HD68450
Direct Memory Access Controller (NMOS)

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.

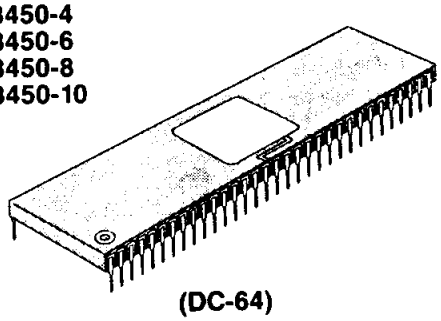
The HD68450 Direct Memory Access Controller (DMAC) is designed specifically to complement the performance and architectural capabilities of the HD68000 MPU by providing the following features:

- HD68000 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 for Requesting, Acknowledging, and Incidental Control of the Peripheral Devices
- Variable System Bus Bandwidth Utilization
- Programmable Channel Prioritization
- 2 Vectored interrupts for each Channel
- Auto-Request and External-Request Transfer Modes
- +5 Volt Operation

The DMAC functions by transfering a series of operands (data) between memory and peripheral 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.

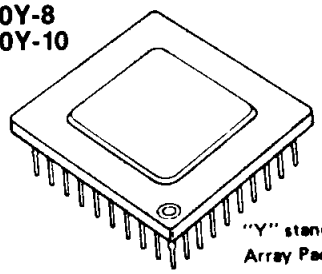
Type No.	Bus Timing	Packaging
HD68450-4	4HMz	DC-64
HD68450-6	6HMz	DC-64
HD68450-8	8HMz	DC-64
HD68450-10	10HMz	DC-64
HD68450Y-4	4HMz	PGA-68
HD68450Y-6	6HMz	PGA-68
HD68450Y-8	8HMz	PGA-68
HD68450Y-10	10HMz	PGA-68

HD68450-4
HD68450-6
HD68450-8
HD68450-10



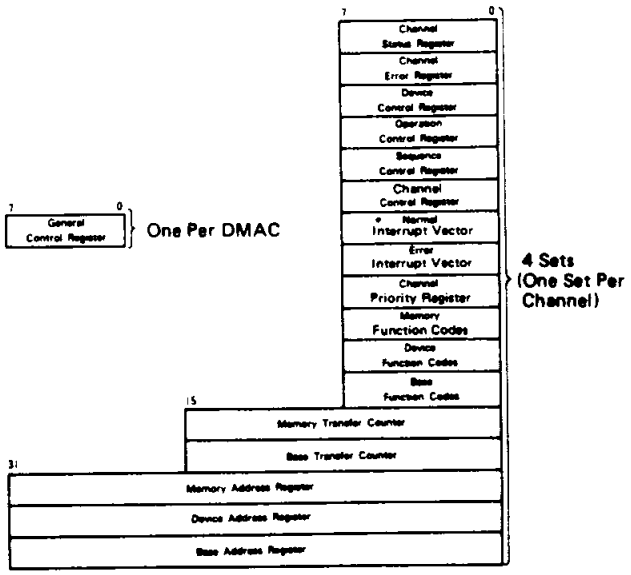
(DC-64)

HD68450Y-4
HD68450Y-6
HD68450Y-8
HD68450Y-10



"Y" stands for Pin Grid Array Package.

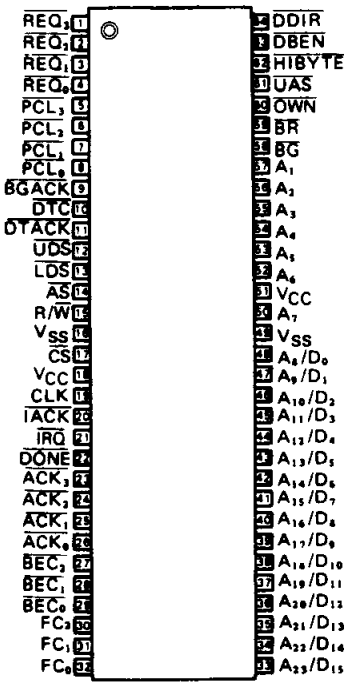
(PGA-68)



HITACHI
Hitachi America, Ltd. • Hitachi Plaza • 2000 Sierra Point Pkwy. • Brisbane, CA 94005-1819 • (415) 589-8300

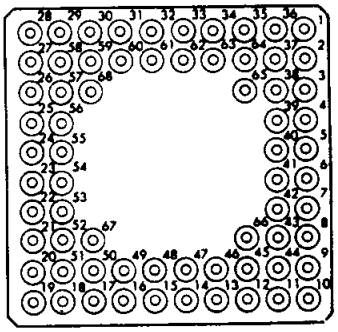
■ PIN ARRANGEMENT

- **HD68450**



(Top View)

- **HD68450Y**



(Bottom View)

Pin No.	Function	Pin No.	Function	Pin No.	Function	Pin No.	Function
1	N/C	18	$\overline{PCL_1}$	35	A_{19}/D_{11}	52	$\overline{BGACK}$
2	A_{13}/D_5	19	$\overline{DTACK}$	36	A_{17}/D_9	53	$\overline{LDS}$
3	A_{11}/D_3	20	$\overline{UDS}$	37	A_{15}/D_7	54	V_{SS}
4	A_{10}/D_2	21	$\overline{AS}$	38	A_{12}/D_4	55	V_{CC}
5	A_8/D_0	22	R/$\overline{W}$	39	A_9/D_1	56	$\overline{DONE}$
6	A_7	23	N/C	40	V_{SS}	57	$\overline{IRQ}$
7	A_6	24	$\overline{CS}$	41	V_{CC}	58	$\overline{ACK_2}$
8	A_5	25	CLK	42	A_4	59	$\overline{BEC_2}$
9	A_3	26	$\overline{IACK}$	43	A_2	60	$\overline{BEC_0}$
10	N/C	27	$\overline{ACK_3}$	44	$\overline{BG}$	61	FC_0
11	$\overline{BR}$	28	$\overline{ACK_0}$	45	$\overline{OWN}$	62	A_{21}/D_{13}
12	$\overline{UAS}$	29	$\overline{BEC_1}$	46	$\overline{HIBYTE}$	63	A_{18}/D_{10}
13	$\overline{DBEN}$	30	FC_2	47	$\overline{DDIR}$	64	A_{16}/D_8
14	$\overline{REQ_3}$	31	FC_1	48	$\overline{REQ_1}$	65	A_{14}/D_6
15	$\overline{REQ_2}$	32	A_{23}/D_{15}	49	$\overline{PCL_2}$	66	A_1
16	$\overline{REQ_0}$	33	A_{22}/D_{14}	50	$\overline{PCL_0}$	67	$\overline{DTC}$
17	$\overline{PCL_3}$	34	A_{20}/D_{12}	51	N/C	68	$\overline{ACK_1}$

■ 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	$\overline{IRQ}$, $\overline{DONE}$	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	
	$\overline{IRQ}$, $\overline{DONE}$	V_{OL}	I_{OL} = 8.9 mA	–	–	0.5	
Power Dissipation		P_D	f = 8 MHz, V_{CC} = 5.0 V, Ta = 25°C	–	1.4	2.0	W
Capacitance		C_{in}	V_{in} = 0V, Ta = 25°C, f = 1 MHz	–	–	15	pF

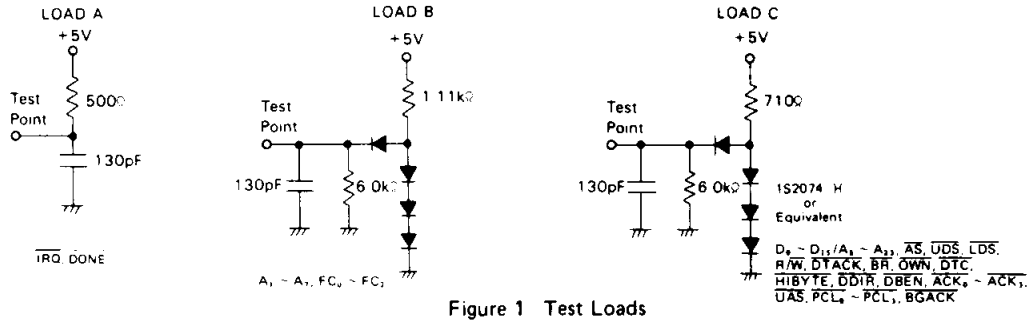


Figure 1 Test Loads

■ AC ELECTRICAL SPECIFICATIONS (V_{CC} = 5V ±5%, V_{SS} = 0V, Ta = 0 ~ +70°C)

No.	Item	Symbol	4MHz HD68450		6MHz HD68450		8MHz HD68450		10MHz HD68450		Unit	?
			min.	max.	min.	max.	min.	max.	min.	max.		
	Frequency of Operation	f	2	4	2	6	2	8	2	10	MHz	
1	Clock Period	t_{cyc}	250	500	167	500	126	500	100	500	ns	
2	Clock Width Low	t_{CL}	115	250	75	250	55	250	45	250	ns	
3	Clock Width High	t_{CH}	115	250	75	250	55	250	45	250	ns	
4	Clock Fall Time	t_{Cf}	—	10	—	10	—	10	—	10	ns	
5	Clock Rise Time	t_{Cr}	—	10	—	10	—	10	—	10	ns	
6	Asynchronous Input Setup Time	t_{ASI}	30	—	25	—	20	—	15	—	ns	
7	Data in to $\overline{DBEN}$ Low	t_{DIDBL}	0	—	0	—	0	—	0	—	ns	
8	$\overline{DTACK}$ Low to Data Invalid	t_{DTLDI}	0	—	0	—	0	—	0	—	ns	
9	Address in to $\overline{AS}$ in Low	t_{AIASL}	0	—	0	—	0	—	0	—	ns	
10	$\overline{AS}$, $\overline{DS}$ in High to Address in Invalid	t_{SIHAIV}	0	—	0	—	0	—	0	—	ns	
10A	$\overline{DS}$ in High to $\overline{CS}$ High	t_{DSHCSH}	—	1.0	—	1.0	—	1.0	—	1.0	clk. per.	
11	Clock High to $\overline{DDIR}$ Low	t_{CHDRL}	—	90	—	80	—	70	—	60	ns	
12	Clock High to $\overline{DDIR}$ High	t_{CHDRH}	—	90	—	80	—	70	—	60	ns	
13	$\overline{DS}$ in High to $\overline{DDIR}$ High Impedance	t_{DSHDRZ}	—	160	—	140	—	120	—	110	ns	
14	Clock Low to $\overline{DBEN}$ Low	t_{CLDBL}	—	90	—	80	—	70	—	54	ns	
15	Clock Low to $\overline{DBEN}$ High	t_{CLDBH}	—	90	—	80	—	70	—	60	ns	
16	$\overline{DS}$ in High to $\overline{DBEN}$ High Impedance	t_{DSHDBZ}	—	160	—	140	—	120	—	110	ns	
17	Clock High to Data Out Valid (MPU read)	t_{CHDVM}	—	290	—	230	—	180	—	160	ns	
18	$\overline{DS}$ in High to Data Out Invalid	t_{DSHDZn}	0	—	0	—	0	—	0	—	ns	
19	$\overline{DS}$ in High to Data High Impedance	t_{DSHDZ}	—	160	—	140	—	120	—	110	ns	
20	Clock Low to $\overline{DTACK}$ Low	t_{CLDTL}	—	90	—	80	—	70	—	60	ns	
21	$\overline{DS}$ in High to $\overline{DTACK}$ High	t_{DSHDTH}	—	160	—	130	—	110	—	110	ns	
22	$\overline{DTACK}$ Width High	t_{DTH}	10	—	10	—	10	—	10	—	ns	
23	$\overline{DS}$ in High to $\overline{DTACK}$ High Impedance	t_{DSHDTZ}	—	220	—	200	—	180	—	160	ns	
24	$\overline{DTACK}$ Low to $\overline{DS}$ in High	t_{DTLDSH}	0	—	0	—	0	—	0	—	ns	
25	$\overline{REQ}$ Width Low	t_{REQL}	2.0	—	2.0	—	2.0	—	2.0	—	clk. per.	
26	$\overline{REQ}$ Low to $\overline{BR}$ Low	t_{RELBRL}	500	—	334	—	250	—	200	—	ns	
27	Clock High to $\overline{BR}$ Low	t_{CHBRL}	—	90	—	80	—	70	—	60	ns	
28	Clock High to $\overline{BR}$ High	t_{CHBRH}	—	90	—	80	—	70	—	60	ns	
29	$\overline{BG}$ Low to $\overline{BGACK}$ Low	t_{BGLBL}	4.5	—	4.5	—	4.5	—	4.5	—	clk. per.	
30	$\overline{BR}$ Low to MPU Cycle End ($\overline{AS}$ in High)	t_{BRLASH}	0	—	0	—	0	—	0	—	ns	
31	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	4.5	5.5	clk. per.	
32	$\overline{REQ}$ Low to $\overline{BGACK}$ Low	t_{REQLBL}	12.0	—	12.0	—	12.0	—	12.5	—	clk. per.	
33	Clock High to $\overline{BGACK}$ Low	t_{CHBL}	—	90	—	80	—	70	—	60	ns	
34	Clock High to $\overline{BGACK}$ High	t_{CHBH}	—	90	—	80	—	70	—	60	ns	
35	Clock Low to $\overline{BGACK}$ High Impedance	t_{CLBZ}	—	120	—	100	—	80	—	70	ns	
36	Clock High to FC Valid	t_{CHFCV}	—	140	—	120	—	100	—	90	ns	
37	Clock High to Address Valid	t_{CHAV}	—	160	—	140	—	120	—	110	ns	
38	Clock High to Address/FC/Data High Impedance	t_{CHAZx}	—	140	—	120	—	100	—	110	ns	
39	Clock High to Address/FC/Data Invalid	t_{CHAZn}	0	—	0	—	0	—	0	—	ns	
40	Clock High to Address High Impedance	t_{CLAZ}	—	140	—	120	—	100	—	90	ns	
41	Clock High to $\overline{UAS}$ Low	t_{CHUL}	—	90	—	80	—	70	—	60	ns	
42	Clock High to $\overline{UAS}$ High	t_{CHUH}	—	90	—	80	—	70	—	60	ns	
43	Clock Low to $\overline{UAS}$ High Impedance	t_{CLUZ}	—	120	—	100	—	80	—	70	ns	
44	$\overline{UAS}$ High to Address Invalid	t_{UHAI}	50	—	40	—	30	—	20	—	ns	
45	Clock High to $\overline{AS}$, $\overline{DS}$ Low	t_{CHSL}	—	80	—	70	—	60	—	55	ns	
46	Clock Low to $\overline{DS}$ Low (write)	t_{CLDSL}	—	80	—	70	—	60	—	55	ns	
47	Clock Low to $\overline{AS}$, $\overline{DS}$ High	t_{CLSH}	—	90	—	80	—	700	—	60	ns	

No.	Item	Symbol	4MHz HD68450		6MHz HD68450		8MHz HD68450		10MHz HD68450		Unit	?
			min.	max.	min.	max.	min.	max.	min.	max.		
48	Clock Low to AS, DS High Impedance	tCLSZ	—	120	—	100	—	80	—	70	ns	
49	AS Width Low	tASL	545	—	350	—	255	—	195	—	ns	
50	DS Width Low	tDSL	420	—	265	—	190	—	145	—	ns	
51	AS, DS Width High	tSH	285	—	180	—	150	—	105	—	ns	
52	Address FC Valid to AS, DS Low	tAVSL	50	—	40	—	30	—	20	—	ns	
53	AS, DS High to Address FC Data Invalid	tSHAZ	50	—	40	—	30	—	20	—	ns	
54	Clock High to R W Low	tCHRL	—	90	—	80	—	70	—	60	ns	
55	Clock High to R W High	tCHRH	—	90	—	80	—	70	—	60	ns	
56	Clock Low to R W High Impedance	tCLRZ	—	120	—	100	—	80	—	70	ns	
57	Address FC Valid to R W Low	tAVRL	100	—	40	—	20	—	10	—	ns	
58	R W Low to DS Low (write)	tRLSL	285	—	170	—	120	—	90	—	ns	
59	DS High to R W High	tSHRH	60	—	50	—	40	—	20	—	ns	
60	Clock Low to OWN Low	tCLOL	—	90	—	80	—	70	—	60	ns	
61	Clock Low to OWN High	tCLOH	—	90	—	80	—	70	—	60	ns	
62	Clock High to OWN High Impedance	tCHOZ	—	120	—	100	—	80	—	70	ns	
63	OWN Low to BGACK Low	tOLBL	50	—	40	—	30	—	20	—	ns	
64	BGACK High to OWN High	tBHOH	50	—	40	—	30	—	20	—	ns	
65	OWN Low to UAS Low	tOLUL	50	—	40	—	30	—	20	—	ns	
66	Clock High to ACK Low	tCHACL	—	90	—	80	—	70	—	60	ns	
67	Clock Low to ACK Low	tCLACL	—	90	—	80	—	70	—	660	ns	
68	Clock High to ACK High	tCHACH	—	90	—	80	—	70	—	60	ns	
69	ACK Low to DS Low	tACLDSL	230	—	140	—	100	—	80	—	ns	
70	DS High to ACK High	tDSHACH	50	—	40	—	30	—	20	—	ns	
71	Clock High to HIBYTE Low	tCHHIL	—	90	—	80	—	70	—	60	ns	
72	Clock Low to HIBYTE Low	tCLHIL	—	90	—	80	—	70	—	60	ns	
73	Clock High to HIBYTE High	tCHHIH	—	90	—	80	—	70	—	60	ns	
74	Clock Low to HIBYTE High Impedance	tCLHIZ	—	120	—	100	—	80	—	70	ns	
75	Clock High to DTC Low	tCHDTL	—	90	—	80	—	70	—	60	ns	
76	Clock High to DTC High	tCHDTH	—	90	—	80	—	70	—	60	ns	
77	Clock Low to DTC High Impedance	tCLDTZ	—	120	—	120	—	80	—	70	ns	
78	DTC Width Low	tDTCL	230	—	147	—	105	—	80	—	ns	
79	DTC Low to DS High	tDTLDH	95	—	50	—	30	—	20	—	ns	
80	Clock High to DONE Low	tCHDOL	—	90	—	80	—	70	—	60	ns	
81	Clock Low to DONE Low	tCLDOL	—	90	—	80	—	70	—	60	ns	
82	Clock High to DONE High	tCHDOH	—	150	—	140	—	130	—	120	ns	
83	Clock Low to DDIR High Impedance	tCLDRZ	—	120	—	100	—	80	—	70	ns	
84	Clock Low to DBEN High Impedance	tCLDBZ	—	120	—	100	—	80	—	70	ns	
85	DDIR Low to DBEN Low	tDRLDBL	50	—	40	—	30	—	20	—	ns	
86	DBEN High to DDIR High	tDBHDRH	50	—	40	—	330	—	20	—	ns	
87	DBEN Low to Address Data High Impedance	tDBLAZ	—	17	—	17	—	17	—	17	ns	
88	Clock Low to PCL Low (1/8 clock)	tCLPL	—	90	—	80	—	70	—	60	ns	
89	Clock Low to PCL High (1/8 clock)	tCLPH	—	90	—	80	—	70	—	60	ns	
90	PCL Width Low (1/8 clock)	tPCLL	4.0	—	4.0	—	4.0	—	4.0	—	clk. per.	
91	DTACK Low to Data In (setup time)	TDALDI	—	320	—	200	—	150	—	115	ns	
92	DS High to Data Invalid (hold time)	tSHDI	0	—	0	—	0	—	0	—	ns	
93	DS High to DTACK High	tSHDAH	0	240	0	160	0	120	0	90	ns	
94	Data Out Valid to DS Low	tDOSL	0	—	0	—	0	—	0	—	ns	
95	Data In to Clock Low (setup time)	tDICL	30	—	25	—	15	—	15	—	ns	
96	BEC Low to DTACK Low	tBECDAL	50	—	50	—	50	—	50	—	ns	
97	BEC Width Low	tBECL	2.0	—	2.0	—	2.0	—	2.0	—	clk. per.	
98	Clock High to IRQ Low	tCHIRL	—	90	—	80	—	70	60	ns		
99	Clock High to IRQ High	tCHIRH	—	150	—	140	—	130	—	120	ns	
100	READY In to DTC Low (Read)	tRALDTL	270	—	180	—	145	—	120	—	ns	
101	READY In to DS Low (Write)	tRALDSL	395	—	240	—	205	—	170	—	ns	
102	DS High to READY High	tDSHRAH	0	240	0	160	0	120	0	90	ns	
103	DONE In Low to DTACK Low	tDOLDAL	50	—	50	—	50	—	50	—	ns	
104	DS High to DONE In High	tDSHDOH	0	240	0	160	0	120	0	90	ns	
105	Asynchronous Input Hold Time	tASIH	15	—	15	—	15	—	15	—	ns	

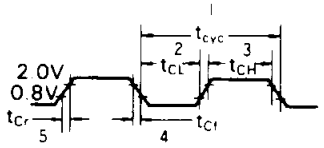


Figure 2 Input Clock Waveform

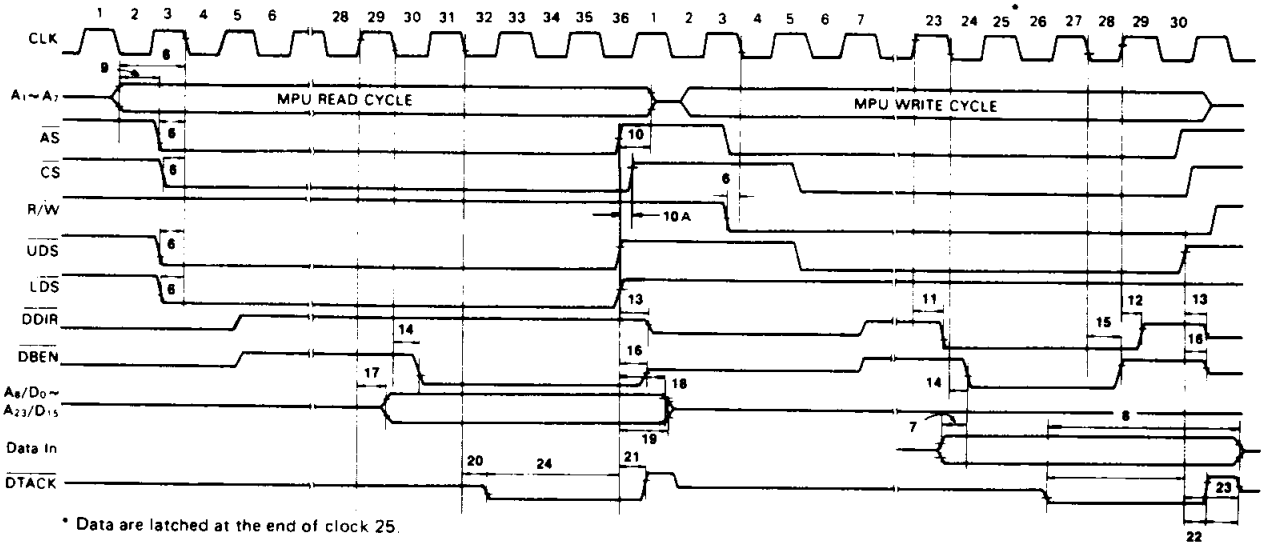
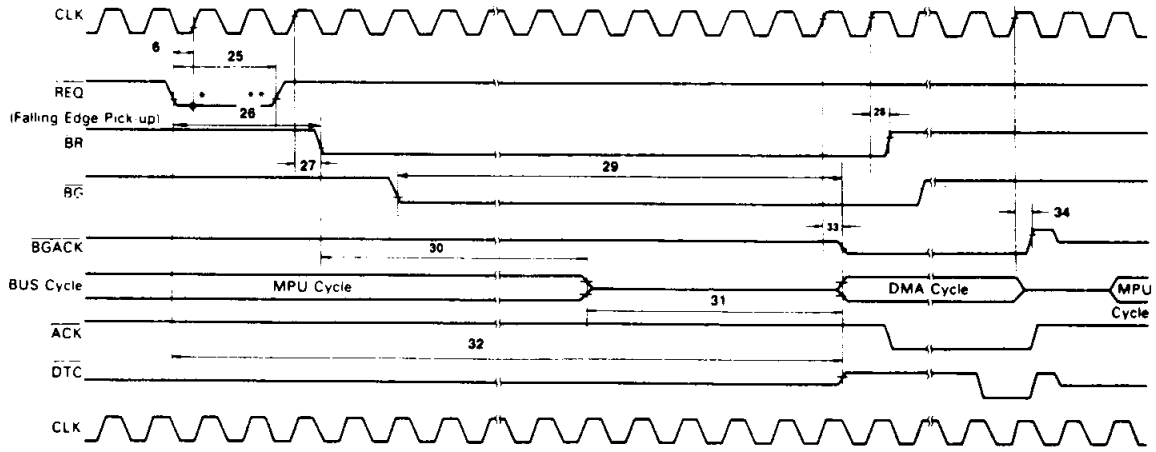
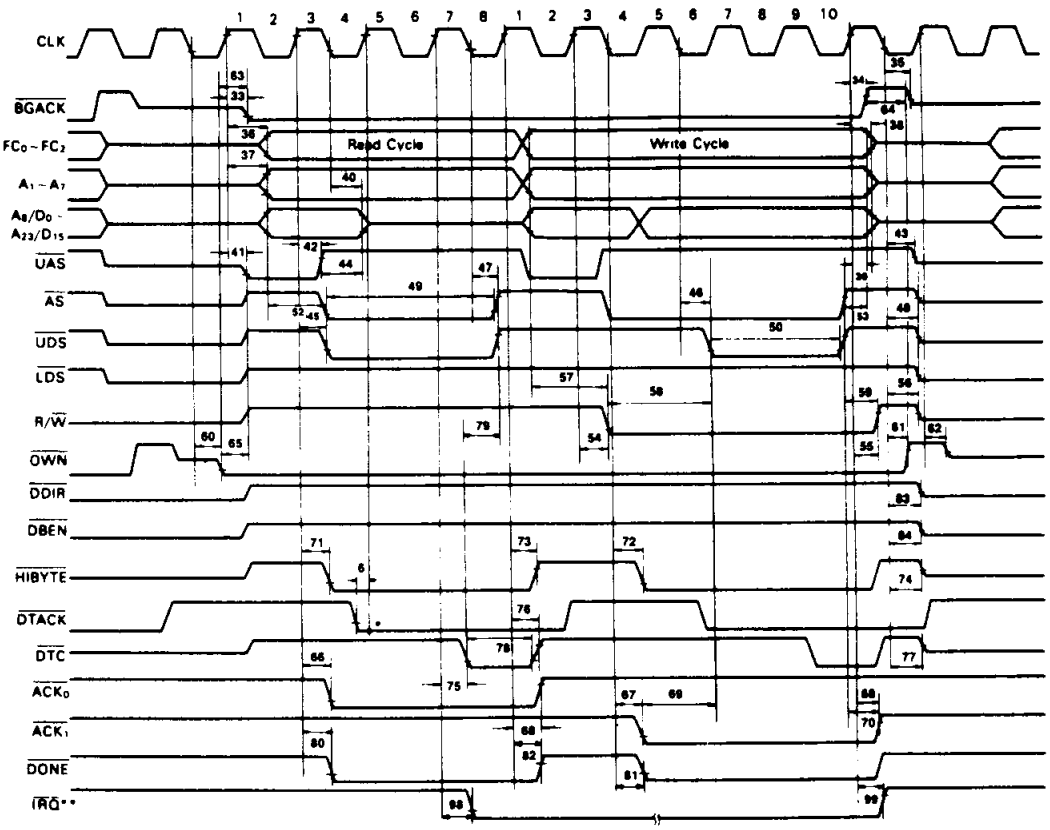


Figure 3 AC Electrical Waveforms – MPU Read/Write



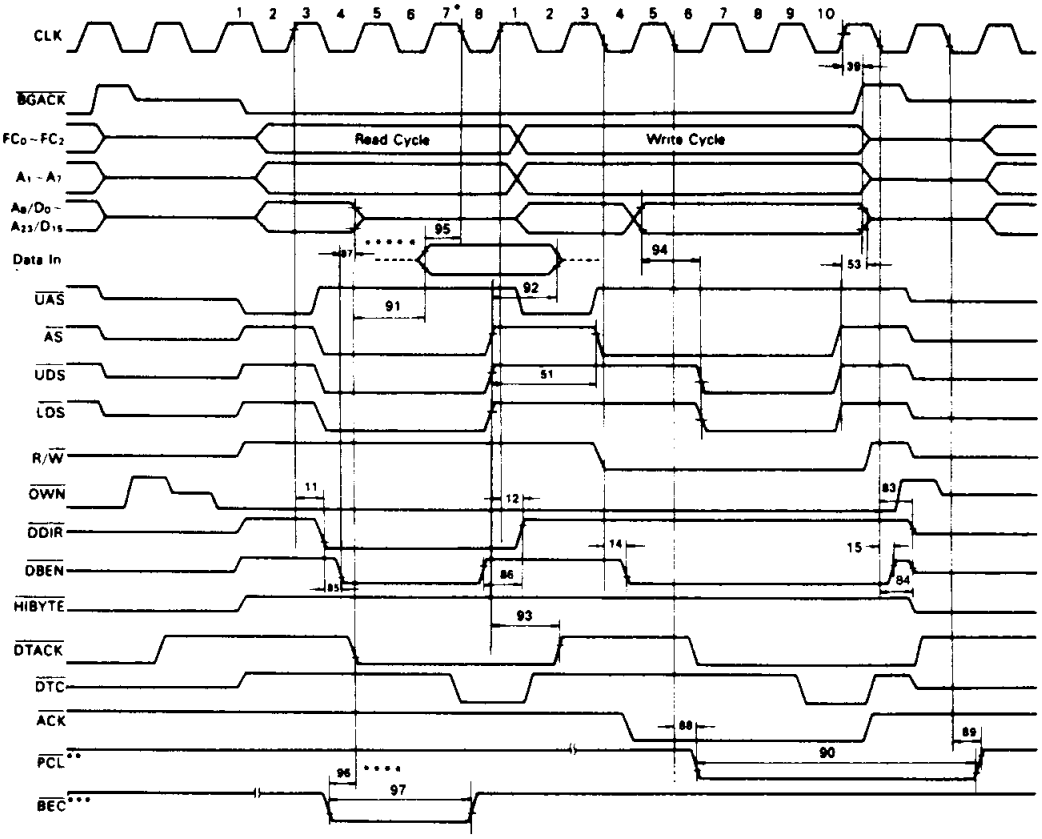
* $\overline{REQ}$ is picked up at the rising edge of CLK in cycle steal and Burst modes.
** $\overline{BR}$ isn't asserted while some $\overline{BEC}$ exception condition exists or DMAC is accessed by MPU.

Figure 4 AC Electrical Waveforms – Bus Arbitration



* $\overline{\text{DTACK}}$ is picked up at the rising edge of CLK. This is different from HD68000.
** This timing is not related to DMA Read/Write (Single Cycle) sequence.

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



* Data are latched at the end of clock 7. This timing is the same as HD68000.
** This timing is not related to DMA Read/Write (Dual Cycle) sequence. This timing is only applicable when 1/8 clock pulse mode is selected.
*** This timing is applicable when a bus exception occures.
**** If #6 is satisfied for both $\overline{DTACK}$ and $\overline{BEC}$, #96 may be 0ns.
***** If the propagation delay of the external bidirectional buffer LS245 is less than 17nsec, the conflict may occur between the address output of the DMAC and the system data bus. In this case, the output of $\overline{DBEN}$ must be delayed externally.

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

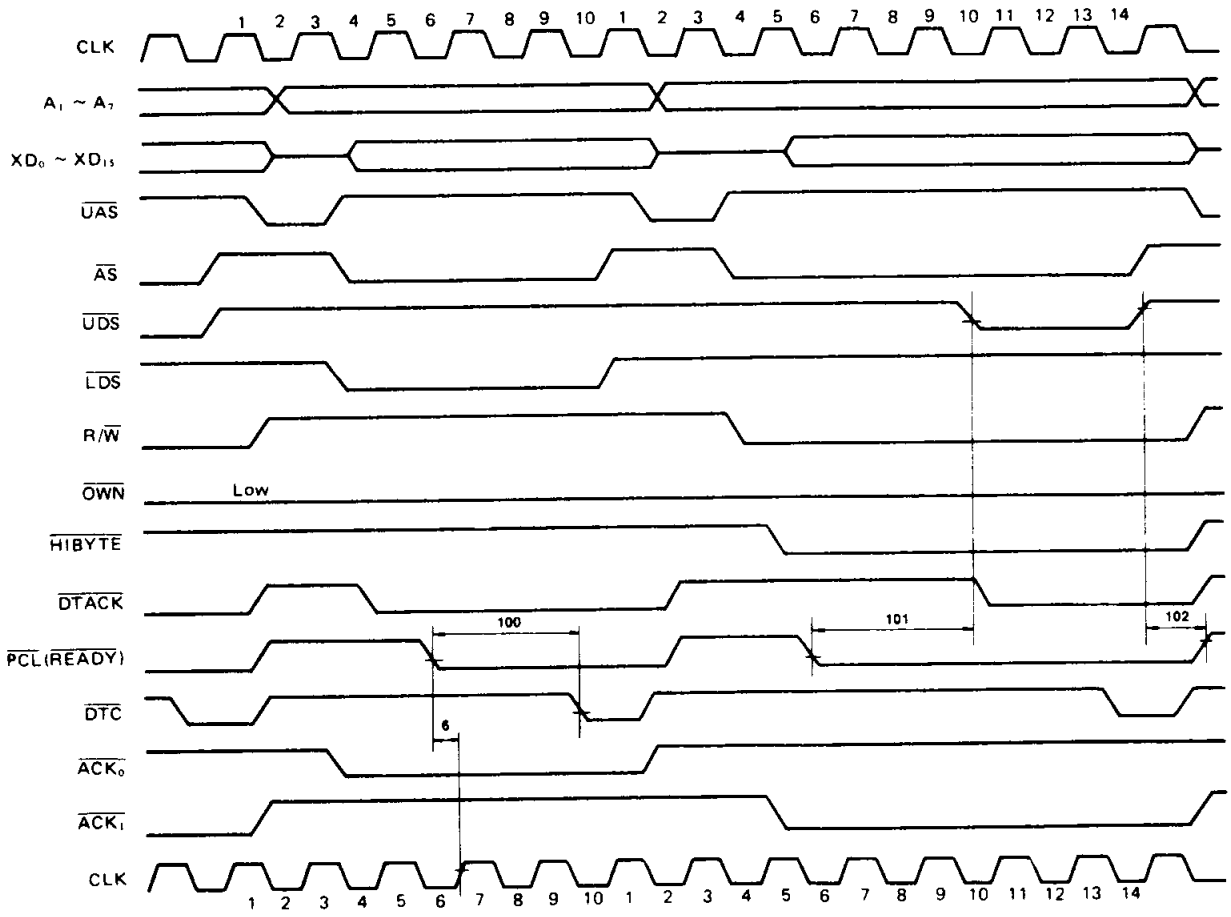


Figure 7 AC Electrical Waveforms—DMA Read/Write (Single Cycle with $\overline{ACK}$ and $\overline{PCL}$)

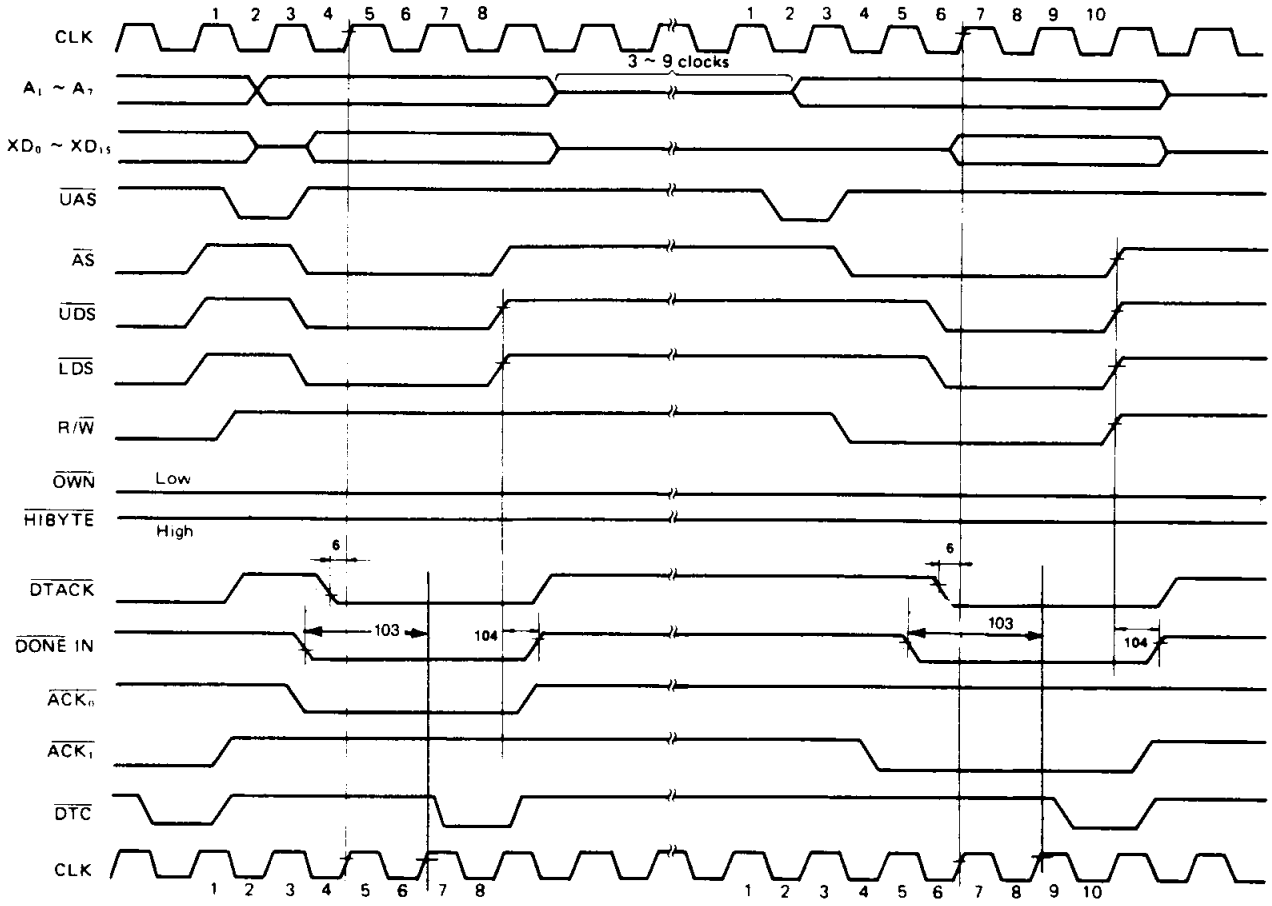


Figure 8 AC Electrical Waveforms – $\overline{\text{DONE}}$ Input

(NOTES for Figure 3 through 8)

1) Setup time for the asynchronous inputs $\overline{BG}$, $\overline{BGACK}$, $\overline{CS}$, $\overline{IACK}$, $\overline{AS}$, $\overline{UDS}$, $\overline{LDS}$, and R/$\overline{W}$ guarantees their recognition at the next falling edge of the clock. Setup time for $\overline{BEC_0}$ ~ $\overline{BEC_2}$, $\overline{REQ_0}$ ~ $\overline{REQ_3}$, $\overline{PCL_0}$ ~ $\overline{PCL_3}$, $\overline{DTACK}$, and $\overline{DONE}$ guarantees their recognition at the next rising edge of the clock.
2) Timing measurements are referenced to and from a low voltage of 0.8 volts and a high voltage of 2.0 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.

■ SIGNAL DESCRIPTION

The following section identifies the signals used in the DMAC. In the definitions, "MPU mode" refers to the state when the DMAC is chip selected by MPU. 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. Moreover, the DMA bus cycle refers to the bus cycle that is executed by the DMAC in the "DMA mode".

NOTE) In this data sheet, the state of the signals is described with these words: active or assert, inactive or negate.

This is done to avoid confusion when dealing with a mixture of "active-low" and "active-high" signals. The term assert or assertion is used to indicate that a signal is active or true independent of whether that voltage is low or high. The term negate or negation is used to indicate that a signal is inactive or false.

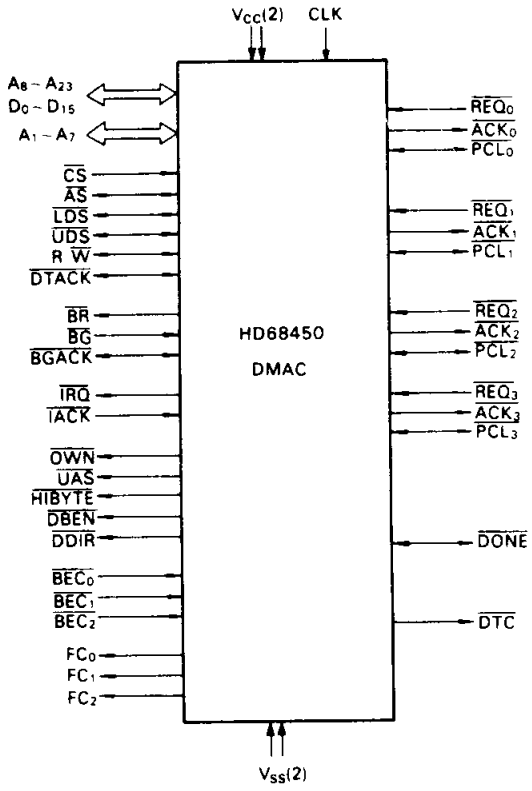


Figure 9 Input and Output Signals

- **Address/Data Bus (A_8/D_0 through A_{23}/D_{15})**

Input/Output Active-high	Three-statable

These lines are time multiplexed for address and data bus. The lines $\overline{DDIR}$, $\overline{DBEN}$, $\overline{UAS}$ and $\overline{OWN}$ are used to control the demultiplexing of the data and address lines externally. Demultiplexing is explained in the later section. The bi-directional data bus is used to transfer data between DMAC, MPU, memory and I/O devices.

Address lines are outputs to address memory and I/O devices.

- **Address Bus (A_1 through A_7)**

Input/Output Active-high	Three-statable

In the MPU mode, the DMAC internal registers are accessed with these lines and $\overline{LDS}$, $\overline{UDS}$. The address map for these registers is shown in Table 1. During a DMA bus cycle, A_1-A_7 are outputs containing the low order address bits of the location being accessed.

- **Function Code (FC_0 through FC_2)**

Output Active-high	Three-statable

These output signals provide the function codes during DMA bus cycles. They are three-stated except in the DMA bus cycles. They are used to control the HD68000 memories. *[See Attention on Usage, Note (6).]*

- **Clock (CLK)**

Input

This is the input clock to the HD68450, and should never be terminated at any time. This clock can be different from the MPU clock since HD68450 operates completely asynchronously.

- **Chip Select ($\overline{CS}$)**

Input Active low

This input signal is used to chip select the DMAC in "MPU" mode. If the $\overline{CS}$ input is asserted during a bus cycle which is generated by the DMAC, the DMAC internally terminates the bus cycle and signals an address error. This function protects the DMAC from accessing its own register. *[See Attention on Usage, Note (5).]*

- **Address Strobe ($\overline{AS}$)**

Input/Output Active low	Three-statable

In the "MPU mode", this line is an input indicating valid address input, and during the DMA bus cycle it is an output indicating valid the address output from the DMAC on the address bus.

The DMAC monitors these input lines during bus arbitration to determine the completion of the bus cycle by the MPU or other bus masters.

- **Upper Address Strobe ($\overline{UAS}$)**

Output Active low	Three-statable

This line is an output to latch the upper address lines on the multiplexed data/address lines. It is three-stated except in the "DMA mode".

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

Output Active low	Three-statable

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

- **Data Direction ($\overline{DDIR}$)**

Outputs	Three-statable
Active low	(when data direction is input to the DMAC)
Active high	(when the data direction is output from the DMAC)

This line controls the direction of data through the bidirectional buffer which is used to demultiplex the data/address lines. It is three-stated during the "IDLE mode"

- **Data Bus Enable ($\overline{DBEN}$)**

Output	Three-statable
Active low	

This line controls the output enable line of bidirectional buffers on the multiplexed data/address lines. It is a 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 single addressing mode. It is asserted when data is present on the upper eight bits of the data bus. It is used to control the output of bidirectional buffers which connects the upper eight bits of the data bus with the lower eight bits. It is three-stated during the "MPU mode" and the "IDLE mode"

- **Read/Write (R/$\overline{W}$)**

Input/Output	Three-statable
Active low (write)	
Active high (read)	

This line is an input in the "MPU mode" and an output during the "DMA mode". It is three-stated during the "IDLE mode". It is used to control the direction of data flow.

- **Upper Data Strobe ($\overline{UDS}$), 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 of the addressed word are being addressed. These lines combined corresponds to address line A_0 in table 1.

- **Data Transfer Acknowledge ($\overline{DTACK}$)**

Input/Output	Three-statable
Active low	

In the "MPU mode", this line is an output indicating the completion of Read/Write bus cycle by the MPU.

In the "DMA mode", the DMAC monitors this line to determine when a data transfer has completed. In the event that a bus exception is requested, except for $\overline{HALT}$, prior to or concurrent with $\overline{DTACK}$, the $\overline{DTACK}$ response is ignored and the bus exception is honored. In the "IDLE mode", this signal is three-stated.

- **Bus Exception Controls ($\overline{BEC_0}$ through $\overline{BEC_2}$)**

Input
Active low

These lines provide an encoded signal input indicating an exceptional condition in the DMA bus cycle. See bus exception section for details.

- **Bus Request ($\overline{BR}$)**

Output
Active low

This output line is used to request ownership of the bus by the DMAC. *[See Attention on Usage, Notes (8), (9).]*

- **Bus Grant ($\overline{BG}$)**

Input
Active low

This line is used to indicate to the DMAC that it is to be the next bus master. The DMAC cannot assume bus ownership until both $\overline{AS}$ and $\overline{BGACK}$ becomes 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 limited rate auto-request mode, to determine whether or not the current bus master is a DMA device or not. $\overline{BGACK}$ is also monitored during bus arbitration in order to assume bus ownership. *[See Attention on Usage, Notes (8), (9).]*

- **Interrupt Request ($\overline{IRQ}$)**

Output	Open drain
Active low	

This line is used to request an interrupt to the MPU.

- **Interrupt Acknowledge ($\overline{IACK}$)**

Input
Active low

This line is an input to the DMAC indicating that the current bus cycle is an interrupt acknowledge cycle by the MPU. The

DMAC responds the interrupt vector of the channel with the highest priority requesting an interrupt. There are two kinds of the interrupt vectors for each channel: normal (NIV) or error (EIV). $\overline{IACK}$ is not serviced if the DMAC has not generated $\overline{IRQ}$.

- **Channel Request ($\overline{REQ_0}$ through $\overline{REQ_3}$)**

Input Active low or falling edge

These lines are the DMA transfer request inputs from the peripheral devices.

These lines are falling edge sensitive inputs when the request mode is cycle steal. They are low-level sensitive when the request mode is burst.

- **Channel Acknowledge ($\overline{ACK_0}$ through $\overline{ACK_3}$)**

Output Active low

These lines indicate to the I/O device requesting a transfer that the request is acknowledged and the transfer is to be performed. These lines may be used as a part of the enable circuit for bus interface to the peripheral.

- **Peripheral Control Line ($\overline{PCL_0}$ through $\overline{PCL_3}$)**

Input/Output Active low	Three-statable

The four lines ($\overline{PCL_0} \sim \overline{PCL_3}$) are multi-purpose lines which may be individually programmed to be a START output, an Enable Clock input, a READY input, an ABORT input, a STATUS input, or an INTERRUPT input. *[See Attention on Usage, Note (2).]*

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

Input/Output Active low	Open Drain

As an output, this line is asserted concurrently with the $\overline{ACK_X}$ timing to indicate the last data transfer to the peripheral device. As an input, it allows the peripheral device to request a normal termination of the DMA transfer. *[See Attention on Usage, Note (2).]*

- **Device Transfer Complete ($\overline{DTC}$)**

Output Active low	Three-statable

This line is asserted when the DMA bus cycle has terminated normally with no exceptions. It may be used to supply the data latch timing to the peripheral device. In this case, data is valid at the falling edge of $\overline{DTC}$.

■ INTERNAL ORGANIZATION

The DMAC has four 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.

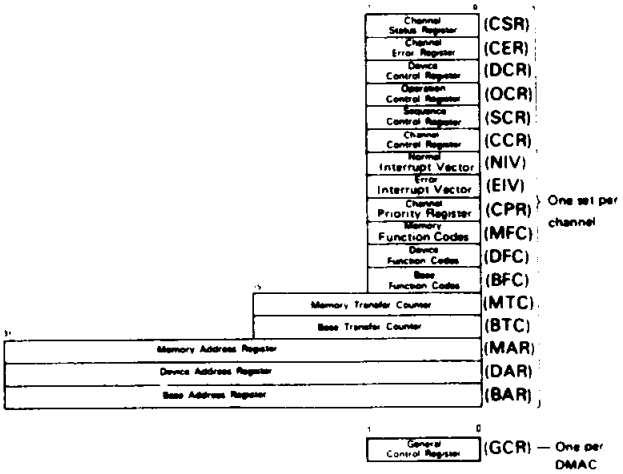


Figure 10 Internal Registers

- **Register Organization**

The internal register addresses are represented in Table 1. Address space not used within the address map is reserved for future expansion. A read from an unused location in the map results in a normal bus cycle with all ones for data. A write to one of these locations results in a normal bus cycle but no write occurs.

Unused bits of the defined registers in Table 1 read as zeros.

Table 1 Internal Register Addressing Assignments

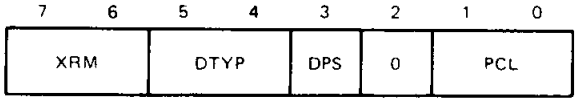
Register	Address Bits 7	6	5	4	3	2	1	0	Mode	
Channel Status Register	c	c	0	0	0	0	0	0	R	W*
Channel Error Register	c	c	0	0	0	0	0	1	R	
Device Control Register	c	c	0	0	0	1	0	0	R	W
Operation Control Register	c	c	0	0	0	1	0	1	R	W
Sequence Control Register	c	c	0	0	0	1	1	0	R	W
Channel Control Register	c	c	0	0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

- **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 HD6800 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 $\overline{READY}$ input. The DPS bit defines the port size (eight or sixteen bits) of peripheral device. (A port size is the largest data which the peripheral device can transfer during a DMA bus cycle.) the PCL bits define the function of the $\overline{PCL}$ line. If the DTYP bits are programmed to be HD6800 device, or Device with $\overline{ACK}$ and $\overline{READY}$, these definitions are ignored. The XRM bits are ignored if an auto-request mode (REQG =

00 or 01 in Operation Control Register) is selected.



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 HD68000 compatible device, explicitly addressed (dual addressing mode)
- 01 HD6800 compatible device, explicitly addressed (dual addressing mode)
- 10 Device with $\overline{ACK}$, implicitly addressed (single addressing mode)
- 11 Device with $\overline{ACK}$ and $\overline{READY}$, implicitly addressed (single addressing mode)

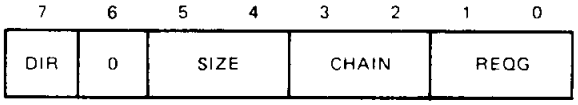
DPS (DEVICE PORT SIZE)
- 0 8 bit port
- 1 16 bit port

PCL (PERIPHERAL CONTROL LINE)
- 00 Status Input
- 01 Status Input with Interrupt
- 10 Start Pulse
- 11 Abort Input

Bit 2 Not Used

- **Operation Control Register (OCR)**

The OCR is an operation control register. The DIR bit defines the direction of the transfer. The SIZE bits define the size of the operand. The CHAIN bits define the type of the CHAIN mode. The REQG bits define how requests for transfers are generated.



DIR (DIRECTION)
- 0 Transfer from memory to device (transfer from MAR address to DAR address)
- 1 Transfer from device to memory (transfer from DAR address to MAR address)

SIZE (OPERAND SIZE)
- 00 Byte (8 bits)
- 01 Word (16 bits)
- 10 Long Word (32 bits)
- 11 See Note Below

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

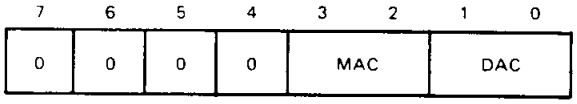
REQG (DMA REQUEST GENERATION METHOD)
- 00 Auto-request at transfer rate limited by General Control Register (Limited Rate Auto-Request)
- 01 Auto-request at maximum rate
- 10 $\overline{REQ}$ line requests an operand transfer
- 11 Auto-request the first operand, external request for subsequent operands

Bit 6 Not Used

NOTE: If the DMAC is set to dual addressing mode, port size 8 bits, external request mode, and the data transfer is from peripheral device to memory, set SIZE = 11 in the Operation Control Register (OCR).

- **Sequence Control Register (SCR)**

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



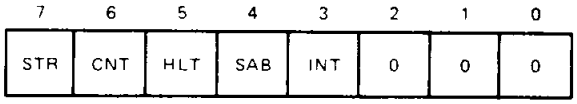
MAC (MEMORY ADDRESS COUNT)
- 00 Memory address register does not count
- 01 Memory address register counts up
- 10 Memory address register counts down
- 11 (undefined, reserved)

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

Bits 7, 6, 5, 4 Not Used

- **Channel Control Register (CCR)**

The CCR is used to start or terminate the operation of a channel. This register also determines if an interrupt request is to be generated. Setting the STR bit causes immediate activation of the channel; the channel will be ready to accept request immediately. The STR and CNT bits of the register cannot be reset by a write to the register. The SAB bit is used to terminate the operation forcedly. Setting the SAB bit will reset STR and CNT. Setting the HLT bit will halt the channel operation, and clearing the HLT bit will resume the operation. Setting start bit must be done by byte access. Otherwise, timing error occurs.



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

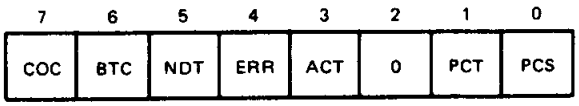
SAB (SOFTWARE ABORT)
- 0 Channel operation not aborted
- 1 Abort channel operation

INT (INTERRUPT ENABLE)
- 0 No interrupts enabled
- 1 Interrupts enabled

Bits 2, 1, 0 Not Used

- **Channel Status Register (CSR)**

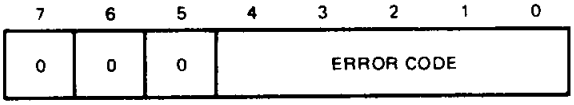
The CSR is a register containing the status of the channel.



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 by $\overline{\text{DONE}}$ input
1 Device terminated operation normally by $\overline{\text{DONE}}$ input
ERR (ERROR BIT)
0 No errors
1 Error as coded in CER
ACT (CHANNEL ACTIVE)
0 Channel not active
1 Channel active
PCT ($\overline{\text{PCL}}$ TRANSITION)
0 No $\overline{\text{PCL}}$ transition occurred
1 $\overline{\text{PCL}}$ transition occurred
PCS (THE STATE OF THE $\overline{\text{PCL}}$ INPUT LINE)
0 $\overline{\text{PCL}}$ "Low"
1 $\overline{\text{PCL}}$ "High"
Bit 2 Not Used

- **Channel Error Register (CER)**

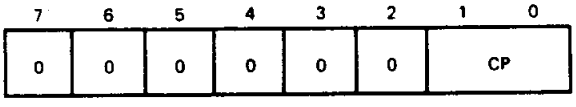
The CER is an error condition status register. The ERR bit of CSR indicates if there is an error or not. Bits 0-4 indicate what type of error occurred. *[See Attention on Usage, Note (3).]*



Error Code
00000 No error
00001 Configuration error
00010 Operation timing error
00101 Address error in MAR
00110 Address error in DAR
00111 Address error in BAR
01001 Bus error in MAR
01010 Bus error in DAR
01011 Bus error in BAR
01101 Count error in MTC
01111 Count error in BTC
10000 External abort
10001 Software abort
Bits 7, 6, 5 Not Used

- **Channel Priority Register (CPR)**

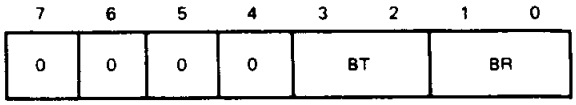
The CPR is used to define the priority level of the channel. Priority level 0 is the highest and priority level 3 is the lowest priority.



CP (CHANNEL PRIORITY)
00 Priority level 0
01 Priority level 1
10 Priority level 2
11 Priority level 3
Bit 7 through 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. GCR is also used to specify the hold time for cycle steal mode with hold.



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 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).

The hold time for cycle steal mode with hold is defined to be minimum of 1 sample interval and maximum of 2 sample intervals. A sample interval is defined to be $2^{(BT+BR+5)}$ (two to the BT + BR + 5 power) clock cycles.

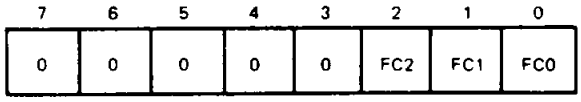
Bits 7 through 4 Not Used

- **Address Registers (MAR, DAR, BAR)**

Three 32-bit registers are utilized to implement the Memory Address Register, Device Address Register, and the Base Address Register. Only the least significant twenty-four bits are connected to the address output pins. The content of the MAR is outputted when the memory is accessed in single or dual adressing mode. The content of the DAR is outputted when the peripheral device is accessed. The contents of the BAR is outputted when reading chain information from memory in the Array Chaining Mode or the Linked Array Chaining Mode. It is also used to set the top address of the next block transfer in Continue mode.

- **Function Code Registers (MFC, DFC, BFC)**

The DMAC has three function code registers per channel: the Memory Function Code Register (MFC), Device Function Code Register (DFC), and the Base Function Code Register (BFC). The contents of these registers are outputted from FC_0 through FC_2 lines when an address is outputted from MAR, DAR, or BAR, respectively. The BFC is also used to set the MFC for the transfer of the next data block in the Continue mode.



Bits 3 through 7 Not Used

- **Transfer Count Registers (MTC, BTC)**

Each channel has two 16-bit counters: the Memory Transfer Counter (MTC) and the Base Transfer Counter (BTC). The MTC

counts the number of transfer words in one block, and is decreased by one for every operand transfer.

The BTC is used to count the number of data blocks in the Array Chaining Mode. BTC is also used to set the number of operands to transfer for the next data block in the Continue Mode.

- **Interrupt Vector Registers (NIV, EIV)**

Each channel has a Normal Interrupt Vector register and an Error Interrupt Vector register.

When an interrupt acknowledge cycle occurs, an interrupt vector is outputted from one of those registers. If the error bit (CSR) is set for the channel with interrupt pending, then content of EIV is outputted, otherwise content of NIV is outputted.

■ OPERATION DESCRIPTION

A DMAC channel operation proceeds in three principal phases. During the initialization phase, the MPU sets the channel control registers, supply the initial address and the number of transfer words, 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.

This section describes DMAC operations. A description of the MPU/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.

- **Read/Write of the DMAC Registers by MPU**

The MPU reads and writes the DMAC internal registers and controls the DMA transfer. Figure 11 indicates the timing diagram when the MPU reads the contents of the DMAC register. The MPU outputs A_1-A_{23}, FC_0-FC_2, $\overline{AS}$, R/$\overline{W}$, $\overline{UDS}$, and $\overline{LDS}$, and accesses the DMAC internal register. The specific internal register is selected by A_1-A_7, $\overline{LDS}$ and $\overline{UDS}$. The $\overline{CS}$ and $\overline{IACK}$ lines are generated by the external circuit with A_8-A_{23} and FC_0-FC_2. The DMAC outputs data on the data bus, together with $\overline{DDIR}$, $\overline{DBEN}$ and $\overline{DTACK}$. The $\overline{DDIR}$ and $\overline{DBEN}$ control the bidirectional buffer on the bus and the $\overline{DTACK}$ indicates that the data has been sent or received by the DMAC. Read Cycle is eighteen CLKs. Figure 12 shows the MPU write cycle. Write cycle is fifteen CLKs.

Note the following points.

(1) The clock reference shown in this figure is the DMAC input clock.
(2) The $\overline{DDIR}$ and the $\overline{DBEN}$ are three-stated at the beginning which detects $\overline{CS}$ and the ending of the cycle.
(3) During the MPU read cycle, the $\overline{DTACK}$ is asserted after the data is valid on the system bus.
(4) During the MPU write cycle, the $\overline{DDIR}$ line will be driven low to direct the data buffers toward to DMAC before the buffers are enabled.
(5) During the MPU write cycle, the DMAC will latch the data before asserting $\overline{DTACK}$. Then it will negate $\overline{DBEN}$ and $\overline{DDIR}$ in the proper order.
(6) After the MPU cycle and the $\overline{LDS}$ and the $\overline{UDS}$ are negated by the MPU, the DMAC will put $\overline{DBEN}$, $\overline{DDIR}$ and the address data lines to a high impedance state.
(7) $\overline{DTACK}$ will once go "High" and then to a high impedance state after negating $\overline{LDS}$ and $\overline{UDS}$.

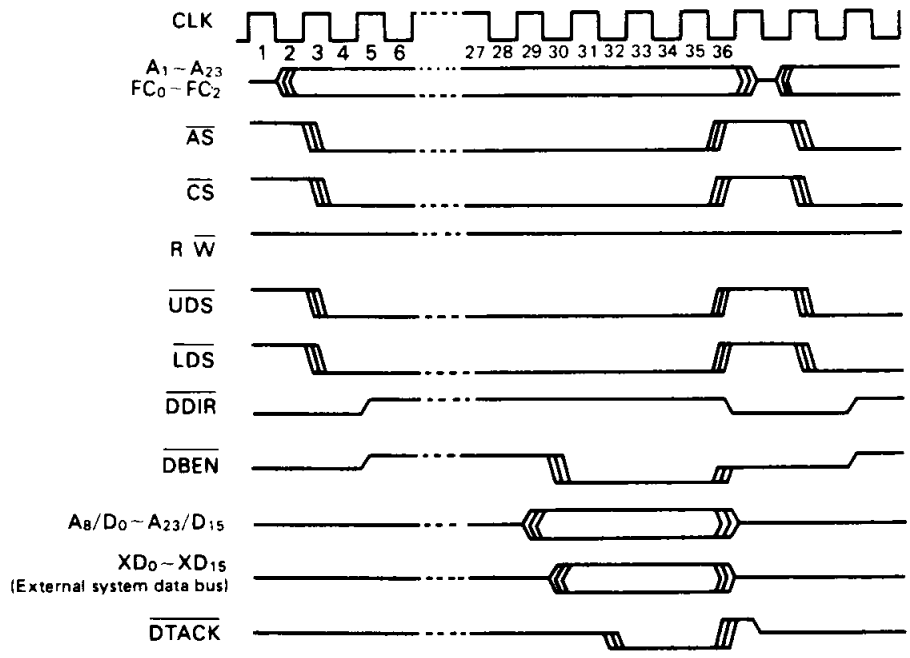


Figure 11 MPU Read from DMAC – Word

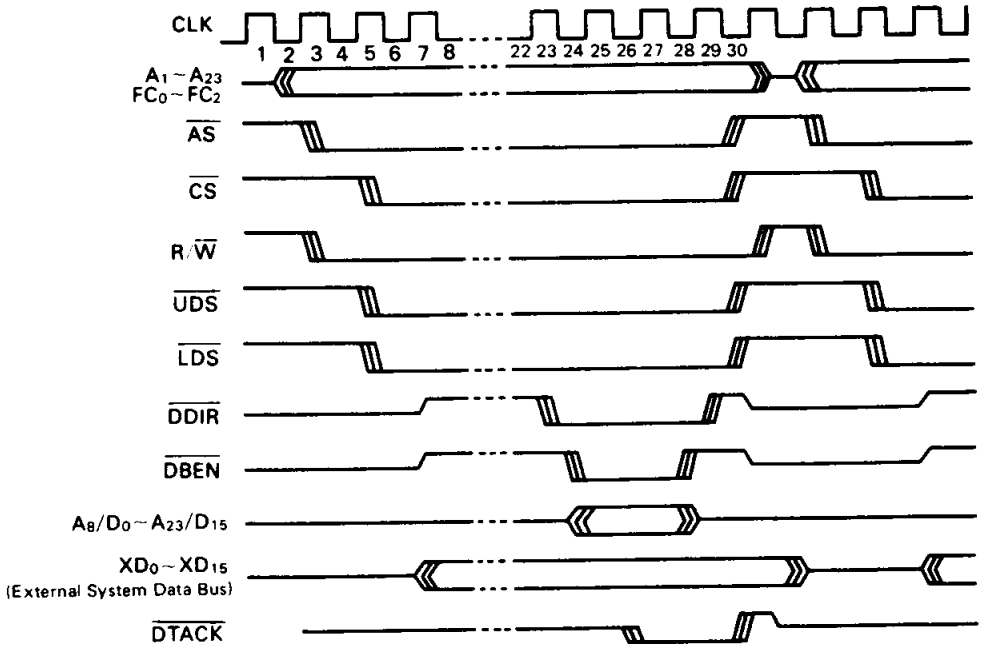
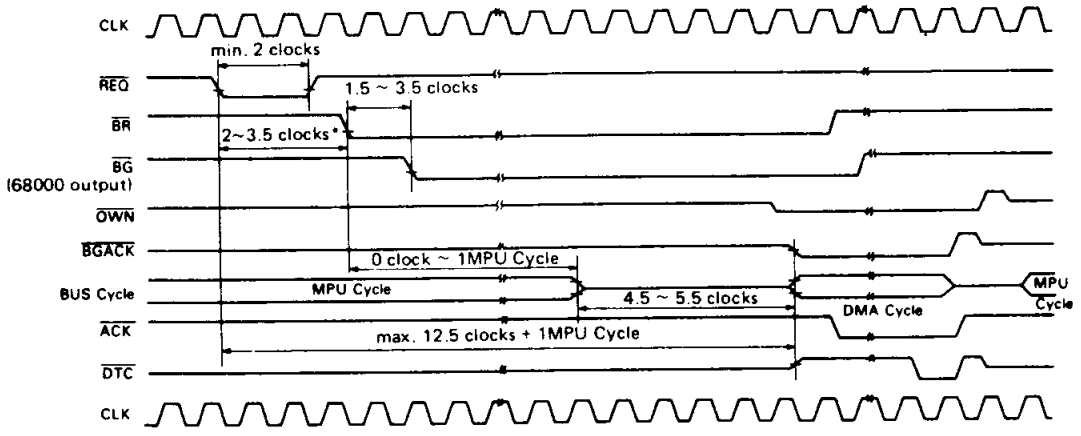


Figure 12 MPU Write to DMAC – Word

- **Bus Arbitration**

The following is the description of the bus arbitration. The DMAC must obtain the ownership of the bus in order to transfer. data. Figure 13 indicates the DMAC bus arbitration timing. It is completely compatible with that of HD68000 MPU. The DMAC asserts the Bus Grant (BG) to request the bus mastership. The MPU recognizes the request and asserts BG, then it grants the ownership in the next bus cycle. After the end of the current cycle (AS is negated), the MPU relinquishes the bus to the DMAC. The DMAC asserts the bus grant acknowledge (BGACK) to indicate that it has the bus ownership. A half clock before BGACK is asserted, the DMAC asserts OWN. OWN is kept asserted for a half clock after BGACK is negated at the end of the DMA cycle. BR is negated one clock after BGACK is asserted.



* This case assumes that no exception condition exists and DMAC isn't accessed by MPU.

Figure 13 DMAC Bus Arbitration Timing

- **Device/DMAC Communication**

Communication between peripheral devices and the DMAC is accomodated by five signal lines. Each channel has $\overline{REQ}$, $\overline{ACK}$ and $\overline{PCL}$, and the last two lines the $\overline{DONE}$ and $\overline{DTC}$ lines, are shared among the four channels.

(1) Request ($\overline{REQ}$)

The peripheral devices assert $\overline{REQ}$ to request data transfers. See the "Requests" section for details.

(2) Acknowledge ($\overline{ACK}$)

This line is used to implicitly address the device which is transferring the data (This device is not selected by address lines.) It is also asserted when the content of DAR is outputted during memory-to-memory transfer except for the auto-request mode at a limited rate or at the maximum rate.

(3) Peripheral Control Line ($\overline{PCL}$)

The function of this line is quite flexible and is determined by the DCR (Device Control Register).

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 $\overline{PCL}$ line is an Enable clock (E clock) input. If the DTYP bits are programmed to be a device with $\overline{READY}$, the PCL definition as ignored and the $\overline{PCL}$ line is a ready input.

$\overline{PCL}$ As a Status Input

The $\overline{PCL}$ line may be programmed as a status input. The status level of this line can be determined by the PCS bit in the CSR, regardless of the PCL function determined by the DCR. If a negative transition occurs and remains stable for a minimum of two clocks, the PCT bit of the CSR is set. This PCT bit is cleared by resetting the DMAC or the writing "1" to the PCT bit.

$\overline{PCL}$ As an Interrupt

The $\overline{PCL}$ line may 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. When using this function, it is necessary to reset the PCT bit in the CSR before the PCL bit in the DCR is set to interrupt, in order to avoid assertion of IRQ line at this time.

$\overline{PCL}$ As a Starting Pulse

The $\overline{PCL}$ line may be programmed to output a starting pulse. This active low starting pulse is outputted when a channel is activated, and is "Low" for a period of four clock cycles.

$\overline{PCL}$ As an Abort Input

The $\overline{PCL}$ line may be programmed to be a negative transition above input which terminates an operation by setting the external abort error in CER. It is necessary to reset the PCT bit in the CSR before activating the channel (Setting the ACT bit of CCR) so that the channel operation is not immediately aborted. *[See Attention on Usage, Note (2).]*

$\overline{PCL}$ As an Enable Clock (E Clock) Input

If the DTYP bits are programmed to be a HD6800 device, the PCL definition is ignored and the $\overline{PCL}$ 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 DMAC's clock.

$\overline{PCL}$ As a $\overline{READY}$ 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 $\overline{READY}$ input. The $\overline{READY}$ is an active low input.

(4) DONE ($\overline{DONE}$)

This line is an active low Input/Output signal with an open drain. It is asserted when the memory transfer count is exhausted in a single block transfer. In the chaining operation, $\overline{DONE}$ is asserted only at the last transfer to the peripheral device of the last data block. In the continue mode, $\overline{DONE}$ is asserted for each data block. It is asserted and negated in coincident with the $\overline{ACK}$ line for the last data transfer to the peripheral device. It is also outputted in coincident with the $\overline{ACK}$ line of the last bus cycle, in which the address is outputted from the DAR, in the memory-to-memory transfer (dual addressing mode) that uses the $\overline{ACK}$ line.

The DMAC also monitors the state of the $\overline{DONE}$ line during the DMA bus cycle. If the device asserts $\overline{DONE}$ during $\overline{ACK}$ active, the DMAC will terminate the operation after the transfer of the current operand. If $\overline{DONE}$ is asserted on the first byte of 2 byte operation or the first word of long word operation, the DMAC does not terminate the operation before the whole operand transfer is completed. If $\overline{DONE}$ is asserted, then the DMAC terminates the operation by clearing the ACT bit of the CSR, and setting the COC and NDT bits of the CSR. If both the DMAC and the device assert $\overline{DONE}$, the device termination is not recognized, but the channel operation does terminate. $\overline{DONE}$ is outputted again for the retry exceptions bus cycles.

(5) Data Transfer Complete ($\overline{DTC}$)

$\overline{DTC}$ is an active low signal which is asserted when the actual data transfer is accomplished. It is also asserted in the bus cycle which read a chain information from memory in the Chaining mode. However, if exceptions are generated and the DMA bus cycle terminates, $\overline{DTC}$ is not asserted. $\overline{DTC}$ is asserted one half clock before $\overline{LDS}$ and $\overline{UDS}$ are negated, and negated one half clock after $\overline{LDS}$ and $\overline{UDS}$ are negated.

- **Requests**

Requests may be externally generated by circuitry in the peripheral device, or internally generated by the auto-request mechanism. The REQG bits of the OCR determine these modes. The DMAC also supports an operation in which the DMAC auto-requests the first transfer and then wait for the peripheral device to request the following 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 and limited-rate. In the former case, the channel always has a request pending.

The limited rate auto-request functions by monitoring the bus utilization.

Limited-rate Auto-request

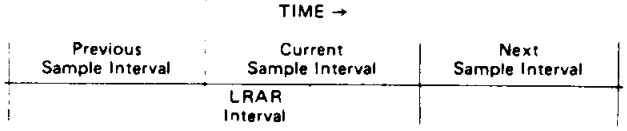


Figure 14 DMAC Sample Intervals

In the limited-rate auto-request the DMAC devides time into equal length sample intervals by counting clock cycles. The end of one sample interval makes the beginning of the next. During a sample interval, the DMAC monitors by means of $\overline{BGACK}$ pin the system bus activity of the DMAC and other bus master devices. At the end of the sample interval, decision is made whether or not to perform the channel's data transfer during the next sample interval. Namely, based on the activity of the DMAC or other bus master devices during the current 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 (the limited-rate auto-request interval) are controlled by the BT and BR bits in the GCR. The length in clock cycles of the limited-rate auto-request interval is $2^{(BT+4)}$ (2 raised to the BT+4 power). For example, if BT equals 2 and the DMA utilization of the bus was low during the previous sample interval, then the DMAC generates the auto-request transfers during the first 64 clock cycles.

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. The ratio of the system bus utilization of the MPU to other bus master devices including the DMAC is $2^{(BR+1)}$ (2 raised to the BR+1 power). If the fraction of DMA clock cycles during the sample interval exceeds the programmed utilization level, the DMAC will not allow limited-rate auto-requests during the next sample interval.

For example, if BR equals 3, then at most one out of 16 clock cycles during a sample interval can be used by the DMAC and other bus master devices, and still the DMAC would allow limited rate auto-request during the next sample interval. Therefore, from the viewpoint of long period, the ratio of the system bus utilization of the MPU to I/O devices including the DMAC is about 16:1. The sample interval length is not a direct parameter, but it is equal to $2^{(BT+BR+5)}$ clock cycles. Thus, the sample interval can be programmed between 32 and 2048 clock cycles.

The DMAC uses the $\overline{BGACK}$ to differentiate between the MPU bus cycle and DMAC or other bus master devices. If $\overline{BGACK}$ is active, then the DMAC assumes that the bus is used by a DMAC or other bus master devices. If it is inactive, then the DMAC assumes that it is used by the MPU.

Maximum-rate Auto-request

If the REQG bits in the OCR indicate auto-request at the maximum rate, the DMAC acquires the bus after the start bit is set and keeps it until the data transfer is completed.

If a request is made by another channel of higher priority, the DMAC services that channel and then resumes the auto-request sequence. If two or more channels are set to equal priority level and maximum rate auto-request, then the channels will rotate in a "round robbin" fashion.

If the HD68000 compatible device is connected to a channel, the $\overline{ACK}$ line is held inactive during an auto-request operation. Consequently, any channel may be used for the memory-to-memory transfer with the auto-request function in addition to the operation of data transfer between memory and peripheral device with using the $\overline{REQ}$ pin. Refer to Figure 15 for the timing of the memory-to-memory transfer. In this mode, the $\overline{ACK}$, $\overline{HIBYTE}$ and $\overline{DONE}$ outputs are always inactive.

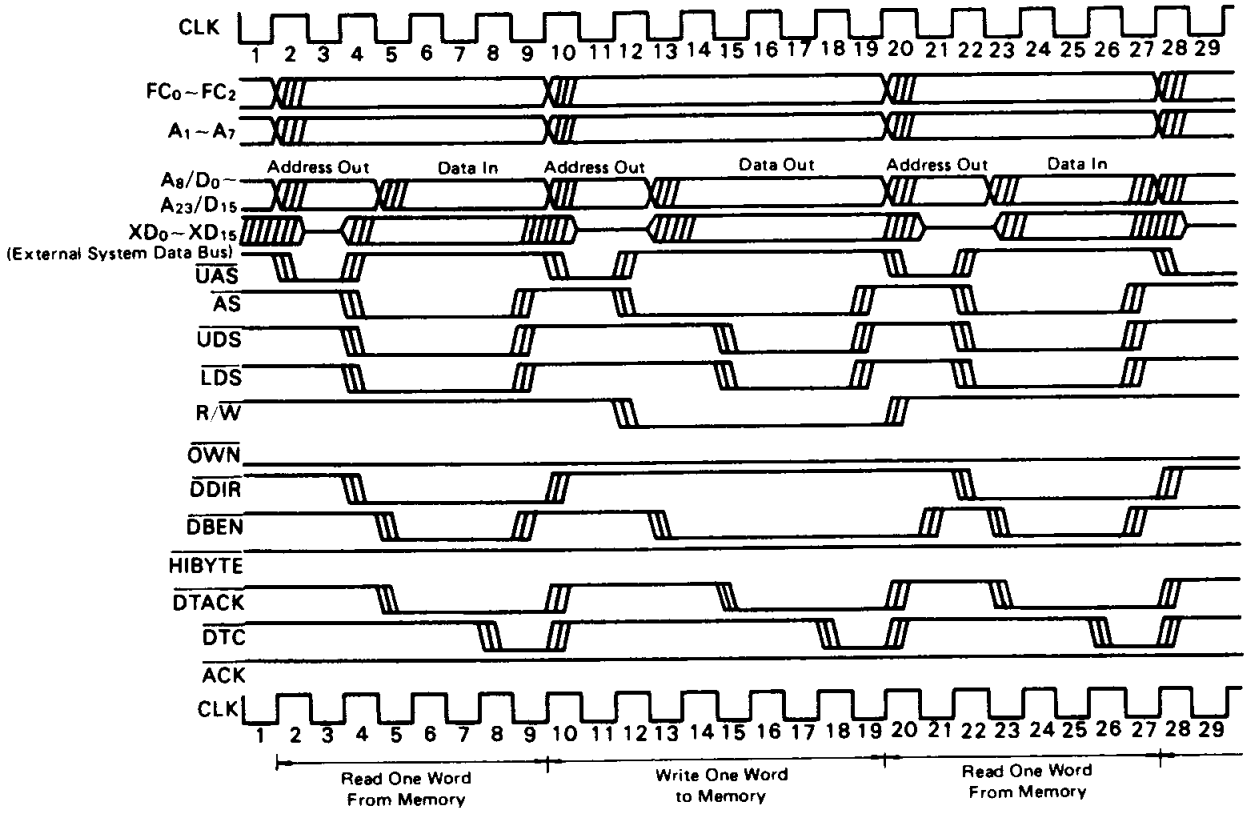


Figure 15 Memory-to-Memory Transfer Read-Write-Read Cycles

(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. The followings are the description of the burst and the cycle steal modes.

Burst Request Recognition

In the burst request mode, the $\overline{REQ}$ line is an active low input. The level sampled at the rising edge of the clock. Once the burst request is asserted, it needs to be held low until the first DMA bus cycle starts in order to insure at least one data transfer operation. In order to stop the burst mode transfer after the current bus cycle, the $\overline{REQ}$ line has to be negated one clock before the $\overline{DTC}$ output clock of this cycle. Refer to Figure 16 or the burst mode timing.

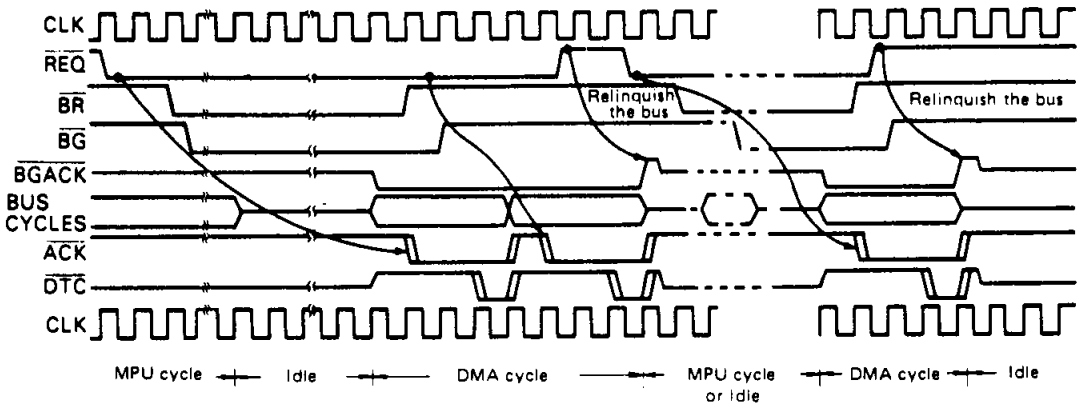


Figure 16 Burst Mode Request Timing
(Only one channel is active)

Cycle Steal Request Recognition

In the cycle steal request mode, the peripheral device requests the DMA transfer by generating an falling edge at the $\overline{REQ}$ line. The $\overline{REQ}$ line needs to be held "low" for at least 2 clock cycles. In the cycle steal mode, if the $\overline{REQ}$ line changes from "High" to "Low" between $\overline{ACK}$ output and one clock before the clock that outputs $\overline{DTC}$, then the next DMA transfer is performed without relinquishing the bus. If the bus is not relinquished, then maximum of 5 idle clocks is inserted between bus cycles. Refer to Figure 17 for the request timing of the cycle steal mode. If the XRM bits specify cycle steal without hold, the DMAC will relinquish the bus. 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. Figure 18 shows the request timing in the cycle steal bus hold. If the $\overline{REQ}$ is inputted during the hold time, the $\overline{ACK}$ is outputted after a maximum of 7.5 clock cycles from the picked-up clock. On the cycle steal with hold mode, the DMAC will hold the bus even when the transfer count is exhausted and the last data has been transferred. If DMA transfer is requested from other channels during this period, they are executed normally.

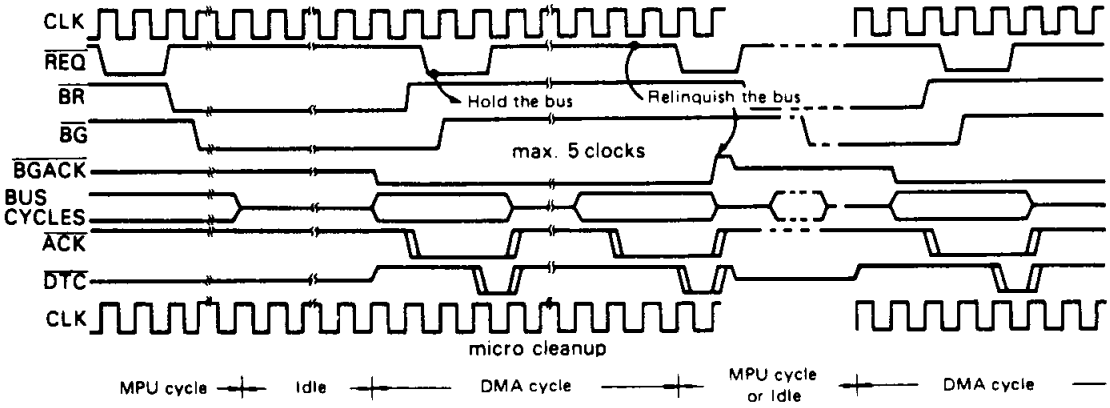


Figure 17 Cycle Steal Mode Request Timing

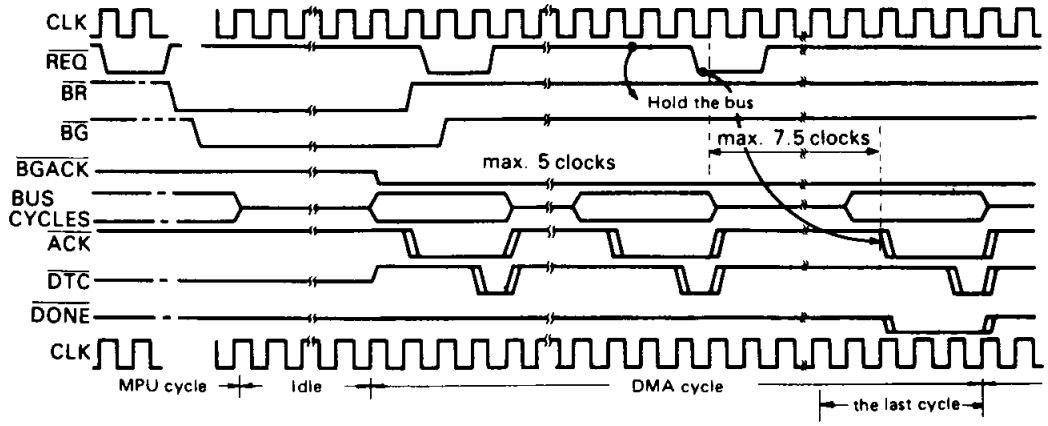


Figure 18 Cycle Steal Bus Hold Mode Request Timing

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. In the cycle steal request mode, when there are two or more than transfers between the DMAC and the peripheral device during one operand transfer, the request is not recognized until the last transfer between the DMAC and the I/O device starts.

(3) Mixed Request Generation

A single channel can mix the two request generation methods. By programming the REQG bits of the OCR to "11", 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.

- **Data Transfers**

All DMAC data transfers are assumed to be between memory and the peripheral device. The word "memory" means a 16-bit HD68000 bus compatible device. By programming the DCR, the characteristics of the peripheral device may be assigned. Each channel can communicate using any of the following protocols.

DTYP	Device Type	
00	HD68000 compatible device	Dual Addressing
01	HD6800 compatible device	
10	Device with $\overline{ACK}$	Single Addressing
11	Device with $\overline{ACK}$ and $\overline{READY}$	

(1) Dual Addressing

HD68000 and HD6800 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 called dual-addressed.

HD68000 Compatible Device Transfers

In this operation, when a request is received, the bus is obtained and the transfer is completed using the protocol as shown in Figures 19 and 20. Figures 21 through 24 show the transfer timings. Figure 21 and 24 show the operation when the memory is the source and the peripheral device is the destination. Figures 22 and 23 show the transfer in the opposite direction. The peripheral device is a 16-bit device in Figures 21 and 22, and a 8-bit device in Figures 23 and 24.

5

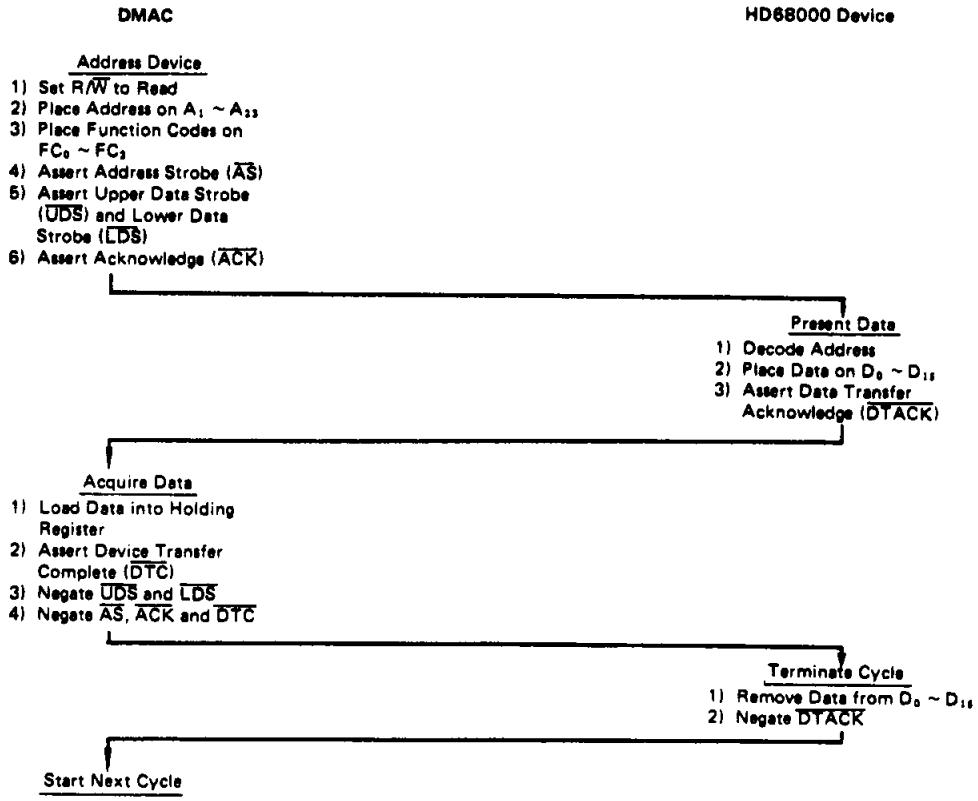


Figure 19 Word Read Cycle Flowchart HD68000 Type Device

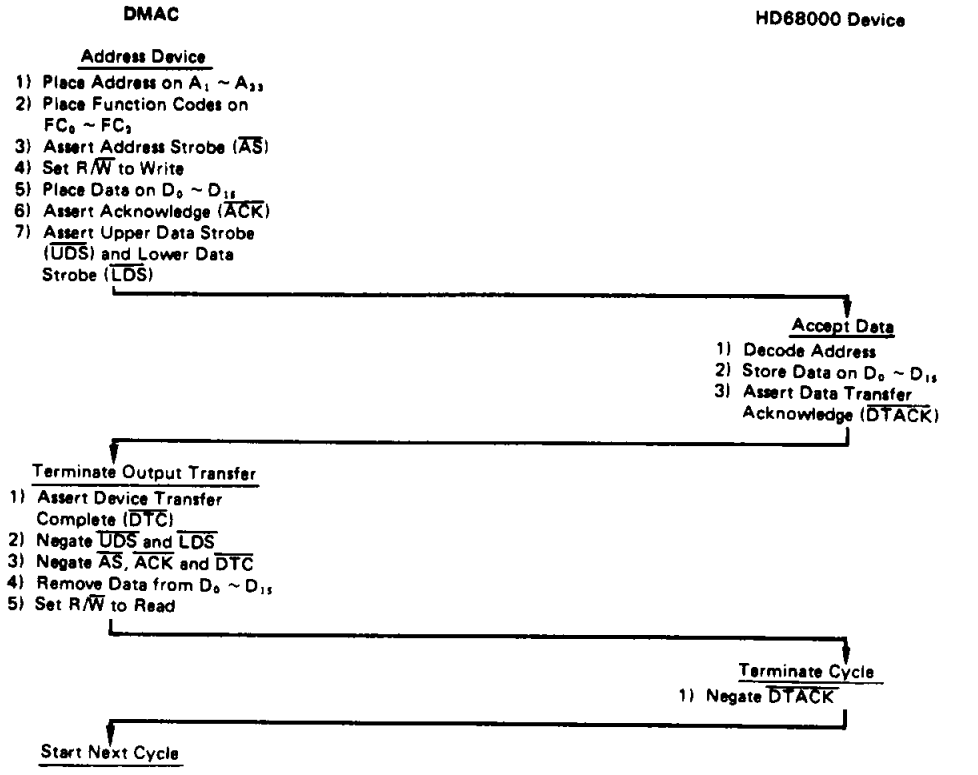


Figure 20 Word Write Cycle Flowchart HD68000 Type Device

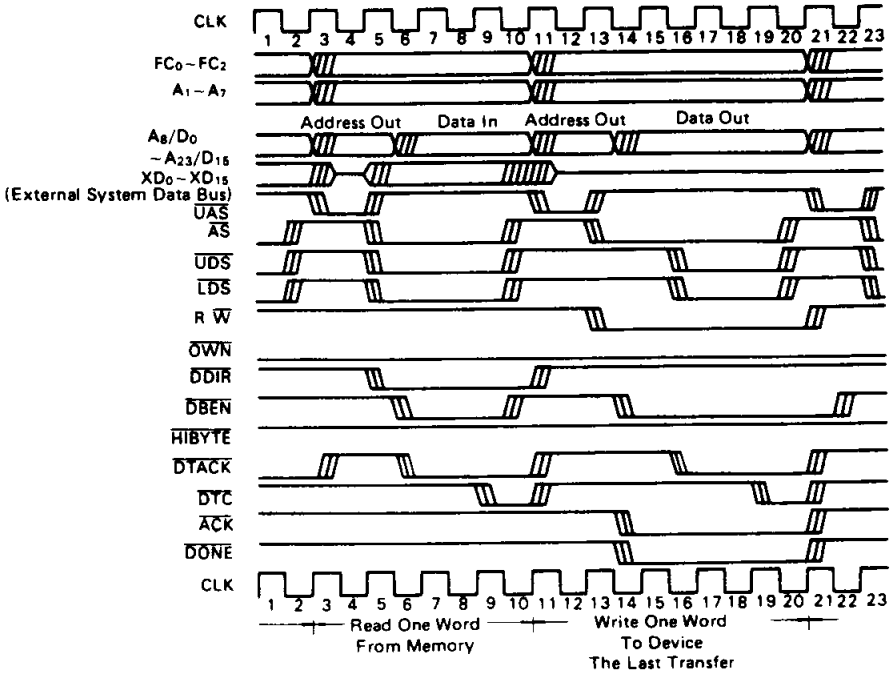


Figure 21 Dual Addressing Mode, Read/Write Cycle, Destination = 16-bit Device, Word Operand

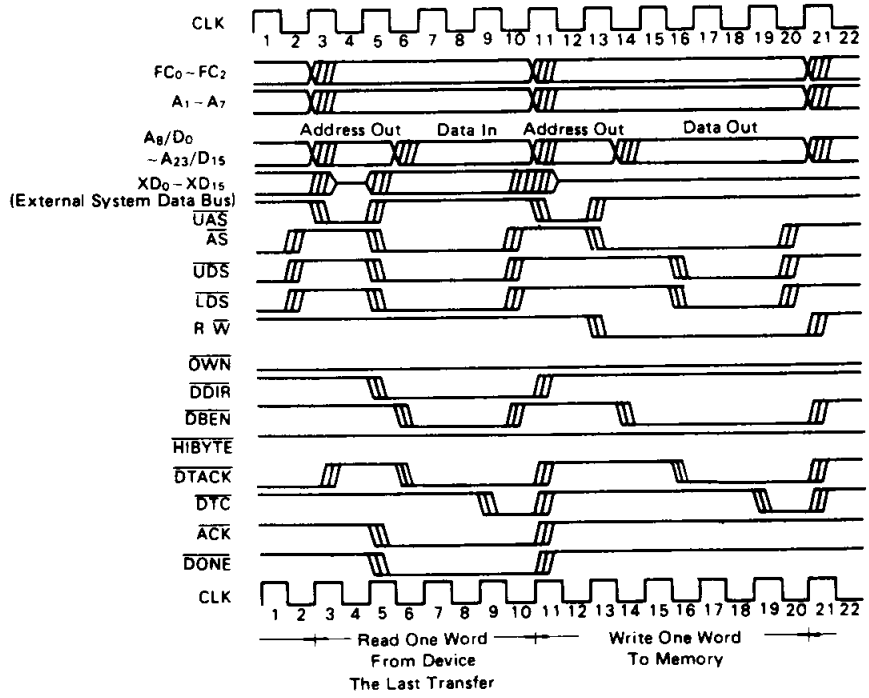


Figure 22 Dual Addressing Mode, Read/Write Cycle, Source = 16-bit Device, Word Operand

5

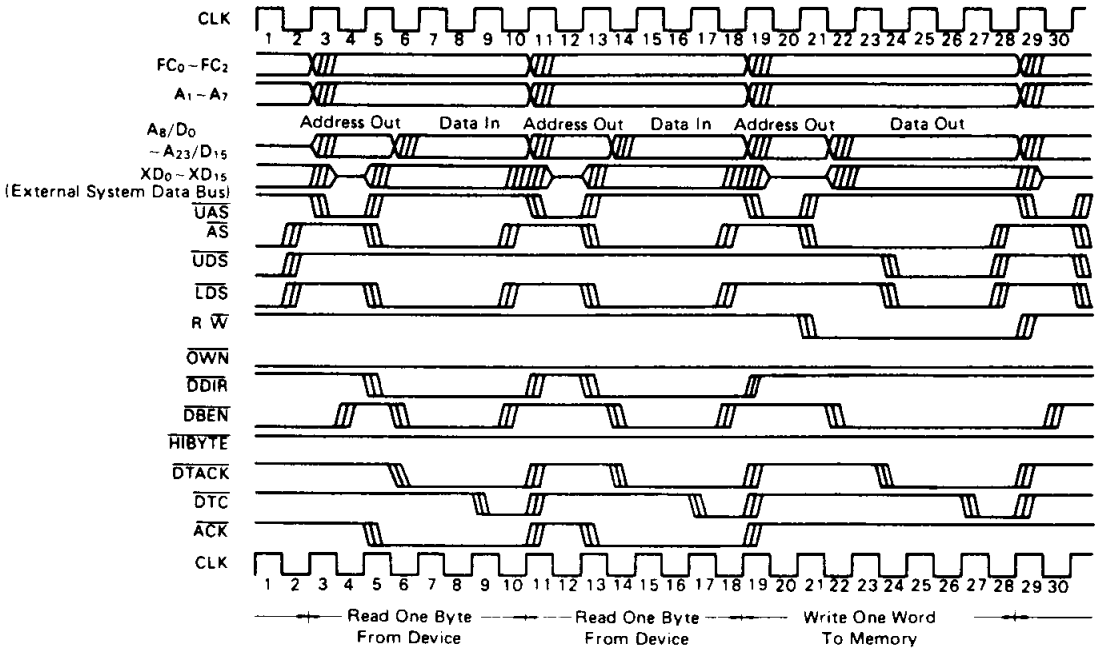


Figure 23 Dual Addressing Mode, Read/Write Cycle
Source = 8-bit Device, Word Operand

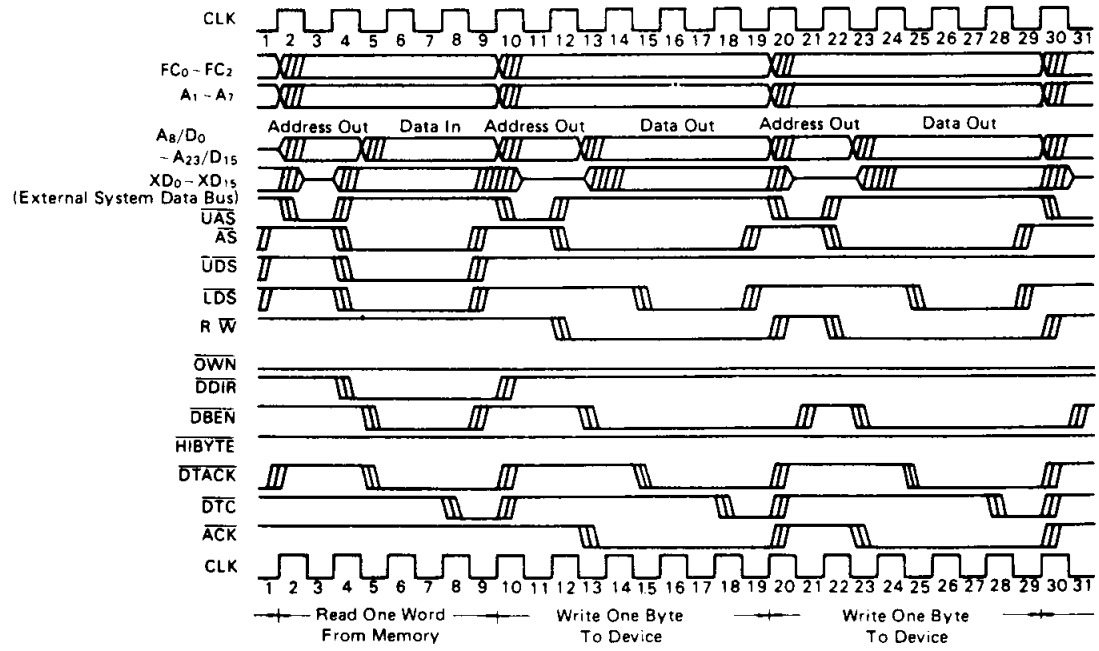


Figure 24 Dual Addressing Mode, Read/Write Cycle,
Destination = 8-bit Device, Word Operand

HD6800 Compatible Device Transfers

When a channel is programmed to perform HD6800 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 peripheral device using the HD6800 bus protocol, with the $\overline{ACK}$ output providing the $\overline{VMA}$ (valid memory address) signal. Figure 25 illustrates this protocol. Refer to Figure 26 for the read cycle timing and Figure 27 for the write cycle timing. In Figure 26, the DMAC latches the data at the falling edge of clock 19, so a latch to hold the data is necessary as shown in Figure 47.

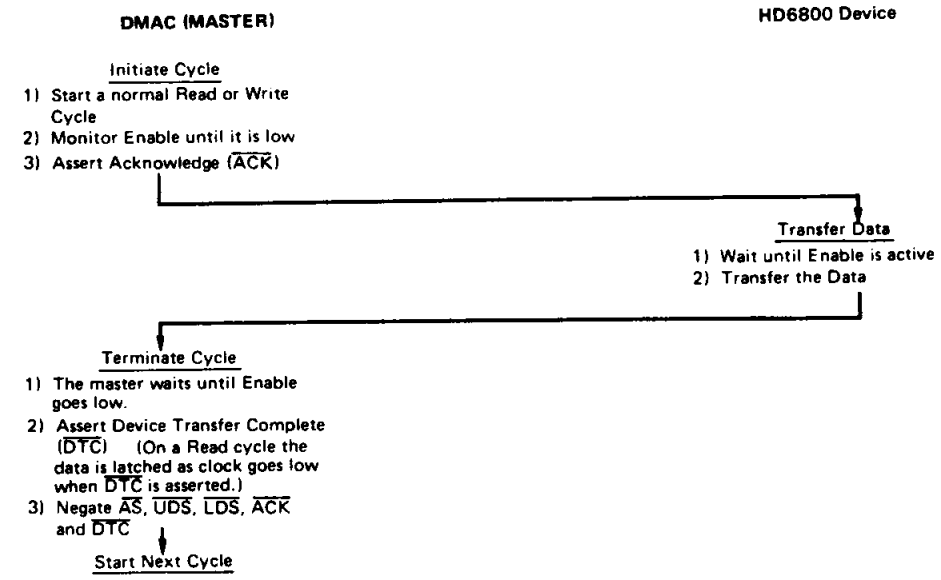


Figure 25 HD6800 Cycle Flowchart

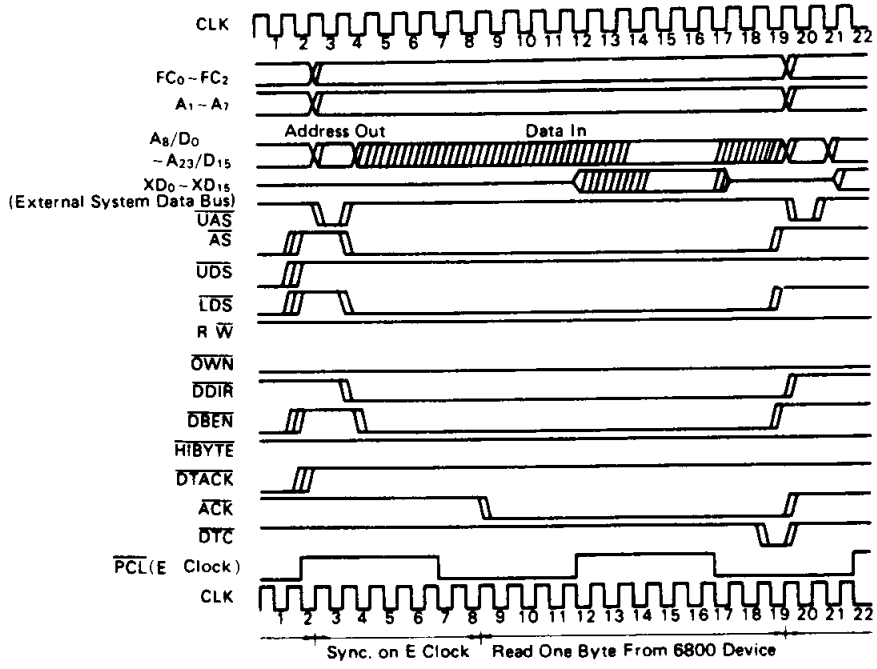


Figure 26 Dual Addressing Mode, HD6800 Compatible Device, Read Cycle

5

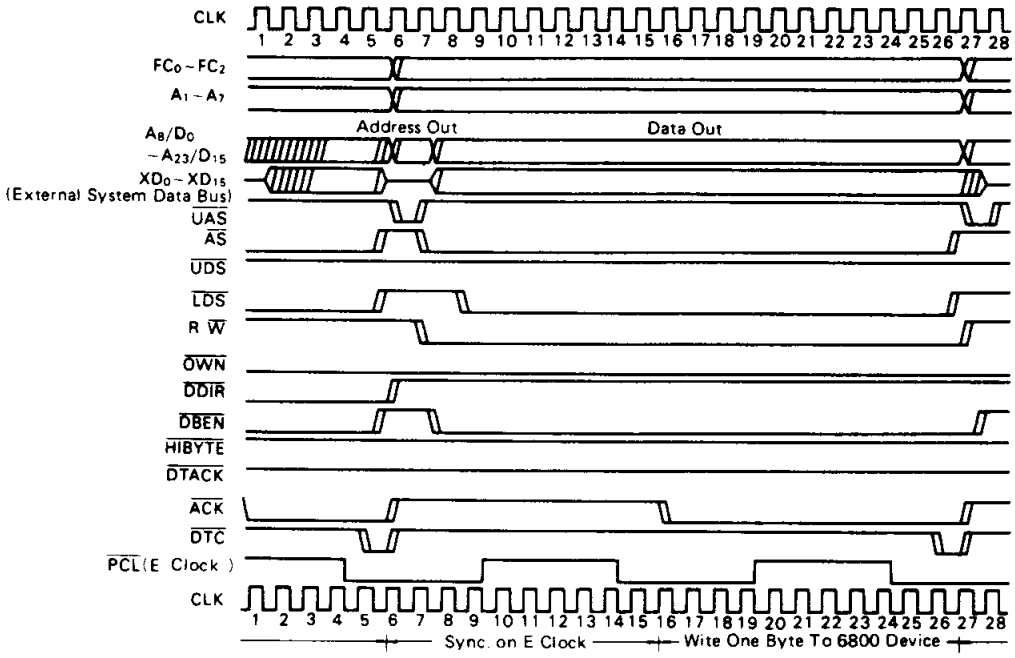


Figure 27 Dual Addressing Mode, HD6800 Compatible Device, Write Cycle

(2) Single Addressing Mode

Implicitly addressed devices are peripheral devices selected not by address but by $\overline{ACK}$. They do not require addressing of data register during data transfer. Transfers between memory and these devices are controlled by the request/acknowledge protocol. Such peripherals require only one bus cycle to transfer data, and the DMAC internal holding register is not used. Because only the memory is addressed during a data transfer and a transfer done in only on bus cycle, this protocol is called single-address.

Device with $\overline{ACK}$ Transfers

Under this protocol, the communication between peripheral device and the DMAC is performed with a two signal $\overline{REQ}/\overline{ACK}$ handshake. When a request is generated using the request method programmed in the DMAC's internal control registers, the DMAC obtains the bus and responds with $\overline{ACK}$. The DMAC asserts all the bus control signals required for the memory access. Refer to Figure 28 for the flowchart of the data transfer from memory to the device with $\overline{ACK}$. Figure 29 shows the flowchart of the data transfer from the device with $\overline{ACK}$ to memory. 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 HD68000 bus control signals needed for the transfer. When the DMAC accepts $\overline{DTACK}$ from memory, it asserts $\overline{DTC}$ and informs the peripheral device of the transfer termination. Figure 30 and 31 show the transfer timings of the device with $\overline{ACK}$: the port size for the former figure is 8-bit and the latter is 16-bit respectively.

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.

When the transfer is from device to memory, data must be valid on the HD68000 bus before the DMAC asserts the data strobes. The data strobes are asserted one clock period after $\overline{ACK}$ is asserted. When the DMAC obtains the bus and starts a DMA cycle, the tri-state of the $\overline{OWN}$ line is cancelled a half clock earlier than other control lines. If the DMA Cycle terminates and the DMAC relinquishes the bus, all the control signals get tri-stated a half clock before $\overline{OWN}$. The $\overline{DDIR}$ and $\overline{DBEN}$ lines are not asserted in the single addressing mode. Four clocks cycle is the smallest bus cycle for the transfer from memory to device. Five clocks cycle is the smallest bus cycle for the transfer from device to memory. If the device port size is 8-bit, either $\overline{LDS}$ or $\overline{UDS}$ is asserted. In the single addressing mode, A_x-A_{23} are outputted for only one and a half clock from the beginning of the DMA bus cycle. Therefore A_x through A_{23} needs to be latched externally just like in the dual addressing mode.

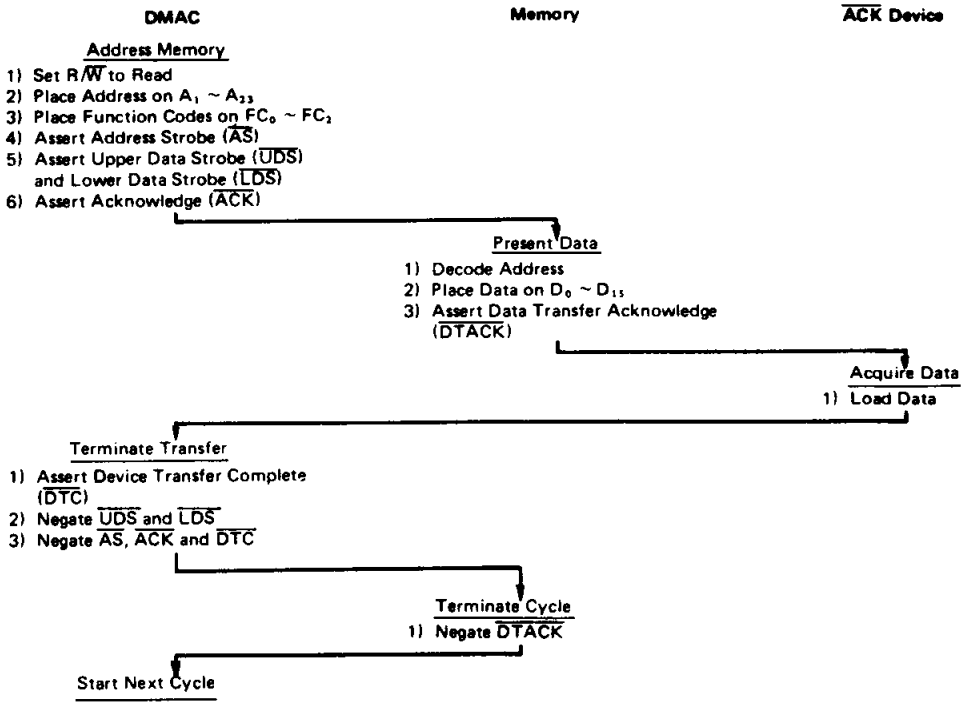


Figure 28 Word from Memory to Device with $\overline{ACK}$

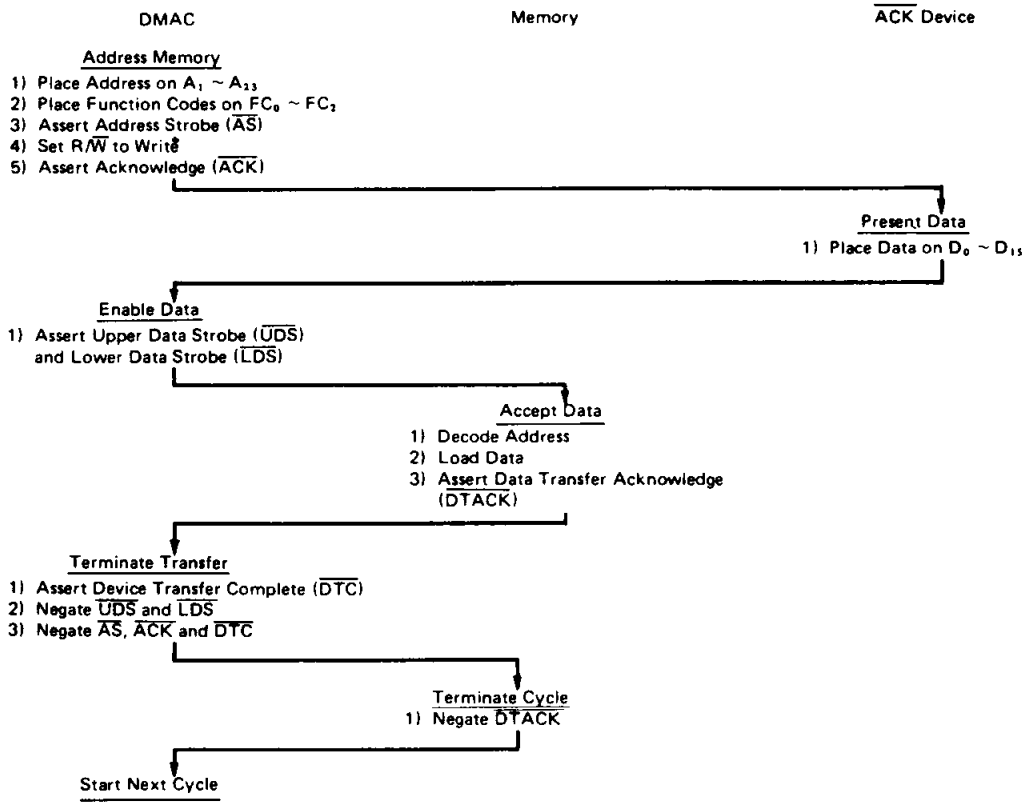


Figure 29 Word from Device with $\overline{ACK}$ to Memory

5

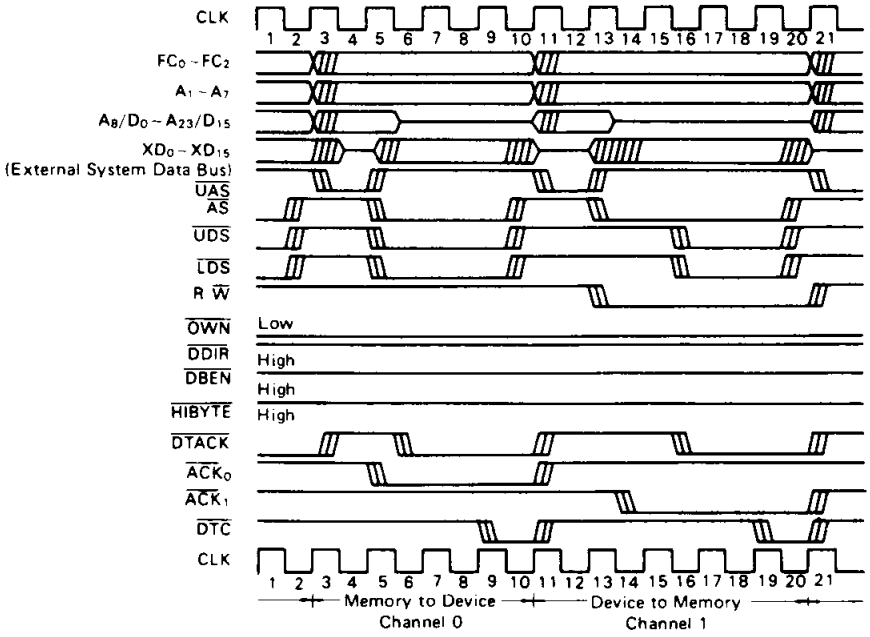


Figure 30 Single Addressing Mode with 16-Bit Devices as Source and Destination (Read-Write Cycles)

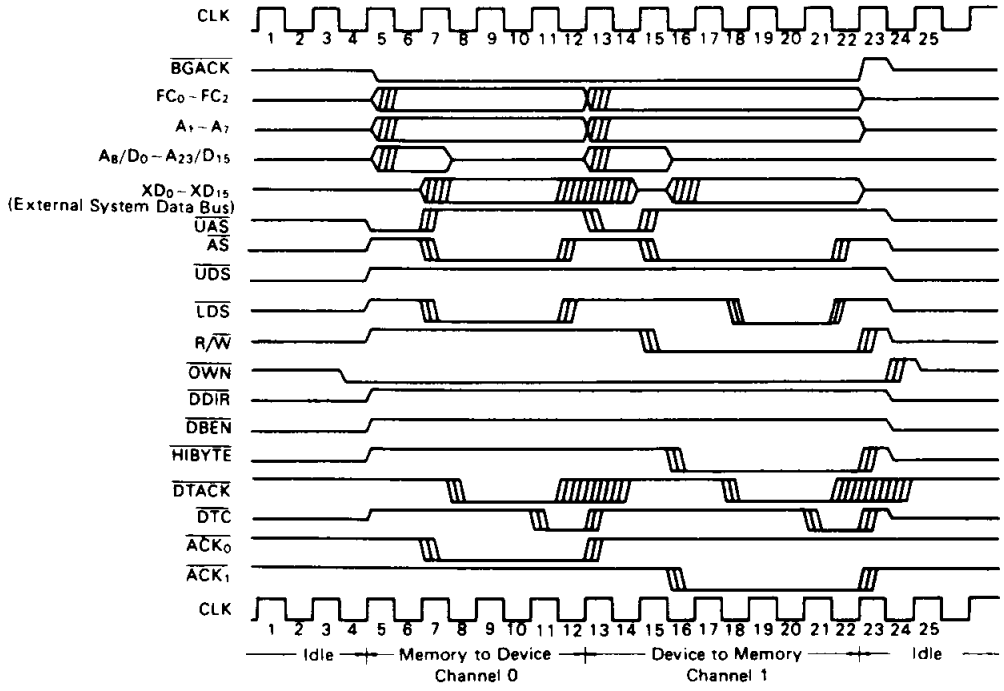


Figure 31 Single Addressing Mode with 8-Bit Device as Source and Destination (Read-Write Cycles)

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

Under this protocol, the communication between peripheral device and the DMAC is performed using a three signal $\overline{REQ}/\overline{ACK}/\overline{READY}$ handshake. The $\overline{READY}$ input to the DMAC is provided by the $\overline{PCL}$ line. The $\overline{READY}$ line is active low. When a request is generated using the request method programmed in the control registers, the DMAC obtains the bus and asserts $\overline{ACK}$ to notify the device that the transfer is to take place. The DMAC waits for $\overline{READY}$ ($\overline{PCL}$ input), which is a response from the device, in addition to $\overline{DTACK}$ which is a response from memory.

When the DMAC accepts both signals, it terminates the transfer. Refer to Figures 33 and 34 for the flowcharts of the data transfer between memory and the device with $\overline{ACK}$ and $\overline{READY}$. Refer to Figure 35 for the transfer timing of the 8-bit device. When the data transfer is from memory to a device, data is valid from the assertion of $\overline{DTACK}$ to the negation of $\overline{LDS}$ and $\overline{UDS}$. $\overline{DTC}$ is asserted a half clock before $\overline{LDS}$ and $\overline{UDS}$ are negated, so this line may be used for latching the data by the peripheral device. In this case, $\overline{READY}$ ($\overline{PCL}$ input) indicates that the device has received the data. Both $\overline{DTACK}$ and $\overline{READY}$ ($\overline{PCL}$ input) signals are needed for terminating the DMA cycle.

When the data transfer is from the device to memory, data must be valid on the bus before the DMAC asserts $\overline{LDS}$ and $\overline{UDS}$. Therefore, $\overline{READY}$ ($\overline{PCL}$ input) is used as the signal to indicate that the peripheral device has outputted the data on the bus. When the DMAC detects PCL (READY input), then it asserts $\overline{LDS}$ and $\overline{UDS}$. After asserting $\overline{LDS}$ and $\overline{UDS}$, the DMAC terminates the cycle when $\overline{DTACK}$ signal from the memory is detected.

When Array Chain or Link Array Chain is set in Device with $\overline{ACK}$ and $\overline{READY}$ Transfer mode, $\overline{READY}$ input is also necessary during DMA bus cycles for reading the chain information from memory. The circuit as shown in Figure 32 may be used in order to generate $\overline{READY}$ input when reading the chain information from memory.

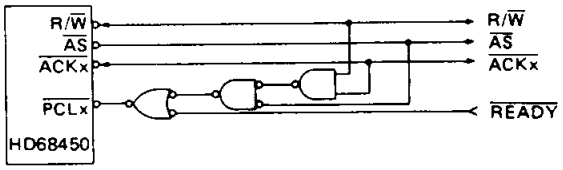


Figure 32 $\overline{READY}$ Circuit When Array or Link Array Chain is set for Device with $\overline{ACK}$ and $\overline{READY}$

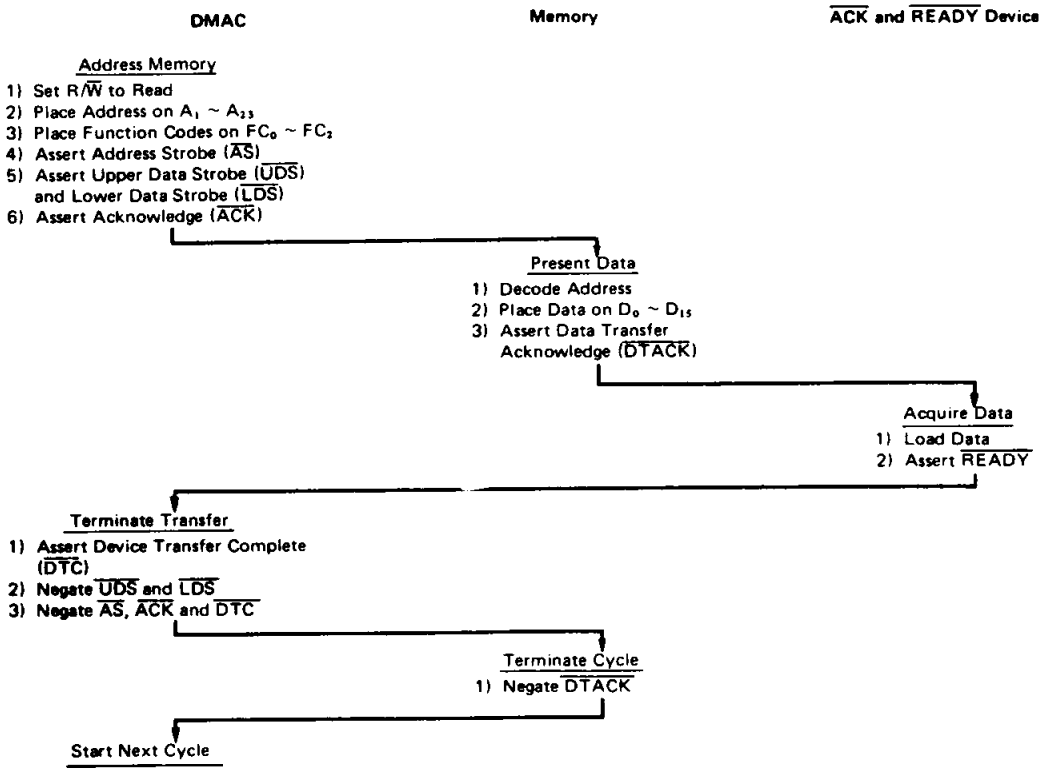


Figure 33 Word from Memory to Device with $\overline{ACK}$ and $\overline{READY}$

5

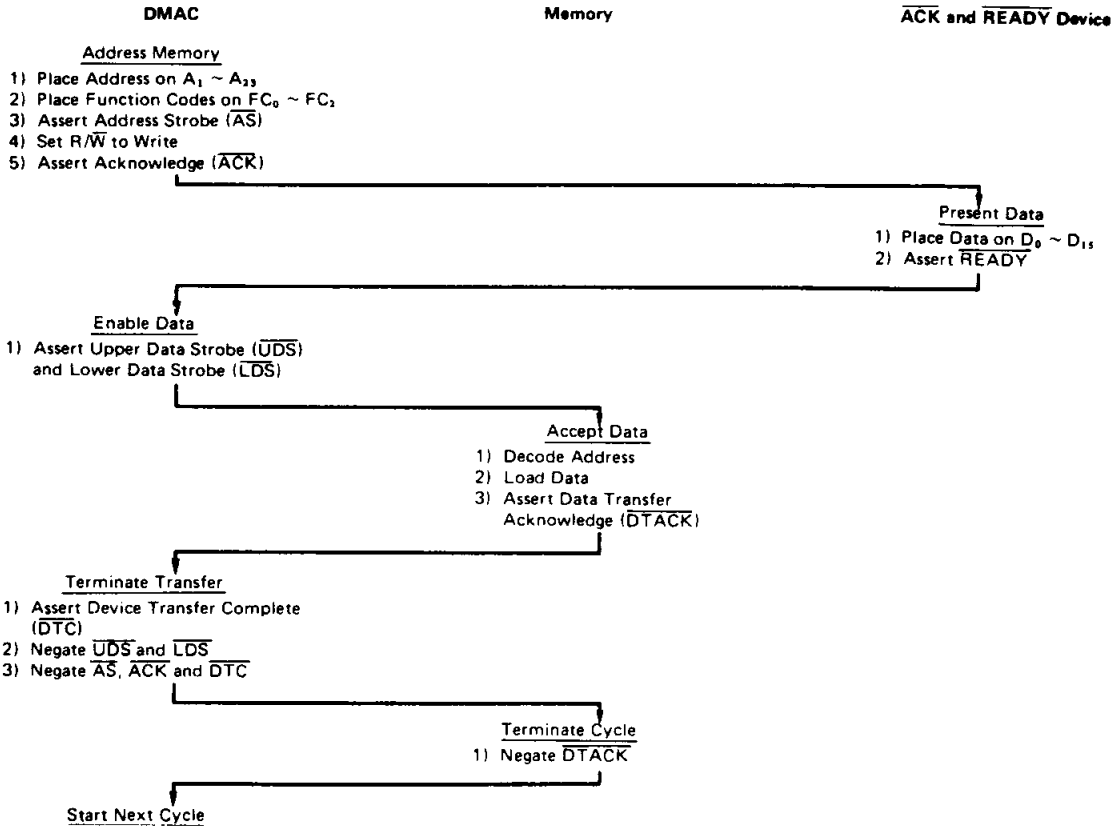


Figure 34 Word from Device with $\overline{ACK}$ and $\overline{READY}$ to Memory

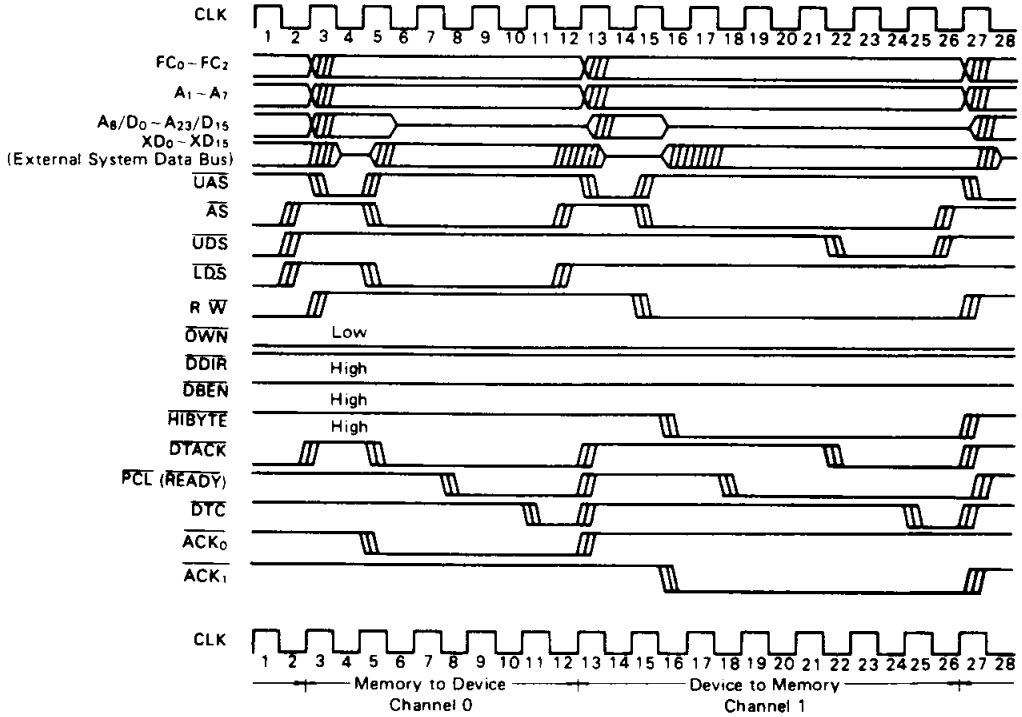


Figure 35 Single Addressing Mode with 8-Bit Devices as Source and Destination with $\overline{PCL}$ Used as a $\overline{READY}$ Input (Read-Write Cycles)

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.

Table 2 indicates the combinations supported by the DMAC about the peripheral devices with different port size and operand sizes in the single and dual addressing mode. In the single addressing mode, port size and operand size must be the same. In the dual addressing mode, byte operand cannot be used when the port size is sixteen and the REQG bit is 10 or 11.

Table 2 Operation Combinations

Addressing	Device Type	Port	Operand			REQG bits of OCR
			Byte	Word	Long Word	
Dual	68000, 6800	8	○	○	○	00, 01, 10, 11
Dual	68000, 6800	16	○	○	○	00, 01
Dual	68000, 6800	16	×	○	○	10, 11
Single	with $\overline{ACK}$ or $\overline{ACK}$ & $\overline{READY}$	8	○	×	×	00, 01, 10, 11
		16	×	○	×	00, 01, 10, 11

○ ; enable × ; disable

(3) 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 executed in the single addressing mode or the dual addressing mode. The memory address count method and the peripheral device address count method is programmed using the Memory address count (MAC) bit and the Device address count (DAC) bit in the Sequence Control Register (SCR).

(i) Single addressing mode

In the single addressing mode, memory address sequencing is shown in Table 3. 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 3 Single Address Sequencing

Port Size	Operand Size	Memory Address Increment		
		+ (increment)	= (unchanged)	- (decrement)
8	Byte	+1	0	-1
16	Word	+2	0	-2

5

(ii) Dual addressing mode

In the dual addressing mode, the operand size need not match the port size. Thus the transfer of an operand may require several DMA bus cycles. Each DMA bus cycle, between memory and DMAC and between DMAC and the device, is called the operand part and transfers a portion or all of the operand. The addresses of the operand parts are in a linear increasing sequence. The step between the addresses of the operand is two. The size of the operand parts is the minimum of the port size and the operand size. The number of the operand part is the operand size divided by the port size. In the dual addressing mode, memory is regarded as a device whose port size is 16-bits.

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. The dual address sequencing is shown in Table 4. *[See Attention on Usage, Note (4).]*

Table 4 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

Pack = byte if packing is not done
= word if packing is done

An Example of a Dual Address Transfer

This section contains an example of a dual address transfer using Table 4 of Dual-Address Sequencing. The table is reproduced here as Table 5. 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 5, 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 5. Table 6 shows the data transfer sequence.

The memory map of this example is shown in Table 7. The operand consists of BYTE A through BYTE D in memory of Table 7. 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 6.

Table 5 Dual-Address Sequencing (Table 4)

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

* Numbers in Table 5 correspond to ones in Table 6 and 7.
** Refer to Address Sequencing on Operand Part Size and PACK.

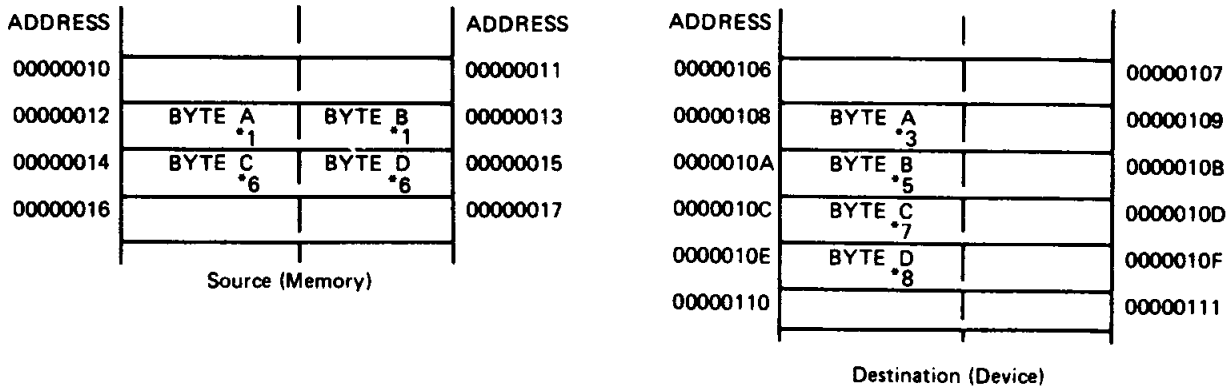
Table 6 An Example of a Data Transfer for One Operand

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 7 Memory Map for the Example of the Data Transfer



• Initiation and Control of Channel Operation

(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 resetting 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 MTC is set to zero in any operation or BTC is set to zero in the array chaining mode, then the count error is signaled and the channel is not activated. 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.

(2) Operation Continuation (Continue Mode)

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.

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.

(3) Operation Halting (Halt)

5

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. However, in the burst request mode, the transfer request should be kept asserted until the initiation of the first transfer after clearing the halt bit.

(4) Operation Abort by Software (Software Abort)

Setting the software abort bit (SAB) in the CCR allows the current operation of the channel to be aborted. In this case, the ERR bit and the COC bit in the CSR are set and the ACT bit is reset. The error code for the software abort is set in the CER. The SAB bit is designed to be reset if the ERR bit is reset. 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. If INT is set, the channel can request interrupts. If it is clear, the channel will not request interrupts.

- **Channel Operation Termination**

As part of the transfer of an operand, the DMAC decrements the memory transfer counter (MTC). If the chaining mode is not used and the CNT bit is not set or the last block is transferred in the chaining mode, the operation of the channel is complete when the last operand transfer is completed and the MTC is zero. The DMAC notifies the peripheral device of the channel completion via the $\overline{DONE}$ output.

However, in the continue mode, $\overline{DONE}$ is outputted at the termination of every data block transfer. When the channel operation has been completed, the ACT bit of the CSR is cleared, and the COC bit of the CSR is set.

The occurrence of errors, such as the bus error, during the DMA bus cycle also terminates the channel operation. In this case, the ACT bit in the CSR is cleared, the ERR and the COC bits are set, and at the same time the code corresponding to the error that occurred is set in the CER.

(1) Channel Status Register (CSR)

The channel status register contains the status of the channel. The register, except for ACT and PCS bits, 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. ACT and PCS bits are unaffected by the write operation.

COC

The channel operation complete (COC) bit is set if the channel operation has completed. The COC bit must be cleared in order to start another channel operation. The COC bit is cleared only by writing a one to this bit or resetting the DMAC.

PCS

The peripheral status (PCS) bit reflects the level of the $\overline{PCL}$ line regardless of its programmed function. If $\overline{PCL}$ is at "High" level, the PCB bit reads as one. If $\overline{PCL}$ is at "Low" level, the PCS bit reads as zero. The PCS bit is unaffected by writing to the CSR.

PCT

The peripheral control transition (PCT) bit is set, if a falling edge transition has occurred on the $\overline{PCL}$ line. (The $\overline{PCL}$ line must remain at "low" level for at least two clock cycles.) The PCT bit is cleared by writing a one to this bit or resetting the DMAC.

BTC

Block transfer complete (BTC) bit is set when the continue (CNT) bit of CCR is set and the memory transfer counter (MTC) is exhausted. The BTC bit must be cleared before the another continuation is attempted (namely, setting the CNT bit again), otherwise an operation timing error occurs. The BTC bit is cleared by writing a one to this bit or resetting the DMAC.

NDT

Normal device termination (NDT) bit is set when the peripheral device terminates the channel operation by asserting the $\overline{DONE}$ line while the peripheral device was being acknowledged. The NDT bit is cleared by writing a one to this bit or resetting the DMAC.

ERR

Error (ERR) bit is set if any errors have been signaled. When the ERR bit is set, the code corresponding to the kind of the error that occurred is set in the CER. The ERR bit is cleared by writing a one to this bit or resetting the DMAC.

ACT

The active (ACT) bit is asserted after the STR bit has been set and the channel operation has started. This bit is remains set until the channel operation is terminated. The ACT bit is unaffected by write operations. This bit is cleared by the termination of the channel or resetting the DMAC.

(2) Interrupts

The DMAC can signal the termination of the channel operation by generating an interrupt request. The INT bit of the CCR determines if an interrupt can be generated. The interrupt request is generated by the following condition.

① INT = 1
and
② COC = 1 or BTC = 1 or ERR = 1 or NDT = 1 or PCT = 1
(the $\overline{PCL}$ line is an interrupt input)

This may be represented as

$$\overline{IRQ} = INT \cdot (COC + BTC + ERR + NDT + PCT^*)$$

(*$\overline{PCL}$ line is programmed as an interrupt input.)

When the $\overline{IRQ}$ line is asserted, changing the INT bit from one to zero to one will cause the $\overline{IRQ}$ output to change from "low" to "high" to "low" again. The $\overline{IRQ}$ should be negated by clearing the COC, the BTC, the ERR, the NDT and the PCT bits.

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

The bus cycle in which the DMAC provides the interrupt vector when receiving an $\overline{IACK}$ from the MPU is called the interrupt acknowledge cycle. 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 occurs, all interrupt vector registers are set to $0F (binary 00001111), the value of the uninitialized interrupt vector. The timing of the interrupt acknowledge cycle is shown in Figure 36. The HD68000 MPU outputs the interrupt level into A_1-A_3 and A_4-A_7 is held "high" during the interrupt acknowledge cycle, but the HD68450 DMAC ignores these signals.

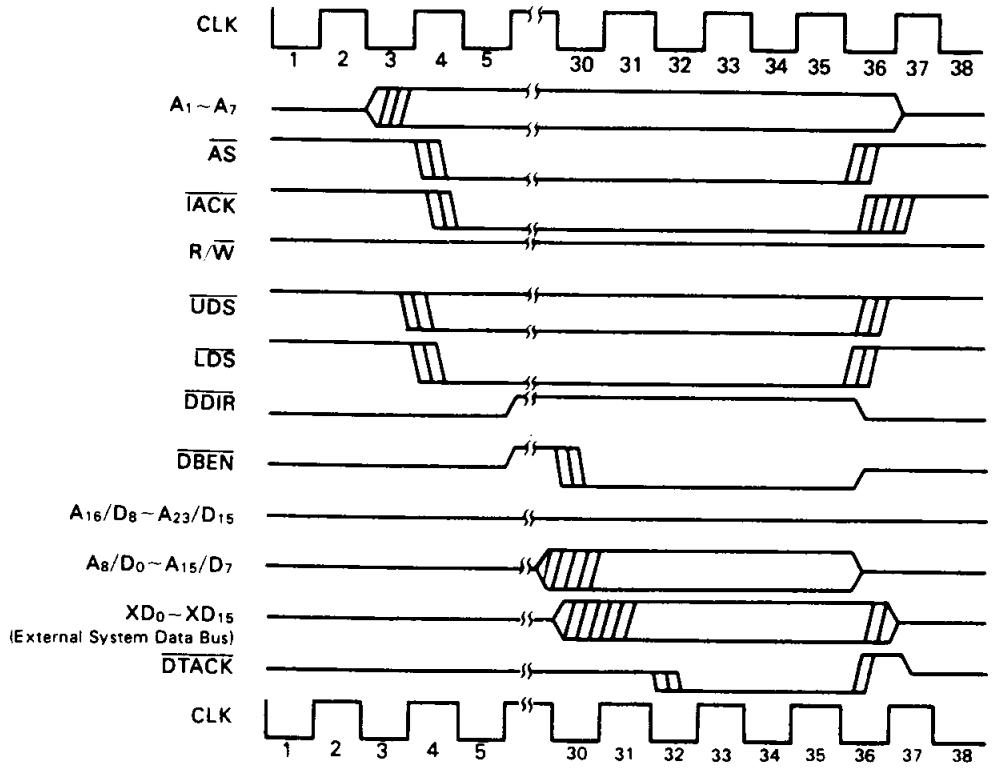


Figure 36 MPU $\overline{IACK}$ Cycle to DMAC

(3) Multiple Data Block Transfer Operation

When the memory transfer counter (MTC) is exhausted, the channel operation still continues if the channel is set to the array chaining mode or the linked array chaining mode and the chain is not exhausted. The channel operation also containes if the continue bit (CNT) of the CCR is set. The DMAC provides the initialization of the memory address register and the memory transfer counter in these cases so that the DMAC can transfer the multiple blocks.

Continued Operation

The continued operation is described in the Initiation and the Control of the Channel Operation section.

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 of zero, the chain is exhausted, and the entry just fetched determines the last block of the channel operation.

An example of the array chaining mode operation and the memory format for supporting for array chaining is shown in Figure 37. 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 or the base 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. Contents of the BFC is outputted as the function code when the DMAC is accessing the memory using the base address register. The value of the function code registers are unchanged in the array chaining operation.

5

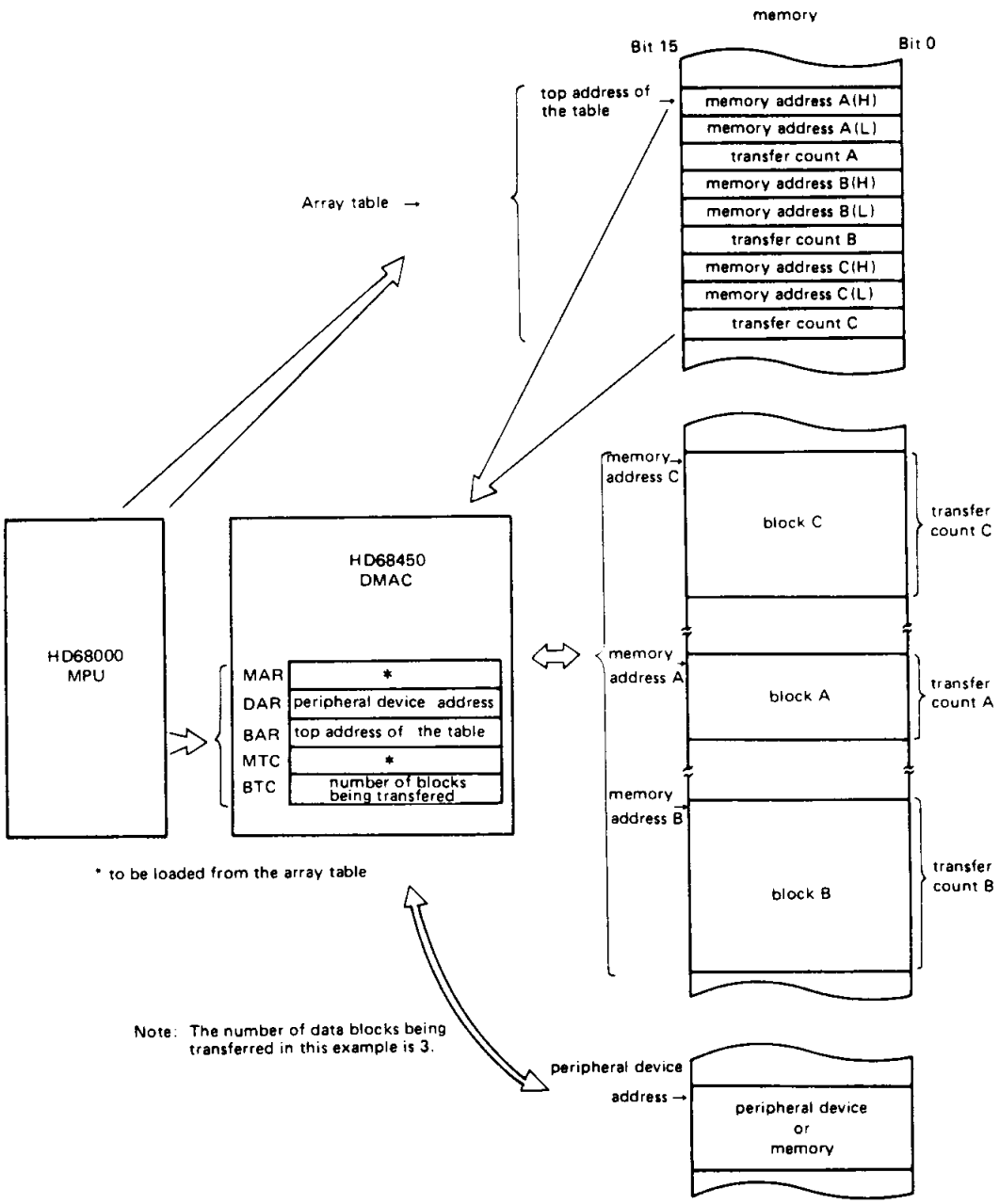


Figure 37 Transfer Example of the Array Chaining Mode

Linked Array 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.

An example of the linked array chaining mode operation and the memory format for supporting it is shown is Figure 38.

In Figure 38, the DMAC transfers data blocks in the order of Block A, Block B, and Block C. In the linked array chaining

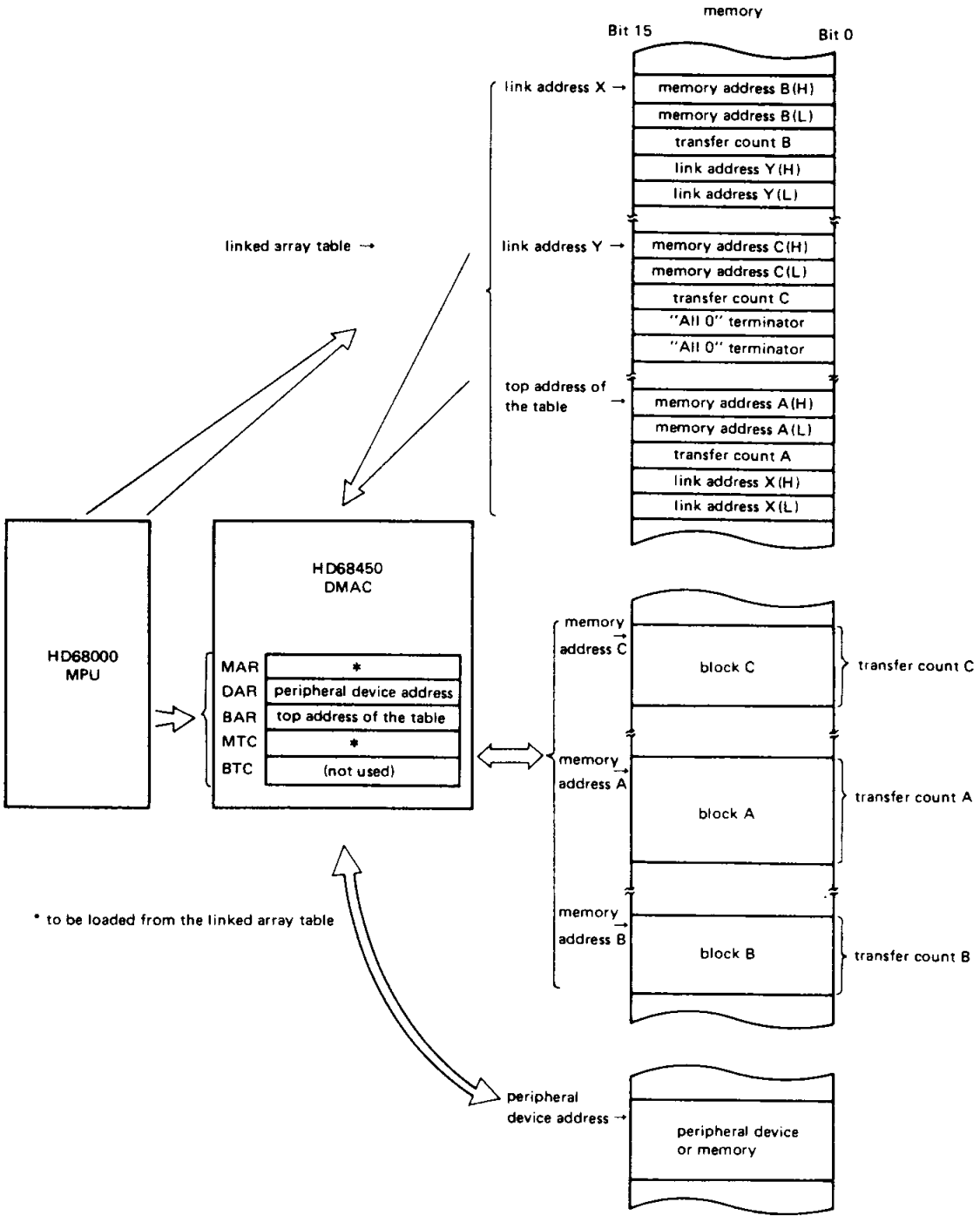


Figure 38 Transfer Example of the Linked Array Chaining Mode

5

mode, the BTC is not used. When the DMAC refers to the linked array table, the value of the BFC is outputted as the function code. The values of the function code registers are unchanged by the linked array chaining operation.

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.

The linked array table must start at an even address in the linked array chaining mode. Starting the table at an odd address results in an address error. If "0" is initially loaded to the MTC, the count error is signaled. Because the MPU can read all of the DMAC registers, all necessary error recovery information is available to the operating system.

The comparision of both chaining modes is shown in Table 8.

Table 8 Chaining Mode Address/Count Information

Chaining Mode	Base Address Register	Base Transfer Counter	Completed When
Array Chaining	address of the array table	number of data blocks being transferred	Base Transfer Count = 0
Linked Array Chaining	address of the linked array table	(unused)	Linked Address = 0

(4) Bus Exception Conditions

The DMAC has three lines for inputting bus exception conditions called $\overline{BEC_0}$, $\overline{BEC_1}$, and $\overline{BEC_2}$. The priority encoder can be used to generate these signals externally. These lines are encoded as shown in Table 9.

Table 9

$\overline{BEC_2}$	$\overline{BEC_1}$	$\overline{BEC_0}$	Exception Condition
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

In order to guarantee, reliable decoding, the DMAC verifies that the incoming code has been statable for two DMAC clock cycles before acting on it. The DMAC picks up $\overline{BEC_0}$-$\overline{BEC_2}$ at the rising edge of the clock. If $\overline{BEC_0}$-$\overline{BEC_2}$ is asserted to the undefined code, the operation of the DMAC does not proceed. For example, when the DMAC is waiting for $\overline{DTACK}$, inputting $\overline{DTACK}$ does not result in the termination of the cycle if $\overline{BEC_0}$-$\overline{BEC_2}$ is asserted to the undefined code. In addition, when the transfer request is received, $\overline{BR}$ is not asserted if the $\overline{BEC_0}$-$\overline{BEC_2}$ is not set to no exception condition.

If exception condition, except for HALT, is inputted during the DMA bus cycle prior to, or in coincidence with $\overline{DTACK}$, the DMAC terminates the current channel operation immediately. Here coincident means meeting the same set up requirements for the same sampling edge of the clock. If a bus exception condition exists, the DMAC does not generate any bus cycles until it is removed. However, the DMAC still recognizes requests.

Halt

The timing diagram of halt is shown in Figure 39. This diagram shows halt being generated during a read cycle from the 68000 compatible device in the dual addressing mode. If the halt exception is asserted during a DMA bus cycle, the DMAC does not terminate the bus cycle immediately. The DMAC waits for the assertion of $\overline{DTACK}$ before terminating the bus cycle so that the bus cycle is completed normally. In the halted state, the DMAC puts all the control signals to high impedance and relinquishes the bus to the MPU. The DMAC does not output the $\overline{BR}$ until halt exception is negated. When halt exception is negated, the DMAC acquires the bus again and proceeds the DMA operation. In order to insure a halt exception operation, the $\overline{BEC}$ lines must be set to halt at least until the assertion of $\overline{DTC}$.

If the DMAC has the bus, but is not executing any bus cycle, the DMAC relinquishes the bus as soon as halt exception is asserted.

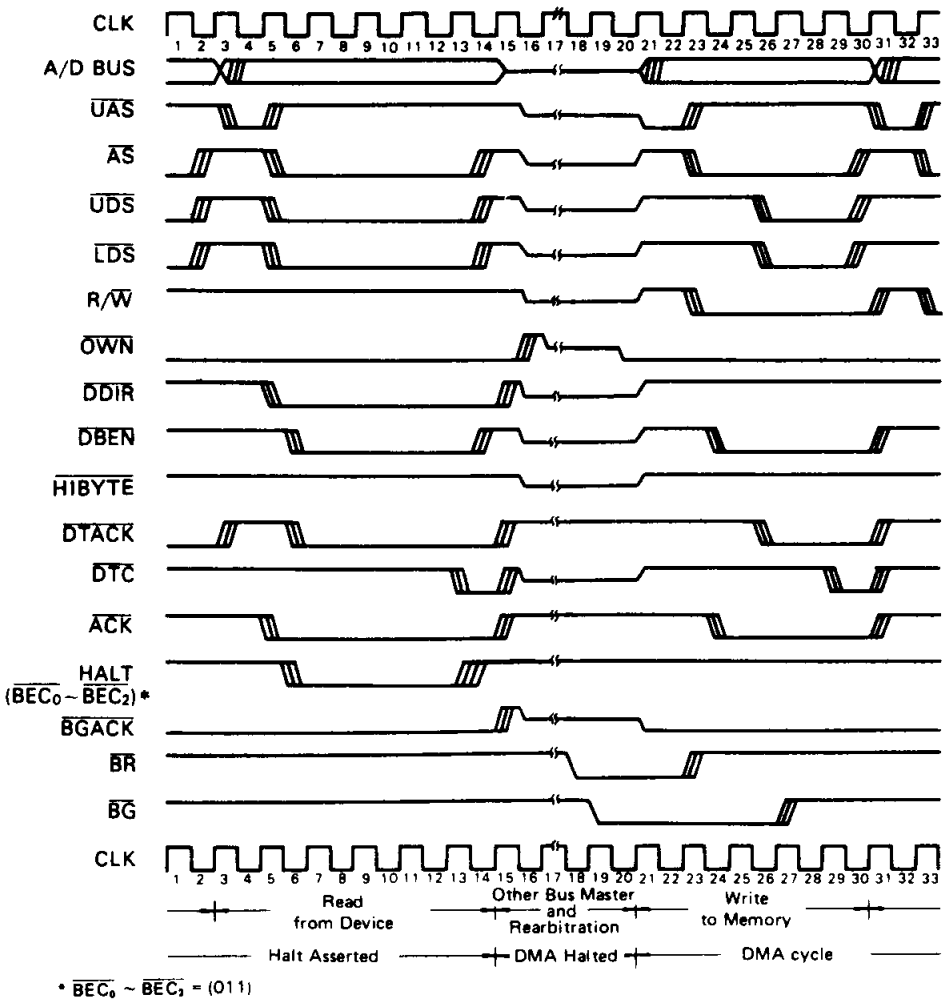


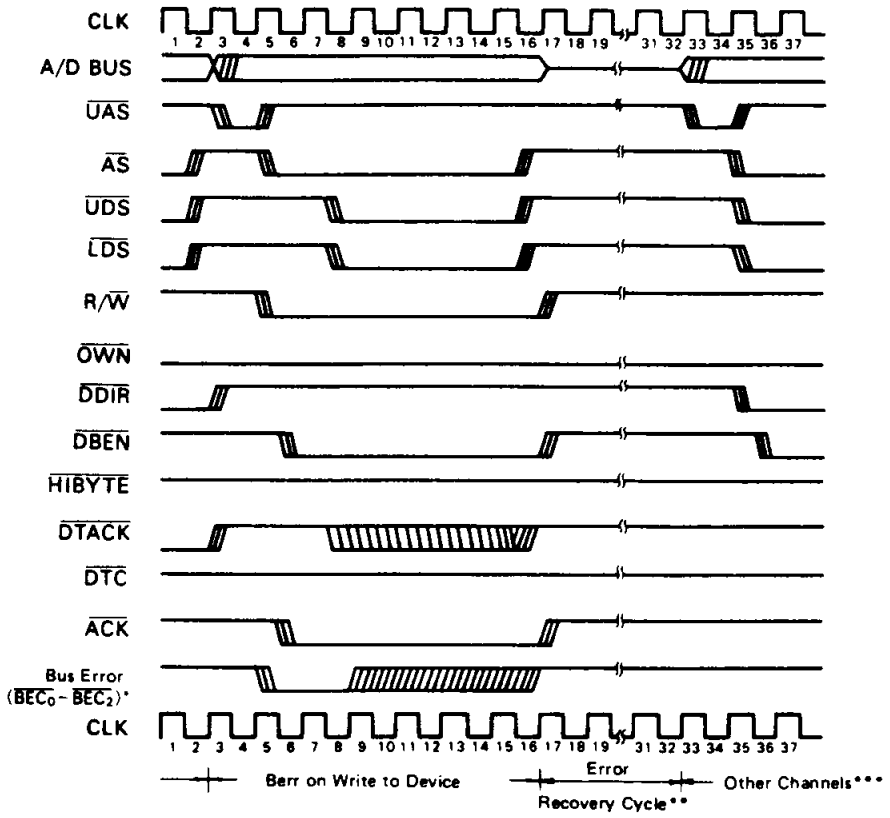
Figure 39 Halt Operation

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. As soon as the DMAC recognizes the bus error exception, the DMAC immediately terminates the bus cycle and proceeds to the error recovery cycle. In this cycle, the DMAC adjusts the values of the MAR, the DAR, the MTC and the BTC to the values when the bus error exception occurred. 25 clocks are required for the error recovery cycle in the single addressing mode and in the read cycle of the dual addressing mode. 29 clocks are required in the write cycle of the dual addressing mode. If the DMAC does not have any transfer request in the other channels after the error recovery cycle, the DMAC relinquishes the bus.

The diagram of the bus error timing is shown in Figure 40.

5



* $\overline{BEC_0}$-$\overline{BEC_2}$ = (101)
** In the single addressing mode and in the read cycle of the dual addressing mode: 25 clocks
In the write cycle of the dual addressing mode: 29 clocks
*** The DMAC keeps the bus because the other channels have requests pending. If other channels do not have requests, the DMAC relinquishes the bus after the error recovery cycle.

Figure 40 Bus Error Operation

Retry

The retry exception causes the DMAC to terminate the present operation and retry that operation when retry is removed, and thus will not honor any requests until it is removed. However, the DMAC still recognizes requests. The retry timing is shown in Figure 41.

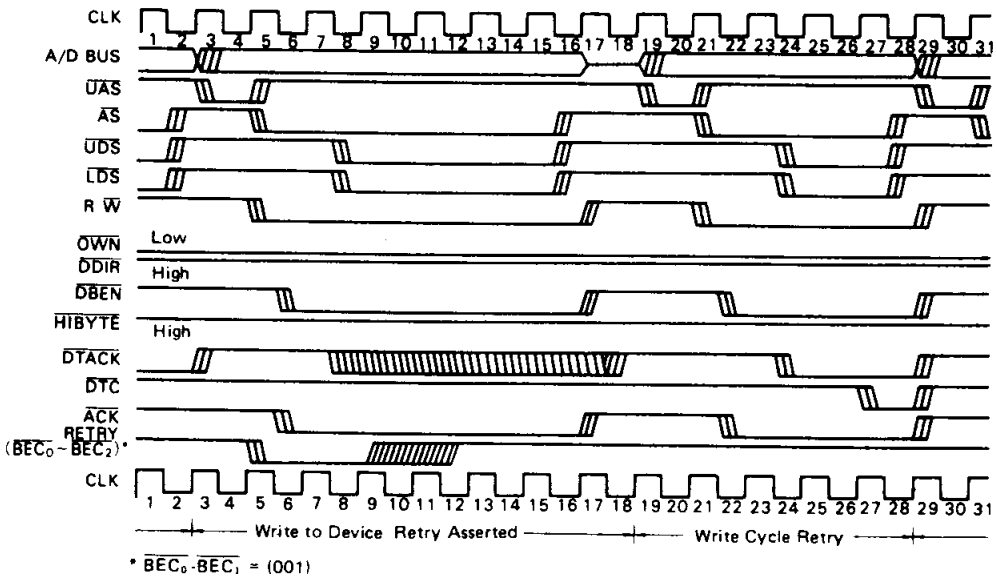


Figure 41 Retry Operation

Relinquish and Retry (R&R)

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

The diagram of the relinquish and retry timing is shown in Figure 42.

5

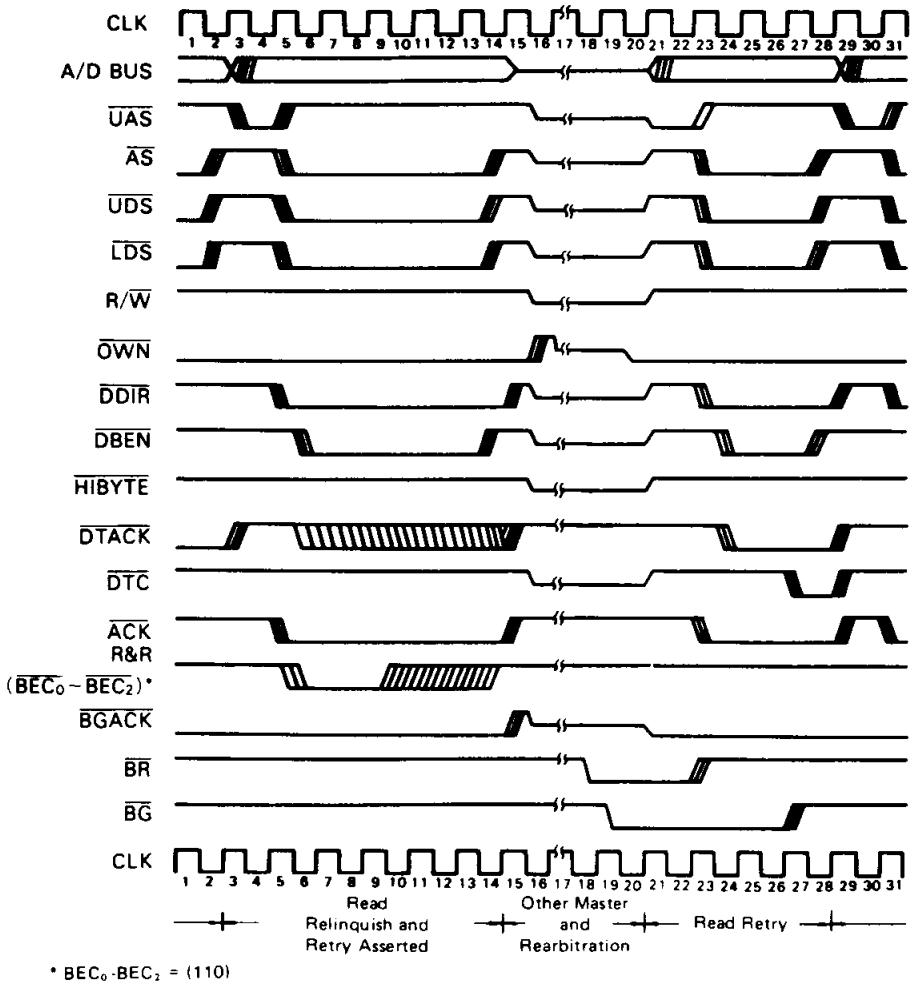


Figure 42 Relinquish and Retry Operation

Reset

The reset provides a means of resetting and initializing the DMAC. If the DMAC is bus master when the reset is asserted, the DMAC relinquishes the bus. Reset clears GCR, DCR, OCR, SCR, CCR, CSR, CPR, and CER for all channels. The NIV and the EIV are all set to $(0F)_{16}$, which is the uninitialized interrupt vector number for the HD68000 MPU. MTC, MAR, DAR, BTC, BAR, MFC, DFC, and BFC are not affected. In order to insure a reset, $\overline{BEC_0} \sim \overline{BEC_2}$ must be kept at "Low" level for at least ten clocks.

(5) 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 is indicated in the CER. All pending operations are cleared, so that both the STR and CNT bits of CCR are cleared.

Enumerated below are the error signals and their sources.

(a) Configuration Error – This error occurs if the STR bit is set in the following cases.
  (i) the CNT bit is set at the same time STR bit in the chaining mode.
  (ii) DTYP specifies a single addressing mode, and the device port size is not the same as the operand size.
  (iii) DTYP specifies a dual addressing mode, DPS is 16 bits, SIZE is 8 bits and REQG is "10" or "11".
  (iv) an undefined configuration is set in the registers. The undefined configurations are: XRM = 01, MAC = 11, DAC = 11, CHAIN = 01, and SIZE = 11.

(b) Operation Timing Error – An operation timing error occurs in the following cases:
  (i) when the CNT bit is set after the ACT bit has been set by the DMAC in the chaining mode, or when the STR and the ACT bits are not set.
  (ii) the STR bit is set when ACT, COC, BTC, NDT or ERR is set.
  (iii) an attempt to write to the DCR, OCR, SCR, MAR, DAR, MTC, MFC, or DFC is made when the STR bit or the ACT bit is set.
  (iv) an attempt to set the CNT bit is made when the BTC and the ACT bits are set.

(c) Address Error – An address error occurs in the following cases:
  (i) an odd address is set for word or long word operands.
  (ii) $\overline{CS}$ or $\overline{IACK}$ is asserted during the DMA bus cycle.

(d) Bus Error – Bus error occurs when a bus error excep-

tion is signaled during a DMA bus cycle.

(e) Count Error – A count error occurs in the following cases:
  (i) The STR bit is set when zero is set in the MTC and the MTC and the chaining mode is not used.
  (ii) the STR bit is set when zero is set in BTC for the array chaining mode.
  (iii) zero is loaded from memory to the BTC or the MTC in the chaining modes or the continue mode.

(f) External Abort – External abort occurs if an abort is asserted by the external circuitry when the $\overline{PCL}$ line is configured as an abort input and the STR or the ACT bit is set.

(g) Software abort – Software abort occurs if the SAB bit is set when the STR or the ACT bit is set.

Error Recovery Procedures

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

The information available to the operating system consists of the present value of the Memory Address, Device Address and Base Address Registers, the Memory Transfer and Base Transfer Counters, the channel status register, the channel error register, and the channel control register. 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. However, in the case of external abort, there are cases in which the previous values are not recovered.

Bus Exception Operating Flow

The bus exception operating flow in the case of multiple exception conditions occurring continuously in sequence is shown in Figure 43. Note that the DMAC can receive and execute the next exception condition. For example, if the retry exception occurs, and next the relinquish and retry exception occurs while the DMAC is waiting for the retry condition to be cleared, the DMAC relinquishes the bus and waits for the exception condition to be cleared. If a bus error occurs during this period, the DMAC executes the bus error exception operation.

The flow diagram of the normal operation without exception operation or errors is shown in Figure 44.

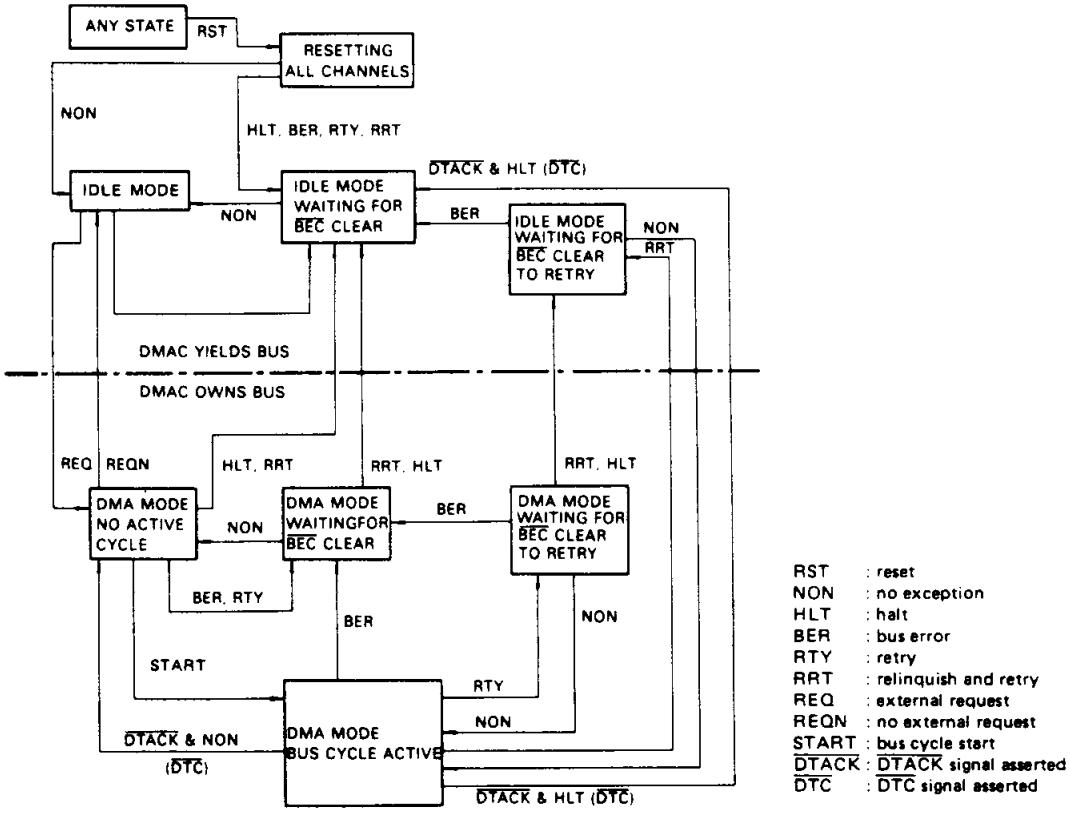


Figure 43 Bus Exception Flow Diagram

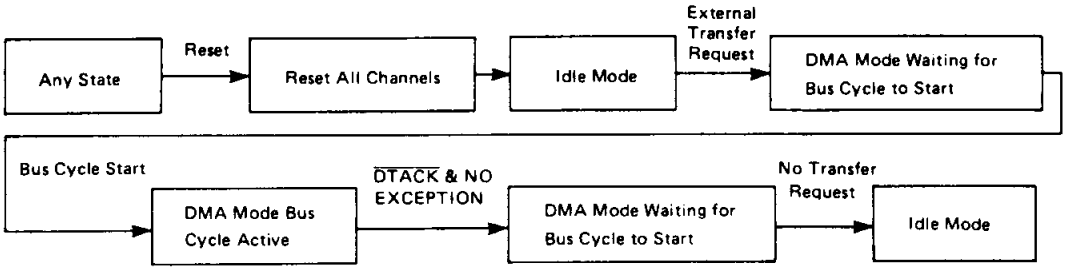


Figure 44 Flow of Normal Operation Without Exception or Error Condition

- **Channel Priorities**

Each channel has a priority level, which is 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.

Resetting the DMAC puts the priority level of all channels to "0", the highest priority level.

■ APPLICATIONS INFORMATION

Examples of how to interface HD68450 to a HD68000 based system are shown in Figure 45 and Figure 46.

Figure 45 shows an example of how to demultiplex the address/data bus. $\overline{OWN}$ and $\overline{UAS}$ are used to control 74LS373 for latching the address. $\overline{DBEN}$ and $\overline{DDIR}$ are used to control the bi-directional buffer 74LS245.

Figure 46 shows an example of inter-device connection in the HD68000 system.

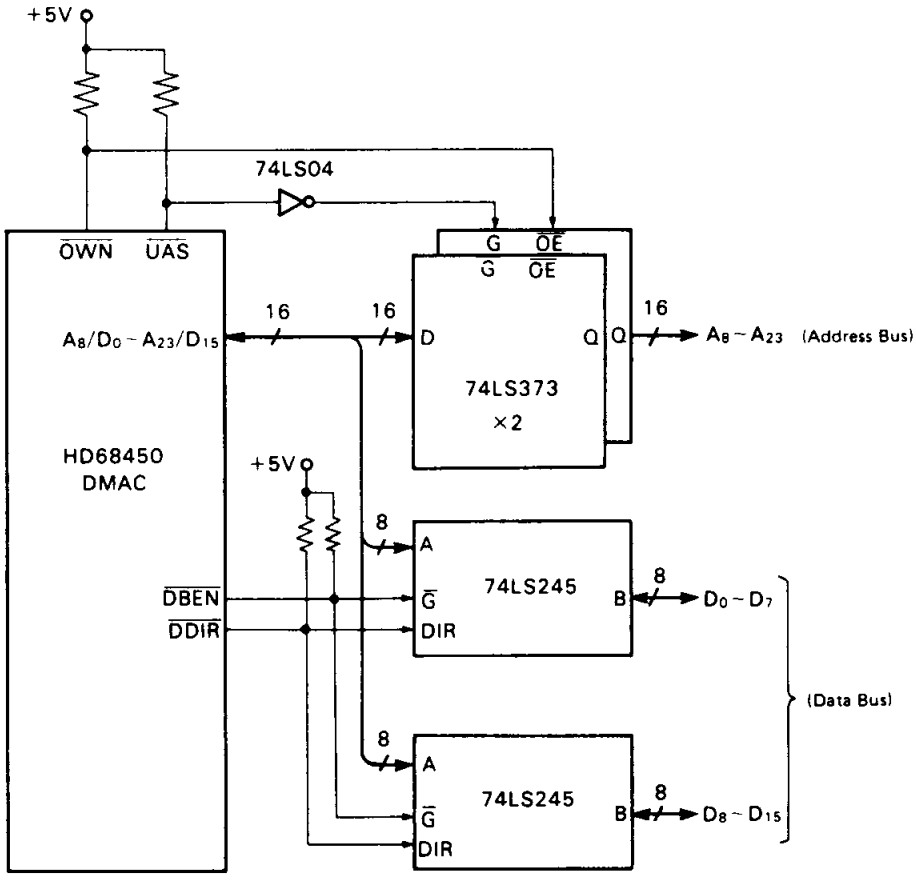
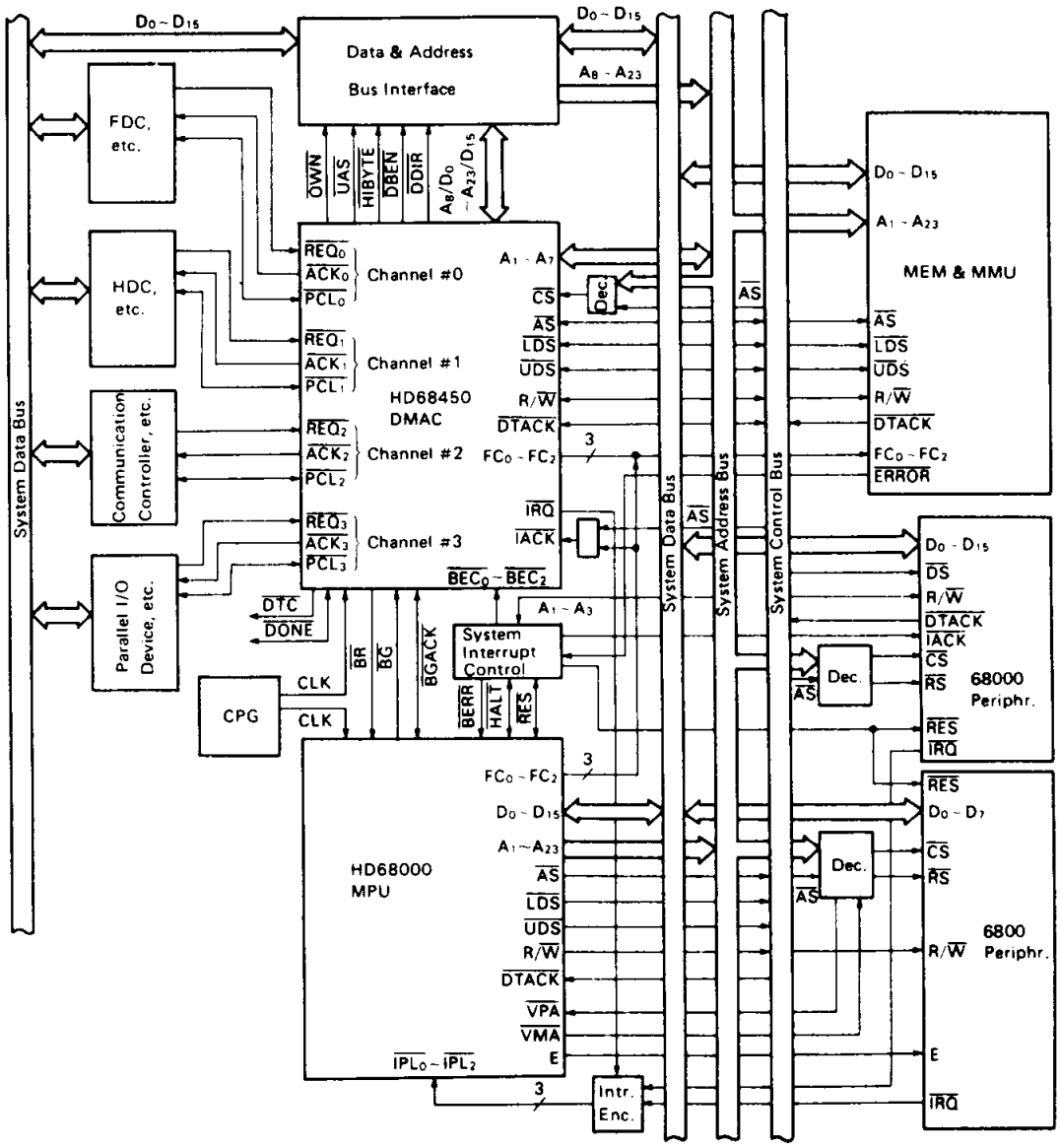


Figure 45 An Example of the Demultiplexed Address Data Bus



The address bus and the system control bus in each device are omitted in this Figure.

Figure 46 An Example of Inter-device Connection in the HD68000 System

5

■ ATTENTION ON USAGE

(1) How to interface various 6800 type peripheral devices to the DMAC based system.

When the DMAC is reading data from the 6800 device, the DMAC latches the data when $\overline{DTC}$ is asserted and not at the falling edge of E clock. The 74LS373 need to be provided externally as shown in Figure 47 so that the data from the 6800 device can be held on the bus for a large period of time until the DMAC can latch the correct data.

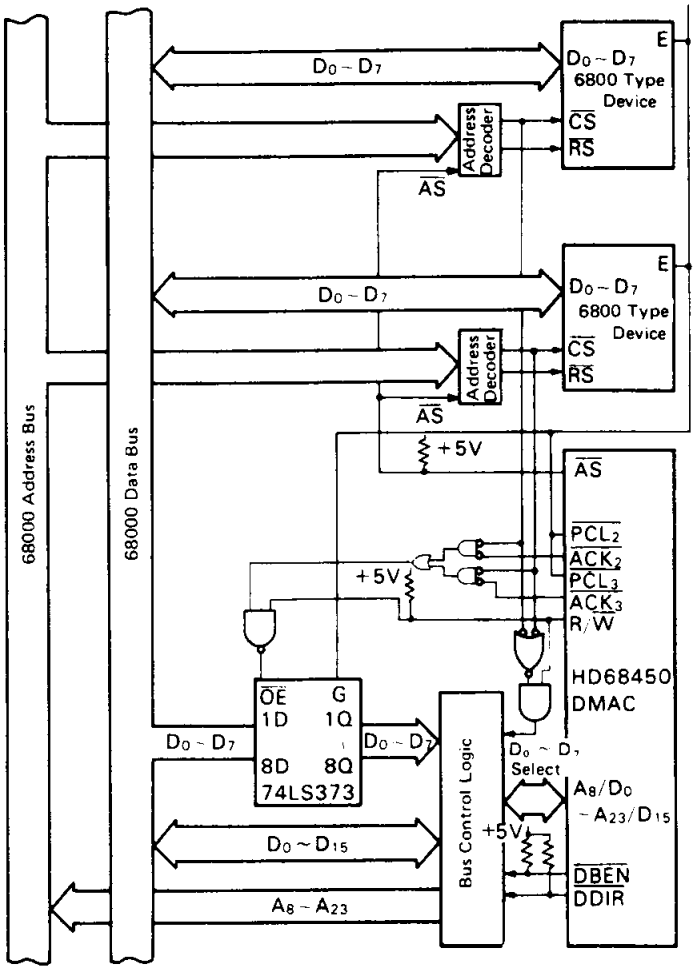


Figure 47 An Example of Connection with 6800 type Peripheral Devices (channel 2 and 3 are used)

(2) When "external abort" is inputted during the $\overline{DONE}$ input cycle

When the transfer direction is from the peripheral device to memory and $\overline{PCL}$ signal is set to the external abort input mode in the dual addressing mode, the external abort will be ignored during the subsequent write cycle from the DMAC's internal holding register to memory if $\overline{DONE}$ is inputted during the read cycle from the peripheral device to the DMAC's internal holding register.

In this case, the channel status register (CSR) and the channel error register (CER) show the normal termination caused by $\overline{DONE}$ Input. The user is able to examine the PCT bit and the ERR bit in order to detect the external abort inputted at the timing described above. If PCT = 1, ERR = 0, and NDT = 1, then an external abort has occurred.

(3) Multiple Errors

The DMAC will log the first error encountered in the channel error register (CER). If an error is pending in the error register and another error is encountered the second error will not be logged. Even though the second error is not logged in the CER, it will still be recognized internally and the channel will not start.

(4) Relinquish & Retry Exception During Dual Address Mode Operation

When the following two conditions occur simultaneously, incorrect data is outputted by the DMAC at the write cycle immediately following the negation of the relinquish & retry (R&R) exception.

(a) R&R is asserted at the write cycle in the dual address mode.
(b) MPU access to the DMAC's internal register is done after the DMAC relinquishes the bus due to R&R exception.

When the R&R exception occurs during the write cycle of the dual addressing mode, and the MPU accesses the DMAC's register, then the DMAC's proper operation sequence should be the folliwing (refer to the Fig. 48):

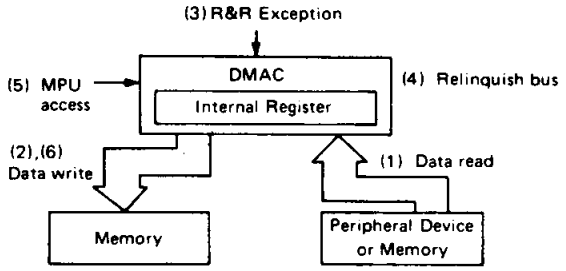


Fig. 48

(1) Data is read by the DMAC during the read cycle
(2) Data read at (1) is outputted at the write cycle
(3) R&R exception is asserted during the write cycle
(4) DMAC relinquishes the bus
(5) MPU accesses the DMAC and it is completed normally
(6) When R&R exception is negated, the DMAC obtains the bus and write cycle is done to output the data read at (1).

But instead, incorrect data is outputted at (6). This is because the data read at (1), which is held internally by the DMAC, is destroyed at (5) when the MPU accesses the DMAC. Avoid occurance of the above condition. For example:

(1) Assert R&R exception only during the read cycle when using dual addressing mode.
(2) If the R&R exception occurred at the write cycle of the dual addressing mode, avoid accessing the DMAC's registers.
(3) Use HALT exception instead of R&R exception to access the DMAC's internal registers.

(5) $\overline{CS}$ Negation Timing

When the $\overline{LDS}$, $\overline{UDS}$ high to $\overline{CS}$ high timing (chip select negation delay) is over 1 clock and the MPU access is long word (32-bit data), then the data stored in the lower word of the register accessed is destroyed and becomes all zeros. In other words, the data in the lower word of the read access is lost and cleared to all zeros. This does not always occur, but on occasional basis due to the asynchronous input timing of the CS signal.

Please observe the timing specification shown in Fig. 49.

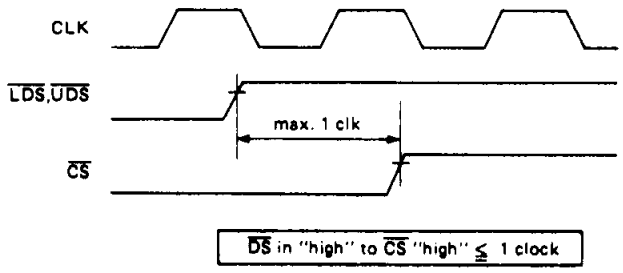


Fig. 49

(6) Unused Function Code (FC_0–FC_2) Lines

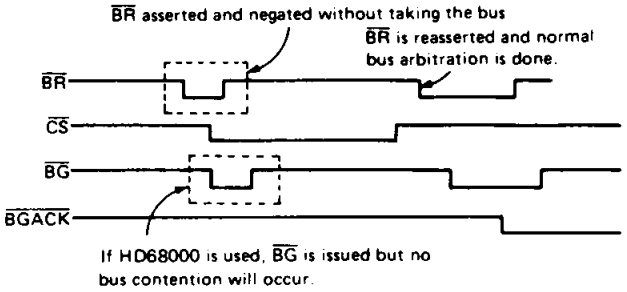
When the FC_0, FC_1, and FC_2 lines are not used, please keep these lines "high" by using a pull-up resistor. If these lines are left unconnected, the HD68450 DMAC may not operate properly.

(7) Vss Wiring

The use of thick wiring is recommended between Vss of the HD68450 and the ground of the circuit board. When a socket is used to install the DMAC on the board, please make sure that the contact of the Vss pins are made well.

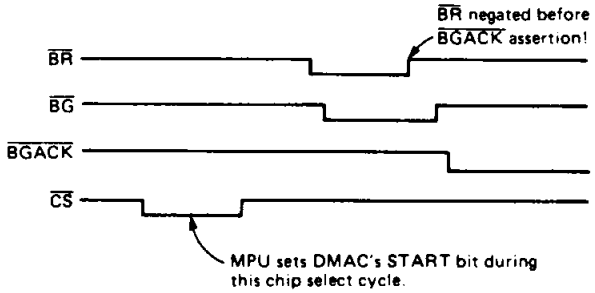
(8) Bus arbitration – $\overline{BR}$ Negation on MPU's $\overline{CS}$.

If the MPU asserts DMAC's $\overline{CS}$ when the DMAC has its $\overline{BR}$ asserted, then the DMAC negates its $\overline{BR}$. This is shown as follows:



(9) $\overline{BR}$ negation before $\overline{BGACK}$ assertion

When the MPU sets the START bit of the DMAC, and a different channel is already active in the limited-rate auto-request mode, the the following bus arbitration timing may occur.



In this timing, the $\overline{BR}$ is negated too early, e.g. before the $\overline{BGACK}$ is asserted. This will cause bus contention between the DMAC and the MPU. For the description of these problems, please contact the sales office.

5