessor.

Each register is accessed by putting the relevant address on lines RS0, RS1 AND RS2 and performing a memory read ($\overline{WE}$ = 1 DBIN = 1) or memory write ($\overline{WE}$ = 0 DBIN = 0) operation. The register addresses and use of each bit is shown in Table 2-1 for the read registers and Table 2-2 for the write registers. A full description of each register is given in the following paragraphs.

Implementation of the functions described by the state diagrams of the IEEE-488 standard is carried out in the IEEE-488 state diagram block. Information is received from the IEEE bus and from the internal registers and is combined with the current status of the device (for example, Talker Active State, TACS) to produce the control signals to load registers or handle the handshake or bus management lines.

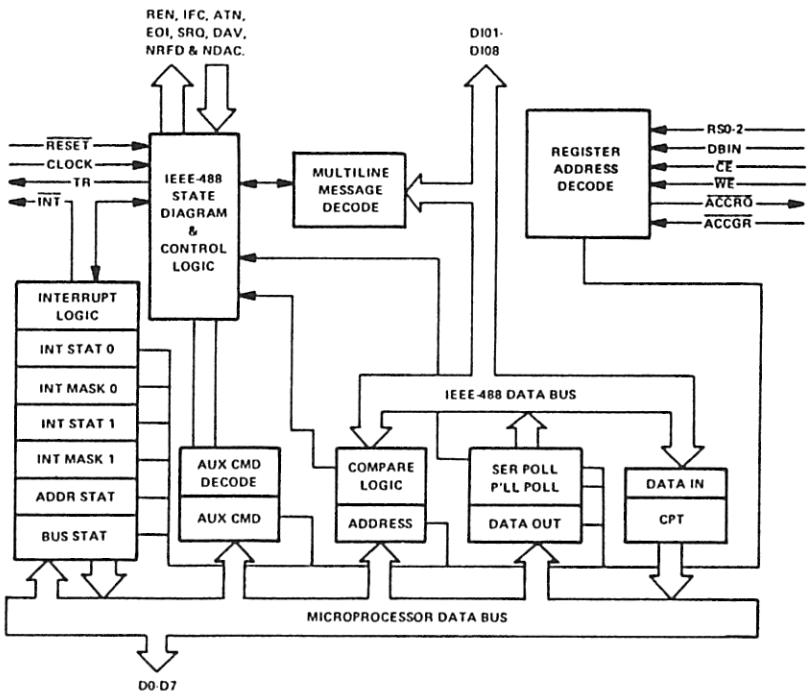


FIGURE 2-1 – SIMPLIFIED BLOCK DIAGRAM

TABLE 2-1 – TMS9914A READ REGISTERS

ADDRESS RS2	RS1	RS0	REGISTER NAME	BIT ASSIGNMENT D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	Int Status 0	INT0	INT1	BI	BO	END	SPAS	RLC	MAC
0	0	1	Int Status 1	GET	ERR	UNC	APT	DCAS	MA	SRQ	IFC
0	1	0	Address Status	REM	LLO	ATN	LPAS	TPAS	LADS	TADS	ulpa
0	1	1	Bus Status	ATN	DAV	NDAC	NRFD	EOI	SRQ	IFC	REN
1	0	0	*								
1	0	1	*								
1	1	0	Cmd Pass Thru	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1
1	1	1	Data In	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1

*The TMS9914A host interface data lines will remain in the high impedance state when these register locations are addressed. An Address Switch Register may therefore be included in the address space of the device at these locations (see Section 1.5).

TABLE 2-2 – TMS9914A WRITE REGISTERS

ADDRESS RS2	RS1	RS0	REGISTER NAME	BIT ASSIGNMENT D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	Int Mask 0			BI	BO	END	SPAS	RLC	MAC
0	0	1	Int Mask 1	GET	ERR	UNC	APT	DCAS	MA	SRQ	IFC
0	1	0	*	xx	xx	xx	xx	xx	xx	xx	xx
0	1	1	Auxiliary Cmd	cs	xx	xx	f4	f3	f2	f1	f0
1	0	0	Address	edpa	dal	dat	A5	A4	A3	A2	A1
1	0	1	Serial Poll	S8	rsvl	S6	S5	S4	S3	S2	S1
1	1	0	Parallel Poll	PP8	PP7	PP6	PP5	PP4	PP3	PP2	PP1
1	1	1	Data Out	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1

*This address is not decoded by the TMS 9914A. A write to this location will have no affect on the device, as if a write had not occurred.

2.1 REGISTERS

2.1.1 Interrupt Mask and Status Registers 0

The Interrupt Mask and Interrupt Status registers operate independently of each other. The status bits will always be set when the appropriate events occur regardless of the state of the corresponding mask bit.

All interrupt bits, with the exception of INTO and INT1 which are not storage bits, are edge triggered and are set when the appropriate condition becomes true. The storage bits are cleared immediately after the corresponding Interrupt Status Register is read by the host MPU. If an interrupt condition becomes true during this read operation, then the event is stored. The corresponding bit is set when the read operation ends, hence no interrupts are lost. In addition to being cleared by a read operation, the BO interrupt is also cleared by writing to the Data Out Register, and the BI interrupt is cleared by reading the Data In Register.

The interrupt status bits are cleared and held in the 0 condition while Software Reset (swrst) is set.

The corresponding bit of the Interrupt Mask register must be set to a 1 if an interrupt status bit is to cause an external interrupt ($\overline{INT}$ Low) when it is set (i.e., $\overline{INT} = \overline{INT\ STATUS.INT\ MASK}$). The mask register is not cleared by 'swrst' or the Hardware Reset pin ($\overline{RESET}$) and will power on in a random state. It must, therefore, be written to by the host MPU before 'swrst' is cleared to avoid extraneous interrupts (see Section 2.1.6 for operation of 'swrst').

The INTO and INT1 bits of the Interrupt Status Register are not true status bits. INT1 will be true if there are any unmasked interrupt status bits set to a 1 in Interrupt Status Register 1. INTO will be true if any of bits 2-7 of Interrupt Status Register 0 are unmasked and set to a 1. If either INT1 or INTO is true, then the external interrupt pin ($\overline{INT}$) will be pulled low provided that the Disable All Interrupts feature (dai) has not been set.

The individual bits of Interrupt Status and Interrupt Mask Register 0 are described are in the following paragraphs. The conditions which set these bits, shown in parentheses, are given in terms of the state diagrams described in Section 3. Each bit is set on the rising edge of the condition shown.

INTERRUPT MASK/STATUS REGISTER 0

xx	xx	BI	BO	END	SPAS	RLC	MAC	INT	MASK 0
INTO	INT1	BI	BO	END	SPAS	RLC	MAC	INT	STATUS 0
D0	D1	D2	D3	D4	D5	D6	D7		MPU BUS

NOTE: A 0 masks and a 1 unmasks the bits in the interrupt mask registers.

INT1 This will be a 1 when an unmasked status bit in Interrupt Status Register 1 is set to a 1.

INTO This will be a 1 when any of bits 2-7 of Interrupt Status Register 0 is unmasked and set to a 1.

BI Byte In. A data byte has been received in the Data In register. If the mask bit is not set, then no interrupt is generated but a RFD holdoff will still occur before the next data byte is accepted. If the Shadow Handshake feature is used, then this status bit will not be set. This bit is cleared by reading the Data In Register as well as after Interrupt Status Register 0 has been read. (Set On: ACDS1.LACS)

BO Byte Out. This is set when the Data Out Register is available to send a byte over the GPIB. This byte may be either a command if the device is a controller or data if the device is a talker. It is set when the device becomes an active talker or controller but will not occur if the Data Out register has been loaded with a byte which has not been sent. Subsequently, it will occur after each byte has been sent and the TMS9914A returns to SGNS. This bit is cleared by writing to the Data Out Register as well as by reading Interrupt Status Register 0.
(Set On: SGNS.CACS + SGNS.TACS.$\overline{SHFS}$)

NOTE

When a controller addresses itself as a talker and then goes to standby, there will be a momentary transition of the source handshake into SIDS before TACS becomes true and it reenters SGNS. Under these circumstances, the TMS9914A is guaranteed to give a BO interrupt on reentering 'SGNS'. The TMS9914, however, may not, and a controller going to standby as a talker should write the first byte of data into the Data Out Register immediately after writing the gts auxiliary command, without waiting for a BO interrupt (see Appendix B for details).

END This indicates that a byte just received by a listener was the last byte in a string, that is, it was received with the EOI line true. It is set at the same time as the BI interrupt. (Set On: (ACDS1.LACS.EOI)

SPAS This indicates that the TMS9914A has requested service via rsv1 or rsv2 (in the Serial Poll Register or Auxiliary Command Register) and has been polled in a serial poll. It is set on the false transition of STRS when the serial poll status byte is sent. (Set On: STRS.SPAS.(APRS1 + APRS2)

RLC Remote/Local Change. This is set by any transition between local and remote states in the Remote/Local function.
(Set On: (LOCS-REMS) + (REMS-LOCS) + (LWLS-RWLS) + (RWLS-LWLS)

MAC My Address Change. This indicates that a command has been received from the GPIB which has resulted in the addressed state of the TMS9914A to change. It will not occur if secondary addressing is being used, nor indicate that the TMS9914A has been readdressed on its other primary address.
(Set On: ACDS1.(MTA.TADSUNT + OTA.$\overline{TADS}$ + MLA.LADS + UN.$\overline{LADS}$)

2.1.2 Interrupt Mask and Status Registers 1

The operation of Interrupt Mask and Status Register 1 is similar to that of Interrupt Mask and Status Register 0 except that all bits are true storage bits. The status bits are cleared only following the register being read and by 'swrst'.

There is one distinct group of interrupts in this register: GET, UNC, APT, DCAS, MA. These are all set in response to commands received over the bus and if unmasked, a Data Accepted (DAC) holdoff will occur when the interrupt in question is set. It may be released with a 'dacr' auxiliary command. This is further discussed in Section 3.2.

The mask bit of the APT interrupt is further used in the talker and listener functions. When the interrupt is unmasked, the talker and listener functions of the TMS9914A implement the extended talker and extended listener functions of IEEE-488. Otherwise these functions implement the talker and listener functions of IEEE-488.

The individual bits of Interrupt Status and Interrupt Mask Register 1 are described below. The conditions which set these bits, shown in parentheses, are given in terms of the state diagrams described in Section 3.

INTERRUPT MASK/STATUS REGISTER 1

GET	ERR	UNC	APT	DCAS	MA	SRQ	IFC	INT	MASK 1
GET	ERR	UNC	APT	DCAS	MA	SRQ	IFC	INT	STATUS 1
D0	D1	D3	D3	D4	D5	D6	D7	MPU	BUS

GET — This is set if a Group Execute Trigger command is received. A DAC holdoff occurs if the interrupt is unmasked. The TR pin becomes high when this command is received and persists high for the duration of a DAC holdoff if one occurs. If the interrupt is masked, the TR pin becomes high for approximately five clock cycles.
(Set On: GET.LADS.ACDS1)

ERR — Error. This is set if the source handshake becomes active and finds that the NDAC and NRFD lines are both high. This indicates that, for whatever reason, there are no acceptors on the bus.
(Set On: SERS)

UNC — Unrecognized Command. This is set if a command has been received which has no meaning to the TMS9914A. Unrecognized addressed commands will only cause this interrupt if the device is LADS except for TCT which will only interrupt in TADS. Secondary commands will only cause this interrupt if the 'pts' auxiliary command has been set previously. A DAC holdoff will occur if this interrupt is unmasked which effectively enables the command pass through feature. Unrecognized commands may be inspected in the Command Pass Through Register before this holdoff is released.
(Set On: ACDS1.(UCG.$\overline{\text{LLO}}$.$\overline{\text{SPE}}$.$\overline{\text{SPD}}$.$\overline{\text{DCL}}$ + ACG.$\overline{\text{GET}}$.$\overline{\text{GTL}}$.$\overline{\text{SDC}}$.$\overline{\text{TCT}}$.LADS + TCT.TADS + SCG.pts))

APT — Address Pass Through. Unmasking this interrupt enables secondary addressing. It is set if a secondary command is received provided that the last primary command received was a primary talk or listen address of the TMS9914A. A DAC holdoff will occur and the secondary address may be read from the Command Pass Through Register. The holdoff may be released by a 'dacr' auxiliary command and the 'cs' bit of the Auxiliary Command Register is used to indicate that a valid (cs = 1) or an invalid (cs = 0) secondary has been identified by the host MPU.
(Set On: ACDS1.SCG.(LPAS + TPAS))

DCAS — Device Clear Active State. This is set when a device clear command (DCL) is received or when a selected device clear (SDC) is received with the TMS9914A in LADS. This will cause a DAC holdoff if unmasked.
(Set On: ACDS1.(DCL + SDC.LADS))

SRQ — Service Request. This is provided for the benefit of the controller which should execute a serial poll in response to this interrupt. It is set when the SRQ line becomes true.
(Set On: SRQ.($\overline{\text{CIDS}}$ + $\overline{\text{CADS}}$))

MA — My Address. This is set when the TMS9914A recognizes its primary talk or listen address. A DAC holdoff will occur if this is unmasked.
(Set On: (MLA + MTA).$\overline{\text{SPMS}}$.aptmk))

IFC — Interface Clear. This is provided for the benefit of devices which are not the System Controller. It is set when the IFC line becomes true and indicates that the TMS9914A has been returned to an idle state. If the device is the System Controller, then the IFC interrupt is not set.
(Set On: IFCIN)

2.1.3 Address Status Register

REM	LLO	ATN	LPAS	TPAS	LADS	TADS	ulpa	
D0	D1	D2	D3	D4	D5	D6	D7	MPU BUS

REM	The device is in the remote state
LLO	Local lockout is in operation
ATN	The attention line is low (true) on the bus
LPAS	TMS9914A is in the listener primary addressed state
TPAS	TMS9914A is the talker primary addressed state
LADS(or LACS)	The device is addressed to listen
TADS(or TACS)	The device is addressed to talk
ulpa	This bit shows the LSB of the last address recognized by the TMS9914A.

2.1.4 Address Register

edpa	dal	dat	A5	A4	A3	A2	A1
D0	D1	D2	D3	D4	D5	D6	D7

edpa	Enable dual primary addressing mode
dal	Disable listener function
dat	Disable talker function
A5-A1	Primary address of the TMS9914A

Bits A5-A1 of this register contain the primary address of the device (denoted AAAAA in Table 3-15). IEEE-488 1975/78 does not allow a device to be assigned the value 11111 for bits A5-A1. When 'swrst' is true at power-up or if set by the host MPU, the TMS9914A is held in an idle state. During this time the host MPU may load the primary address of the device into these bits. Often this will be read from an Address Switch Register (see Section 1.5).

The 'edpa' bit is used to enable the dual addressing mode of the TMS9914A. It causes the LSB of the address to be ignored by the address comparator giving two consecutive primary addresses for the device. The address by which the TMS9914A was selected is indicated by the 'ulpa' bit of the Address Status Register.

The Address Register is not cleared by 'swrst' or hardware reset.

2.1.5 Auxiliary Command Register (see Section 3.1 for Auxiliary Command State Diagram)

cs	xx	xx	F4	F3	F2	F1	F0
D0	D1	D2	D3	D4	D5	D6	D7

f4-f0	Auxiliary command select (see Table 2-3)
cs	Clear or set the feature (where applicable)

Auxiliary commands are used to enable and disable most of the selectable features of the TMS9914A and to initiate many of the actions of the device. The desired feature is selected by writing a byte to this register with the appropriate value in bits f4-f0. These values are given in Table 2-3.

The 'cs' bit is used in most cases when the feature selected by f4-f0 is of the clear/set type. The feature is enabled if 'cs' = '1' and disabled if 'cs' = '0'. The holdoff on all data (hdfa) feature is an example of such a feature. Other auxiliary commands initiate an action of the TMS9914A, such as release RFD holdoff (rhdf). In most cases, the 'cs' bit is unused and ignored by these commands.

All the clear/set auxiliary commands are cleared by the hardware RESET pin except 'swrst,' which is set true by RESET.

The force group execute trigger (fget) and return to local (rtl) auxiliary commands have a clear/set mode of operation and a pulsed mode of operation. They behave as normal clear/set features, but if they are written with 'cs' = '0' when they have not been previously set, then they will pulse true. Using the 'fget' command in this manner will produce a pulse of approximately 1 μs at the TR pin (with a 5 MHz clock). The 'rtl' command used in this way will cause a return to one of the local states (assuming local lockout is not in force) but the TMS9914A may reenter the remote state next time the listen address occurs (see Section 3.6).

TABLE 2-3 – AUXILIARY COMMANDS

c/s	f4	f3	f2	f1	f0	MNEMONIC	FEATURES
0/1	0	0	0	0	0	swrst	Software reset
0/1	0	0	0	0	1	dacr	Release DAC holdoff
na	0	0	0	1	0	rhdf	Release RFD holdoff
0/1	0	0	0	1	1	hdfa	Holdoff on all data
0/1	0	0	1	0	0	hdfe	Holdoff on EOI only
na	0	0	1	0	1	nbaf	New byte available false
0/1	0	0	1	1	0	fget	Force group execute trigger
0/1	0	0	1	1	1	rtl	Return to local
na	0	1	0	0	0	feoi	Send EOI with next byte
0/1	0	1	0	0	1	lon	Listen only
0/1	0	1	0	1	0	ton	Talk only
na	0	1	0	1	1	gts	Go to standby
na	0	1	1	0	0	tca	Take control asynchronously
na	0	1	1	0	1	tcs	Take control synchronously
0/1	0	1	1	1	0	rpp	Request parallel poll
0/1	0	1	1	1	1	sic	Send interface clear
0/1	1	0	0	0	0	sre	Send remote enable
na	1	0	0	0	1	rqc	Request control
na	1	0	0	1	0	rlc	Release control
0/1	1	0	0	1	1	dai	Disable all interrupts
na	1	0	1	0	0	pts	Pass through next secondary
0/1	1	0	1	0	1	stdl	Short T1 settling time
0/1	1	0	1	1	0	shdw	Shadow handshake
0/1	1	0	1	1	1	vstdl	Very short T1 delay
0/1	1	1	0	0	0	rsv2	Request Service Bit 2

2.1.6 Description of Auxiliary Commands

Software Reset (swrst) 0/1xx00000

Setting this command causes the TMS9914A to be returned to a known idle state during which it will not take part in any activity on the GPIB. This auxiliary command is set by the power-on RESET and the chip should be configured while 'swrst' is set. Configuration should include writing the address of the device into the Address Register, writing mask values into the Interrupt Mask Registers and selecting the desired features in the Auxiliary Command Register and Address Register. After this, 'swrst' may be cleared at which point the device becomes logically existent on the GPIB. The Serial Poll Register and Parallel Poll Registers may also be written in this period but this is not necessary if there is no status to report as both of these are cleared by the power-on RESET pin. Table 2-4 lists the various states and other conditions forced by 'swrst'.

TABLE 2-4 – SOFTWARE RESET CONDITIONS

MNEMONIC	DESCRIPTION
SIDS	Source idle state
AIDS	Acceptor idle state
TIDS	Talker idle state
TPAS	Talker primary idle state
LIDS	Listener idle state
LPAS	Listener primary state
NPRS	Negative poll response state
LOCS	Local state
CIDS	Controller idle state
SPIS	Serial poll idle state
PPSS	Parallel poll standby state
ADHS	DAC holdoff state
AEHS	RFD holdoff on end state
SHFS	Source holdoff state
ENIS	END idle state

NOTES: 1. See Section 3 for definition of above.
2. All interrupt status bits are held in a 0 state, but interrupt mask bits are not affected.

Release DAC Holdoff (dacr)0/1xx00001

The Data Accepted (DAC) holdoff allows time for the host microprocessor to respond to unrecognized commands, secondary addresses, and device trigger or device clear commands. The holdoff is released by the MPU when the required action has been taken. Normally the command is loaded with the clear/set bit at zero; however, when used with the address pass through feature CS is set to one if the secondary address was valid or to zero if invalid (see APT interrupt in Section 2.1.2).

Release RFD Holdoff (rhdf)naxx00010

Any Ready For Data (RFD) holdoff caused by a 'hdfa' or 'hdfe' is released.

Holdoff on All Data (hdfa) 0/1xx00011

A Ready For Data (RFD) holdoff is caused on every data byte until the command is loaded with CS set to zero. The handshake must be completed after each byte has been received by the MPU using the 'rhdf' command.

Holdoff on End (hdfe)0/1xx00100

A RFD holdoff will occur when an end of data string message (EOI true with ATN false) is received over the interface. This holdoff must be released using 'rhdf'.

Set New Byte Available False (nbaf)naxx00101

If a talker is interrupted before the byte just stored in the data out register is sent over the interface, this byte will normally be transmitted as soon as the ATN line returns to the false state. If, as a result of the interrupt, this byte is no longer required, its transmission may be suppressed using the 'nbaf' command.

Force Group Execute Trigger (fget)0/1xx00110

The state of the TR output from the TMS9914A is affected when this command is executed. If the CS bit is zero, the line is pulsed high for approximately 5 clock cycles (1 μs at 5 MHz). If CS is one, the TR line goes high until 'fget' is sent with CS equal to zero. No interrupts or handshakes are initiated.

Return to Local (rtl)0/1xx00111

Provided the local lockout (LLO) has not been enabled, the remote/local status bit is reset, and an interrupt is generated (if enabled) to inform the host microprocessor that it should respond to the front panel controls. If the CS bit is set to one the 'rtl' command must be cleared (CS = 0) before the device is able to return to remote control. If CS is set to zero, the device may return to remote without first clearing 'rtl'.

Force End or Identify (feoi) naxx01000

This command causes the EOI message to be sent with the next data byte. The EOI line is then reset.

Listen Only (lon) 0/1xx01001

The listener state is activated until the command is sent with CS set to 0 or until deactivated by a bus command.

Talk Only (ton) 0/1xx01010

The talker state is activated until the command is sent with CS set to 0 or until deactivated by a bus command.

NOTE

'ton' and 'lon' are included for use in systems without a controller. However, where the TMS9914A is being used as a controller, it utilizes the 'lon' and 'ton' functions to set itself up as a listener or talker, respectively. Care must therefore be taken to ensure these functions are reset if sending UNL or OTA.

Go to Standby (gts)naxx01011

Issued by the controller in charge to set the ATN line false.

Take Control Synchronously (tcs)naxx01101

Control is again taken by the controller in charge, and ATN is asserted. If the controller is not a true listener, the shadow handshake command must be used to monitor the handshake lines so that the TMS9914A is synchronous with the talker/ listeners and only sends ATN true at the end of byte transfer. This ensures that no data is lost or corrupted.

Request Parallel Poll (rpp)0/1xx01110

This is executed by the controller in charge to send the parallel poll command over the interface (the TMS9914A must be in the Controller Active State so that the Attention line is asserted). The poll is completed by reading the Command Pass Through Register to obtain the status bits, then sending 'rpp' with the CS bit at zero.

Take Control Asynchronously (tca)naxx01100

This command is used by the controller in charge to set the attention line true and to gain control of the interface. The command is executed immediately and data corruption or loss may occur if a talker/listener is in the process of transferring a data byte.

Send Interface Clear (sic)0/1xx01111

The IFC line is set true when this command is sent with CS set to one. This must only be sent by the system controller and should be reset (CS = 0) after the IEEE minimum time for IFC has elapsed (100 μs). The system controller is put into the controller active state.

Send Remote Enable (sre)0/1xx10000

Issued by the system controller to set the REN line true and send the remote enable message over the interface. REN is set false by sending 'sre' with CS at zero.

Request Control (rqc)naxx10001

When the TCT command has been recognized via the unidentified command pass through, this command is sent by the MPU. The TMS9914A waits for the ATN line to go false and then enters the controller active state (CACS).

Release Control (ric)naxx10010

This command is used after TCT has been sent and handshake completed to release the ATN line and pass control to another device.

Disable All Interrupts (dai)0/1xx10011

The $\overline{\text{INT}}$ line is disabled, but the interrupt registers and any holdoffs selected are not affected.

Pass Through Next Secondary (pts)naxx10100

This feature may be used to carry out a remote configuration of a parallel poll. The parallel poll configure command (PPC) is passed through the TMS9914A as an unrecognized addressed command and is identified by the MPU. The 'pts' command is loaded, and the next byte received by the TMS9914A is passed through via the Command Pass Through Register. This would be the parallel poll enable (PPE), which is read by the microprocessor.

Set T1 Delay (std1)1xx10101

The T1 delay time can be set to 6 clock cycles (1.2 μs at 5 MHz) if this command is sent with the CS bit at one. The TI delay time is 11 clock cycles (2.2 μs at 5 MHz) following a power-on reset or if the command is sent with CS set to zero.

Shadow Handshake (shdw)0/1xx10110

This feature enables the controller in charge to carry out the listener handshake without participating in a data transfer. The Data Accepted line (DAC) is pulled true a maximum of 3 clock cycles after Data Valid (DAV) is received, and Not Ready For Data (NRFD) is allowed to go false as soon as DAV is removed.

The shadow handshake function allows the 'tcs' command to be synchronized with the Acceptor Not Ready State (ANRS) so that ATN can be re-asserted without causing the loss or corruption of data byte. The END interrupt can also be received and causes a RFD holdoff to be generated.

Very Short T1 Delay (vstd1)0/1xx10111

If this feature is enabled, the GPIB settling time (T1) will be reduced to 3 clock cycles (600 ns at 5 MHz) on the second and subsequent data bytes when ATN is false. Otherwise, the GPIB settling time is determined by the std1 feature.

Request Service Bit 2 (rsv2)0/1xx11000

The rsv2 bit performs the same function as the rsv1 bit (see Section 2.1.8) but provides a means of requesting service which is independent of the Serial Poll Register.

This allows minor updates to be made to the Serial Poll Register without affecting the state of the request service.

In addition, rsv2 is cleared when the serial poll status byte is sent to the controller during a serial poll. It is therefore used in situations where a service request is simply a request from an instrument for the controller to poll its status. As soon as this happens, rsv2 is cleared since the reason for requesting service has been satisfied. This eliminates the burden of clearing the bit from the host MPU but also guarantees that rsv2 is cleared before another serial poll can occur. If this were not so, there would be a possibility of a second status byte being sent with the RQS message true, which could result in confusion for the controller. (rsv2 is cleared on: SPAS.(APRS1 + APRS2).STRS). It should be noted that the vstd1 and rsv2 features were not present on the TMS9914.

2.1.7 Bus Status Register

ATN	DAV	NDAC	NRFD	EOI	SRQ	IFC	REN	
D0	D1	D2	D3	D4	D5	D6	D7	MPU BUS

The host MPU may examine the status of the GPIB management lines at the time of reading.

The IFC bit of this register does not indicate a true value if the device is a system controller using the 'sic' auxiliary command.

2.1.8 Serial Poll Register

S8	rsv1	S6	S5	S4	S3	S2	S1	
DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	GPIB
D0	D1	D2	D3	D4	D5	D6	D7	MPU BUS

S8, S6-S0 Device status
rsv1 Request service bit 1

Bits S8, S6-S1 of this register are sent out over the GPIB when the device is addressed during a serial poll. They are cleared by a hardware reset but not by 'swrst' and may therefore be set up during configuration of the chip. These bits are fully double buffered and if the register is written to while the device is addressed during a serial poll (serial poll active state, SPAS), the value written is saved, and these bits are updated when SPAS is terminated.

The rsv1 bit provides an input to the service request function of the TMS9914A and is used to instruct this to request that the controller service the device. When rsv1 is set true, the SRQ line is pulled true on the GPIB, and the controller typically responds by setting up a serial poll to obtain the status of all instruments on the bus that may require service. When the TMS9914A is addressed to send its status byte, SRQ is set false, and the status byte is sent with the RQS message true on DIO7. The rsv1 bit must then be cleared and set true again if service is to be requested a second time. The SPAS interrupt is set immediately following the status byte being sent.

The rev1 bit is also cleared by the hardware reset pin but not by 'swrst'. It is not double-buffered but the service request function comprehends changes in the state of rsv1 while the device is in SPAS. The Serial Poll Register may therefore be written to any time. Section 3.5 contains more information on the service request function. Note that the rev1 bit of the TMS9914 was simply referred to as 'rsv' since this device did not have an rsv2 auxiliary command.

2.1.9 Command Pass Through Register

DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	GPIB
D0	D1	D2	D3	D4	D5	D6	D7	MPU BUS

This provides a means of directly inspecting the GPIB data lines (DIO(8-1)). It has no storage and should only be used when the data lines are known to be in a steady state such as will occur during a DAC holdoff or in CPWS during a parallel poll. It is used to read unrecognized commands and secondaries following a UNC interrupt or to read secondary addresses following an APT interrupt. In addition, an active controller uses this register to read the results of a parallel poll at least 2μs after setting the 'rpp' auxiliary command.

2.1.10 Parallel Poll Register

PP8	PP7	PP6	PP5	PP4	PP3	PP2	PP1	
DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	GPIB
D0	D1	D2	D3	D4	D5	D6	D7	MPU BUS

When a controller initiates a parallel poll, the contents of this register are presented to the GPIB data lines. If all bits of the register are cleared, then none of the lines DIO(8-1) will be pulled low during a parallel poll which corresponds to the Parallel Poll Idle State (PPIS) of IEEE-488. If it is desired to participate in a parallel poll, then the bit corresponding to the desired parallel poll response is set to a 1.

The Parallel Poll Register is fully double buffered. If it is written to during a parallel poll, the new value is held until the parallel poll ends, at which point the register is updated. This permits the host MPU to update the parallel poll response completely asynchronously to the GPIB.

If this register is cleared by the hardware RESET pin but not by 'swrst,' it may be loaded while the chip is being configured with 'swrst' set.

2.1.11 Data In Register

DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	GPIB
D0	D1	D2	D3	D4	D5	D6	D7	MPU BUS

This register is used to hold data received by the TMS9914A when it is a listener. It is loaded during Accept Data State (ACDS1) and, following this, an RFD holdoff will occur. This will normally be released when the byte is read by the host MPU, but if the Holdoff On All Data (hdfa) feature is selected, this holdoff must be released by the 'rhdf' auxiliary command.

If the Holdoff On End (hdfe) feature is selected, the RFD holdoff will be released by reading the Data In Register. But if the EOI line is true when the byte is received, reading the data byte will not release the holdoff and rhdf must be used.

As the Data In Register is loaded, the BI interrupt is set. The END interrupt is set simultaneously if the byte is accompanied by a true EOI line.

2.1.12 Data Out Register

DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	GPIB
D0	D1	D2	D3	D4	D5	D6	D7	MPU BUS

The Data Out register is used by a controller or talker for sending interface messages and device dependent messages. When the TMS9914A enters the Talker Active State (TACS) or the Controller Active State (CACS), the contents of the Data Out Register are presented to the GPIB data lines (DIO(8-1)), and the byte is sent over the bus under the control of the Source Handshake. Each time a byte is written, the source handshake is enabled, and the byte is sent. If the handshake is interrupted before the byte can be sent, then it will be sent next time the Source Handshake becomes active unless a new byte available false (nbaf) auxiliary command is written. This has the effect of clearing an unsent byte from the Data Out Register, and although the register itself is not cleared, the TMS9914A behaves as if it had not been loaded.

Each time the source handshake becomes active and there is no unsent byte in the Data Out Register, a BO interrupt will occur informing the host MPU that the Data Out Register is available for use.

The Data In Register and Data Out Register operate independently. The Data Out Register is not double buffered, and its contents are output directly to the data lines of the GPIB.

2.2 DIRECT MEMORY ACCESS

The TMS9914A can operate in DMA using the $\overline{ACCRQ}$ (DMA request) and $\overline{ACCGR}$ (DMA grant) DMA handshake lines. The operation is automatic within the TMS9914A and needs no 'mpu' configuration.

The $\overline{ACCRQ}$ signal is set by (BO.$\overline{CACS}$ + BI) and can therefore not be used by a controller while ATN is asserted. It is reset by 'swrst' readin data in register, writing to the data out register and $\overline{ACCGR}$. It is not cleared by reading interrupt status register 0.

If using DMA, the internal CE and addressing is disabled by the $\overline{ACCGR}$ signal going low and $\overline{ACCGR}$ will automatically address either the data in register (DBIN = 0) or the data out register (DBIN = 1).

NOTE

The sense of DBIN is inverted for DMA operation.

At the end of a DMA read from memory sequence, the $\overline{ACCRQ}$ will be left low (also BO bit set). It may be necessary for the 'mpu' to clear this in some circumstances, e.g., starting DMA write to memory sequence.

In DMA it is recommended that the MA interrupt be unmasked to prevent errors due to interrupted data streams.

If DMA is not being utilized, the $\overline{ACCGR}$ signal must be held high. In this case, the $\overline{ACCRQ}$ signal can be used as a separate interrupt line for BO and BI. This allows faster 'mpu' transfers to take place as it is not necessary to read the interrupt register to find the cause of the interrupt. Figure 2-2 shows a typical DMA configuration.

2.3 TERMINAL ASSIGNMENTS AND FUNCTIONS

The IEEE-488 standard uses the negative logic convention for the GPIB lines. The FALSE state (0) is represented by a high voltage (> 2.0 V); the TRUE state (1) is represented by a low voltage (> 0.8 V). The GPIB terminations of the TMS9914A are in agreement with this convention. For example, if Data Valid is true (1), the DAV line is pulled low by the device. These terminations are connected to the bus via noninverting buffers to obtain the correct signal polarity.

Note that the terminations on the microprocessor side of the device are in positive logic (true state (1) = high voltage . false state (0) = low voltage). This is in agreement with the logic convention used by most microprocessors. Thus if:

D0(MSB)							D7(LSB)
0	1	1	0	1	0	0	1

is written into the data out register, it will appear as:

DIO8(MSB)							DIO1(LSB)
HIGH	LOW	LOW	HIGH	LOW	HIGH	HIGH	LOW

on the IEEE-488 D10 lines.

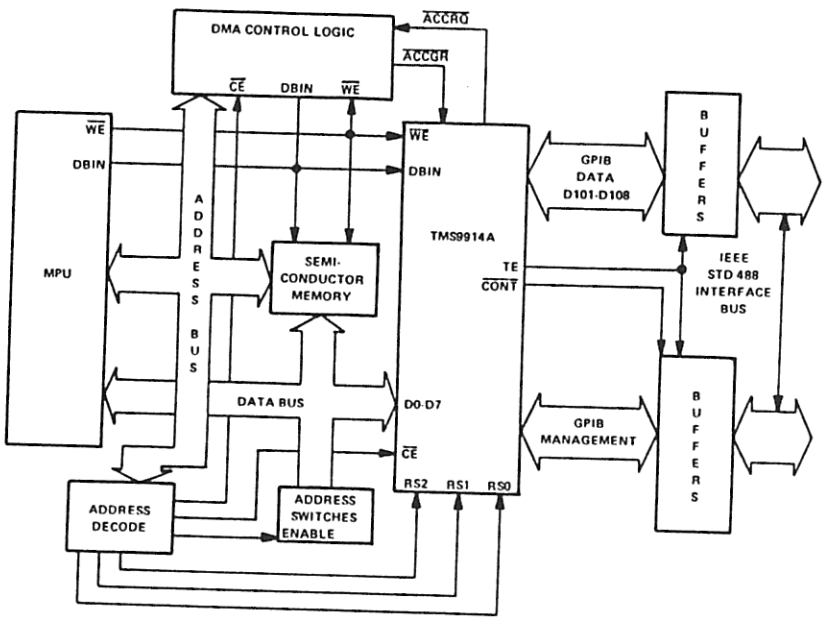
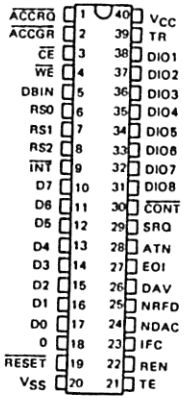


FIGURE 2-2 – DMA CONFIGURATION

TERMINAL ASSIGNMENTS AND FUNCTIONS

SIGNATURE	PIN	I/O (TYPE)	DESCRIPTION
DIO8	31	I/O(p/p)	DIO8 through DIO1 are the data input/output lines on the GPIB side. These pins connect to the IEEE-488 bus via non-inverting transceivers.
DIO7	32	I/O(p/p)	
DIO6	33	I/O(p/p)	
DIO5	34	I/O(p/p)	
DIO4	35	I/O(p/p)	
DIO3	36	I/O(p/p)	
DIO2	37	I/O(p/p)	
DIO1	38	I/O(p/p)	
DAV	26	I/O(p/p)	DATA VALID: handshake line controlled by source to show acceptors when valid data is present to the bus.
NDAC	24	I/O(p/p)	NOT DATA ACCEPTED: handshake line. Acceptor sets this false (high) when it has latched the data from the I/O lines.
NRFD	25	I/O(p/p)	NOT READY FOR DATA: handshake line. Sent by acceptor to indicate readiness for the next byte.
ATN	28	I/O(p/p)	ATTENTION: sent by controller in charge. When true (low), interface commands are being sent over the DIO lines. When false (high), these lines carry data.
REN	22	I/O(o/d)	REMOTE ENABLE: sent by system controller to select control either from the front panel or from the IEEE bus.
IFC	23	I/O(o/d)	INTERFACE CLEAR: sent by the system controller to set the interface system into a known quiescent state. The system controller becomes the controller in charge.
SRQ	29	I/O(p/p)	SERVICE REQUEST: set true (low) by a device to indicate a need for service.
EOI	27	I/O(p/p)	END OR IDENTIFY: if ATN is false (high), this indicates the end of of a message block. If ATN is true (low), the controller is requesting a parallel poll.
$\overline{CONT}$	30	O(p/p)	Indicates if a device is controller in charge. It is used to control direction of SRQ and ATN in pass control systems. Logically, it is (CIDS + CADS).
TE	21	O(p/p)	TALK ENABLE: controls the direction of the transfer of the line transceivers. Logically, it is:(CACS + TACS + EIO.ATN.(CIDS + CADS). $\overline{SWRST}$)
D0	17	I/O(p/p)	Data transfer lines on the MPU side of the device
D1	16	I/O(p/p)	
D2	15	I/O(p/p)	
D3	14	I/O(p/p)	
D4	13	I/O(p/p)	
D5	12	I/O(p/p)	
D6	11	I/O(p/p)	
D7	10	I/O(p/p)	
RS0	6	I	REGISTER SELECT LINES: determine which register is addressed by the MPU during a read or write operation.
RS1	7	I	
RS2	8	I	



TERMINAL ASSIGNMENTS AND FUNCTIONS (continued)

SIGNATURE	PIN	I/O (TYPE)	DESCRIPTION
CE	3	I	CHIP ENABLE: $\overline{CE}$ low allows access of read and write registers. If $\overline{CE}$ is high, D0-D7 are in high impedance unless $\overline{ACCGR}$ is low.
$\overline{WE}$	4	I	WRITE ENABLE: when active (low), indicates to the TMS9914A that data is being written to one of its registers.
DBIN	5	I	DATA BUS IN: an active (high) state indicates to the TMS9914A that a read is about to be carried out by the MPU.
$\overline{INT}$	9	O(o/d) (no pullup)	INTERRUPT: sent to the MPU to cause a branch to a service routine.
$\overline{ACCRQ}$	1	O(p/p)	ACCESS REQUEST: this pin becomes active (low) to request a direct memory access.
$\overline{ACCGR}$	2	I	ACCESS GRANTED: when received from the direct memory access control logic this enables the byte onto the data bus. ACCGR must be high when not participating in DMA transfer.
$\overline{RESET}$*	19	I	INITIALIZES the TMS9914A at power-on.
TR	39	O(p/p)	TRIGGER: activated when the GET command is received over the interface or the fget command is given by the MPU.
$\overline{0}$	18	I	CLOCK input: 500 kHz to 5 MHz. Need not be synchronous to system clock.
V_{SS}	20		Ground reference voltage.
V_{CC}	40		Supply voltage (+5 V nominal).

(p/p) = push/pull output.
(o/d) = open drain output with internal pull up.
* The hardware RESET pin has the following effect on the TMS9914A:
- Serial and Parallel Poll registers cleared
- All clear/set auxiliary commands cleared except 'swrst'
- 'swrst' auxiliary command set. This holds the TMS9914A in known states, as described in Section 2.1.6.

2.4 TRANSCEIVER CONNECTIONS

There are three linear transceivers designed to work with the TMS9914A: the SN75160, SN75161, and SN75162. Data sheets for these are included as Appendix C. Figure 2-3 shows the possible transceiver connections. Note that there is a corresponding pinout between the TMS9914A and the transceivers. This allows the whole GPIB interface to be laid out in a very small area of printed circuit board.

The SN75160 is a 20 pin device used to buffer the IEEE-488 data lines (DIO(8-1)) in all applications. The direction of the buffers is controlled by the Talk Enable (TE) output of TMS9914A. This active high signal becomes true whenever there is an interface function of the TMS9914A not sending the NUL message on DIO(8-1), that is, when the device is in TACS, CACS, SPAS, or PPAS. The Pull-Up Enable (PE) input of the SN75160 is an active high input which selects whether the 'DIO(8-1)' lines are driven by open collector or push/pull buffers. A push/pull buffer is required if faster data rates are required and the 'atdl' and/or the 'vatdl' features are used. Open collectors must be used if parallel polling is being used in a particular GPIB environment. If only one of these features is desired the PE input may be hardwired otherwise it must be derived from ATN and EOI, as shown in Figure 2-3.

The SN75161 is a 20-pin device used to buffer the IEEE-488 management lines. It may be used for a talker/listener device or for a controller which does not pass control. The direction of the handshake line buffers NRFD, NDAC, DAV are again controlled by the TE signal. However, the SRQ, ATN, REN, and IFC buffers are controlled by the DC input of the SN75161, which connects to the Controller Active ($\overline{CONT}$) output of the TMS9914A. $\overline{CONT}$ becomes low whenever the TMS9914A is an active controller, that is, when it is not in CIDS or CADS. The SN75161 also includes the logic necessary to control the direction of the EOI buffer. This is dependent on the TE signal when ATN is false (high) and the DC signal when ATN is true (low).

The SN75162 is a 22-pin device which may be used to buffer the IEEE-488 management lines in all applications including devices which pass control. The SN75162 has a separate pin to control the direction of the REN and IFC buffers, but is otherwise identical to the SN75161 in all other respects. This input is the System Controller input (SC) which may be hardwired or switchable to determine whether or not the instrument in question is a system controller or not. Note that a device which has its buffers configured as a non-system controller should never use the 'sic' and 'sre' auxiliary commands for reasons explained in Section 3.8.3.

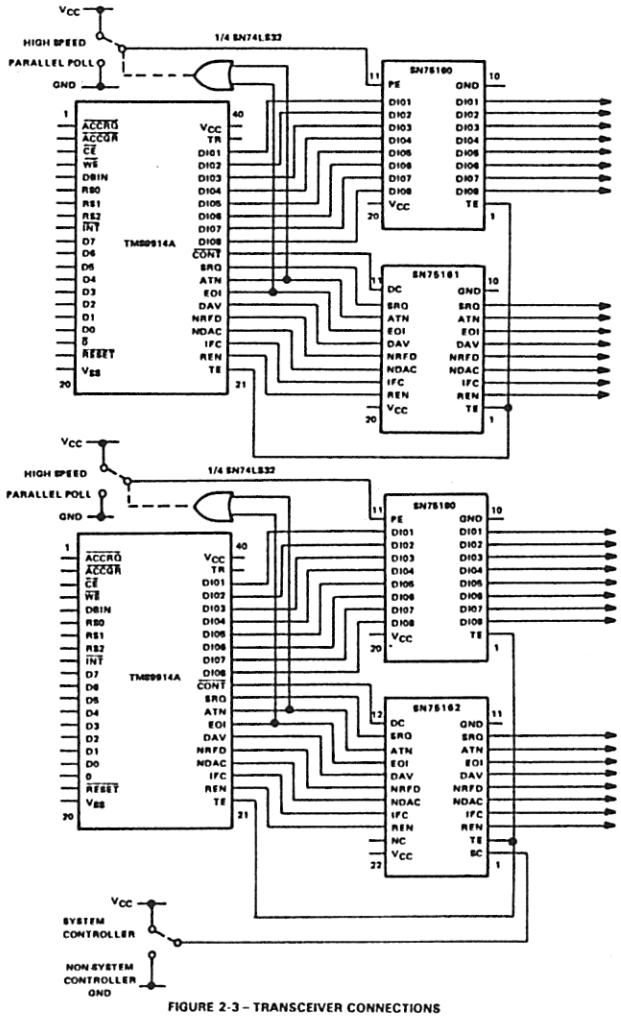


FIGURE 2-3 – TRANSCEIVER CONNECTIONS

3. STATE DIAGRAM IMPLEMENTATION

This section presents the state diagrams for the TMS9914A.

Where equivalent, the names of TMS9914A states are the same as those of IEEE-488. In some cases, IEEE-488 states have been divided, for example, ACDS of the IEEE-488 has been split into ACDS1 and ACDS2. The convention of lower case characters for local messages and upper case for remote messages and interface states is retained.

State diagrams with remote message outputs are supplemented with tables. T is used to represent a true output and F a false output. Parentheses denote a passive output; otherwise, it is active. The outputs shown are the values presented to the bus and assume the use of the SN75160 and SN75161 or SN75162 transceivers or their logical equivalents. The symbol (NUL) associated with DIO(1-8) indicates that each of these lines is sent passive false by the function in question.

NOTE

> An arrow into a state with no state as its origin represents a transition from every other state on the diagram. Note, however, that this does not imply that all exit conditions from the destination state are overridden. If such an entry condition is true and, simultaneously, an exit condition is true then this represents an illegal situation and should be avoided. Such situations will not occur in normal operation of the device.

No maximum timings are discussed. The TMS9914A with its recommended transceivers meets all IEEE-488 maximum timing requirements as may be determined from Section 4.4 and Appendix C. If the TMS9914A is used with other transceivers, then it must be ensured that these requirements are still met.

3.1 AUXILIARY COMMANDS

There are two basic types of commands implemented in the auxiliary command register: immediate execute and clear/set.

The clear/set commands are used to enable and disable the various features of the TMS9914A. The particular feature is selected by the code on f0-f4 (see Section 2.1.6) and it is set or cleared according to the value on the cs bit. For the purposes of the state diagrams, the mnemonic of a clear/set command simply represents its current state.

The immediate execute auxiliary commands remain active for the duration of a strobe signal after the auxiliary command register has been written to. This is represented in the form of a state diagram in Figure 3-1. Note that writes to the auxiliary command register must be spaced by at least five clock cycles. For the purposes of the remaining state diagrams, the immediate execute commands are represented as the mnemonic gated by the auxiliary command strobe state (AXSS).

The clear/set bit of the auxiliary command register is used by several of the immediate execute commands, for example, 'dacr' uses it to differentiate between valid and not valid secondary addresses when releasing a DAC holdoff on a secondary address. The 'lon' and 'ton' auxiliary commands are also considered immediate execute, as described in Section 3.4.

The 'fget' and 'rtl' auxiliary commands are both immediate execute and clear/set. They may be cleared or set in the normal way, but if they are cleared when they are already in the false state, they will pulse true for the duration of AXSS. In the following state diagrams, however, these are simply included in their clear/set form.

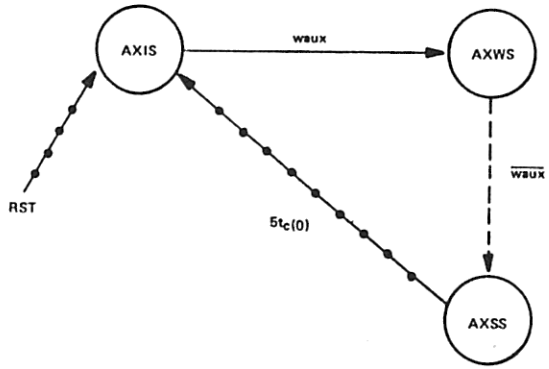


FIGURE 3-1 – TMS9914A AUXILIARY COMMAND STATE DIAGRAM

TABLE 3-1 – AUXILIARY COMMAND STATE DIAGRAM MNEMONICS

MESSAGES			STATES		
waux	=	write to auxiliary command register	AXIS	=	auxiliary command register idle state
$t_{c(0)}$	=	clock cycle time	AXWS	=	auxiliary command write state
			AXSS	=	auxiliary command strobe state

3.2 ACCEPTOR HANDSHAKE

The TMS9914A acceptor handshake is shown in Figure 3-2. The main variation from IEEE-488 to note is that the device remains in AIDS while the controller function is in CACS. The TMS9914A, therefore, does not monitor the commands which it sends over the bus and this places some restrictions on the user which are outlined in Section 3.8.

The accept data state of IEEE-488 (ACDS) is divided into two states. The first, (ACDS1) is used to strobe data into the Data In Register or to sequence the decoding of commands from the bus. All interrupts generated by the acceptor handshake (GET, MA, MAC, DCAS, APT, UCG, BI, and END) are generated by this state. The second (ACDS2) is used as a holding state where the device will remain in the event of a DAC holdoff.

As discussed in Section 2.1.2, certain of the commands will cause interrupts in ACDS1 and, if the interrupts are unmasked, a DAC holdoff will occur. The interrupts concerned are GET, MA, DCAS, UCG, and APT. This is represented in the state diagram by the signal SAHF which becomes true when one of the above interrupts is set if it is unmasked. It persists for the duration of ACDS1. This event is stored by causing the ADHS to become active which inhibits the transition from ACDS2 to AWNS. ADHS is cleared by 'dacr'. Table 3-15 shows the response of the TMS9914A to the various bus commands.

If a GET command is received in ACDS1, then the TR pin will be set high. This high condition persists throughout ACDS1 and ACDS2, which means that if a DAC holdoff occurs, the TR pin will remain high until the holdoff is released by a 'dacr' auxiliary command.

Two additional state diagrams are included to record the type of data received in ACDS1 when ATN is false. ANHS indicates that a data byte has been received and that an RFD holdoff should be caused before the next data byte is accepted. The holdoff may be released by reading the Data In Register unless the 'hdfa' feature is enabled in which case 'rhdf' must be used. AEHS shows that the last data byte was accepted with the EOI message true and the 'hdfe' feature set. This will cause an RFD holdoff which can only be released by 'rhdf'.

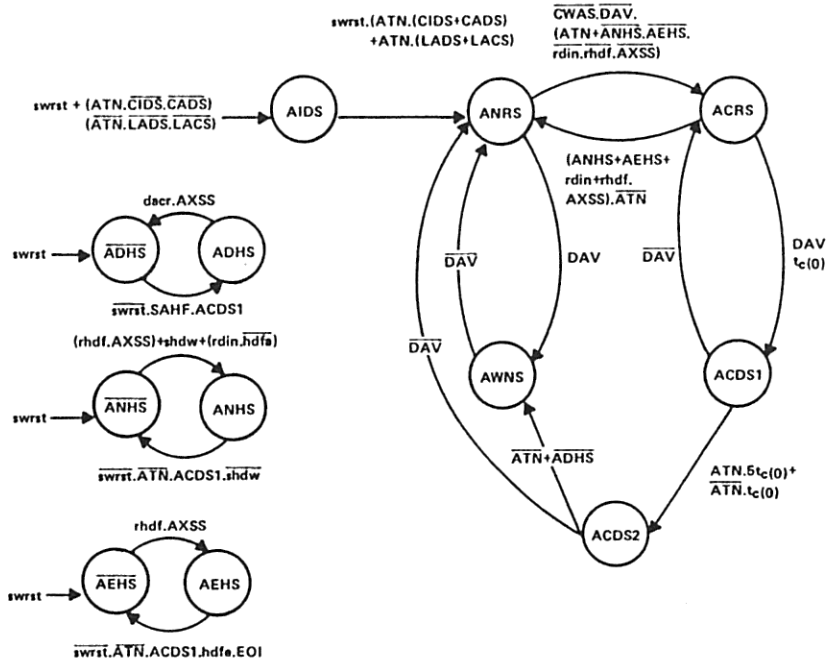


FIGURE 3-2 – TMS9914A ACCEPTOR HANDSHAKE STATE DIAGRAM

TABLE 3-2 – ACCEPTOR HANDSHAKE MNEMONICS

MESSAGES			STATES		
swrst	=	software reset	AIDS	=	acceptor idle state
dacr	=	DAC release	ANRS	=	acceptor not ready state
rhdf	=	release RFD holdoff	ACRS	=	acceptor ready state
shdw	=	shadow handshake	ACDS1	=	accept data state 1
rdin	=	read data in register	ACDS2	=	accept data state 2
hdfe	=	enable RFD holdoff after END messages received	AWNS	=	acceptor wait for new cycle state
hdfa	=	enable RFD holdoff on all data	ADHS	=	accept data holdoff state
ATN	=	attention	ANHS	=	acceptor not ready holdoff state
DAV	=	data valid	AEHS	=	acceptor not ready holdoff after 'END'
EOI	=	end or identify state	CWAS	=	controller wait for ANRS state (controller function)
RFD	=	ready for data	AXSS	=	auxiliary command strobe state (auxiliary command register)
DAC	=	data accepted	LADS	=	listener addressed state (listener function)
SAHF	=	set accept data holdoff state	LACS	=	listener active state (listener function)
$t_{c(0)}$	=	clock cycle time	CIDS	=	controller idle state (controller function)
			CADS	=	controller addressed state (controller function)

TABLE 3-3 – ACCEPTOR HANDSHAKE MESSAGE OUTPUTS

STATE	REMOTE MESSAGES SENT RFD	REMOTE MESSAGES SENT DAC	OTHER ACTIONS
AIDS	(T)	(T)	
ANRS	F	F	
ACRS	(T)	F	
ACDS1	F	F	ATN False: – data entered into Data In Register – BI interrupt generated – end interrupt generated if EOI is true. ATN true: – commands decoded – command related interrupts set – sahf set if command requires a DAC holdoff – TR pin set true if GET message is received – 'pts' feature cleared after UNC interrupt set
ACDS2	F	F	TR pin set true if GET command was received in ACDS1
AWNS	F	(T)	

3.3 SOURCE HANDSHAKE

The TMS9914A source handshake state diagram is shown in Figure 3-3. IEEE-488 states SIWS and SWNS have been removed. These record the false then true transition of 'nba' (new byte available) as the old data byte is removed and a new data byte is made ready. Instead the TMS9914A uses a separate state (SHFS) to record the availability of a data byte in the Data Out Register. This state is exited when a byte is written to the Data Out Register which enables the transition from SGNS to SDYS and the subsequent transmission of the byte. The SHFS is reentered as the byte is sent in STRS, but if the handshake is interrupted before this, then the fact that the byte has not been sent is recorded until the source handshake again becomes active. If, however, the byte in the data out register is to be disregarded, then 'nbaf' may be used to return the device to SHFS.

The status byte in the Serial Poll Register is continually available. The transition from SGNS to SDYS is not dependent on SHFS during a serial poll, that is, while SPAS is active. By separately recording the availability of a byte in the Data Out Register, a talker sending data may be interrupted for a serial poll without risk of a byte being lost.

The additional state SERS is included to detect an error condition on the bus. This will be entered when the source handshake tries to send a byte but finds both the NRFD and NDAC lines false at the same time. This condition will normally indicate for a controller that there are no devices powered up on the bus, or for a talker that there are no devices addressed to listen on the bus.

The state VSTS will be entered after the first data byte of a talker has been sent if the 'vstdl' feature is enabled. This enables a very short bus settling time ($4t_{c(0)}$) for all subsequent bytes until ATN next becomes true. The TMS9914A will not use the short bus settling time when it is an active controller.

NOTE

The TMS9914 did not implement the 'VSTS'. The bus settling time was therefore either $8t_{c(0)}$ or $12t_{c(0)}$ for 'stdl' set and not set respectively.

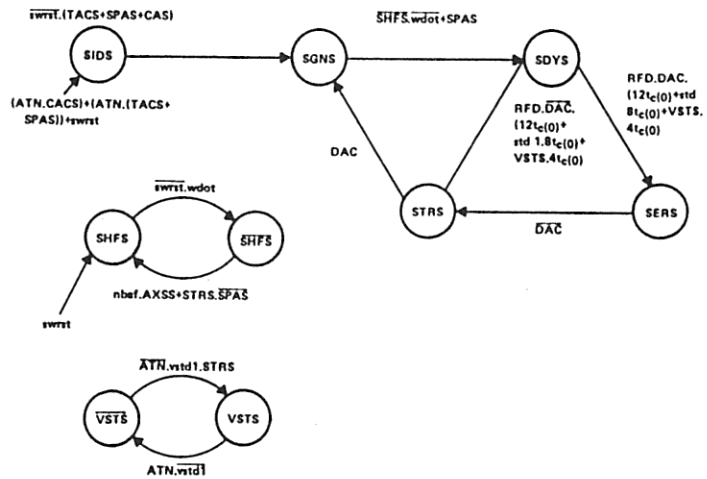


FIGURE 3-3 – TMS9914A SOURCE HANDSHAKE STATE DIAGRAM

TABLE 3-4 – SOURCE HANDSHAKE MNEMONICS

MESSAGES			STATES		
swrst	=	software reset	SIDS	=	source idle state
nbaf	=	new byte available false	SGNS	=	source generate state
wdot	=	write to the data out register	SDYS	=	source delay state
stdl	=	enable short bus settling time	SERS	=	source error state
vstdl	=	enable very short bus settling time	STRS	=	source transfer state
ATN	=	attention	SHFS	=	source holdoff state
RFD	=	ready for data	VSTS	=	very short bus settling time state
DAC	=	data accepted	TACS	=	talker active state (talker function)
$t_{c(0)}$	=	clock cycle time	CACS	=	controller active state (controller function)
			SPAS	=	serial poll active state (talker function)
			AXSS	=	auxiliary command strobe state (auxiliary command register)

TABLE 3-5 – SOURCE HANDSHAKE MESSAGE OUTPUTS

STATE	REMOTE MESSAGES SENT DAV	OTHER ACTIONS
SIDS	(F)	BO interrupt and ACCRQ set
SGNS	F	true if SHFS is false and SPAS is not true
SDYS	F	
SERS	F	ERR interrupt set true
STRS	T	

3.4 TALKER AND LISTENER FUNCTIONS

Figures 3-4 and 3-5 show the TMS9914A listener and talker state diagrams, which serve the purpose of the listener and talker or extended listener and extended talker functions of IEEE-488, depending on the state of the APT interrupt mask bit.

The TMS9914A does not recognize secondary addresses on-chip and these must be passed through to the host MPU for verification. Secondary addressing is enabled by unmasking the APT interrupt. A secondary address will cause this interrupt if the last primary command received was a primary address of the device, that is, it is in TPAS or LPAS. A DAC holdoff will also occur. The host MPU must respond to the interrupt by reading the secondary from the Command Pass Through Register and identifying it as being valid or not valid. The holdoff may then be released with a 'dacr' auxiliary command, the sense of the 'cs' bit being used to indicate a valid (cs = 1) or not valid (cs = 0) secondary. If a valid secondary address is indicated then the TMS9914A will enter TADS or LADS depending on whether it is in TPAS or LPAS.

The 'lon' and 'ton' auxiliary commands together with the clear/set bit (cs) have a direct influence on the appropriate state diagrams. Therefore, although they appear as ordinary clear/set auxiliary commands, they can be effectively cleared by other bus events. For example, if a TMS9914A addresses itself as a listener via the 'lon' command it may be returned to LIDS by an UNL command from the bus at a later time.

The 'lon' and 'ton' auxiliary commands are used to implement two features of IEEE-488. First, talk only and listen only are used in situations where there is no active controller on the bus. Note that the 'lon' and 'ton' commands are linked with these features to indicate to the user that these commands are not enabled by CAS as are 'ltn' and 'lun' of IEEE-488.

Second, the 'lon' and 'ton' auxiliary commands are used by an active controller to address itself. IEEE-488 provides for a controller to address itself to listen via the 'ltn' and 'lun' message but there is no corresponding message for the talker. Hence, when a controller addresses itself to talk via 'ton,' it must send its talk address over the bus and similarly, if it sends another talk address over the bus then it must un-address itself by writing 'ton' false.

When the TMS9914A enters SPAS, the contents of the serial poll register are sampled and presented on DIO(8-1). These will remain unchanged until SPAS is exited. The source handshake will, however, send this status byte as many times as the controller will accept it.

The internal IFC signal of the TMS9914A (IFCIN) is suppressed when the device itself is sending IFC in order to simplify implementation of the controller function (see Section 3.8.3). Therefore, the send interface clear (sic) auxiliary command is included with IFCIN to return the talker and listener functions to their idle states and allow a system controller to clear its own interface.

A separate state diagram is included to control the sending of the END message of IEEE-488. If the 'feoi' auxiliary command is written followed by loading a byte into the Data Out Register, the TMS9914A will enter ERAS, and the EOI line will be asserted as 'DIO(8-1)' begin to change. The function will enter ENAS as soon as the source handshake begins to send this byte, and EOI will be released when the Data Out Register is next loaded. If it is desired to send EOI true with the next byte as well, then 'feoi' may be written before the Data Out Register returns the device to ERAS.

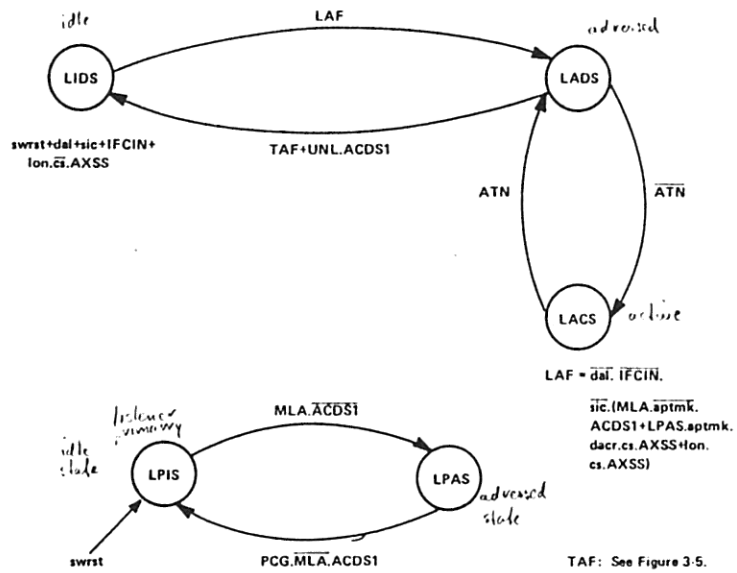
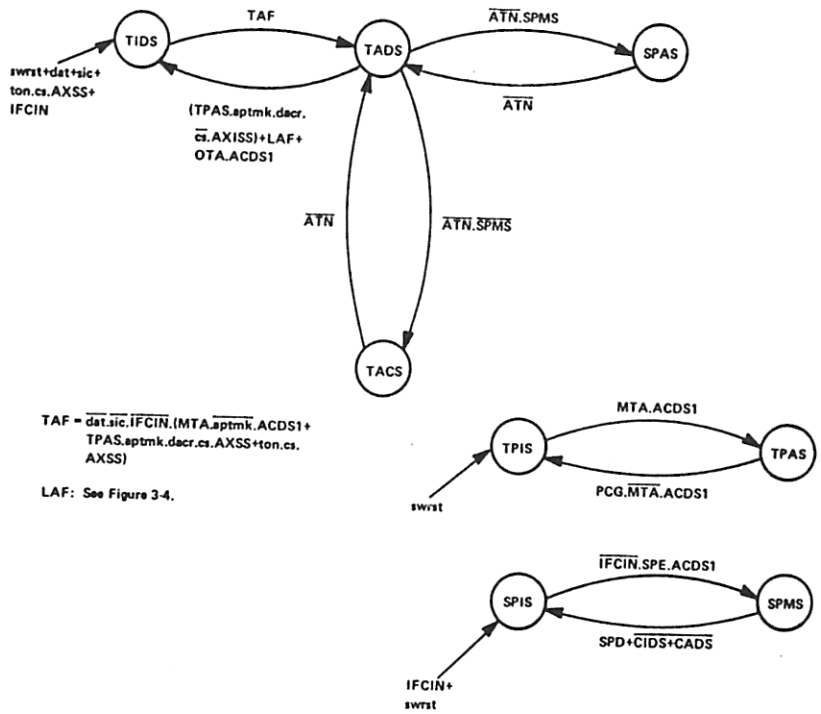


FIGURE 3-4 – TMS9914A LISTENER STATE DIAGRAM



TAF = dat.sic.IFCIN.(MTA.aptmk.ACDS1+
TPAS.aptmk.dacr.cs.AXSS+ton.cs.
AXSS)

LAF: See Figure 3-4.

FIGURE 3-5 — TMS9914A TALKER STATE DIAGRAM

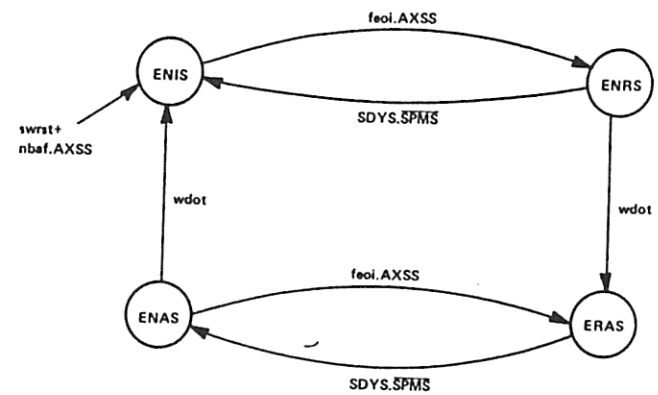


FIGURE 3-5 – TMS9914A TALKER STATE DIAGRAM (Continued)

TABLE 3-6 – TALKER AND LISTENER MNEMONICS

MESSAGES			STATES		
swrst	=	software reset	LIDS	=	listener idle state
dal	=	disable listener	LADS	=	listener addressed state
dat	=	disable talker	LACS	=	listener active state
sic	=	send interface clear	LPIS	=	listener primary idle state
lon	=	listen only	LPAS	=	listener primary addressed state
ton	=	talk only	TIDS	=	talker idle state
cs	=	clear/set bit of the auxiliary command register	TADS	=	talker addressed state
dacr	=	release 'DAC' holdoff	TACS	=	talker active state
aptmk	=	address pass through interrupt mask	SPAS	=	serial poll active state
nbaf	=	new byte available false	SPIS	=	serial poll idle state
feoi	=	force 'EOI'	SPMS	=	serial poll mode state
wdot	=	write to the Data Out Register	TPIS	=	talker primary idle state
ATN	=	attention	TPAS	=	talker primary addressed state
IFCIN	=	internal interface clear message (a debounced signal, suppressed by 'sic')	ENIS	=	end idle state
EOI	=	end or identify	ENRS	=	end ready state
PCG	=	primary command group	ERAS	=	end ready and active state
MLA	=	my listen address	ENAS	=	end active state
MTA	=	my talk address	SDYS	=	source delay state (source handshake)
OTA	=	other talk address	CIDS	=	controller idle state (controller function)
SPE	=	serial poll enable	CADS	=	controller addressed state (controller function)
SPD	=	serial poll disable	ACDS1	=	accept data state 1 (acceptor handshake)
UNL	=	unlisten	AXSS	=	auxiliary command strobe state (auxiliary command register)
PCG	=	primary command group			

TABLE 3-7 – TALKER FUNCTION MESSAGE OUTPUTS

STATE	QUALIFIER	REMOTE MESSAGES SENT		OTHER ACTIONS
		RQS	EOI	DIO(8-1)
TIDS		(F)	(F)	(NUL)
TADS		(F)	(F)	(NUL)
TACS	ENIS.ENRS	(F)	F	DATA OUT REG
TACS	ENAS.ERAS	(F)	T	DATA OUT REG
SPAS	NPRS.SRQS	F	F	SERIAL POLL REG
SPAS	APRS1.APRS2	T	F	SERIAL POLL REG

3.5 SERVICE REQUEST FUNCTION

Figure 3-6 shows the state diagram for the TMS9914A service request function. The device has two means of implementing the request service (rsv) local message of IEEE-488: the first, 'rsv1', is bit 7 of the Serial Poll Register; the second is the auxiliary command 'rsv2'. These are simply ORed together to provide an input to the service request function, and, in any particular application, only one would normally be used, the other being left in its hardware reset state.

The affirmative poll response state (APRS) of IEEE-488 is split into two states on the TMS9914A for the following reason: Consider the case where a device has requested service, has been serial polled, and then wishes to request service again. The host MPU must clear the 'rsv' message and then set it true again. Now suppose this temporary false condition happens within one occurrence of SPAS. If the service request function has been implemented exactly as per IEEE-488, it will not be recognized, and SRQ will not be asserted a second time. Therefore, 'rsv' may only be cleared when the device is known not to be in SPAS, which can only happen if it is cleared as a consequence of some pre-arranged action of the controller. This action would normally be a part of the service routine executed by the controller as a response to the request for service. For example, if service was requested by an instrument which had some data to send for processing or to a printing device then 'rsv' could be cleared when it is addressed to talk and send its data over the bus.

For many applications, the fact that the device has been serial polled after requesting service is considered sufficient response from the controller. The 'rsv' local message therefore simply becomes a request for the controller to read its serial poll status byte. It is then desirable to be able to clear and reassert 'rsv' at any time after the serial poll status byte has been polled and the SPAS interrupt set. The TMS9914A is able to record a false transition of 'rsv1' or 'rsv2' by moving from APRS1 to APRS2 even if the device is in SPAS. This makes the above approach to serial polling possible.

To further support this approach, the 'rsv2' auxiliary command is automatically cleared when the serial poll status byte is polled, ensuring that "rsv2' is cleared before a second serial poll can occur. If this were not the case, then the same status byte might be polled twice by the controller with the RQS bit true, which may indicate that two reasons for requiring service have arisen.

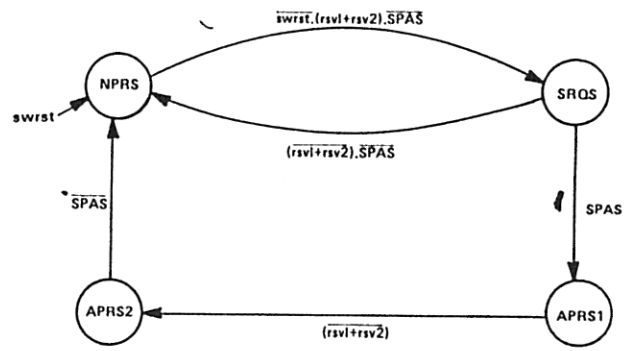


FIGURE 3-6 – SERVICE REQUEST STATE DIAGRAM

NOTE

The TMS9914 service request function was implemented exactly as per IEEE-488. Also, it had only one 'rsv' bit which was equivalent of the TMS9914A's 'rsv1.'

The TMS9914A will only send one serial poll status byte during each active period of SPAS., However, it will send this status byte as many times as the controller is prepared to accept it. Therefore, the controller should only read the status byte once per serial poll; otherwise, each time a status byte is sent with the RQS message true, the SPAS interrupt will be generated and 'rsv2' will be cleared.

TABLE 3-8 – SERVICE REQUEST MNEMONICS

MESSAGES			STATES		
swrst	=	software reset	NPRS	=	negative poll response state
srv1	=	request service 1 (bit 7 of serial poll register)	SRQS	=	service request state
rsv2	=	request service 2 (auxiliary command register)	APRS1	=	affirmative poll state 1
			APRS2	=	affirmative poll state 2
			SPAS	=	serial poll active state (talker function)

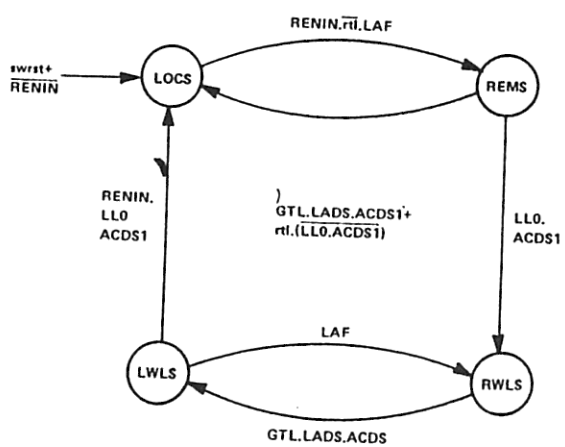
TABLE 3-9 – SERVICE REQUEST MESSAGE OUTPUTS

STATE	REMOTE MESSAGES SENT SRQ	OTHER ACTIONS
NPRS	(F)	
SRQS	T	
APRS1	(F)	– rsv2 cleared if in SPAS and STRS – SPAS interrupt set if in SPAS when STRS is exited
APRS2	(F)	– same as APRS1

3.6 REMOTE/LOCAL FUNCTION

The TMS9914A remote local state diagram is shown in Figure 3-7. It differs little from that of IEEE-488.

The complete listener function (LAF) is used to effect the transition from LOCS to REMS or from LWLS to RWLS. This means that if the APT interrupt is masked, the device will enter one of the remote states in response to its listen address, but if secondary addressing is enabled, then this will not happen until 'dacr' is written with 'cs' true in response to a valid secondary address. In addition, the transition to one of the remote states will occur if 'lon' is used to address the device to listen.



LAF: See Figure 3-4

FIGURE 3-7 – TMS9914A REMOTE LOCAL STATE DIAGRAM

TABLE 3-10 – REMOTE/LOCAL MNEMONICS

MESSAGES			STATES		
swrst	=	software reset	LOCS	=	local state
rtl	=	return to local	REMS	=	remote state
RENIN	=	internal remote enable message (debounced)	RWLS	=	remote with lockout state
GTL	=	go to local	LWLS	=	local with lockout state
LLO	=	local lockout	LADS	=	listener addressed state (listener function)
			ACDS1	=	accept data state 1 (acceptor handshake)

3.7 PARALLEL POLL FUNCTION

The parallel poll function of the TMS9914A only nominally supports logically-configured parallel poll. With a suitable software package, remotely-configured parallel poll may also be easily implemented. The state diagram is shown in Figure 3-8.

When the EOI and ATN lines become true simultaneously (the Identify message), the contents of the Parallel Poll Register are output to DIO(8-1). If parallel poll is to be used in a particular bus environment, then the Pull-Up Enable (PE) input of the SN75160 must be held low so that the DIO(8-1) are driven by open collector buffers. Parallel Poll, occurring when the Parallel Poll Register is in the hardware reset condition of all zeros, will result in none of DIO(8-1) being pulled low. This corresponds to the parallel poll idle state (PPIS). If it is desired to participate in a parallel poll, then the bit corresponding to the desired parallel poll response is set true. This implements the parallel poll standby state (PPSS), and, when the Identify message becomes true, the appropriate line of DIO(8-1) is pulled low. This is equivalent to the parallel poll active state (PPAS). Only one bit of the parallel Poll Register should be set true at once.

3.7.1 Remotely Configured Parallel Poll

The parallel poll configure command (PPC) is treated by the TMS9914A as an unrecognized addressed command. It is passed through when the TMS9914A is in LADS. If an instrument is to be remotely configured for parallel poll, then the pass through next secondary (pts) auxiliary command should be written before releasing the DAC holdoff. This will cause the next command received to also set a UNC interrupt if it is a secondary command. The secondary command will be either the parallel poll enable command (PPE) or the parallel poll disable command (PPD) and should be read from the Command Pass Through Register and identified. If it is the PPE command, then the attendant bits (S, P1, P2, P3) should be extracted and stored by the host MPU (see Section 2.9.3 of IEEE-488 1978). The S bit should then be matched against the individual status of the instrument (represented by 'ist'), and if they are the same, the bit corresponding to the parallel poll response, specified by P1, P2, P3, should be set true in the Parallel Poll Register. If this is not the case, then the Parallel Poll Register should be cleared if it is not already clear. After this, each time the individual status of the device changes, the 'ist' should again be matched against the S bit and the Parallel Poll Register updated accordingly until PPD or PPU is received.

If a PPD command is passed through after the 'pts' feature has been written, the Parallel Poll Register should be cleared before the DAC holdoff is released. The PPC command that precedes PPD is an address command; it is a means of eliminating individual members of a parallel poll. The parallel unconfigure command is treated by the TMS9914A as an unrecognized universal command. When it is passed through, the host MPU should clear its Parallel Poll Register before releasing the DAC holdoff. This command will clear all members of a parallel poll.

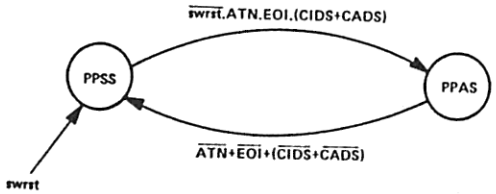


FIGURE 3-8 – TMS9914A PARALLEL POLL STATE DIAGRAM

TABLE 3-11 – PARALLEL POLL MNEMONICS

MESSAGES			STATES		
swrst	=	software reset	PPSS	=	parallel poll standby state
ATN	=	attention	PPAS	=	parallel poll active state
EOI	=	end or identify	CIDS	=	controller idle state (controller function)
			CADS	=	controller addressed state (controller function)

TABLE 3-12 – PARALLEL POLL MESSAGE OUTPUTS

STATE	REMOTE MESSAGES SENT	OTHER ACTIONS
	DIO(8-1)	
PPSS	(NUL)	
PPSS	PARALLEL POLL REG*	

* If there is a true bit in the Parallel Poll Register, it must be sent active; any false bit must be sent passive.

3.8 CONTROLLER FUNCTION

The controller function of the TMS9914A is greatly simplified compared with that of IEEE-488. It relies heavily on software support but, with suitable software, it enables all subsets of the controller function to be implemented. With this approach the controller logic is reduced to a small proportion of the chip area which means that the device may be economically used in situations where a talker/listener only is required.

Figure 3-9 shows the controller function state diagram. With suitable software, it will perform the full controller function, as described in the IEEE-488A 1980 supplement to the IEEE-488 1978. It therefore includes the additional state CSHS, which allows time for DAV to be recognized false by all devices on the bus before ATN is asserted. The 'tcs' local message is implemented by an immediate execute auxiliary command. The state CWAS is therefore added to record the occurrence of this command until the acceptor handshake enters ANRS and the device can enter CSHS. The 'tcs' auxiliary command also causes entry into CSHS although IEEE-488A 1980 allows it to move directly from CSBS to CSWS. This is done for convenience of implementation and results in the 'tcs' auxiliary command taking an extra 1.6 microseconds to assert ATN.

The delay between CSWS and CAWS is slightly less than specified in IEEE-488A 1980 but the total time taken in moving from CSWS to CACS is still greater than the specified minimum.

The Controller Parallel Poll State (CPPS) is not included on the TMS9914A. To conduct a parallel poll, a TMS9914A based controller must set the 'rpp' clear/set auxiliary command true when it is in CACS, moving it to CPWS which sends EOI true. The host MPU must then wait 2 microseconds before reading back the parallel poll responses via the Command Pass Through Register. The 'rpp' auxiliary command can then be cleared, EOI will go false, and the parallel poll is complete. The host MPU will receive a BO interrupt as soon as the TMS9914A reenters CACS and the source handshake becomes active.

3.8.1 Controller Self Addressing

As discussed in Section 3.2, the acceptor handshake does not operate when the controller is active. This means commands being sent are not monitored, and special precautions are required as a consequence of this when addressing devices and when passing control.

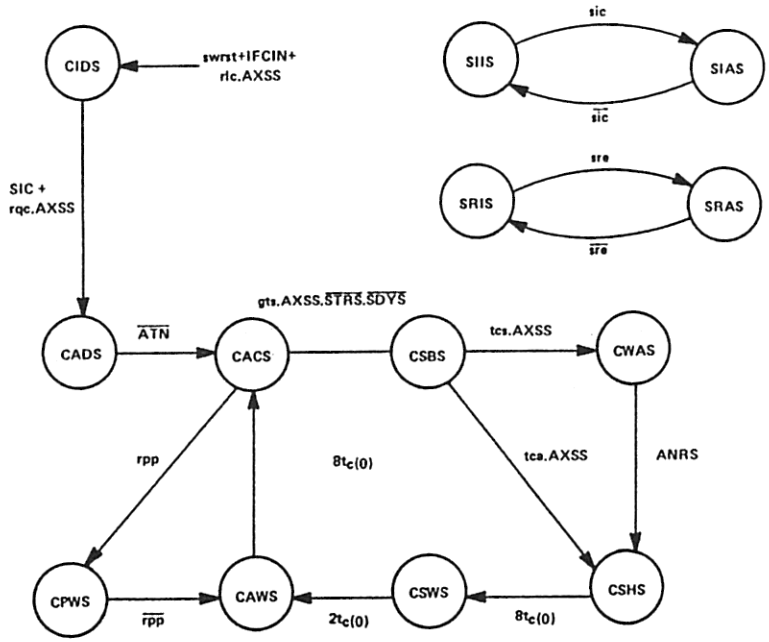


FIGURE 3-9 – TMS9914A CONTROLLER STATE DIAGRAMS

TABLE 3-13 – CONTROLLER FUNCTION MNEMONICS

MESSAGES			STATES		
swrst	=	software reset	CIDS	=	controller idle state
sic	=	send interface clear	CADS	=	controller addressed state
sre	=	send remote enable	CACS	=	controller active state
rqc	=	request control	CSBS	=	controller standby state
rlc	=	release control	CSHS	=	controller standby hold state
gts	=	go to standby	CSWS	=	controller synchronous wait state
tcs	=	take control synchronously	CAWS	=	controller active wait state
tca	=	take control asynchronously	CPWS	=	controller parallel poll wait state
rpp	=	request parallel poll	ANRS	=	acceptor not ready state (acceptor handshake)
IFCIN	=	internal interface clear message (a debounced signal which is suppressed if 'sic' is true)	SDYS	=	source delay state (source handshake)
ATN	=	attention	STRS	=	source transfer state (source handshake)
$t_{c(0)}$	=	clock cycle time	AXSS	=	auxiliary command strobe state (auxiliary command register)
			LWAS	=	controller wait for ANRS state

TABLE 3-14 – CONTROLLER FUNCTION MESSAGE OUTPUTS

STATE	REMOTE MESSAGE SENT			OTHER ACTIONS
	ATN	EOI	DIO(8-1)	
CIDS	(F)	(F)	(NUL)	
CADS	(F)	(F)	(NUL)	
CACS	T	F	DATA OUT REG	Data Out Reg. may contain any of the commands in Table 3-15
CSBS	F	(F)	(NUL)	
CWAS	F	(F)	(NUL)	
CSHS	F	(F)	(NUL)	
CSWS	T	F	(NUL)	
CAWS	T	F	(NUL)	
CPWS	T	T	(NUL)	DIO(8-1) may be read via the Command Pass Through Register

STATE	REMOTE MESSAGES SENT	OTHER ACTIONS
	IFC	
SIIS*	(F)	
SIIS	F	Internal interface clear message IF-CIN is held false
SIAS	T	

STATE	REMOTE MESSAGES SENT	OTHER ACTIONS
	REN	
SRIS*	(F)	
SRIS	F	
SRAS	T	

* Buffers not configured for a system controller; otherwise, buffers are configured for system controller.

When the controller is active, it uses 'ton' or 'lon' to address and unaddress itself. IEEE-488 provides for the controller to locally address itself to listen, but there is no corresponding local message for the talker. The TMS9914A should always accompany a 'ton' auxiliary command with 'cs' true with its own talk address or an UNT command sent over the bus. Similarly, if the TMS9914A sends the talk address of another device over the bus, it should ensure that it is in TIDS by writing the 'ton' auxiliary command false.

Appendix B shows some typical sequences of events when the controller addresses itself, goes to standby, takes control again, etc.

3.8.2 Passing Control

As Figure 3-9 shows, the controller transfer state (CTRS) of IEEE-488 is not present, and all transitions associated with the TCT command have been removed. Instead, two immediate execute auxiliary commands are included. Request control (rqc) will cause a transition from CIDS to CADS, and the release control command (rlc) will return the function to CIDS. The TCT command is treated similarly to an unrecognized addressed command but will cause a UNC interrupt if the device is in TADS.

Figure 3-10 is a representation of the sequence of events involved in passing control from one TMS9914A based device to another. The device passing control must initially ensure that it is not in TADS; then it should send out the talk address of the device to receive control. The receiving device will enter TADS, and after any DAC holdoff has been released, the host MPU of the device passing control will set a BO interrupt indicating that it may then send the TCT command. The TCT command will cause a UNC interrupt to the host MPU of the receiving device, and also a DAC holdoff will occur. The host MPU of the receiving device must examine its Command Pass Through Register, and upon identifying TCT, should write the auxiliary command 'rqc' to put its TMS9914A into CADS. The receiving device may then release DAC with a 'dacr' auxiliary command causing another BO interrupt at the device passing control. This indicates that the 'rlc' auxiliary command may then be used by the host MPU of the device passing control to return its TMS9914A to CIDS and allowing ATN to go false. The receiving device then enters CACS, asserts ATN, and its host MPU gets a BO interrupt as the source handshake becomes active. The passing of control is complete.

3.8.3 System Controller

The TMS9914A has no on-chip means of determining whether or not it is the system controller. Instead, this is determined by the software and by the configuration of the buffers to the IEEE-488 bus.

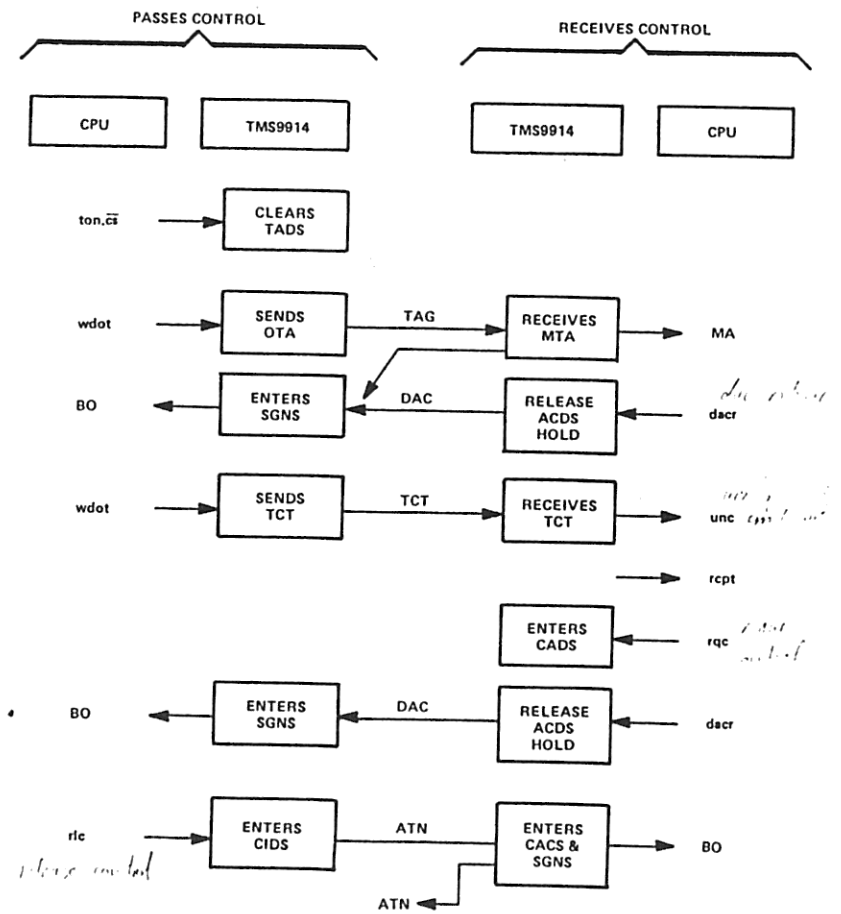
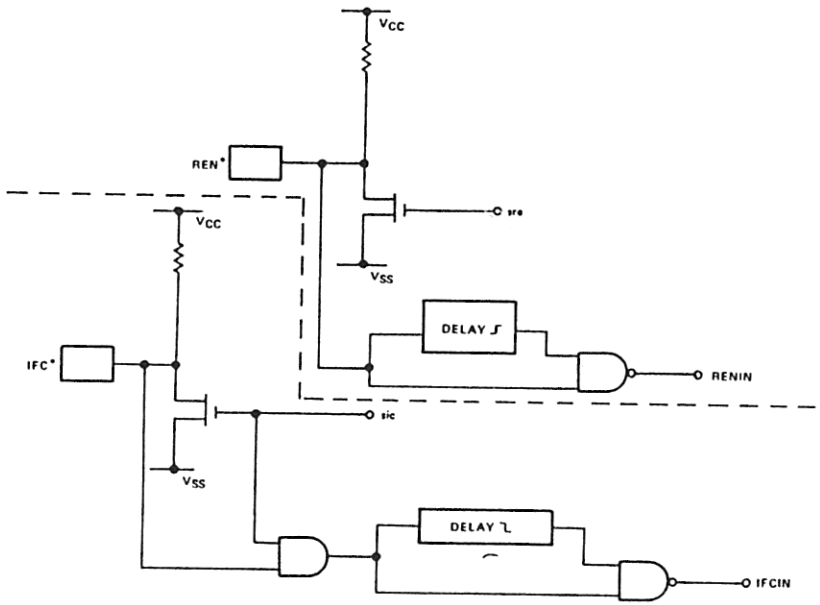


FIGURE 3-10 – PASSING CONTROL BETWEEN TMS9914s

The REN and IFC outputs of the TMS9914A are controlled by the auxiliary commands 'sre' and 'sic'. These should never be used by the host MPU of a device unless it is the system controller. As may be seen from Figure 3-11, the REN and IFC outputs of the TMS9914A are open drains with internal pull-ups. This means that the outputs are capable of driving the inputs of the buffers if the device is a system controller. If not, the buffers will drive into the REN and IFC pins and override the pull-ups. Hence, no direction control is required.

The false transition of REN and the true transition of IFC are both debounced to prevent noise on these lines from causing permanent state changes on the TMS9914A. In addition, the internal interface clear signal (IFCIN) is held false if the TMS9914A is sending IFC. Figure 3-9 shows the reason for this. If the device is not a system controller, then the occurence of IFC will return the controller function to CIDS. If, however, the device is a system controller, when it asserts IFC and is in CIDS, the 'sic' auxiliary command will cause it to enter CADS. As IFCIN is suppressed, it will not be forced back into CIDS, and there will be no conflict.



* The REN and IFC signals are at the pins of the TMS9914A and are therefore negative logic signals. The remaining signals are conventional positive logic signals.

FIGURE 3-11 – IFC AND REN PINS

TABLE 3-15 – MULTILINE INTERFACE MESSAGES

COMMAND	SYMBOL	DIO 8 – 1	CLASS	INTERRUPT (1,2)	DAC (3) HOLDOFF	NOTE
ADDRESSED COMMAND GROUP	ACG	000XXXX	AC	–	–	
DEVICE CLEAR	DCL	X0010100	UC	DCAS	YES	
GROUP EXECUTE TRIGGER	GET	X0001000	AC	GET	YES	
GO TO LOCAL	GTL	X0000001	AC	RLC	NO	14
LISTEN ADDRESS GROUP	LAG	X01XXXXX	AD	–	–	
LOCAL LOCKOUT	LLO	X0010001	UC	NONE	NO	
MY LISTEN ADDRESS	MLA	X01AAAAA	AD	MA,MAC,RLC	MA ONLY	4,14
MY TALK ADDRESS	MTA	X10AAAAA	AD	MA,MAC	MA ONLY	4
MY SECONDARY ADDRESS	MSA	X11SSSSS	SE	APT	YES	5,6
OTHER SECONDARY ADDRESS	OSA	SCG.MSA	SE	APT	YES	6,7
OTHER TALK ADDRESS	OTA	TAG.MTA	AD	MAC	NO	
PRIMARY COMMAND GROUP	PCG	ACG + UCG + LAG + TAG	–	–		
PARALLEL POLL CONFIGURE	PPC	X0000101	AC	UNC	YES	8
PARALLEL POLL ENABLE	PPE	X110SPPP	SE	UNC	YES	9,10
PARALLEL POLL DISABLE	PPD	X111DDDD	SE	UNC	YES	9,11
PARALLEL POLL UNCONFIGURE	PPU	X0010101	UC	UNC	YES	12
SECONDARY COMMAND GROUP	SCG	X11XXXXX	SE	–	–	
SELECTED DEVICE CLEAR	SDC	X0000100	AC	DCAS	YES	
SERIAL POLL DISABLE	SPD	X0011001	UC	NONE	NO	
SERIAL POLL ENABLE	SPE	X0011000	UC	NONE	NO	
TAKE CONTROL	TCT	X0001001	AC	UNC	YES	13
TALK ADDRESS GROUP	TAG	X10XXXXX	AD	–	–	
UNLISTEN	UNL	X0111111	AD	MAC	NO	
UNTALK	UNT	X1011111	AD	–	–	
UNIVERSAL COMMAND GROUP	UCG	X001XXXX	UC	NONE	NO	

Classes:
- UC – universal command
- AC – addressed command
- AD – address
- SE – secondary command

Symbols:
- 0 – logical zero (high level on GPIB)
- 1 – logical one (low level on GPIB)
- × – don't care (received message)

NOTES:
1. Interrupts listed are as a direct consequence of the command received. They are set during ACDS1 (see Section 3.2) and will cause the INT pin to be pulled low if unmasked.
2. The addressed commands will only cause their corresponding interrupt if the device is in LADS with the exception of TCT.
3. A DAC holdoff will only be caused if the corresponding interrupt is unmasked.
4. AAAAA represents the primary address of a device.
5. SSSSS represents the secondary address of a device.
6. Secondary addresses are handled via address pass through (APT interrupt). The host MPU should respond by writing the 'dacr' auxiliary command with 'cs' false.
7. If OSA is passed through via the APT interrupt, the host MPU should respond by writing the 'dacr' auxiliary command with 'cs' false.
8. PPC is not recognized by the TMS9914A and is therefore treated as an unrecognized addressed command.
9. PPE and PPD are secondary commands. These may be passed through to the host MPU using the 'pts' auxiliary command. When the PPC command is received the 'pts' auxiliary command should be written. PPE or PPD will then cause an APT interrupt.
10. SPPP specifies the sense bit, and the desired parallel poll response is a remotely configured parallel poll (see Section 3.7.1).
11. DDDD specifies don't care bits which must be sent as zeros but need not be decoded by the host MPU of the receiving devices.
12. PPU is not recognized by the TMS9914A and will cause a UNC interrupt.
13. TCT is not recognized directly by the TMS9914A. It will cause a UNC interrupt when the device is in TADS.
14. RLC is set if MLA or GTL causes an appropriate transition in the Remote/Local function.

4. TMS9914A ELECTRICAL SPECIFICATIONS

4.1 ABSOLUTE MAXIMUM RATINGS OVER OPERATING FREE-AIR TEMPERATURE RANGE (UNLESS OTHERWISE NOTED)*

Supply voltage, V_{CC} (see Note 1) . . . . . −0.3 V to 20 V
All input and output voltages . . . . . −0.3 V to 20 V
Continuous power dissipation . . . . . 0.8 W
Operating free-air temperature range . . . . . 0 °C to 70 °C
Storage temperature range . . . . . −55 °C to 150 °C

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

NOTE 1: Under absolute maximum ratings voltage values are with respect to V_{SS}.

4.2 RECOMMENDED OPERATING CONDITIONS

	MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}	4.75	5	5.25	V
Supply voltage, V_{SS}		0		V
High-level input voltage, V_{IH}	2		V_{CC} + 1	V
Low-level input voltage, V_{IL}	V_{SS} − 0.3		0.8	V
Operating free-air temperature, T_A	0		70	°C

4.3 ELECTRICAL CHARACTERISTICS OVER FULL RANGE OF RECOMMENDED OPERATING CONDITIONS

PARAMETER			TEST CONDITIONS	MIN	TYP†	MAX	UNIT
V_{OH}	High-level output voltage	Except REN,IFC,INT	I_{OH} = −400 μA	2.4		V_{CC}	V
		REN,IFC only	I_{OH} = −100 μA	2.2		V_{CC}	
V_{OL}	Low-level output voltage		I_{OL} = 2 mA	V_{SS}		0.4	V
I_I	Input current (any input)		V_I = 2 V to V_{CC}			±10	μA
I_{CC}	V_{CC} supply current					150	mA
C_I	Input capacitance (any input)		f = 1 MHz, unmeasured pins at 0 V			15	pF

† All typical values are at T_A = 25 °C and nominal voltage.

4.4 TIMING CHARACTERISTICS AND REQUIREMENTS

Timing characteristics and requirements are given in Section 4.4.1 through Section 4.4.6; relevant timing diagrams are shown in Figure 4-1 through Figure 4-9.

4.4.1 Clock and Host Interface Timing Requirements Over Full Range of Operating Conditions

PARAMETER		MIN	TYP	MAX	UNIT
$t_{c(\phi)}$	Clock cycle time	200		2000	ns
$t_{w(\phi H)}$	Clock high pulse width	100		1955	ns
$t_{w(\phi L)}$	Clock low pulse width	45			ns
$t_{su(AD)}$	Address setup time	0			ns
$t_{su(DBIN)}$	DBIN setup time	0			ns
$t_{su(CE)}$	$\overline{CE}$ setup time	100			ns
$t_{su(WE)}$	$\overline{WE}$ setup time	0			ns
$t_{w(WE)}$	$\overline{WE}$ low pulse width	80			ns
$t_{su(DA)}$	Data setup time	60			ns
$t_{h(DA)}$	Data hold time	0			ns
$t_{h(AD)}$	Address hold time	0			ns
$t_{h(DBIN)}$	DBIN hold time	0			ns
$t_{h(CE)}$	$\overline{CE}$ hold time	80			ns
$t_{su(GR)}$	$\overline{ACCGR}$ setup time	100			ns
$t_{h(GR)}$	$\overline{ACCGR}$ hold time	80			ns

4.4.2 Host Interface Timing Characteristics Over Full Range of Operating Conditions

PARAMETER		MIN	TYP	MAX	UNIT
$t_{a(CE)}$	Access time from $\overline{CE}$			150	ns
$t_{a(DBIN)}$	Access time from DBIN			150	ns
$t_{su(AD)}$	Address setup time to $\overline{CE}$	0			ns
$t_{z(DBIN)}$	Hi-Z time from DBIN		50	100	ns
$t_{z(CE)}$	Hi-Z time from $\overline{CE}$		50	100	ns
$t_{a(GR)}$	Access time from $\overline{ACCGR}$			150	ns
$t_{z(GR)}$	Hi-Z time from $\overline{ACCGR}$		50	100	ns
$t_{d(GR/RQ)}$	Delay of $\overline{ACCRQ}$ high from $\overline{ACCGR}$			100	ns

4.4.3 Source Handshake Timing Characteristics Over Full Range of Operating Conditions (see Note 1)

PARAMETER		TEST CONDITIONS	MIN	MAX	UNIT
t_{d1}	Delay of DAV true from end of write operation to data out register	Normal T_1 (see Note 2)	12(ϕ)†	12(ϕ)† + 310	ns
		Short T_1 (see Note 2)	8(ϕ)†	8(ϕ)† + 310	ns
		Very short T_1 (see Note 2)	4(ϕ)†	4(ϕ)† + 310	ns
t_{d2}	Delay of valid GPIB data lines from end of write cycle			140	ns
t_{d3}	Delay of BO interrupt from DAC true	BO interrupt unmasked		300	ns
t_{d4}	Delay of $\overline{ACCRQ}$ DAC true			300	ns
t_{d5}	Delay of DAV false from DAC true			160	ns

NOTES: 1. The timing of the source handshake is the same whether ATN is true or false, i.e., whether the device is in TACS, CACS, or SPAS.
2. A very short bus settling time (T_1) occurs on the second and subsequent data byte sent when ATN is false if the 'std1' feature is set. A slightly longer bus settling time takes place if 'std1' is set unless there is a very short bus settling time. In all other instances, a normal bus settling time occurs.

4.4.4 Acceptor Handshake Timing Characteristics Over Full Range of Operating Conditions

PARAMETER		TEST CONDITIONS	MIN	MAX	UNIT
t_{d6}	Delay of BI interrupt from DAV true	BI interrupt unmarked ATN = false device is in LACS	2(ϕ)†	2(ϕ)† + 415	ns
t_{d7}	Delay of $\overline{ACCRQ}$ from DAV true	ATN = false device is in LACS	2(ϕ)†	2(ϕ)† + 290	ns
t_{d8}	Delay of NDAC false from DAV true	ATN = false device is in LACS	3(ϕ)†	3(ϕ)† + 445	ns
t_{d9}	Delay of NRFD false from end of read operation of Data In register	ATN = false device is in LACS		220	ns
t_{d10}	Delay of interface message interrupt from DAV true (see Note 3)	ATN = true device not in CACS all interface message interrupts (except UNO)	2(ϕ)†	2(ϕ)† + 415	ns
		UNO interrupt only	5(ϕ)†	5(ϕ)† + 415	ns
t_{d11}	Delay of NDAC false from DAV true	ATN = true device not in CACS no DAC holdoff	7(ϕ)†	7(ϕ)† + 415	ns
t_{d12}	Delay of NDAC false from end of write operation			230	ns
t_{d13}	Delay of NRFD false from DAV false	ATN = true device not in CACS		180	ns

NOTE 3: The interrupts generated by interface messages are shown in Table 4-1.

4.4.5 ATN, EOI, and IFC Timing Characteristics Over Full Range of Operating Conditions

PARAMETER		TEST CONDITIONS	MIN	MAX	UNIT
t_{d14}	Delay of NDAC true from ATN true	Device is not in CACS		195	ns
t_{d15}	Delay of TE high from EOI true	Device is not in CACS		125	ns
t_{d16}	Delay of valid data from EOI true	Device is not in CACS		140	ns
t_{d17}	Delay of TE low from EOI false	Device is not in CACS		125	ns
t_{d18}	Delay of NRFD true from ATN false	Device is in LADS/LACS		140	ns
t_{d19}	Response time to IFC		$16t_{c(0)}$	$30t_{c(0)}$	ns

4.4.6 Controller Timing Characteristics Over Full Range of Operating Conditions

PARAMETER		TEST CONDITIONS	MIN	MAX	UNIT
t_{d20}	Delay of ATN true from end of t_{cs} aux command		$8t_{c(0)}$	$10(\phi)1+220$	ns
t_{d21}	Delay of BO interrupt from end of t_{cs} aux command		$18t_{c(0)}$	$22(\phi)1+415$	ns
t_{d22}	Delay of ATN true from end of t_{cs} aux command	BO unmasked device is in ANRS	$8t_{c(0)}$	$10(\phi)1+220$	ns
t_{d23}	Delay of BO interrupt from end of t_{cs} aux command	BO unmasked device is in ANRS	$18t_{c(0)}$	$22(\phi)1+415$	ns
t_{d24}	Delay of EOI true from r_{pp} aux command set			230	ns
t_{d25}	Delay of EOI false from r_{pp} aux command cleared			230	ns
t_{d26}	Delay of EOI from r_{pp} aux command cleared	BO unmasked	$8t_{c(0)}$	$10(\phi)1+415$	ns
t_{d27}	Delay of ATN false from sts aux command	Device is not in SDYS or STRS		210	ns

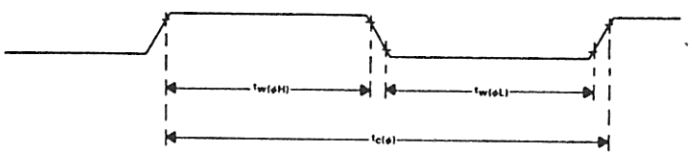


FIGURE 4-1 – TMS9914A CLOCK CYCLE TIMING

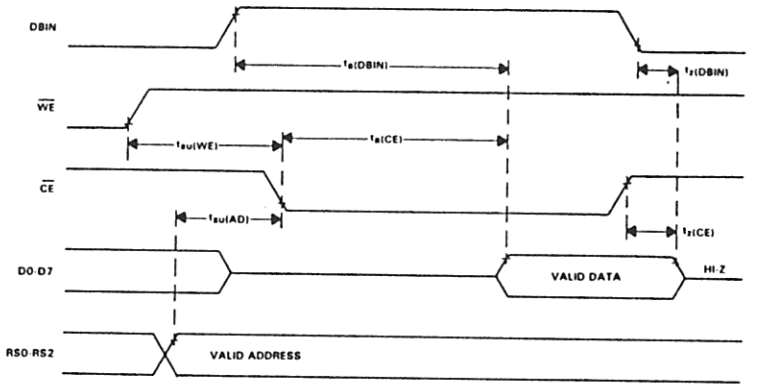
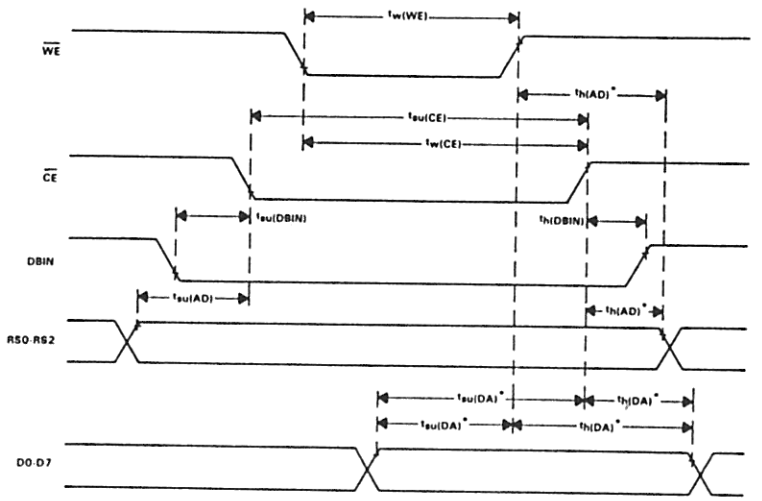
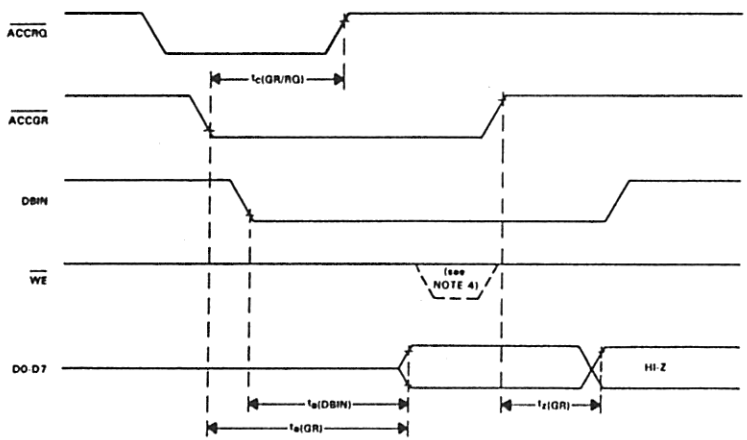


FIGURE 4-2 – TMS9914A READ CYCLE TIMING



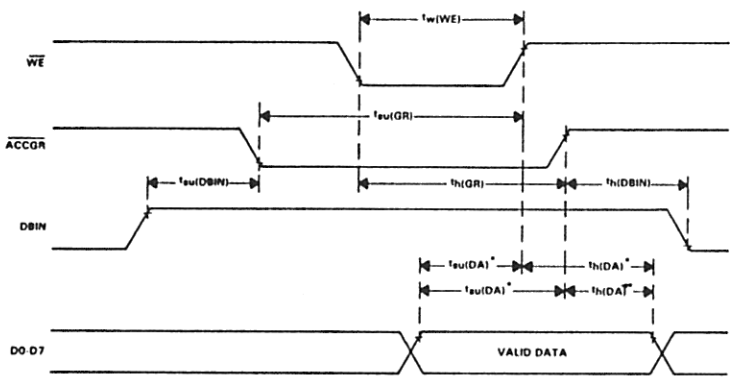
$t_{su(DA)}$, $t_{h(DA)}$, and $t_{h(AD)}$ are only applicable to the first signal to become inactive, whether it is $\overline{WE}$ or $\overline{CE}$.

FIGURE 4-3 – TMS9914A WRITE CYCLE TIMING



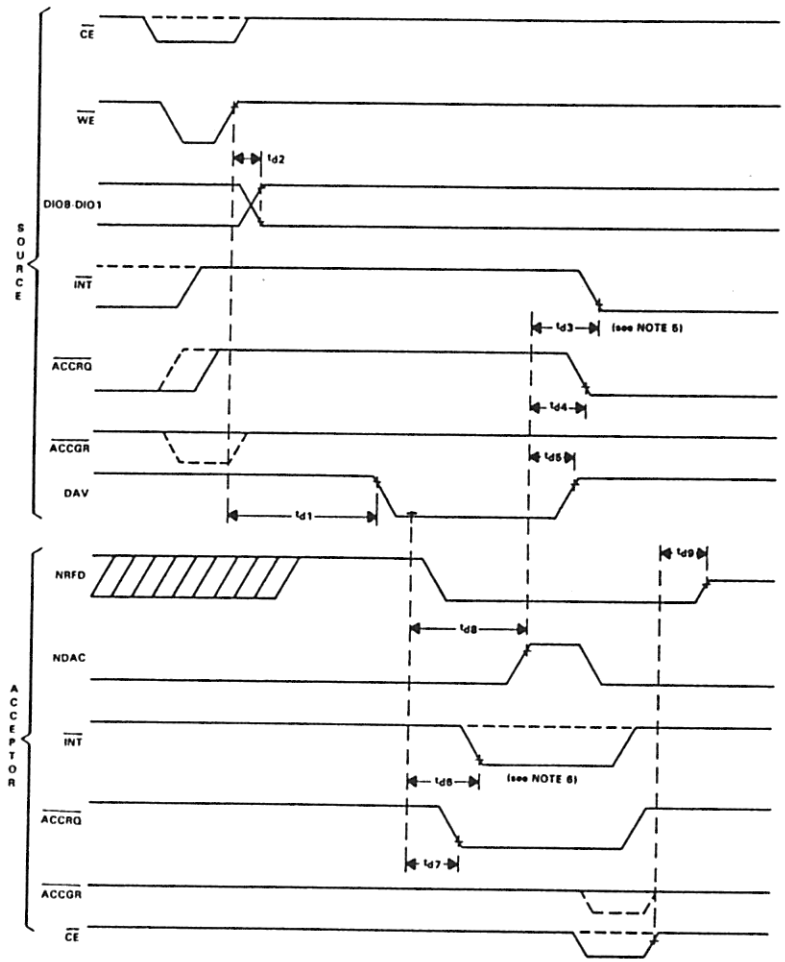
NOTE 4: A write enable pulse may occur in a DMA read operation. A write enable pulse may therefore be provided for system memory and need not be suppressed at the TMS9914A.

FIGURE 4-4 – TMS9914A DMA READ OPERATION



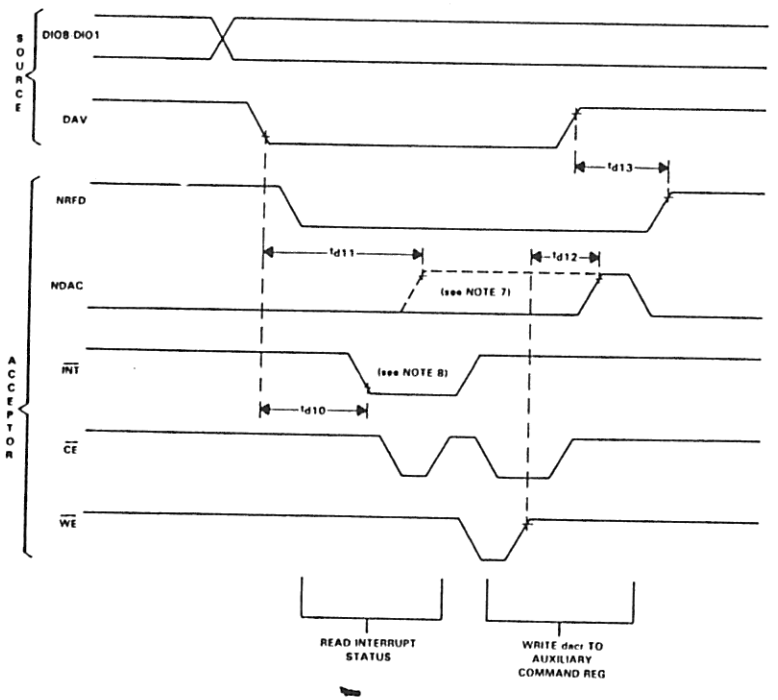
* $t_{su(DA)}$ and $t_{h(DA)}$ are only applicable to the first signal to become inactive, whether it is $\overline{WE}$ or $\overline{ACCGR}$.

FIGURE 4-5 – TMS9914A DMA WRITE OPERATION



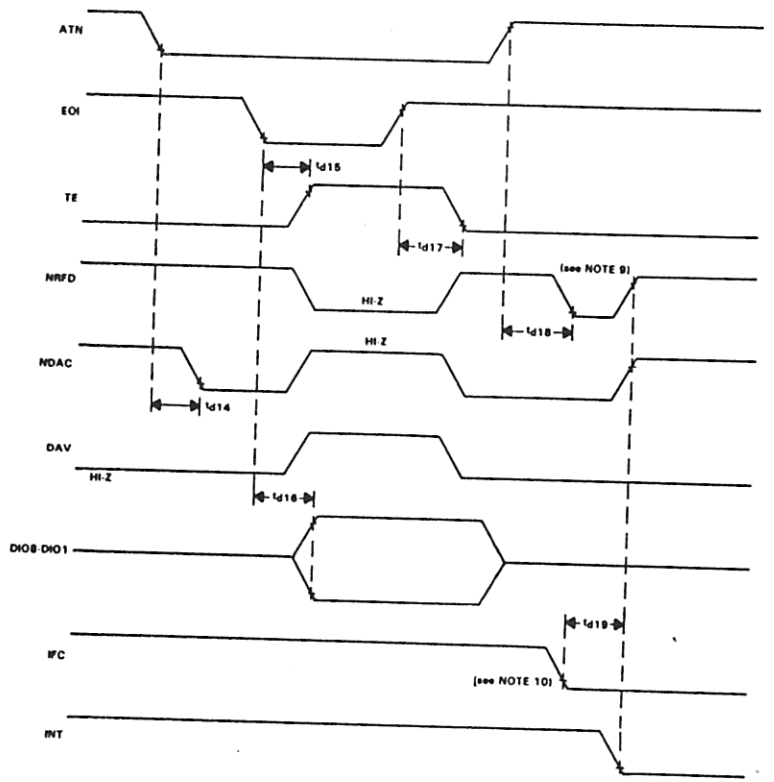
NOTES: 5. The interrupt line is taken low by a BO interrupt.
6. The interrupt line is taken low by a BI interrupt.

FIGURE 4-6 – TMS9914A SOURCE AND ACCEPTOR HANDSHAKE TIMING(S)



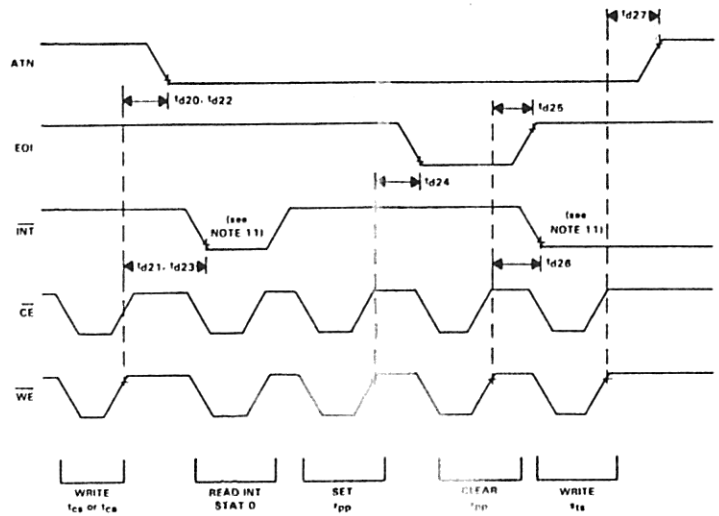
NOTES: 7. The broken line shows the waveform if there is no DAC holdoff. The solid lines assume there is a DAC holdoff.
8. The interrupts generated by interface messages are shown in Table 4-1.

FIGURE 4-7 – TMS9914A ACCEPTOR HANDSHAKE TIMING "ATN" TRUE



NOTES: 9. This assumes that an RFD holdoff occurs.
10. IFC causes the TMS9918A to be unaddressed and an IFC interrupt occurs.

FIGURE 4-8 – TMS9914A RESPONSE TO 'ATN' AND 'EOI'



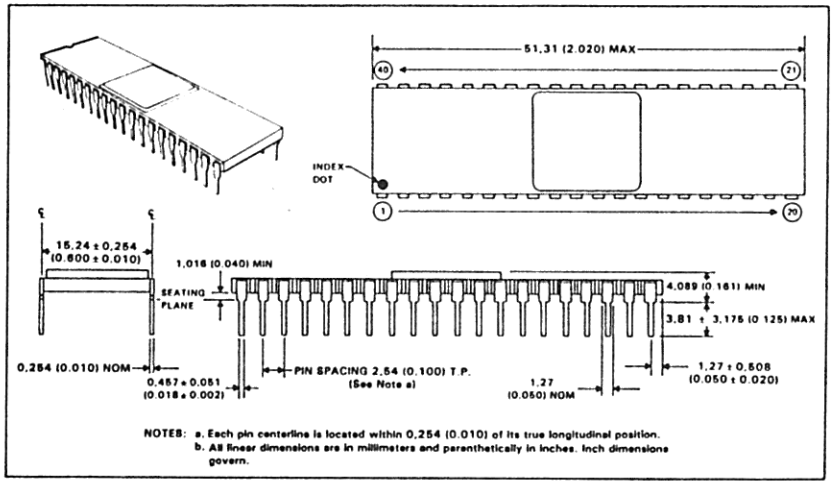
NOTE 11: A BD interrupt occurs as the TMS9914A enters CACS.

FIGURE 4-9 – TMS9914A CONTROLLER TIMING

5. MECHANICAL SPECIFICATIONS

5.1 TMS9914AJL – 40-PIN CERAMIC PACKAGE

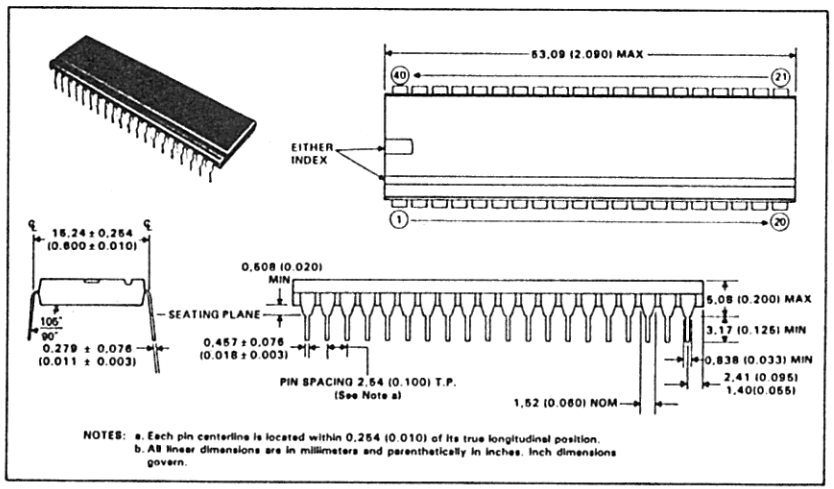
Ceramic packages with side-brazed leads and metal, epoxy, or glass lid seal



NOTES: a. Each pin centerline is located within 0,254 (0.010) of its true longitudinal position.
b. All linear dimensions are in millimeters and parenthetically in inches. Inch dimensions govern.

5.2 TMS9914ANL – 40-PIN PLASTIC PACKAGE

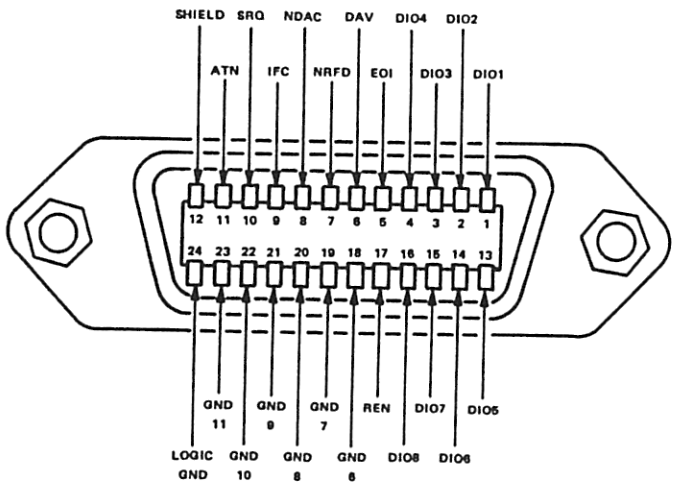
Plastic packages



NOTES: a. Each pin centerline is located within 0,254 (0.010) of its true longitudinal position.
b. All linear dimensions are in millimeters and parenthetically in inches. Inch dimensions govern.

APPENDIX A

IEEE-488 STANDARD CONNECTOR



APPENDIX B

TYPICAL SEQUENCES OF EVENTS FOR THE CONTROLLER

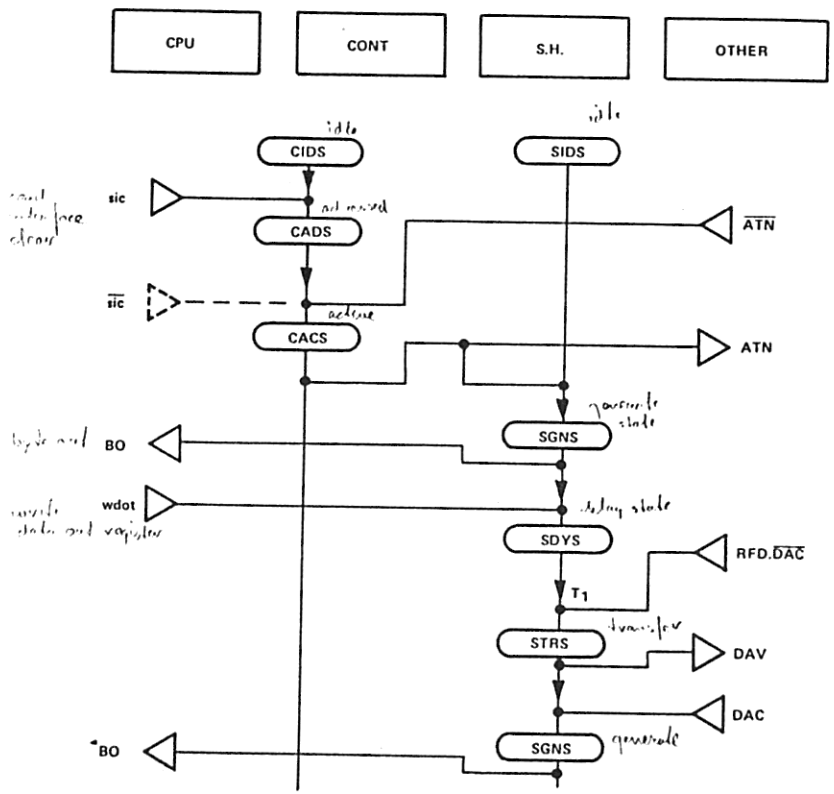
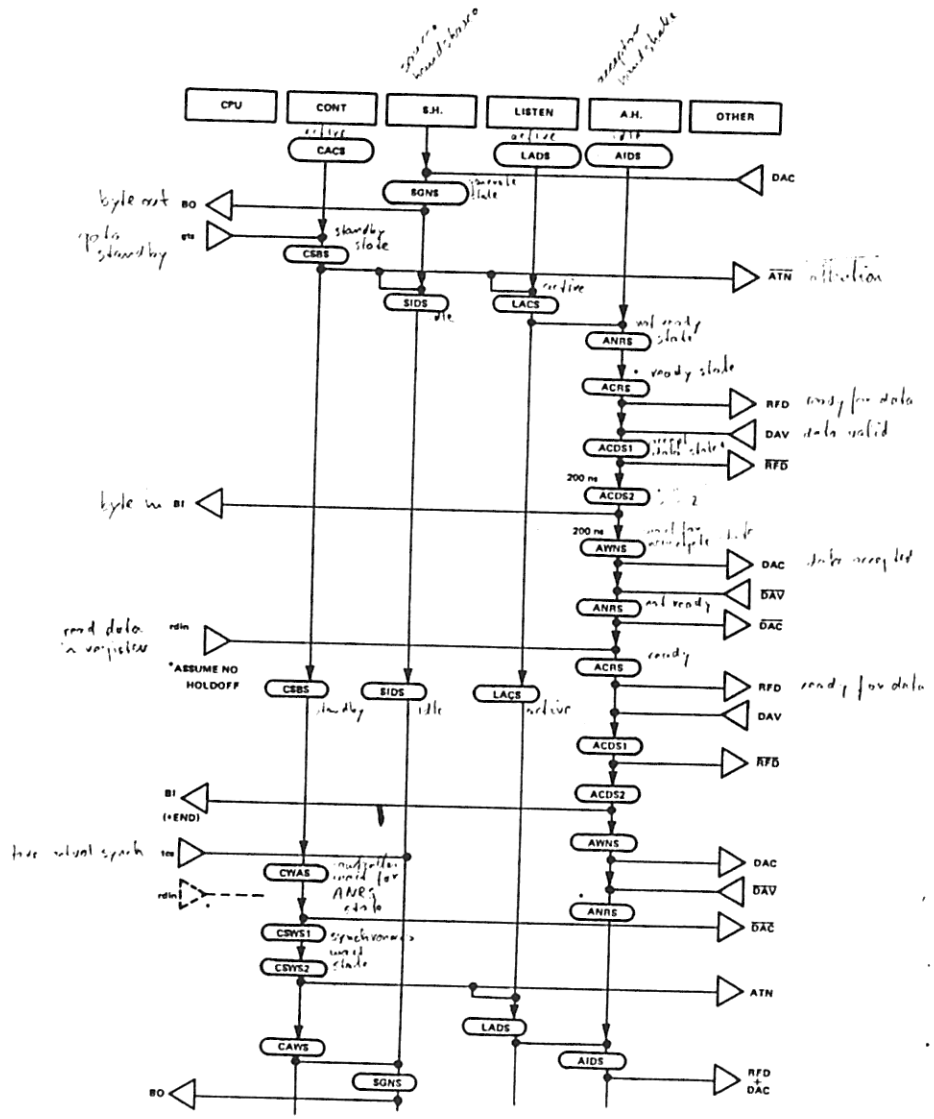
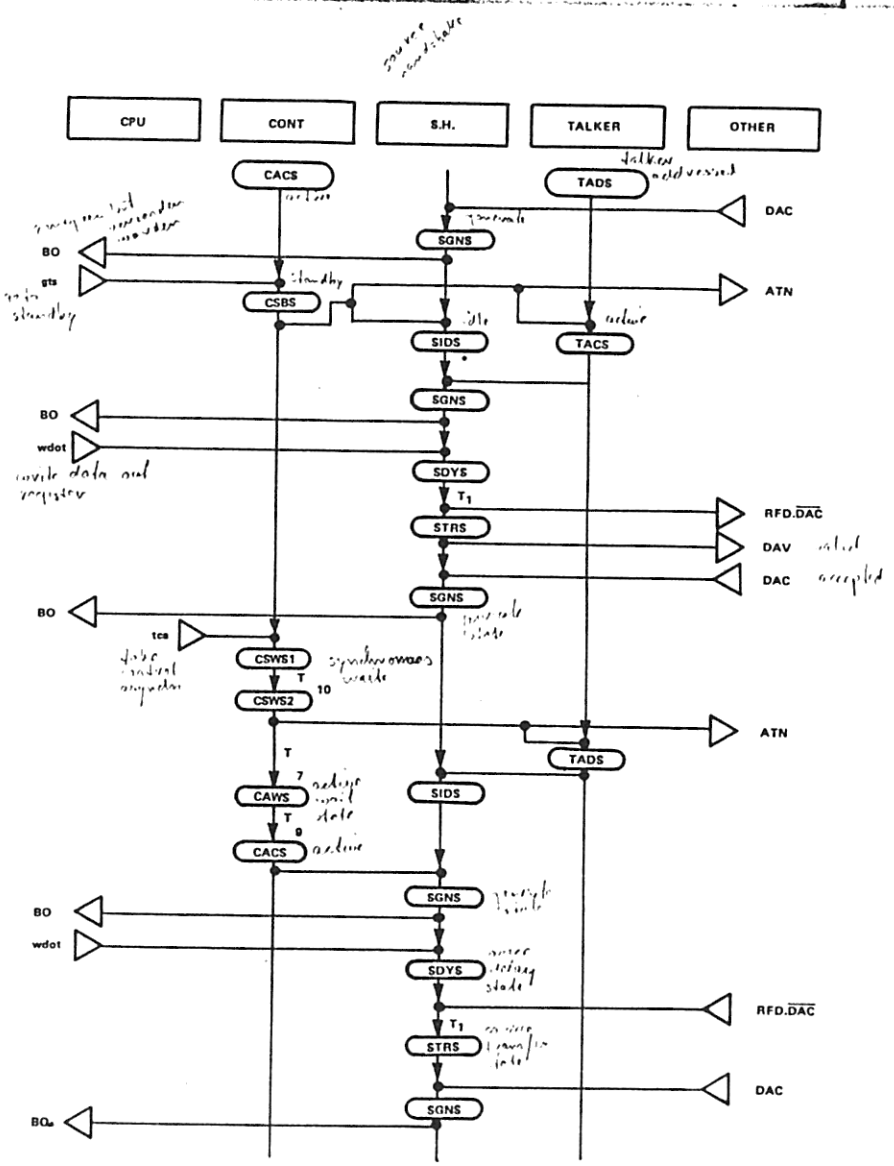


FIGURE B-1 – CONTROLLER TAKING CONTROL



*CWAS inhibits ANRS → ACRS, so rdin can occur before ATN is set.

FIGURE B-2 – CONTROLLER AS A LISTENER (GOING TO STANDBY)



*Momentary transition following BO interrupt may not occur on the TMS9914 but is guaranteed on the TMS9914A.

FIGURE B-3 – CONTROLLER AS A TALKER (GOING TO STANDBY)

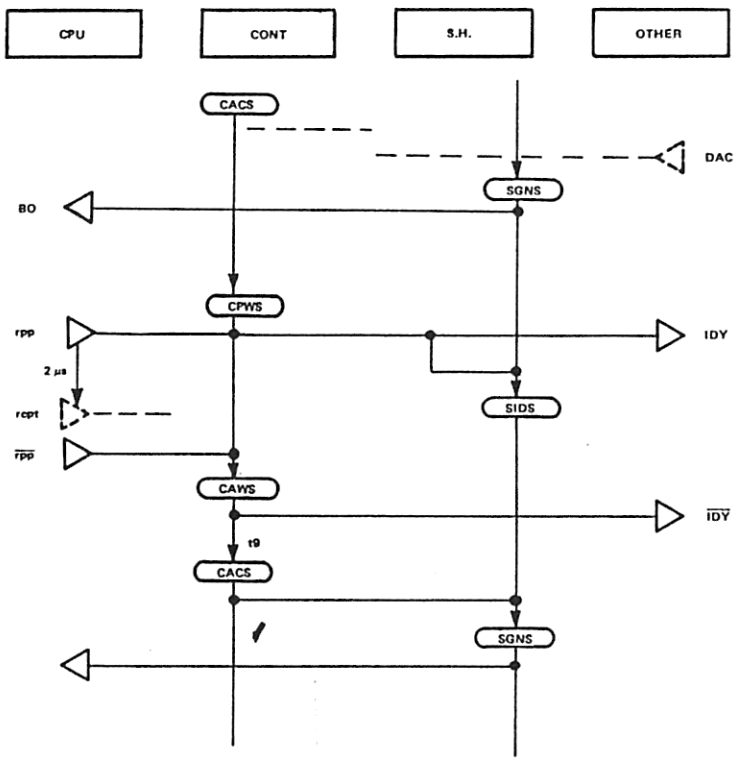


FIGURE B-4 – CONTROLLER PARALLEL POLLING

APPENDIX C

SN75160/161/162 DATA SHEETS

Texas Instruments SN75160 family of bus transceivers are designated to provide the interface between the bus and the bus controller. These transceivers may be used with the TI TMS9914 Bus Controller chip or any of the other GPIB controllers commercially available. They provide the simplest method of interfacing to the bus, because each part is tailored to either the 8-line data bus or 8-line control bus, so they require no extra logic or complicated board layout. With the SN75160 family, it takes only two 20-pin DIP packages to get on the GPIB. The new improved SN75160A series is pin-for-pin compatible with the original SN75160 series but with lower power and faster speeds, as shown in Figure C-1.

All transceivers in the SN75160 family have several features in common. Each driver output has built into it the termination network required by IEEE Standard 488. This termination is designed so that when power is removed from the transceiver, the output presents a high impedance to the bus. Also, each receiver has a minimum of 400 mV hysteresis for additional noise margin.

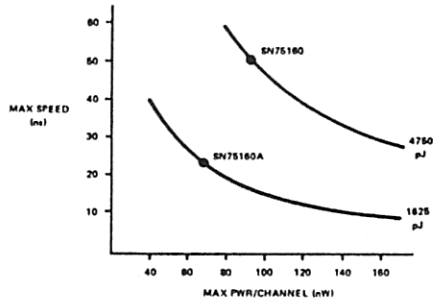


FIGURE C-1

The SN75160A is designed to implement the 8-line data bus. The direction of data flow is controlled by the Talk Enable (TE) input. All eight channels are simultaneously in the receive mode when the TE is low and data is received fron the bus and transferred to the bus controller. When the TE is in the high state, all eight channels go to the transmit mode, and data will be transmitted onto the bus. Each driver features a totem-pole output which can actively drive the bus high or low to give the fastest data rates possible. The SN75160A has a Pull-up Enable (PE) input which, when taken low, disables the upper stage of the driver outputs turning all eight driver outputs into open-collector type outputs. The open-collector output mode does not allow as fast a data rate as with the totem-pole, but it does allow more than one instrument to be transmitting on the bus at the same time. This feature is used in parallel polling where up to eight instruments may be polled simultaneously, each responding on one line of the eight-line data bus, greatly speeding the polling process. They may then be switched back to the totem-pole mode for regular data transmission.

The SN75161A is used to implement the 8-line control bus. Included in it is the necessary logic which, combined with the Talk-Enable (TE) and Direction Control (DC) inputs, insures that each channel is enabled in the correct direction for exchange of bus management and handshaking signals. Three of the channels, NDAC, NRFD, and SRQ, have open-collector driver outputs as required by the IEEE Standard 488. These lines are always used in a wired-OR configuration. The other five channels have totem-pole outputs. The SN75162A offers an alternate method of implementing the control bus. The SN75162A is identical to the SN75161A except that the direction of the REN and IFC channels is controlled by a separate input called the System Controller (SC). With this additional flexibility, control of the entire Bus System may be transferred from one instrument to another (multiple controller systems). Because of this extra input, the SN75162A package has 22 pins.

SN75160A, SN75161A AND SN75162A OCTAL IEEE-488 GPIB BUS TRANSCEIVERS

features

- 8-channel bidirectional transceivers
- Meet IEEE standard 488 1978
- Low power dissipation (65 mW max per channel)
- High-impedance PNP inputs
- Receiver hysteresis (500 mV typ)
- Open-collector driver output option (SN75160A)
- Bus-terminating resistors provided on driver outputs
- No loading of bus when device is powered down (V_{CC} = 0 V)
- SN75161A for single-controller systems; SN75162A for multi-controller systems

NOTE: Integrated Schottky-Barrier diode-clamped transistor is patented by Texas Instruments. U.S. Patent Number 3,463,975.

description

These octal bus transceivers are designed to provide communication on the general-purpose interface bus (GPIB) between operating units of the instrumentation system. The sixteen signal lines are normally required by the SN75162A in systems with more than one controller. An active turn-off feature has been incorporated into the bus-terminating resistors so that the devices exhibit a high impedance to the bus when V_{CC} = 0 V.

When PE is low, the bus outputs of the SN75160A have the characteristics of open-collector outputs. They act as three-state ports when PE is high. Taking TE low places those ports in the free-state, wherein they can be driven by the bus lines, and enables the D outputs.

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

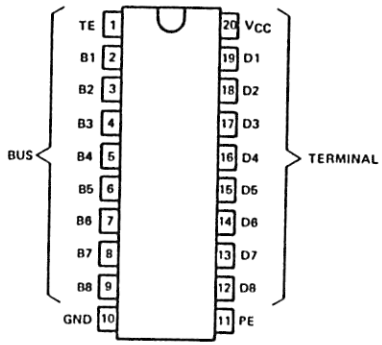
Supply voltage, V_{CC} (see Note 1)	7 V
Input voltage	5.5 V
Low-level driver output current	100 mA
Continuous total dissipation at (or below) 25 °C free-air temperature (see Note 2)	1150 mW
Operating free-air temperature range	0 °C to 70 °C
Storage temperature range	−65 °C to 150 °C
Lead temperature 1/16 inch (1.6 mm) from case for 10 seconds	260 °C

NOTES: 1. All voltage values are with respect to network ground terminal.
2. For operation above 25 °C free-air temperature, derate linearly at the rate of 9.3 mW/°C to 740 mW at 70 °C.

table of abbreviations

CLASS	NAME	IDENTITY
CONTROL INPUTS	DC	Direction Control
	PE	Pull-up Enable
	TE	Talk Enable
SN75161A I/O PORTS	B	Bus side of device
	D	Terminal side of device
SN75161A/162A SIGNAL MNEMONICS	ATN	Attention
	DAV	Data Valid
	EOI	End or Identify
	IFC	Interface Clear
	NDAC	Not Data Accepted
	NRFD	Not Ready for Data
	REN	Remote Enable
	SRQ	Service Request
	SC	System Controller

SN75160A N DUAL-IN-LINE PACKAGE (TOP VIEW)

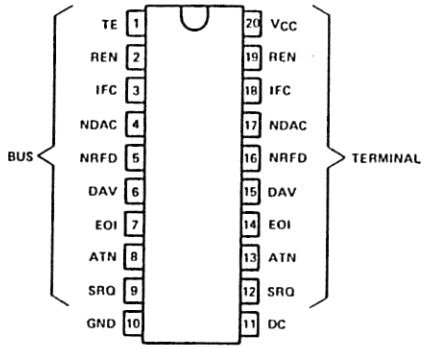


SN75160A function table

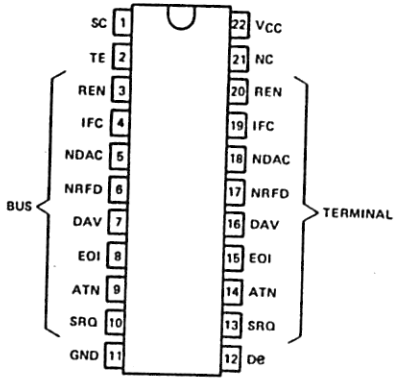
DRIVERS				RECEIVERS			
INPUTS			OUTPUTS	INPUTS			OUTPUTS
D	TE	PE	B	B	TE	PE	D
H	H	H	H	L	L	X	L
L	H	H	L	H	L	X	H
H	X	L	F	X	H	X	Z
L	H	L	L				
X	L	X	F				

F = free state*, H = high level, L = low level, X = irrelevant, Z = high-impedance state.
*This is the high-impedance state of a normal 3-state output modified by the internal resistors to V_{CC} and ground.
modified by the internal resistors to V_{CC} and ground.

SN75161A N DUAL-IN-LINE PACKAGE (TOP VIEW)



SN75162A N DUAL-IN-LINE PACKAGE (TOP VIEW)



SN75161A function table

CONTROLS†				DIRECTION OF DATA‡						
TE	DC	ATN LEVEL	DIRECTION	EOI	REN	IFC	SRQ	NRFD	NDAC	DAV
H	H	H	R	T	R	R	T	R	R	T
H	H	L	R	R	R	R	T	R	R	T
H	L	X	T	T	T	T	R	R	R	T
L	H	X	R	R	R	R	T	T	T	R
L	L	H	T	R	T	T	R	T	T	R
L	L	L	T	T	T	T	R	T	T	R

H = high level, L = low level, R = receive, T = transmit, X = irrelevant

† ATN is a normal transceiver channel that functions additionally as an internal direction control or talk enable for EOI whenever the TE and DC inputs are in the same state. When TE and DC are in opposite states, the ATN channel functions as an independent transceiver only.

‡ Direction of data transmission is from the terminal side to the bus side, and the direction of data receiving is from the bus side to the terminal side. Data transfer is noninverting in both directions.

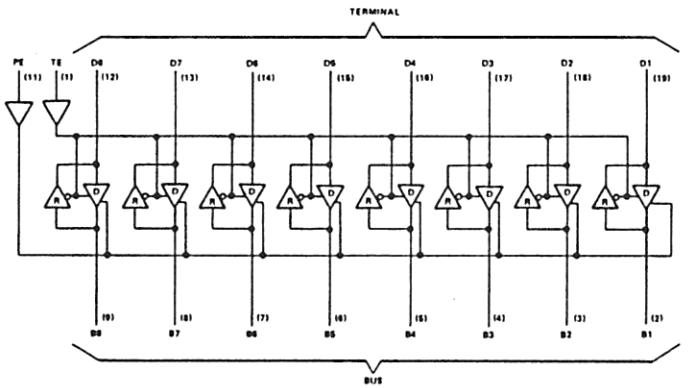
SN75162A function table

CONTROLS				DIRECTION OF DATA						
TE	DE	SC	ATN LEVEL	DIRECTION	EOI	REN	IFC	SRQ	NRFD	NFAC	DAV
H	H	L	H	R	T	R	R	T	R	R	T
H	H	L	L	R	R	R	R	T	R	R	T
H	L	L	X	T	T	R	R	R	R	R	T
L	H	L	X	R	R	R	R	T	T	T	R
L	L	L	H	T	R	R	R	R	T	T	R
L	L	L	L	T	T	R	R	R	T	T	R
H	H	H	H	R	T	T	T	T	R	R	T
H	H	H	L	R	T	T	T	T	R	R	T
H	L	H	X	T	T	T	T	R	R	R	T
L	H	H	X	R	R	T	T	R	T	T	R
L	L	H	H	T	R	T	T	R	T	T	R
L	L	H	L	T	T	T	T	R	T	T	R

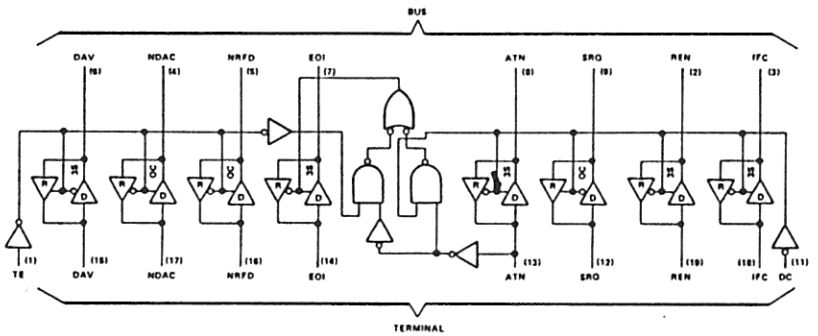
H = high, L = low, R = receive, T = transmit, X = irrelevant.

functional block diagrams

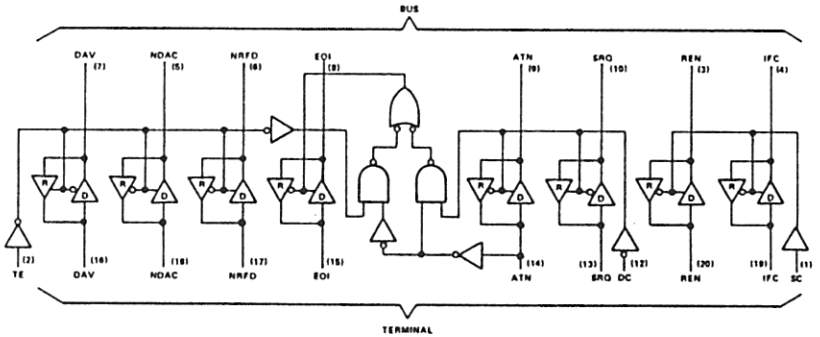
SN75160A



SN75161A



SN75162A



switching characteristics, V_{CC} = 5 V, C_L = 15 pF, T_A = 25°C (unless otherwise noted)

PARAMETER		FROM	TO	TEST CONDITIONS	SN75160A MIN	SN75160A TYP	SN75160A MAX	SN75161A MIN	SN75161A TYP	SN75161A MAX	SN75162A MIN	SN75162A TYP	SN75162A MAX	UNIT
t_{PLH}	Propagation delay time, low-to-high-level output	Terminal	Bus	C_L = 30 pF, R_L = 38.3 Ω to 2.3 V		14	20		17	25		17	25	ns
t_{PHL}	Propagation delay time, high-to-low-level output					14	20		17	25		17	25	
t_{PLH}	Propagation delay time, low-to-high-level output	Bus	Terminal	C_L = 30 pF, R_L = 240 Ω to 5 V		12	20		16	25		16	25	ns
t_{PHL}	Propagation delay time, high-to-low-level output					15	22		16	25		16	25	
t_{PZH}	Output enable time to high level	TE or DC	Bus	R_L = 480 Ω to 0 V		25								ns
t_{PHZ}	Output disable time from high level					12								
t_{PZL}	Output enable time to low level			R_L = 38.3 Ω to 2.3 V		22								
t_{PLZ}	Output disable time from low level					21								
t_{PZH}	Output enable time to high level	TE or DC	Terminal	R_L = 3kΩ to 0 V		20								ns
t_{PHZ}	Output disable time from high level					13								
t_{PZL}	Output enable time to low level			R_L = 280 Ω to 0 V		23								
t_{PLZ}	Output disable time from low level					19								
	Output pull-up enable time	PE	Terminal	R_L = 480 Ω to 0 V		15								ns
	Output pull-up disable time					13								

APPENDIX D

EXAMPLE SOFTWARE

DESIGN EXAMPLE

DESIGN OBJECTIVES

- Illustrate the procedures and protocol of the remote and local messages used to configure a controller.
- Illustrate the procedures used to configure a remote instrument to acquire data and then transmit the data over the GPIB bus to some other device.
- Demonstrate the use of the TMS9914 as a controller, talker, and listener.
- Show the software necessary to drive the TMS9914 when it is interfaced to TM990 products.
- Show the elements of hardware design which must be dealt with when interfacing the TMS9914 to an MPU.
- Show the hardware necessary to interface the TMS9914 to the GPIB bus.

STATEMENT OF THE EXAMPLE DESIGN PROBLEM

- TM990/101-based system
- Use the TMS9914 to communicate with the GPIB.
- Configure an HP3455A digital voltmeter to take two-wire resistance measurements.
- Trigger the meter to take a measurement and send the data over the GPIB to the TM990 system.
- Decode the ASCII data into speech and provide a vocal annunciation of the resistance measurement.

SYSTEM COMPONENTS CHOSEN

- Hewlett Packard model 3438A digital voltmeter
  - IEEE-488 compatible device
  - Range and function selection locally selected and indicated in measurement data
  - Remote measurement trigger capability
- TI TM990/101M microcomputer CPU module
  - TMS9900 16-bit CPU
  - 4K bytes of RAM mappable to either the bottom or top of the memory address space
  - Up to 8K bytes of EPROM at the bottom of memory space
  - TMS9901 programmable system interface providing up to 16 prioritized maskable interrupts and interval timer
  - Two serial I/O ports (RS232C compatible)
- TI TM990/306 speech module
  - TM990 series bus compatible
  - 180-word vocabulary
  - 2.5 W amplifier for direct speaker drive
  - Programmed through TMS9900 CRU interface
  - Serviced on interrupt or polled basis
  - Additional edge connector for interfacing to non-TM990 devices
- TI TMS9914-based interface board
  - Designed to illustrate the interfacing of the TMS9914 to a CPU and to the IEEE-488 bus
  - Interfaces directly to the components described above

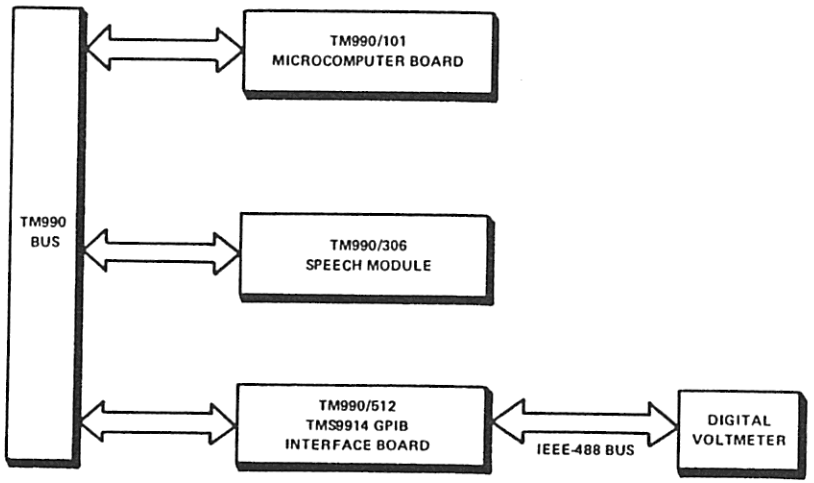


FIGURE D-1 – APPLICATIONS HARDWARE BLOCK DIAGRAM

PROGRAMMER'S MODEL OF THE TM990/306 SPEECH BOARD

- 14-bit address selects word to speak
  - Bits 16 to 29 on the CRU interface
- 1 bit to enable EPROM speech data
  - Set to 1 during initialization
- 1 bit to start/stop speaking
  - 0 starts/enables speech
  - 1 stops/disables speech
- 1 bit to indicate the busy status of speech board
  - 0 means speech board busy
  - 1 means speech board not busy

METER DATA OUTPUT CHARACTERISTICS

- Data is output as a string of ASCII characters.
  - Format of the string is +D.DDDE+D,R 0D)A
- R indicates the range setting of the meter.
  '1' = DC volts
  '2' = AC volts
  '3' = DC amps
  '4' = AC amps
  '5' = ohms
- EOI message is sent with LINE FEED character.

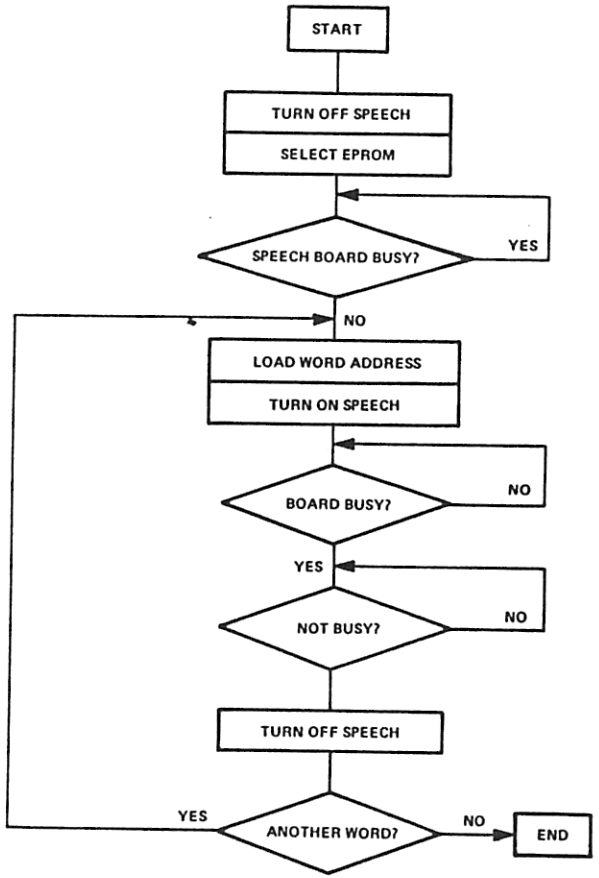


FIGURE D-2 – TYPICAL SPEECH BOARD USE

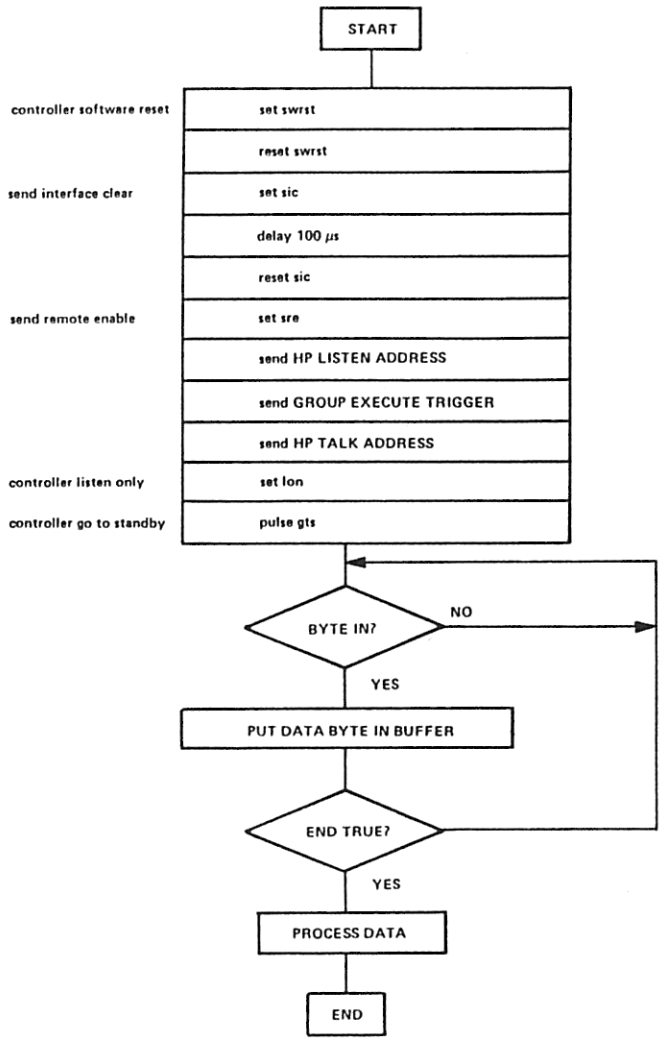


FIGURE D-3 – TYPICAL SOFTWARE TO CONTROL DIGITAL VOLTMETER

SOFTWARE LISTING

```
        IDT     'DEM3438A'
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                               *
*       GPIB DEMONSTRATION      THIS VERSION FOR HP 3438A       *
*                                                               *
*       "TALKING CONTROLLER"    REVISION 4/14-NOV-80/PNK        *
*                                                               *
*             TMS9914 BASE ADDRESS:     >5540                   *
*             HP 3438A DEVICE ADDRESS:  >17                     *
*             TM990/306 CRU ADDRESS:    >1FE0                   *
*                                                               *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
*       TMS 9914 REGISTER EQUATES
*
BASE    EQU     >5540           BASE ADDRESS OF TMS 9914
IMASK0  EQU     BASE+0          INTERRUPT MASK 0
ISTAT0  EQU     BASE+0          INTERRUPT STATUS 0
IMASK1  EQU     BASE+2          INTERRUPT MASK 1
ISTAT1  EQU     BASE+2          INTERRUPT STATUS 1
ADSTAT  EQU     BASE+4          ADDRESS STATUS REGISTER
BUSTAT  EQU     BASE+6          BUS STATUS REGISTER
AUXCMD  EQU     BASE+6          AUXILARY COMMAND REGISTER
ADRSWI  EQU     BASE+8          ADDRESS SWITCH "REGISTER"
ADDRES  EQU     BASE+8          ADDRESS REGISTER
SERPOL  EQU     BASE+10         SERIAL POLL REGISTER
CMDPAS  EQU     BASE+12         COMMAND PASS THROUGH REGISTER
PARPOL  EQU     BASE+12         PARALLEL POLL REGISTER
DATIN   EQU     BASE+14         DATA FROM BUS REGISTER
DATOUT  EQU     BASE+14         DATA TO BUS REGISTER
*
*       TMS 9914 INTERRUPT AND POLLING MASKS
*
INT0    EQU     >8000           INTERRUPT GROUP 0
INT1    EQU     >4000           INTERRUPT GROUP 1
BI      EQU     >2000           BYTE READY FOR INPUT
BO      EQU     >1000           BYTE READY FOR OUTPUT
ENDB    EQU     >0800           BYTE IS LAST ONE
SRQ     EQU     >0200           SERVICE REQUEST
MASK0   EQU     BI+BO+ENDB      ENABLE INTERRUPTS GIVEN
```

```
*
*       SELECTED TMS 9914 AUXILARY COMMANDS
*
SWRST   EQU     >8000           SET SOFTWARE RESET
SWRSTC  EQU     >0000           CLEAR SOFTWARE RESET
FEOI    EQU     >0800           FORCE END OR IDENTIFY
LON     EQU     >8900           SET LISTEN ONLY
LONC    EQU     >0900           CLEAR LISTEN ONLY
TON     EQU     >8A00           SET TALK ONLY
TONC    EQU     >0A00           CLEAR TALK ONLY
GTS     EQU     >0B00           GO TO STANDBY
TCA     EQU     >0C00           TAKE CONTROL ASYNCHRONOUSLY
SIC     EQU     >8F00           SET SEND INTERFACE CLEAR
SICC    EQU     >0F00           CLEAR SEND INTERFACE CLEAR
SRE     EQU     >9000           SET SEND REMOTE ENABLE
SREC    EQU     >1000           CLEAR SEND REMOTE ENABLE
*
*       SELECTED MULTILINE I/F MESSAGES
*
GET     EQU     >0800           GROUP EXECUTE TRIGGER (ADDRESSED)
LLO     EQU     >1100           LOCAL LOCKOUT (UNIVERSAL)
SDC     EQU     >0400           DEVICE CLEAR (ADDRESSED)
DCL     EQU     >1400           DEVICE CLEAR (UNIVERSAL)
SPE     EQU     >1800           SERIAL POLL ENABLE (UNIVERSAL)
SPD     EQU     >1900           SERIAL POLL DISABLE (UNIVERSAL)
UNL     EQU     >3F00           UNLISTEN COMMAND (UNIVERSAL)
UNT     EQU     >5F00           UNTALK COMMAND (UNIVERSAL)
*
*       HP3455A DVM ADDRESSES
*
HPLA    EQU     >3700           LISTEN ADDRESS
HPTA    EQU     >5700           TALK ADDRESS
*
        AORG    >0000           STARTING ADDRESS
*
*       INTERRUPT VECTOR(S)
*
RESET   DATA    MAINWS          INITIAL WP
        DATA    START           INITIAL PC
*
```



```
        AORG    >0100           PROGRAM ADDRESS
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                   *
*       GPIB SUBROUTINE DEFINITIONS                                 *
*                                                                   *
*            BYTOUT - SENDS BYTE IN R0 OVER GPIB                    *
*            BYTIN  - RECEIVES BYTE FROM GPIB INTO R0               *
*            STROUT - SENDS STRING POINTED TO BY R0 OVER GPIB       *
*                     SENDING EOI WITH LAST BYTE (INDICATED BY      *
*                     FOLLOWING BYTE WITH BYTE = 00)                *
*            STRIN  - RECEIVES STRING FROM GPIB INTO BUFFER         *
*                     POINTED TO BY R0                              *
*            DOAUX  - PERFORMS 9914 AUX CMD SPECIFIED IN BYTE       *
*                     AFTER SUBROUTINE CALL                         *
*            DELAY  - DELAYS NUMBER OF MS INDICATED BY THE          *
*                     CONTENTS OF R0                                *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
BYTOUT  MOV     @ISTAT0,R1      CHECK IF BO FLAG SET
        ANDI    R1,BO
        JEQ     BYTOUT          WAIT UNTIL IT IS
        MOV     R0,@DATOUT      SEND BYTE IN R0 OVER GPIB
        RT
*
BYTIN   MOV     @ISTAT0,R0      CHECK IF BI FLAG SET
        MOV     R0,R1           COPY TO R1
        ANDI    R0,BI
        JEQ     BYTIN           WAIT UNTIL IT IS
        MOV     @DATIN,R0       PUT BYTE FROM GPIB INTO R0
        RT
*
STROUT  MOV     R11,R10         SAVE RETURN ADDRESS
        MOV     R0,R2           SAVE POINTER
        MOVB    *R2+,R0         GET FIRST BYTE TO SEND
STROU1  MOVB    *R2+,R4         GET NEXT BYTE TO SEND
        JNE     STROU2          SKIP IF NEXT IS NOT STOP FLAG
        BL      @DOAUX          PERFORM AUXILARY COMMAND
        DATA    FEOI            FORCE END WITH THIS BYTE
STROU2  BL      @BYTOUT         ELSE SEND THE BYTE
        MOVB    R4,R0           MOVE NEXT BYTE INTO R0
        JNE     STROU1          KEEP SENDING IF NOT STOP FLAG
        B       *R10
*
STRIN   MOV     R11,R10         SAVE RETURN ADDRESS
        MOV     R0,R2           COPY BUFFER ADDRESS
STRIN1  BL      @BYTIN          GET BYTE FROM GPIB
        MOVB    R0,*R2+         COPY BYTE INTO BUFFER
        ANDI    R1,ENDB         CHECK IF LAST BYTE
        JEQ     STRIN1          KEEP GETTING BYTES IF NOT
        SB      *R2,*R2         CLEAR BYTE AFTER LAST ONE
        B       *R10
```

```
*
DOAUX   MOV     *R11,@AUXCMD    SEND AUX CMD IN WORD AFTER CALL
        INCT    R11             BUMP RETURN ADDRESS
        RT
*
DELAY   LI      R1,150          DELAY NUMBER OF MS IN R0
DEL01   DEC     R1              DECREMENT MS TIMER
        JNE     DEL01           COUNT DOWN 1 MS
        DEC     R0              DECREMENT MAIN TIMER
        JGT     DELAY           LOOP UNTIL TIME IS UP
        RT
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                             *
*       TALKING SUBROUTINE DEFINITIONS (TM990/306 BOARD)      *
*                                                             *
*            TLKSET - INITIALIZES TALKING ROUTINE WORKSPACE   *
*            TLKWRD - SPEAKS WORD AT SPEECH ADDRESS INDICATED *
*                     BY (R0) IF (R0) >= 0 OR DELAYS NUMBER OF *
*                     MS INDICATED BY -(R0) IF (R0) < 0       *
*            TLKSEN - SPEAKS SENTENCE POINTED TO BY R1, HALTING *
*                     WHEN WORD TO SPEAK IS >FFFF             *
*            DIGCVT - CONVERTS ASCII CODE FOR DECIMAL DIGITS  *
*                     INTO SPEECH ADDRESSES                   *
*            REDTLK - SPEAKS VALUE OF HP3455A FORMATTED OUTPUT *
*                     DATA POINTED TO BY R2                   *
*                                                             *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
TLKCRU  EQU     >1FE0           TM990/306 BOARD HI CRU BASE ADDRESS
TLKBSY  EQU     -15             BUSY STATUS BIT
TLKNTK  EQU     14              TALK/NOTALK BIT
TLKEPM  EQU     15              TALK EPROM ENABLE BIT
*
TLKSET  LI      R12,TLKCRU      INITIALIZE TALK CRU
        SBO     TLKEPM          SET EPROM ENABLE BIT
        SBO     TLKNTK          AND TURN OFF TALKING
        RTWP
*
TLKGO   DATA    TALKWS          BLWP XFER VECTOR
        DATA    TLKSET
*
TLKWRD  MOV     *R13,R0         GET WORD FROM OLD WS
        ABS     R0              CHECK IF WORD OR DELAY
        JLT     TLKW03          SKIP IF DELAY
        LDCR    R0,14           SELECT WORD TO SPEAK
        SBZ     TLKNTK          TURN ON TALKING
TLKW01  TB      TLKBSY          WAIT FOR 306 TO START
        JNE     TLKW01
TLKW02  TB      TLKBSY          WAIT FOR 306 TO STOP
        JEQ     TLKW02
        SBO     TLKNTK          TURN OFF TALKING
        RTWP
TLKW03  BL      @DELAY          PERFORM DELAY
        RTWP
```



```
*
TALK    DATA    TALKWS          BLWP XFER VECTOR
        DATA    TLKWRD
*
TLKSEN  MOV     *R1+,R0         COPY WORD TO R0
        CI      R0,-1           QUIT IF STOP FLAG
        JEQ     TLKS01
        BLWP    @TALK           SAY ONE WORD OR DELAY
        JMP     TLKSEN          GET NEXT WORD
TLKS01  RT
*
DIGCVT  CI      R1,>30          CHECK IF ASCII DIGIT
        JLT     DIGC01
        CI      R1,>3A
        JLT     DIGC02
DIGC01  LI      R0,-1           RETURN DELAY IF NOT
        RT
DIGC02  ANDI    R1,>000F        ISOLATE BCD VALUE
        SLA     R1,1            MULTIPLY BY 2
        MOV     @DIGTLK(R1),R0  GET SPEECH VALUE
        RT
DIGTLK  DATA    >2E94           ZERO
        DATA    >06FC           ONE
        DATA    >0900           TWO
        DATA    >249A           THREE
        DATA    >0372           FOUR
        DATA    >03C0           FIVE
        DATA    >355E           SIX
        DATA    >23AE           SEVEN
        DATA    >02AC           EIGHT
        DATA    >31C2           NINE
*
REDTLK  MOV     R11,R10         SAVE RETURN ADDRESS
REDT01  MOVB    *R2+,R1         GET BYTE TO SPEAK
        JEQ     REDT07          QUIT IF STOP BYTE
        SRL     R1,8            ISOLATE BYTE IN LS
        CI      R1,>2C          CHECK FOR ","
        JEQ     REDT06          SKIP TO END IF SO
        CI      R1,>2D          COMPARE TO "-"
        JNE     REDT02          SKIP IF NOT
        LI      R0,>0F02        INDICATE "MINUS"
        JMP     REDT05          SKIP AHEAD
REDT02  CI      R1,>2E          COMPARE TO "."
        JNE     REDT03          SKIP IF NOT
        LI      R0,>2146        INDICATE "POINT"
        JMP     REDT05          SKIP AHEAD
REDT03  CI      R1,>45          COMPARE TO "E"
        JNE     REDT04          SKIP AHEAD IF NOT
        LI      R1,ETOTHE       SAY "10 TO THE"
        BL      @TLKSEN
        JMP     REDT01
REDT04  BL      @DIGCVT         CONVERT DIGIT
```

```
REDT05  BLWP    @TALK           SAY THE WORD
        JMP     REDT01          GET NEXT BYTE
REDT06  MOVB    *R2+,R1         GET FORMAT BYTE
        ANDI    R1,>0F00        ISOLATE FORMAT CODE
        SRL     R1,7            SHIFT AND CREATE INDEX
        DECT    R1              MAKE INDEX 0 -> 4
        MOV     @UNITS(R1),R1   GET UNITS MESSAGE PTR
        BL      @TLKSEN         AND SAY IT
REDT07  B       *R10
ETOTHE  DATA    -200            DELAY 200MS
        DATA    >2D06           "TEN"
        DATA    >0900           "TO"
        DATA    >087A           "THE"
        DATA    -1
UNITS   DATA    DCV             DC VOLTS
        DATA    ACV             AC VOLTS
        DATA    DCI             DC AMPS
        DATA    ACI             AC AMPS
        DATA    OHMS            OHMS
DCV     EQU     $
        DATA    -200
        DATA    >36FA           "VOLTS"
        DATA    >1740           "D"
        DATA    >15A0           "C"
        DATA    -1
ACV     EQU     $
        DATA    -200
        DATA    >36FA           "VOLTS"
        DATA    >09CC           "A"
        DATA    >15A0           "C"
        DATA    -1
DCI     EQU     $
        DATA    -200
        DATA    >142E           "AMPS"
        DATA    >1740           "D"
        DATA    >15A0           "C"
        DATA    -1
ACI     EQU     $
        DATA    -200
        DATA    >142E           "AMPS"
        DATA    >09CC           "A"
        DATA    >15A0           "C"
        DATA    -1
OHMS    EQU     $
        DATA    -200
        DATA    >2078           OHMS
        DATA    -1
*
```



```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                   *
*       MAIN PROGRAM                                                *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*
START   BLWP    @TLKGO          INITIALIZE 306 BOARD
        LI      R1,SPINTR       GIVEN INTIAL MESSAGE
        BL      @TLKSEN
        BL      @DOAUX          SEND SWRST
        DATA    SWRST
        BL      @DOAUX          RESET SWRST
        DATA    SWRSTC
        LI      R0,MASK0        SET UP INTERRUPTS
        MOV     R0,@IMASK0
        MOV     @ADRSWI,R0      READ ADDRESS SWITCH
        ANDI    R0,>1F00        TRIM TO 5 BITS
        MOV     R0,@ADDRES      INITIALIZE 9914 ADDRESS
        BL      @DOAUX          SEND IFC
        DATA    SIC
        LI      R0,1            DELAY 1 MS
        BL      @DELAY
        BL      @DOAUX          CLEAR IFC/ENTER CACS
        DATA    SICC
        BL      @DOAUX          SEND REMOTE ENABLE
        DATA    SRE
        LI      R0,HPLA         SEND HP3438A LISTEN ADDRESS
        BL      @BYTOUT
        LI      R0,GET          SEND TRIGGER COMMAND
        BL      @BYTOUT
        LI      R0,UNL          SEND UNLISTEN COMMAND
        BL      @BYTOUT
        BL      @DOAUX          TURN ON 9914 LISTENER
        DATA    LON
        LI      R0,HPTA         SEND HP3438A TALK ADDRESS
        BL      @BYTOUT
        BL      @DOAUX          GO TO STANDBY/ATN = 0
        DATA    GTS
        LI      R0,DATBUF       GET READING FROM GPIB
        BL      @STRIN
        BL      @DOAUX          TURN OFF LISTENER
        DATA    LONC
        LI      R2,DATBUF       POINT TO READING
        BL      @REDTLK         AND SAY IT
        BL      @DOAUX          CLEAR THE INTERFACE
        DATA    SIC
        LI      R0,1            DELAY 1 MS
        BL      @DELAY
        BL      @DOAUX          RELEASE INTERFACE CLEAR
        DATA    SICC
        BL      @DOAUX          DISABLE REMOTE OPERATION
        DATA    SREC
```

```
        BL      @DOAUX          RELEASE CONTROL
        DATA    GTS
        LI      R1,SPPMPT       GIVE NEW START MESSAGE
START2  BL      @TLKSEN
        LI      R0,30000        WAIT 30 SECONDS
        BL      @DELAY
        LI      R1,SPPMPT       POINT TO PROMPT
        JMP     START2          AND REPEAT
*
*       SPEECH LISTS
*
SPPMPT  DATA    >2188           PRESS
        DATA    >087A           THE
        DATA    >229C           RED
        DATA    >082C           SWITCH
        DATA    >0900           TO
        DATA    >0EA0           MEASURE
        DATA    -1
*
SPINTR  DATA    >33D6           READY
        DATA    >0900           TO
        DATA    >0D5C           GO
        DATA    -500
        DATA    -1
*
        DORG    >FF00
*
MAINWS  BSS     32              TEMPORARY LOCATIONS
TALKWS  BSS     32
DATBUF  BSS     20              DATA BUFFER
*
        END     START
```