PART 5

TMP68305F-16

Chapter 1 Introduction

TMP68305 uses the 68HC000, the CMOS version of the 68000, as its core processor. It also includes peripheral circuits such as a serial interface, timer, interrupt controller, DMA controller, clock generator, and address decoder.

In addition, TMP68305 can directly use 68000 development environments and software resources.

- Core processor 68HC000
- Minimum instruction execution time : 240 ns (with 16.67 MHz-system clock)
- 17 32-bit registers
- 16M-byte direct addressing
- 56 powerful basic instructions
- 14 addressing modes

- 2-channel asynchronous serial interface
- 1-channel 16-bit timer/counter
- 9-channel interrupt controller (4 external channels, 5 internal channels)
- 2-channel DMA controller (8MB/s max.)
- Clock generator (duty controller built in)
- 4-channel chip-select signal output ($\overline{CS0}$, $\overline{CS1}$, $\overline{CS2}$, $\overline{CS3}$)
- Bus monitor function
- Low power consumption (CMOS)
- 100-pin QFP

TMP68305 has two operating modes: normal operating mode, and emulation mode that enables use of an in-circuit emulator (ICE), a 68000 development tool. In emulation mode, the 68HC000 core built into TMP68305 is disconnected from the bus, and the internal peripheral circuits are controlled by address, data, and control signals from the development tool.

The 68HC000 core built into TMP68305 is the same as the standard TMP68HC000, except that 8-bit peripheral device control signals E, $\overline{VPA}$, and $\overline{VMA}$ are disabled. For 68HC000 operation and instructions, refer to your TLCS-68000 Data Book.

Figure 1.1 is the TMP68305 block diagram.

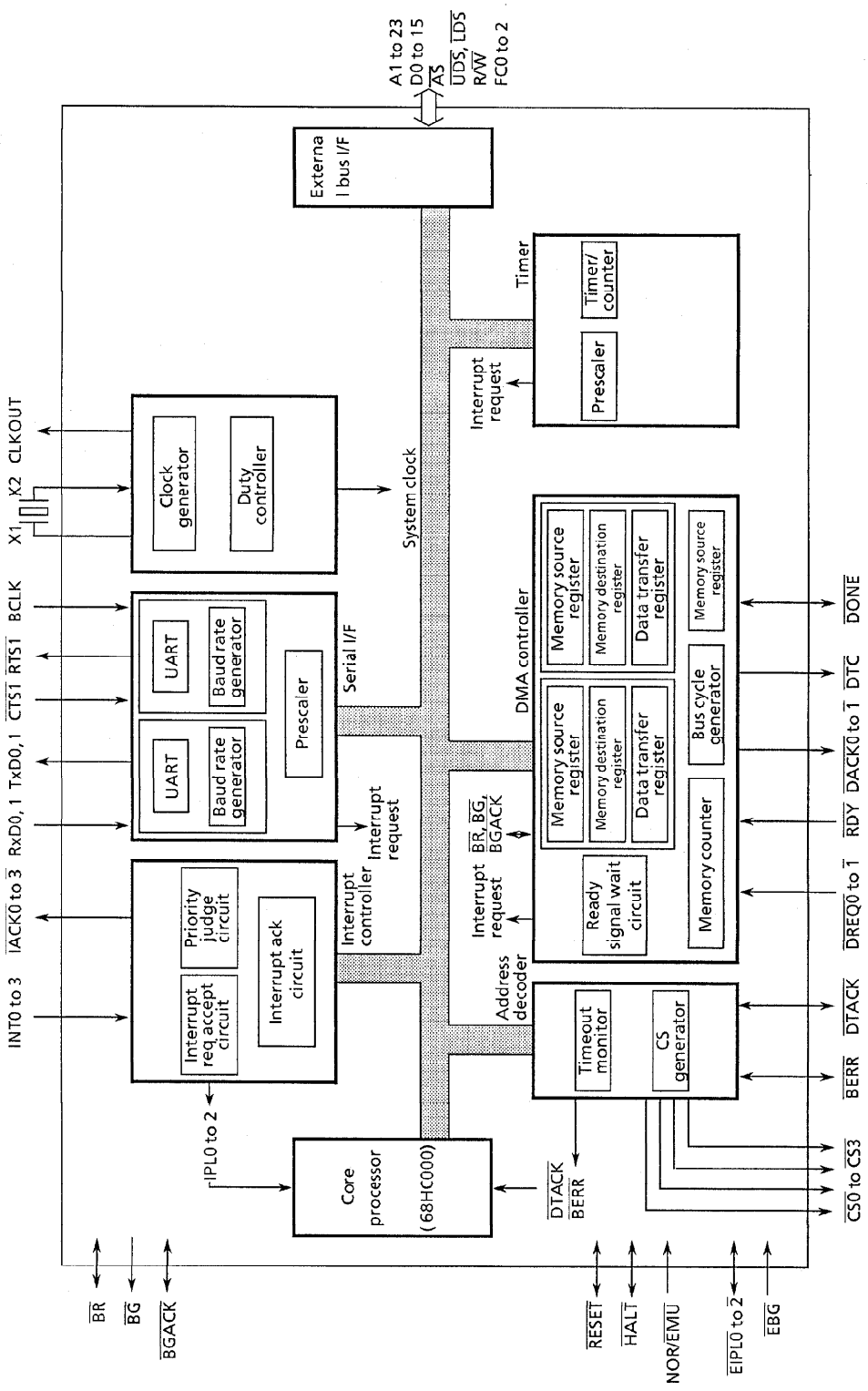


Figure 1.1 TMP68305 Block Diagram

Chapter 2 Signal Description

This section briefly describes input and output signals.

The terms "assert" and "negate" appear frequently. These terms are used to avoid ambiguity when terms such as "active high" and "active low" are used. "Assert" shows that signals are active or true, irrespective of whether the signal is electrically high or low. "Negate" shows that signals are inactive or false.

2.1 Pin Assignments

Figure 2.1 shows the pin assignments.

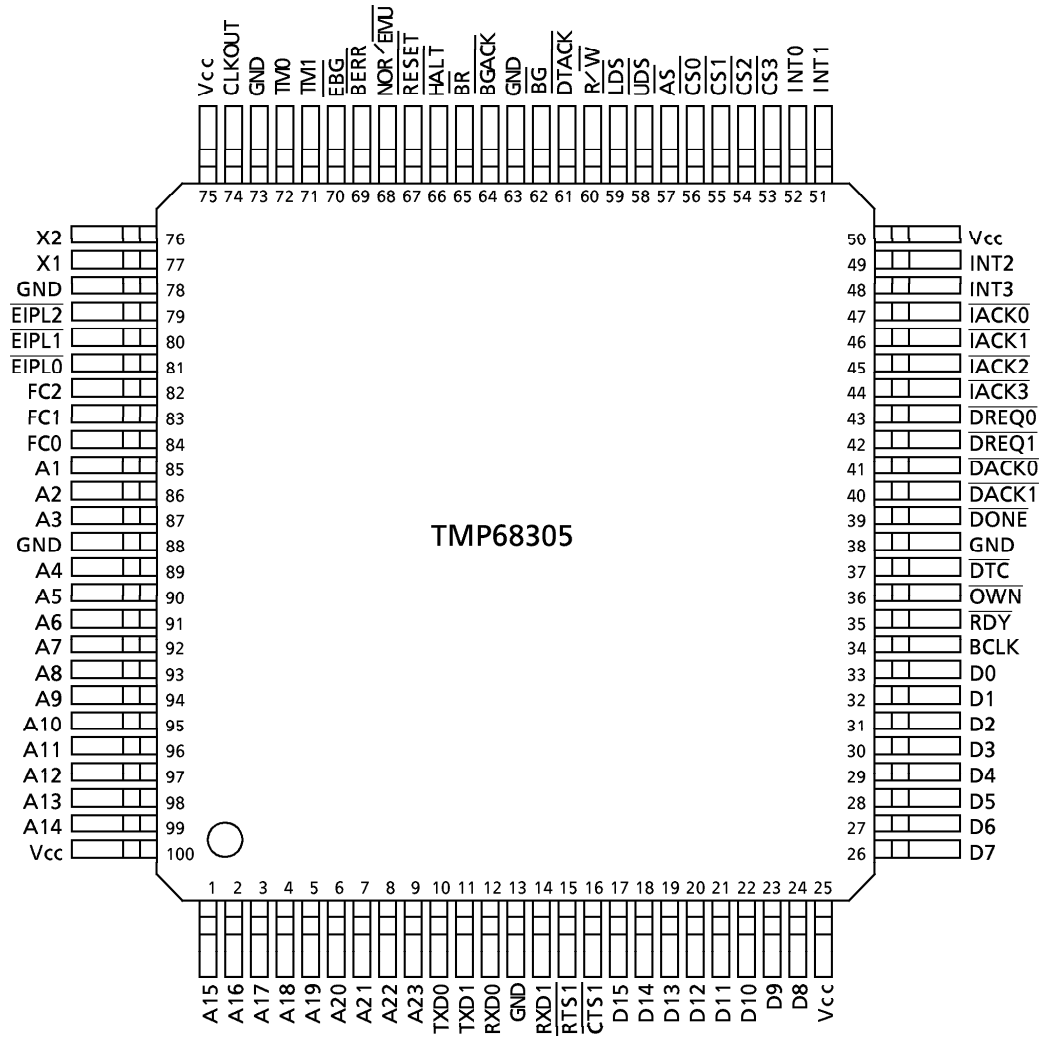


Figure 2.1 Pin Assignments (top view)

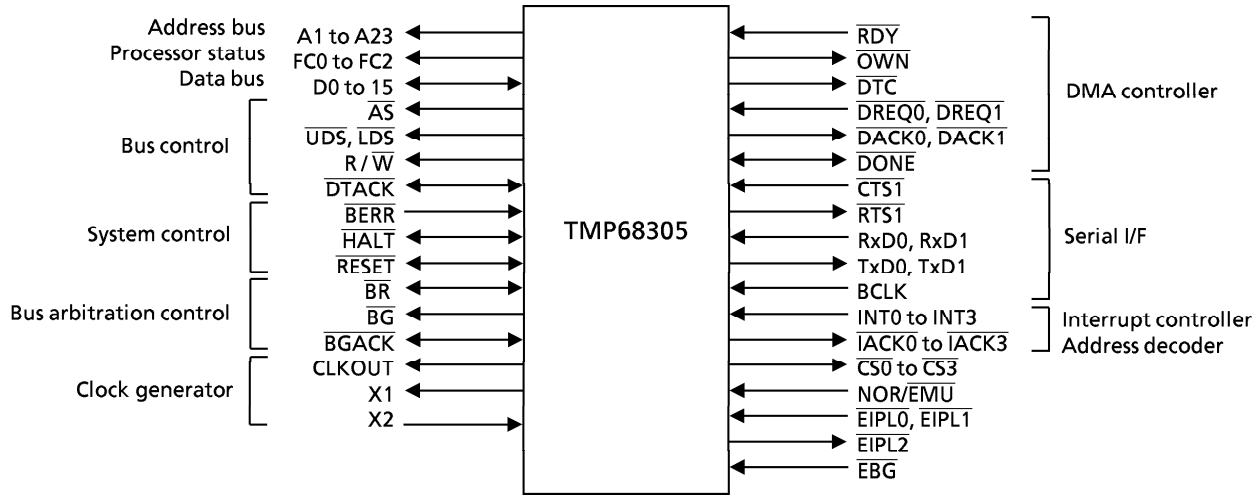


Figure 2.2 Normal Mode Input/Output Signals When Configured as Bus Master

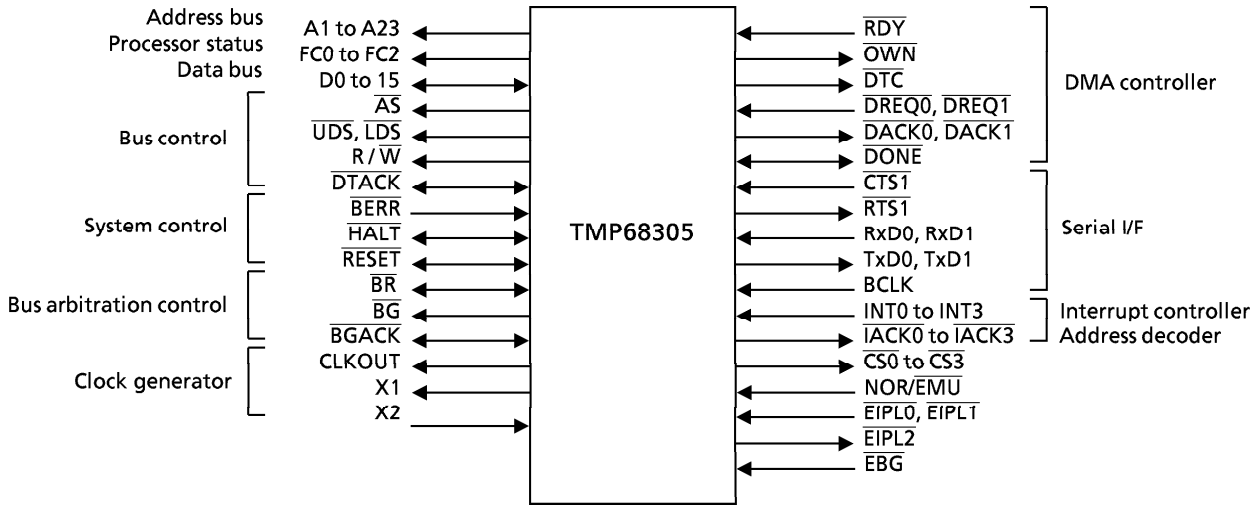


Figure 2.3 Normal Mode Input/Output Signals When Bus Mastership Released

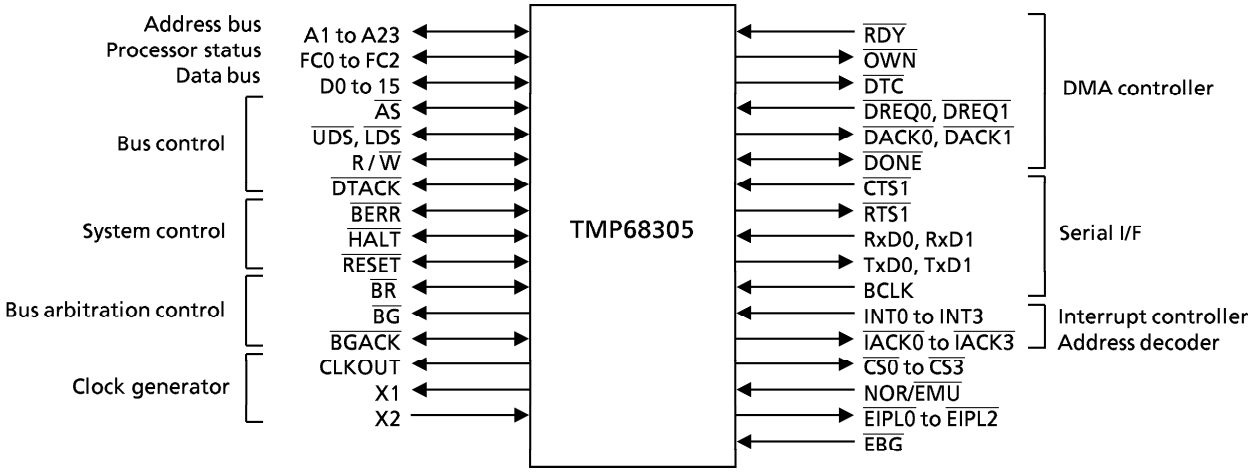


Figure 2.4 Emulation Mode Input/Output Signals

2.2 Pin Names and Functions

The following describes pin states and functions in normal and emulation modes.

NOR : Normal mode, EMU : Emulation mode
O.D : Open drain output, O : Output, I: Input, I/O : Input/output

Signal name	Pin status		Function
	NOR	EMU	
A1 to A23	O	I/O	23-bit address bus. Can directly access 16M bytes of memory.
FC0 to FC2	O	I/O	These signals show the status of the processor and the current cycle type. For details, see Table 2.2.
D0 to D15	I/O	I/O	16-bit general-purpose data bus
$\overline{AS}$	O	I/O	This signal indicates that there is a valid address on the address bus.
R/$\overline{W}$	O	I/O	This signal indicates whether the data transfer is read (high) or write (low).
$\overline{UDS}$ / $\overline{LDS}$	O	I/O	These signals control the data on the data bus. See Table 2.1 for details.
$\overline{DTACK}$	I	O.D	This signal indicates the end of a data transfer.
$\overline{BR}$	I/O.D	I/O.D	This signal is wire-ORed with all other devices that could become bus masters. The signal indicates that another device is requesting bus mastership.
$\overline{BG}$	O	O	This signal indicates to devices that could become bus masters that the processor will release control of the bus at the end of the current bus cycle.
$\overline{BGACK}$	I/O.D	I/O.D	This signal indicates that another device has become the bus master.
$\overline{BERR}$	I	O.D	This signal reports to the processor that there is a problem in the current cycle.
$\overline{RESET}$	I/O.D	I/O.D	In combination with $\overline{HALT}$, this signal resets the processor. If the $\overline{RESET}$ instruction is executed, this signal functions as a reset signal for external devices.
$\overline{HALT}$	I/O.D	I/O.D	As an input, this signal halts the processor when the current bus cycle ends. Also, the signal acts as an output when a double bus error exception occurs.
X1, X2 (CLKIN)	I	I	Crystal oscillator connecting pins. System clock can be input to X2 externally. (In this case, leave X1 open.)
CLKOUT	O	O	Output frequency of signals is the same as that generated by the built-in clock generator.
INT0, INT1, INT2, INT3	I	I	External interrupt request input
$\overline{IACK0}$, $\overline{IACK1}$ $\overline{IACK2}$, $\overline{IACK3}$	O	O	These signals indicate the interrupt acknowledge cycles corresponding to INT0, INT1, INT2, and INT3.
NOR / $\overline{EMU}$	I	I	Normal mode/emulation mode switch signal.
$\overline{EIPL0}$, $\overline{EIPL1}$ $\overline{EIPL2}$	I	O	These signals are interrupt requests output from the internal interrupt controller in emulation mode only. In normal mode, the signals are input but do not have any function. Thus, fix $\overline{EIPL0}$ and $\overline{EIPL1}$ to low; $\overline{EIPL2}$ to high.
$\overline{EBG}$	I	I	In emulation mode, this signal output by the CPU is received by the built-in DMA controller. In normal mode, fix it to high.

NOR : Normal mode, EMU : Emulation mode, O : Output, I : Input, I/O : Input/output

Signal name Pin status Function

Signal name	Pin status		Function
	NOR	EMU	
$\overline{DREQ0}$, $\overline{DREQ1}$,	I	I	DMA request signals to the DMA controller
$\overline{DACK0}$, $\overline{DACK1}$,	O	O	Acknowledge signals to peripheral devices from the DMA controller
$\overline{DONE}$	I/O.D	I/O.D	Disable signal ignoring DMA requests, end signal suspending DMA operation, or output signal indicating transfer complete.
$\overline{DTC}$	O	O	Output signal indicating DMA cycle complete
$\overline{OWN}$	O	O	Output signal indicating built-in DMA controller is bus master.
$\overline{RDY}$	I	I	$\overline{READY}$ signal input at DMA transfer from I/O device with external $\overline{READY}$.
$\overline{CTS1}$	I	I	$\overline{CTS}$ signal for serial interface channel 1
$\overline{RTS1}$	O	O	$\overline{RTS}$ signal for serial interface channel 1
RxD0, RxD1	I	I	Receive data input for serial interface
TxD0, TxD1	O	O	Transmit data output for serial interface
BCLK	I	I	Reference clock input for generating the serial interface baud rate.
$\overline{CS0}$, $\overline{CS1}$, $\overline{CS2}$, $\overline{CS3}$	O	O	Signal used to access the memory area specified by the address decoder.
Vcc	–	–	Power input pin (+ 5 V)
GND	–	–	GND pin (0 V)
TM0, TM1	–	–	Shipment test pins. Fix to open in both normal and emulation modes.

Table 2.1 Data Bus Control By Data Strobe

$\overline{UDS}$	$\overline{LDS}$	R/$\overline{W}$	D8 to D15	D0 to D7
H	H	—	Data invalid	Data invalid
L	L	H	Valid data bits 8 to 15	Valid data bits 0 to 7
H	L	H	Data invalid	Valid data bits 0 to 7
L	H	H	Valid data bits 8 to 15	Data invalid
L	L	L	Valid data bits 8 to 15	Valid data bits 0 to 7
H	L	L	Valid data bits 0 to 7*	Valid data bits 0 to 7
L	H	L	Valid data bits 8 to 15	Valid data bits 8 to 15*

L : LOW H : HIGH

* : This condition is a result of the current implementation and may not apply in the future.

Table 2.2 Function Code Output

FC2	FC1	FC0	Cycle type	
L	L	L	*	
L	L	H	User data	
L	H	L	User program	
L	H	H	*	
H	L	L	*	
H	L	H	Supervisor data	
H	H	L	Supervisor program	
H	H	H	Interrupt acknowledge	(Note)

L : LOW H : HIGH

* : Undefined, for future use

Note : If FC0, FC1, and FC2 are pulled up, when releasing the bus, the state is the same as an interrupt acknowledge cycle, and the interrupt controller malfunctions. To prevent this, pull down one of FC0, FC1, or FC2.

2.3 Signal Summary

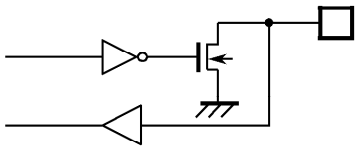
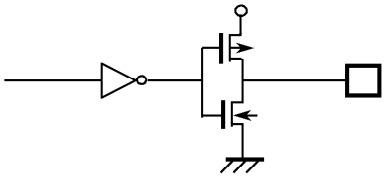
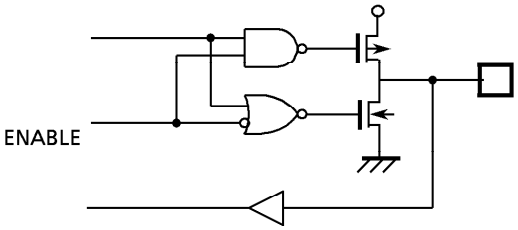
Table 2.3 Signal Summary

Signal name	Mnemonic	Input/ Out	Active state	Tri-state Out	When $\overline{HALT}$ asserted	When $\overline{BG}$ asserted	When $\overline{RESET}$/$\overline{HALT}$ both asserted
Address bus	A1 to A23	Out (In/Out)	H	Y	Hi-Z	Hi-Z	Hi-Z
Data bus	D0 to D15	In / Out (Out / In)	H	Y	Hi-Z	Hi-Z	Hi-Z
Address strobe	$\overline{AS}$	Out (In/Out)	L	Y	H	Hi-Z	Hi-Z
Read/write	R/$\overline{W}$	Out (In/Out)	H (read) L (write)	Y	H	Hi-Z	Hi-Z
Upper and lower data strobe	$\overline{UDS}$, $\overline{LDS}$	Out (In/Out)	L	Y	H	Hi-Z	Hi-Z
Data transfer acknowledge	$\overline{DTACK}$	In/Out	L	O.D.	–	–	–
Bus request	$\overline{BR}$	In/Out	L	O.D.	–	–	–
Bus grant	$\overline{BG}$	Out	L	N	H	L	H
Bus grant acknowledge	$\overline{BGACK}$	In/Out	L	O.D.	–	–	–
Emulation interrupt priority	$\overline{EIPL0}$, $\overline{EIPL1}$, $\overline{EIPL2}$	In/Out	L	N	–	–	–
Bus error	$\overline{BERR}$	In/Out	L	O.D.	–	–	–
Reset	$\overline{RESET}$	In/Out	L	O.D.	O.D.	O.D.	L
Halt	$\overline{HALT}$	In/Out	L	O.D.	L	O.D.	L
Function code	FC0, FC1, FC2	Out (In/Out)	H	Y	–	Hi-Z	Hi-Z
System clock	CLKOUT	Out	H	N	–	–	–
Power supply	V_{CC}	In	–	–	–	–	–
Ground	GND	In	–	–	–	–	–
Chip select	$\overline{CS0}$ to $\overline{CS3}$	Out	L	N	–	–	H
Own	OWN	Out	L	N	–	–	H
Data transfer complete	DTC	Out	L	N	–	–	H
DMA request	$\overline{DREQ0}$ to $\overline{1}$	In	L	–	–	–	–
DMA acknowledge	$\overline{DACK0}$ to $\overline{1}$	Out	L	N	–	–	H
DMA disabled/suspended/transfer complete	$\overline{DONE}$	Out/In	L	O.D.	–	–	H
Clear to send	$\overline{CTS1}$	In	L	–	–	–	–
Request to send	$\overline{RTS1}$	Out	L	N	–	–	H
Receive data	RxD0 to 1	In	H	–	–	–	–
Transmit data	TxD0 to 1	Out	H	N	–	–	H
Baud rate clock	BCLK	In	H	–	–	–	–
Ready	$\overline{RDY}$	In	L	–	–	–	–
Crystal oscillator	X1	Out	H	N	–	–	–
Crystal oscillator	X2 (CLKIN)	In	H	–	–	–	–
Mode switch	NOR / $\overline{EMU}$	In		–	–	–	–
Interrupt request	INT0 to 3	In		–	–	–	–
Interrupt acknowledge	$\overline{IACK0}$ to $\overline{3}$	Out	L	N	–	–	H
Test mode	TM0, TM1	–	–	–	–	–	–
Emulation bus grant	$\overline{EBG}$	In	L	–	–	–	–

The parentheses in the input/output column indicate in emulation mode.

Note : O.D : Open drain, Hi-Z : High impedance, - : Optional,
In : Input,
Out : Output, H : High Y : Yes
In/out : Input/output L : Low N : No

2.4 Pin Input/Output Circuits

Pin	Input/output	circuit Remarks
$\overline{\text{RESET}}$, $\overline{\text{HALT}}$, $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$ $\overline{\text{BR}}$, $\overline{\text{BGACK}}$ $\overline{\text{DONE}}$		Sink open drain output
CLKOUT $\overline{\text{BG}}$ TxD0, TxD1 $\overline{\text{IACK0}}$, $\overline{\text{IACK1}}$, $\overline{\text{IACK2}}$, $\overline{\text{IACK3}}$ $\overline{\text{DACK0}}$, $\overline{\text{DACK1}}$ CS0, CS1, CS2, CS3		Push/pull output
RxD0, RxD1 BCLK INT0, INT1, INT2, INT3 NOR / $\overline{\text{EMU}}$		
A1 to A23, D0 to D15 FC0 to FC2, $\overline{\text{AS}}$, $\overline{\text{UDS}}$, $\overline{\text{LDS}}$ R/$\overline{\text{W}}$ $\overline{\text{EIPL0}}$, $\overline{\text{EIPL1}}$, $\overline{\text{EIPL2}}$ $\overline{\text{EBG}}$ $\overline{\text{RTS1}}$, $\overline{\text{CTS1}}$		Tri-state output

Chapter 3 Address Decoder

3.1 Outline

The address decoder generates two $\overline{CS}$ (chip select) signals to select memory or external devices. The registers in the address decoder can freely select the $\overline{CS}$ memory area from the 16M bytes of address space available to the core processor. The size of the memory area can also be modified.

The address decoder can freely allocate, in boundary units of 1K byte, the internal peripheral circuit registers to the 16M bytes of address space.

Moreover, to allow the decoder to monitor the bus cycles in all memory areas, the address decoder includes a function to automatically generate $\overline{BERR}$ (bus errors) after a set timeout time.

3.2 Area Selection

The address decoder decodes the addresses output from the core processor at each bus cycle. The decoder checks for access to the two memory areas and access to the register area. When a memory area is accessed, the address decoder externally asserts the chip select signal ($\overline{CSn}$).

When a register area is accessed, the address decoder selects an associated internal peripheral circuit and transfers the data. An internal $\overline{DTACK}$ signal is automatically generated and output to the core processor. At this time, a $\overline{DTACK}$ signal from an external source is ignored.

When a register area is accessed, a valid signal is output to the external pins same as when a memory area is accessed. In the read cycle, if external memory overlaps a register area, the data from the internal peripheral circuit are valid; the data from the external source are ignored. In the write cycle, take care when external memory overlaps a register area, as the data are written to both the register area and external memory. To avoid writing to both the register and external memory areas, either ensure that the register area and the external memory do not overlap, or select the external memory using the $\overline{CSn}$ signal. If the memory area selected using the $\overline{CSn}$ signal overlaps with the register area, the access area is determined according to the area priority in Table 3.1. When the register area is accessed, the $\overline{CSn}$ signal is not asserted, thus avoiding accessing the same memory area.

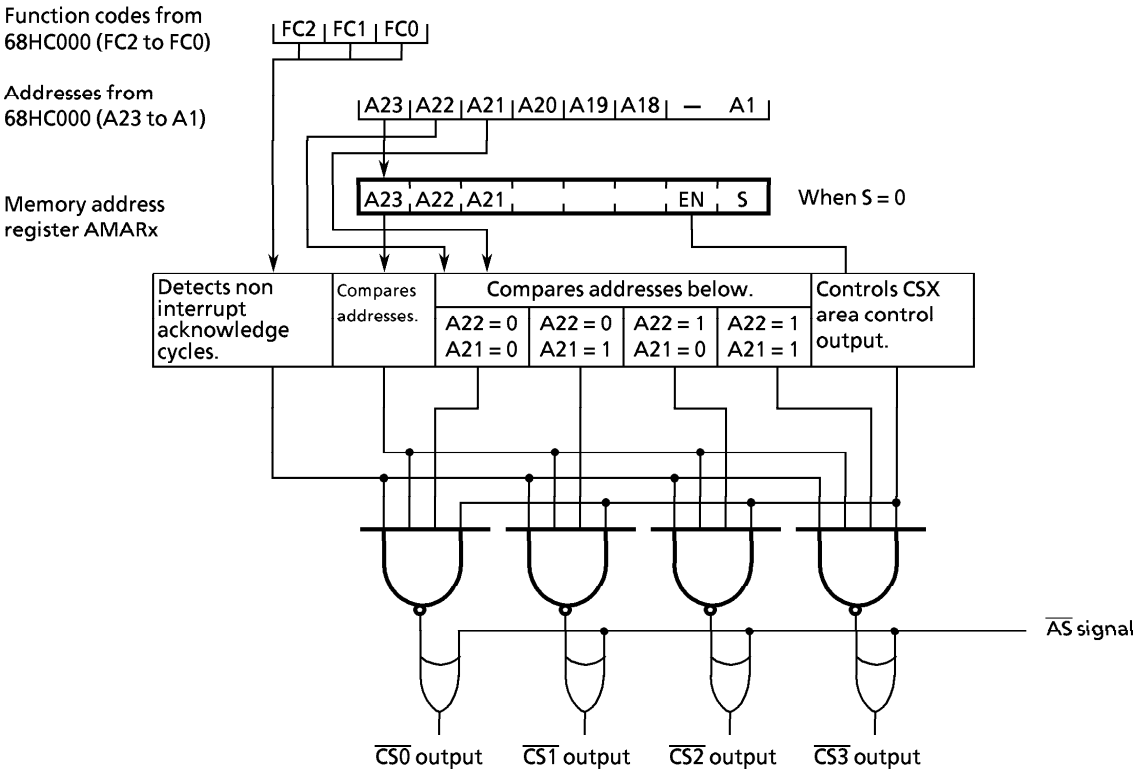
Table 3.1 Area and Access Priority

Area Priority	
Register area	High
	|
Memory area 0, 1, 2, 3	Low

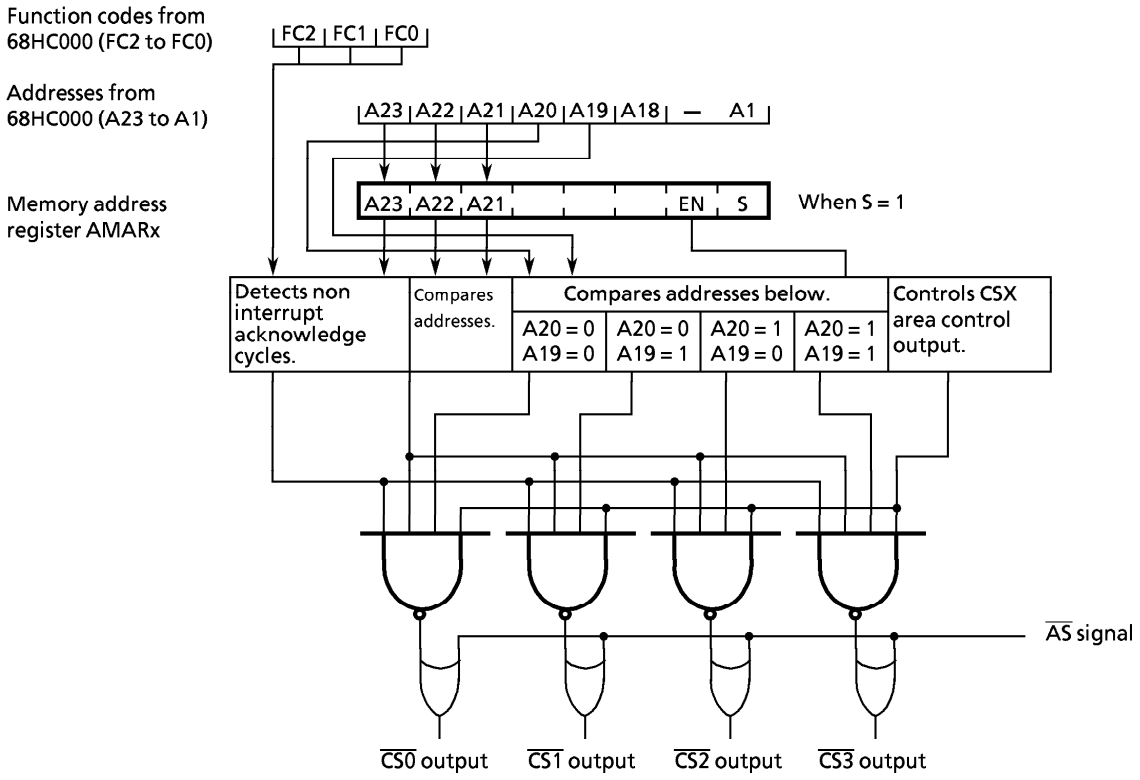
In the interrupt acknowledge cycle (IACK cycle), the core processor starts loading the vector number. At this time, no area selection signal is generated, even when the address generated attempts to access an area. (No register area access signal or CSn signal is asserted.)

This prevents inadvertently accessing areas during the IACK cycle.

Figure 3.1 outlines $\overline{CSn}$ signal generation.



(a) When bit S = 0



(b) When bit S = 1

Figure 3.1 Chip Select Signal Generation

Figure 3.2 shows the output status of area selection signals for overlapping areas.

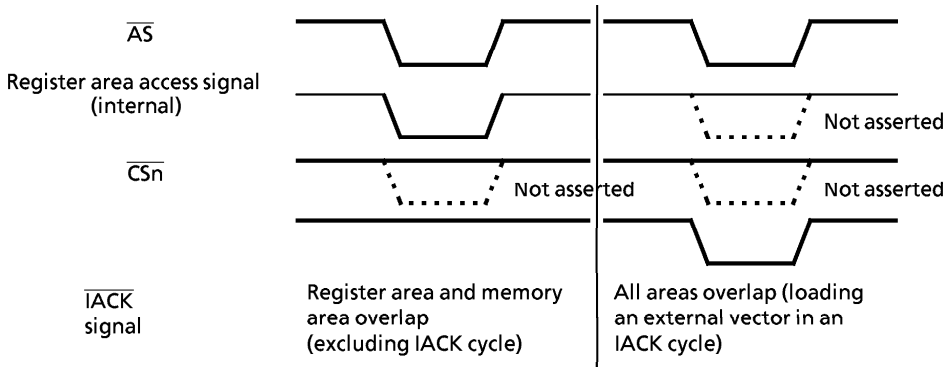


Figure 3.2 Output Status of Area Selection Signals

Figure 3.3 shows the relationship between the $\overline{CSn}$ signal and the $\overline{AS}$ signal.

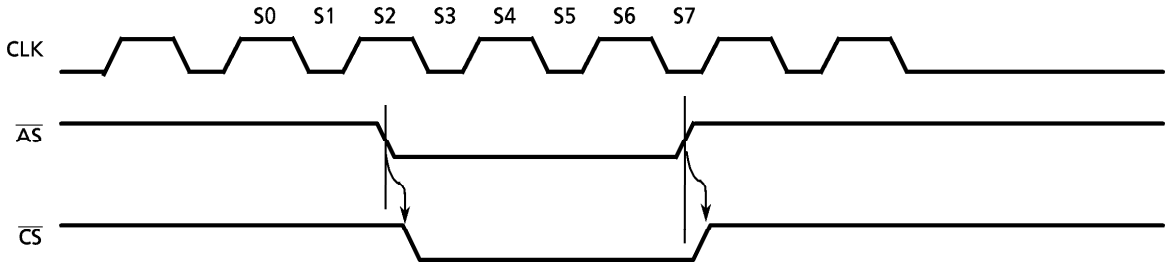


Figure 3.3 $\overline{CS}$ Signal and $\overline{AS}$ Signal Relationship

3.2.1 Memory Area

The memory area register (AMAR) specifies the start address in the memory area. The memory area is divided into four. Every bus cycle, the address decoder compares the address on the bus and the value in the address register. If the values match, the address decoder determines that the memory area is accessed.

The memory address register S bit determines the size of the memory area. Either 2M-byte (one area is 512K bytes) or 8M-byte (one area is 2M bytes) area can be selected.

3.2.2 Register Area

The register area contains internal peripheral circuit registers, including the address decoder registers. The size of the register area is 1K byte. The start address of the register area is specified by the relocation register. Accordingly, the register area can be freely allocated at 1K-byte boundaries within the 16M-byte address space. Figure 3.4 shows actual address generation methods for the registers. The register area after reset release is allocated to $FFFC00 to $FFFFFF (last part of address space). (Figure 3.5.)

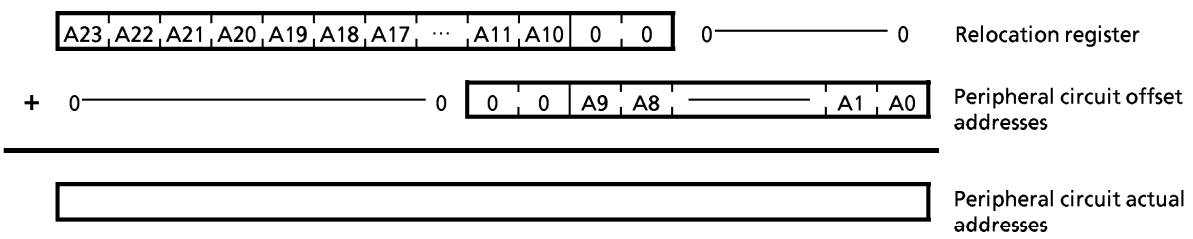


Figure 3.4 Real Register Address Generation

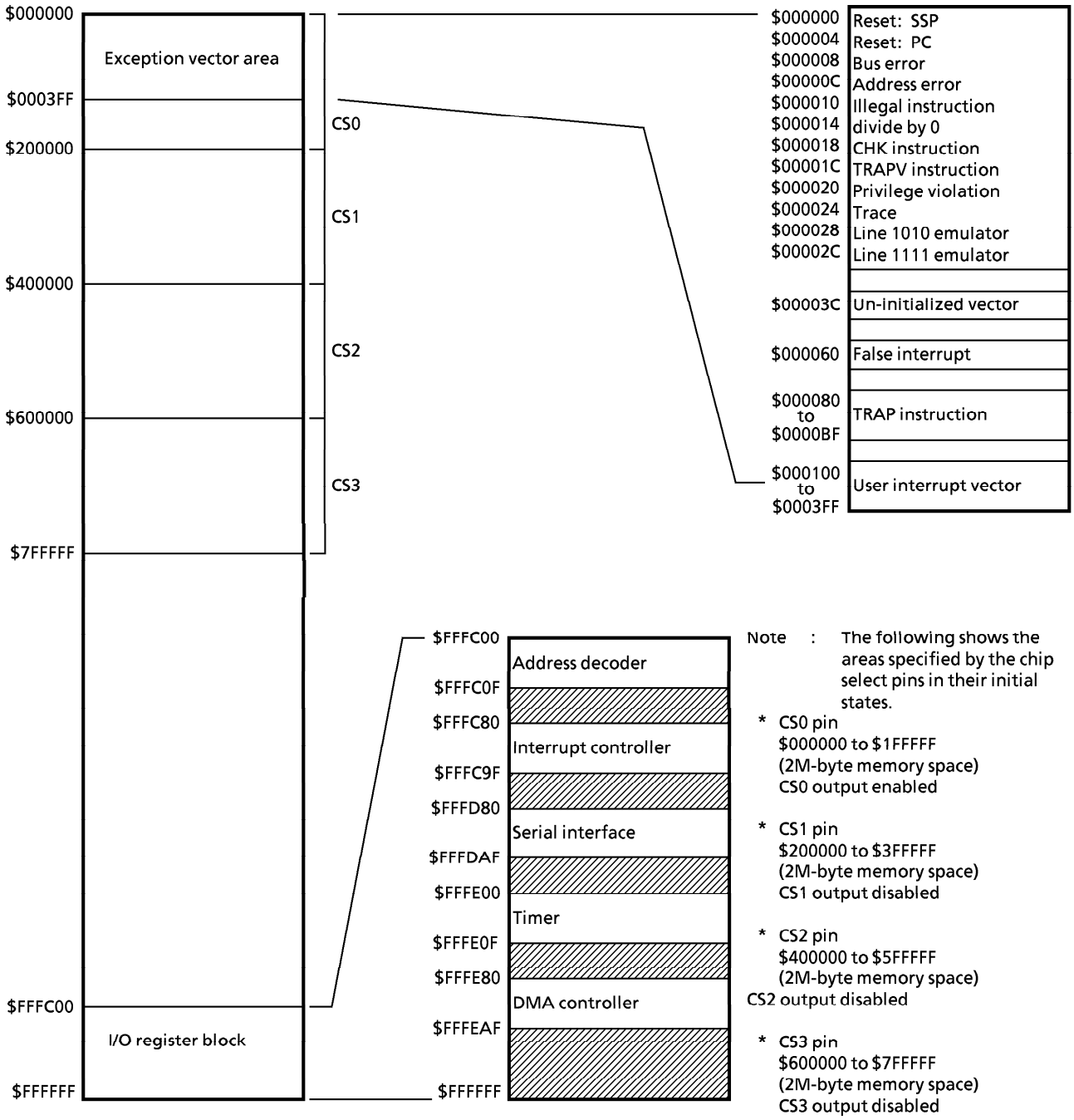


Figure 3.5 Memory Map After Reset Release

3.3 Bus Cycle Monitor

For an address with no memory allocated in the system, the bus cycle stops without returning the $\overline{\text{DTACK}}$ signal. If the length of the bus cycle is abnormal, an address decoder function generates a $\overline{\text{BERR}}$ (bus error) signal. The length of the bus cycle until bus error generation can be selected among 32, 64, 128, or 256 clocks.

To use bus cycles above 256 clocks, an externally generated $\overline{\text{BERR}}$ signal can be validated by disabling generation of the $\overline{\text{BERR}}$ signal by the address decoder. Figure 3.6 illustrates the $\overline{\text{BERR}}$ signal generation circuit. Figure 3.7 shows the $\overline{\text{BERR}}$ signal generation timing.

This bus cycle monitoring applies to all memory areas regardless of the memory area of $\overline{\text{CSx}}$.

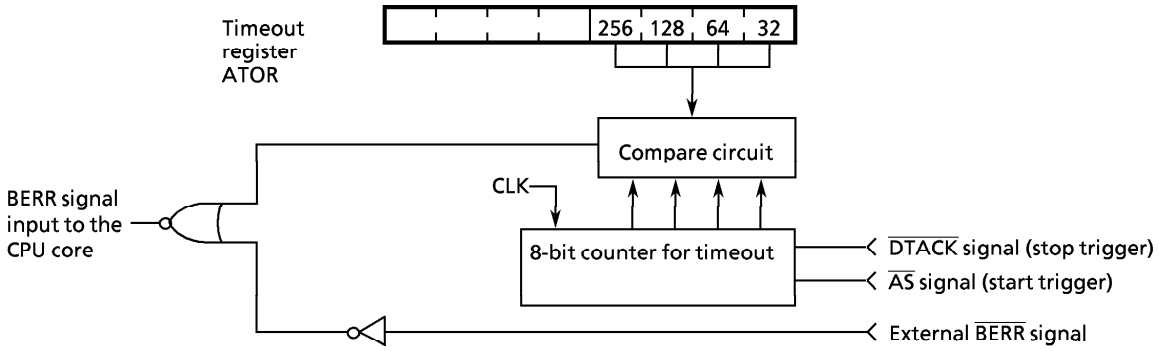


Figure 3.6 $\overline{\text{BERR}}$ Signal Generation Circuit

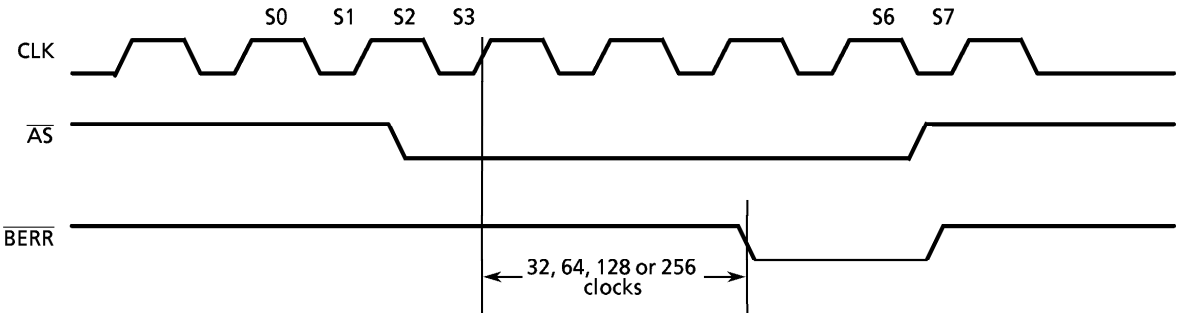


Figure 3.7 $\overline{\text{BERR}}$ Signal

3.4 Register Configuration

3.4.1 Memory Address Register (AMAR)

This register specifies the start address and the size of the memory area. Addresses are specified in three bits, A23 to A21. The start address of the memory area can only be set within the boundary of the memory area, that is, within 2M or 8M bytes. The size of the memory area is specified in bit S of this register.

After reset, AMAR is $02 (start address: $000000, size: 8M bytes, area enable).

Offset address

$000	A23	A22	A21				EN	S	AMAR
	0	0	0	0	0	0	1	0	
				R	R	R			

R : Read Only

EN : Area control
- 0 : Area disable ($\overline{CSn}$ signal not output)
- 1 : Area enable ($\overline{CSn}$ signal output)

S : Area size
- 0 : 8M bytes (2M bytes×4)
- 1 : 2M bytes (512K bytes×4)

3.4.2 Timeout Register

This register specifies the time until $\overline{BERR}$ generation. The time can be selected from among 32, 64, 128, or 256 clocks. If more than one bit is set, only the bit corresponding to the longest time remains set and the other bits are cleared. If no bit is set, no $\overline{BERR}$ is generated by the address decoder. After a hardware reset, the timeout register is set to $08 (256 clocks).

Offset address	7	6	5	4	3	2	1	0	
$00B					256	128	64	32	ATOR
Reset	0	0	0	0	1	0	0	0	
	R	R	R	R					

R : Read only

Set value				Time until $\overline{BERR}$ generation
256	128	64	32	
0	0	0	0	No $\overline{BERR}$ generation
0	0	0	1	Generated after 32 clocks
0	0	1	X	Generated after 64 clocks
0	1	X	X	Generated after 128 clocks
1	X	X	X	Generated after 256 clocks

X: Any value

3.4.3 Relocation Register

This register specifies the start address of the internal register block. Specifiable addresses are A23 to A10. Accordingly, the register block can only specify in 1K-byte boundaries. After a hardware reset, the relocation register is set to $FFFC.

Offset address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$00C	A23	A22	A21	A20	A19	A18	A17	A16	A15	A14	A13	A12	A11	A10			ARELR
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0 R	0 R	

R : Read only

3.5 Bus Cycles Based on External Bus Master

When an external device other than the core processor becomes the bus master, the address decoder functions as if the core processor was the bus master. As a result, the external bus master can access the internal registers of the address decoder. The timing of the bus cycle generated by the external bus master must match the specifications of the bus cycle generated by the core processor. If the timing does not match these specifications, the internal register contents may be overwritten in the write cycle generated by the external bus master.

Chapter 4 Interrupter Controller

4.1 Overview

The interrupt controller provides nine interrupt channels. Five of the channels are for internal peripheral circuits, while the other four are for external interrupts from interrupt request input pins INT0, 1, 2, and 3. Levels of interrupt requests input to the core processor (the pattern of 0s and 1s input to the core processor pins IPL0, 1, and 2) can be independently set for each channel, thus, priorities can be changed freely. The interrupt request input mode (input level, rising/falling edge) for external interrupts can also be set freely. In addition, the external interrupt vector number can be selected as either an internal vector number in the interrupt controller, or an externally input vector number.

If an external vector is input, the IACK output (IACK0, 1, 2, and 3) corresponding to each external interrupt channel is asserted in accordance with the interrupt acknowledge cycle.

The auto vector interrupt function supported by TMP68HC000 is not available because TMP68305 does not have 68000 interface signals.

Figure 4.1 is a block diagram of the interrupt controller.

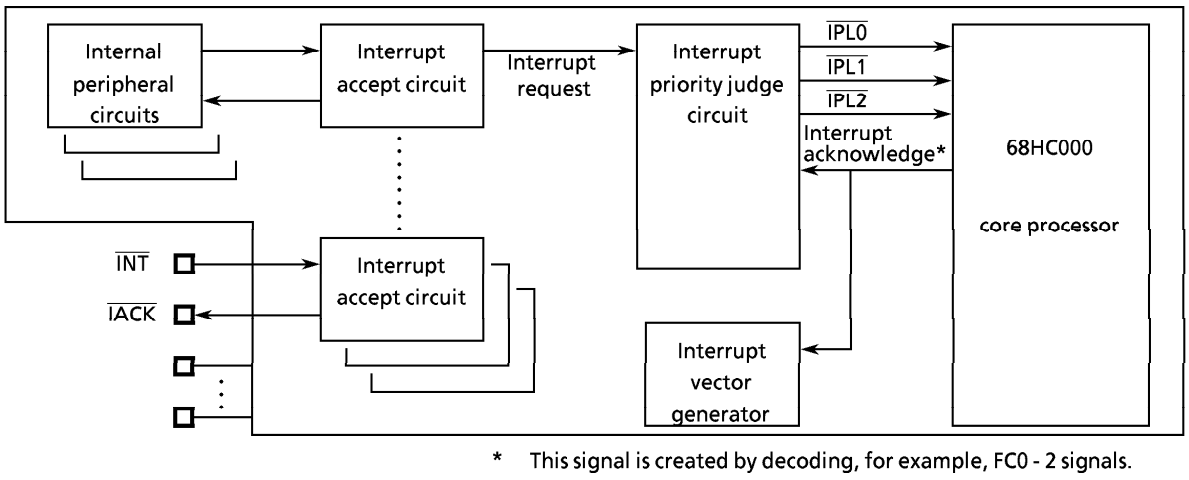


Figure 4.1 Interrupt Controller Block Diagram

4.2 Interrupt Request

When there is an interrupt request to an interrupt channel, the interrupt controller uses the internal $\overline{IPL0}$ to $\overline{IPL2}$ signals (not output to external pins in normal mode) to issue an interrupt request to the core processor at the previously set interrupt level. If requests occur to more than one channel at the same time, the request with the highest priority level is issued.

The following input modes can be selected for external interrupt requests (interrupts using INT0, 1, 2, or 3).

- Low-level interrupt
- High-level interrupt
- Rising-edge interrupt
- Falling-edge interrupt

Interrupt request inputs (INT0, 1, 2, or 3) are detected at the falling edge of the system clock. Level interrupts must be asserted until the $\overline{IACK}$ output ($\overline{IACK0}$, $\overline{1}$, $\overline{2}$, or $\overline{3}$) for the channel corresponding to the interrupt request is asserted. Edge interrupts must be held at the same state for at least two system clocks after the edge is detected to prevent malfunction due to noise.

Note : When the mode switches from level to edge, interrupt requests (pending bits) received in level mode are not cleared. To avoid this, clear any interrupt requests as follows:

1. Mask interrupt requests.
2. Switch from level mode to edge mode.
3. Clear the pending bits.
4. Release the interrupt request mask.

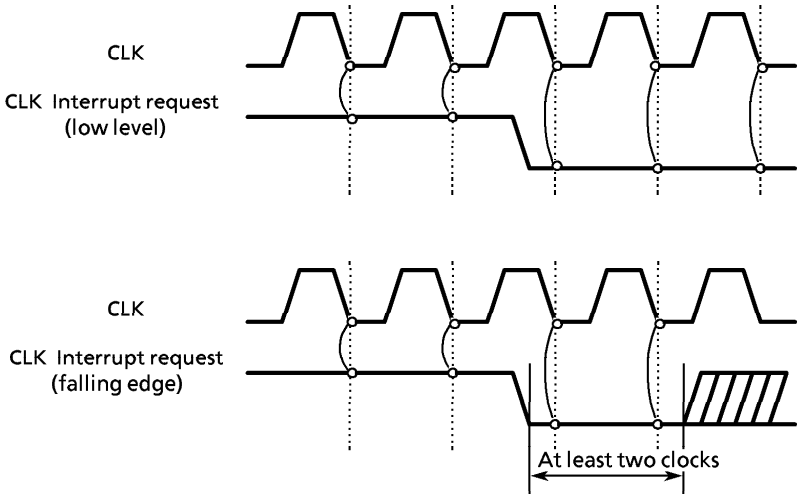


Figure 4.2 External Interrupt Requests

4.3 Priority Between Channels

The interrupt controller can set an interrupt level of IPL0, 1, and 2 for interrupting the core processor using the level bit of the interrupt control register for each channel. This sets the relative priority of each channel. The following priority applies if more than one channel is set to the same interrupt level:

Priority	Channel	
High	External interrupt request	Channel 0
	DMA controller	Channel 0
	Timer	Channel 0
	External interrupt request	Channel 1
	Serial interface	Channel 0
	DMA controller	Channel 1
	External interrupt request	Channel 2
	Serial interface	Channel 1
Low	External interrupt request	Channel 3

Table 4.1 Priority of Channels Set to the Same Interrupt Request Level

4.4 Interrupt Acknowledge Cycle (IACK Cycle)

In 68000 interrupt processing, if an interrupt is accepted, the interrupt acknowledge cycle (IACK cycle) is generated. The level of the interrupt request is output to addresses A1 to A3 and the corresponding interrupt vector number is read from data bus D0 to D7.

TMP68305 internal interrupt controller has a function to automatically generate the vector number read by the core processor during the IACK cycle. This function allows the interrupt controller to automatically perform the above interrupt processing. Also, when an external interrupt is accepted, the interrupt controller can assert the $\overline{\text{IACKn}}$ signal corresponding to the interrupt and obtain the vector externally.

If more than one channel issues requests at the same level, the interrupt controller asserts an interrupt acknowledge signal for the channel with the highest priority in Table 4.1. The interrupt acknowledge signal asserted here only applies internally to 68305 and is not output externally. However, with external interrupts, interrupt acknowledge signals can be output externally as $\overline{\text{IACK}}$ signals. $\overline{\text{IACK}}$ signals are asserted with the same timing as $\overline{\text{AS}}$ signals.

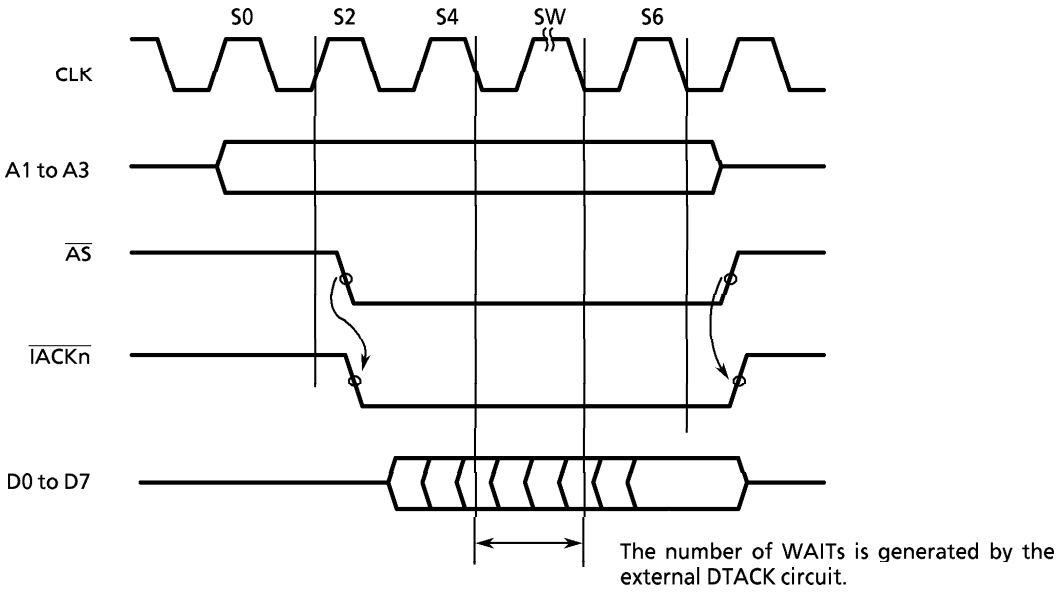


Figure 4.3 $\overline{\text{IACK}}$ Signals for Obtaining External Vector for External Interrupt

4.5 Automatic Generation of Vector Numbers

The interrupt controller automatically generates vector numbers during IACK cycles and these are read by the core processor. The lower five bits of the vector number are determined by the interrupt channel and interrupt cause. The upper three bits are set by the interrupt vector number register (IVNR). Table 4.2 lists vector numbers.

With external interrupt requests, automatic generation or external vector input can be selected by setting the vector number generation mode bit (V bit) of the interrupt control register (ICR0, 1, 2, or 3).

Serial interface and DMA controller interrupt requests generate vector numbers not only by the channel generating the interrupt, but also the interrupt cause.

If an external bus master obtains bus mastership while responding to an interrupt with vector number generation set, inputting the $\overline{\text{AS}}$ signal from the external bus master without any change may output an interrupt vector number from the interrupt controller. In this case, data may contend on the external data bus. Configure the hardware so that $\overline{\text{AS}}$ signals from an external bus master are not input.

Channel	Cause	Vector number
External interrupt Channel 0		XXX00000
External interrupt Channel 1		XXX00001
External interrupt Channel 2		XXX00010
Timer 0 Channel 0		XXX00100
Serial interface Channel 0	Receive error	XXX01000
	Receive complete, break detect	XXX01001
	Transmit ready	XXX01010
	Interrupt cause cleared while interrupt pending (Note 1)	XXX01011
Serial interface Channel 1	Receive error	XXX01100
	Receive complete, break detect	XXX01101
	Transmit ready	XXX01110
	Interrupt cause cleared while interrupt pending (Note 1)	XXX01111
DMA controller Channel 0	Error	XXX10000
	Service complete	XXX10001
DMA controller Channel 1	Error	XXX10000
	Service complete	XXX10001
Others	Default vector (Note 2)	XXX11111

XXX : Set by the upper three bits of the IVNR.

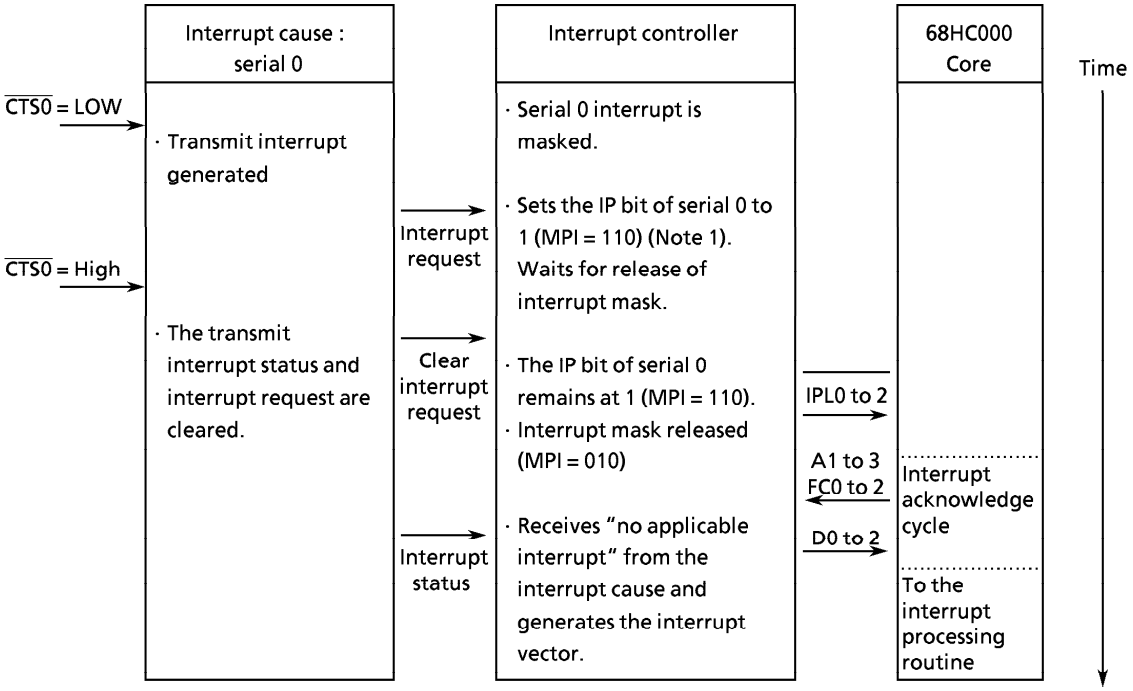
Table 4.2 Vector Number List

Note1 : This vector number is generated because the interrupt cause becomes unclear when the interrupt cause is cleared before the interrupt acknowledge for a pending interrupt is returned.

Note2 : If the CPU accepts an interrupt, the IACK cycle starts when the following instruction is completed. However, if that instruction masks the interrupt, the cause becomes masked and the vector is fixed at 11111 because the vector cannot now be generated in the IACK cycle.

4.5.1 Interrupt Cause Cleared While Interrupt Pending

Interrupts which are cleared during pending occur under the following conditions. The following description is based on the generation of a serial 0 interrupt.



When an interrupt is generated at the interrupt source, an interrupt request is issued to the interrupt controller. As a result, the interrupt controller sets the relevant IP bit to 1. If the relevant interrupt is masked, the interrupt request to the 68HC000 core is pending. (Note 1: MPI represents the bit status of the mask register, pending register, and in-service register for the generated interrupt). During this period, the interrupt condition may be cleared at the interrupt source. (In the above example, when the $\overline{CTS0}$ input changes to high in the transmit interrupt state). Even if the interrupt cause is cleared at the interrupt source, because interrupt is still pending, the interrupt controller sends an interrupt request to the 68HC000 core when the interrupt mask is released. This starts the interrupt sequence and generates the interrupt vector. Because the vector reflects the state of the interrupt source, "no applicable interrupt" indicates that the interrupt cause was cleared at the interrupt source while the interrupt was pending, as mentioned previously. For serial interfaces, these kinds of interrupts occur under the following conditions:

(a) The interrupt is masked by the serial mode register (SMRn) while it is pending.

(b) The $\overline{CTS0}$ input changes to high while a transmit ready interrupt is pending. (When $\overline{CTS0}$ changes to high, TxRDY is set to 0, clearing the transmit interrupt cause).

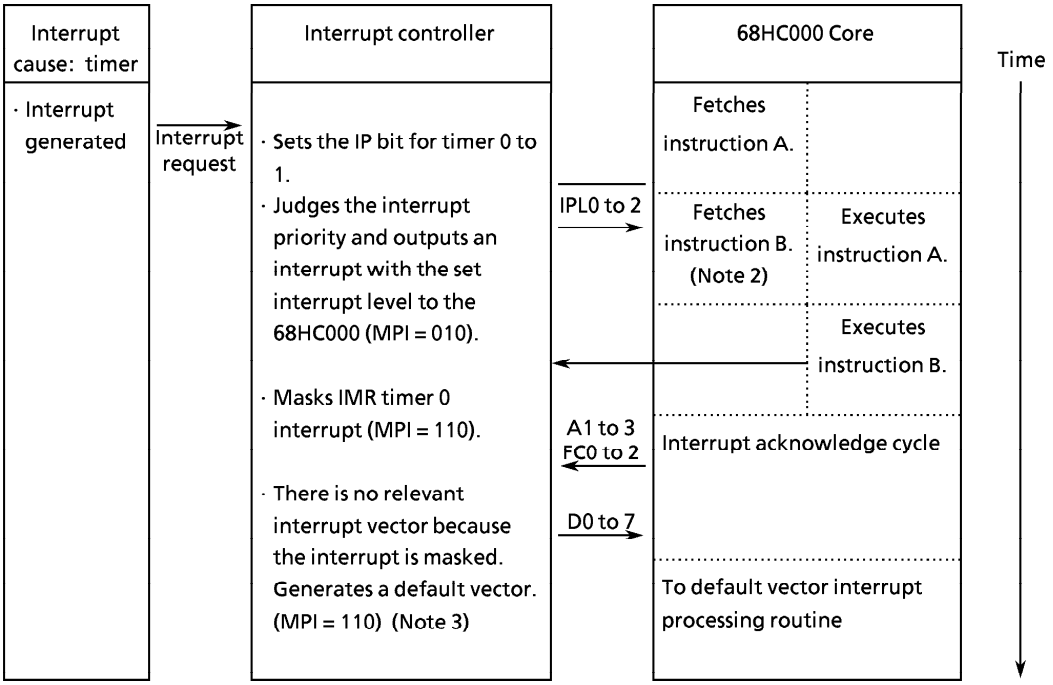
(c) In the serial command register (SCMRn), TxEN is set to 0 while a transmit ready interrupt is pending, or RxEN is set to 0 while a receive complete interrupt is pending.

(d) The receive buffer is read by an error interrupt processing routine.
(If both ERINTM and RxINTM of the serial mode register (SMRn) are set to 0 and both interrupt masks are cleared, both error interrupt and receive interrupt are generated when an error occurs. Because the priority of the error interrupt is higher, the error interrupt processing routine is executed first. If the serial data register (SDRn) is read by this processing routine, the pending receive interrupt is cleared.)

These interrupts occur due to software processing problems as described above. Therefore, find out the causes and write software which does not generate these interrupts.

4.5.2 Default Vector Interrupts

Default vector interrupts occur under the following conditions. The following description is based on the generation of a timer 0 interrupt.



When an interrupt is generated at the interrupt source, an interrupt request is issued to the interrupt controller. As a result, the interrupt controller sets the relevant IP bit to 1, checks that the interrupt has the highest priority, and outputs the interrupt with the set interrupt level to the 68HC000 core ($\overline{IPL0}$ to $\overline{2}$ output). The internal status of the interrupt controller at this time is as follows: mask bit = 0 (M=0), pending bit = 1 (P=1), and in-service bit = 0 (I=0). (Note 1: MPI represents the bit status of the mask register, pending register, and in-service register for the generated interrupt).

When the 68HC000 core accepts the IPL0 to 2 interrupts, it attempts to jump to the interrupt sequence operation. However, the interrupt sequence is delayed until instruction B, for which the 68HC000 core has completed the fetch, is executed. If instruction B masks the generated interrupt (Note 2), the instruction is executed and the interrupt controller interrupt request is cancelled. Subsequently, even if the 68HC000 core starts the interrupt acknowledge cycle, the interrupt controller has no corresponding interrupt. In such a case, the interrupt controller generates a default vector indicating that there is no relevant interrupt and completes the interrupt acknowledge cycle.

Even if default vector interrupts are generated, interrupts generated remain in the interrupt controller in the pending state (Note 3). Accordingly, after clearing the interrupt mask, a normal interrupt sequence can be obtained.

To simply return to the main processing and disable the vector generation before this default interrupt occurs, insert an instruction to set the 68HC000 core interrupt level to the highest level before the interrupt mask instruction (instruction B). This prevents the 68HC000 core from receiving interrupts, apart from interrupt level 7, thus preventing initiation of the interrupt sequence midway through masking the interrupt. However, if the interrupt generated is level 7, the 68HC000 core cannot disable the interrupt request and the default vector is generated. In this case, default vector processing is performed.

4.6 Interrupt Status

The interrupt status for each channel is represented by the mask register (IMR), pending register (IPR), and in-service register (IISR) bits corresponding to the channel. The meaning of each bit is described below. The set (setting the bit to 1) and reset (resetting the bit to 0) methods vary according to each bit.

Mask bit (M)	Control bit for masking interrupt requests
1	Masks the interrupt request.
0	Does not mask the interrupt request.
set	Hardware reset Sets to 1 by software.
reset	Sets to 0 by software.

Pending bit (P)	Bit indicating an interrupt request was generated and is pending (waiting for interrupt processing)
1	An interrupt request was generated and is pending.
0	No interrupt request was generated.
set	An interrupt request was generated (this bit cannot be set to 1 by software).
reset	Hardware reset Sets to 0 by software when the interrupt request is accepted by the core processor (see notes).

In-service bit (I)	Bit indicating whether an interrupt request is accepted by the core
1	Indicates that an interrupt request is accepted.
0	Indicates that no interrupt request is accepted.
set	An interrupt request is accepted by the core processor (this bit cannot be set to 1 by software)
reset	Hardware reset Sets to 0 by software.

Note1 : The function for clearing pending bits by software is used when initializing the whole system or when pending bits are set by unnecessary interrupts. However, if pending bits corresponding to interrupts generated by internal peripheral circuits are cleared to 0 while set to 1, subsequent interrupts will not be accepted. This is because the pending bit is only set when the interrupt request from the interrupt source changes from 0 to 1 (when an interrupt is generated). To re-enable interrupts after the pending bit has been cleared, the interrupt source must also be treated as follows.

Timer

First clear the interrupt request bit (INT bit) of the timer control register (TCRn) to 0, then set to 1.

Serial Interface

First set the interrupt mask bit (INTM bit) of the serial control register (TCRn) to 1, then clear to 0. If, at that time, interrupt requests are generated from channel interrupt generation sources, the corresponding IP bit is set immediately after the INTM bit is cleared to 0. To avoid this, disable interrupt requests for each channel (for example, by setting an interrupt mask for each channel, by reading the receive data, or by performing an error reset) before performing the above procedure.

External Interrupt

With edge interrupts, the IP bit is set the next time the interrupt edge is input. With level interrupts, the IP bit is set the next time the interrupt input is asserted.

Note2 : With level interrupts, clearing the IP bit by software requires first negating the interrupt input. The IP bit cannot be cleared by software while the interrupt input is still asserted.When clearing the pending bit by software, write 1 to all bits except the bit to be cleared. Writing 1 to the pending bit will not affect operation but prevents the pending bit from mistakenly being cleared and interrupts from being disabled.

Table 4.3 shows the interrupt states for the mask bit (M), pending bit (P), and in-service bit (I) values.

M	P	I	
0	0	0	No interrupt request
0	0	1	During interrupt processing routine
0	1	0	Interrupt request generated
0	1	1	A subsequent interrupt request is generated during an interrupt processing routine.
1	0	0	No interrupt request
1	0	1	Interrupt is masked during an interrupt processing routine.
1	1	0	Interrupt request is generated while interrupt is masked.
1	1	1	Interrupt request is generated while interrupt is masked during an interrupt processing routine.

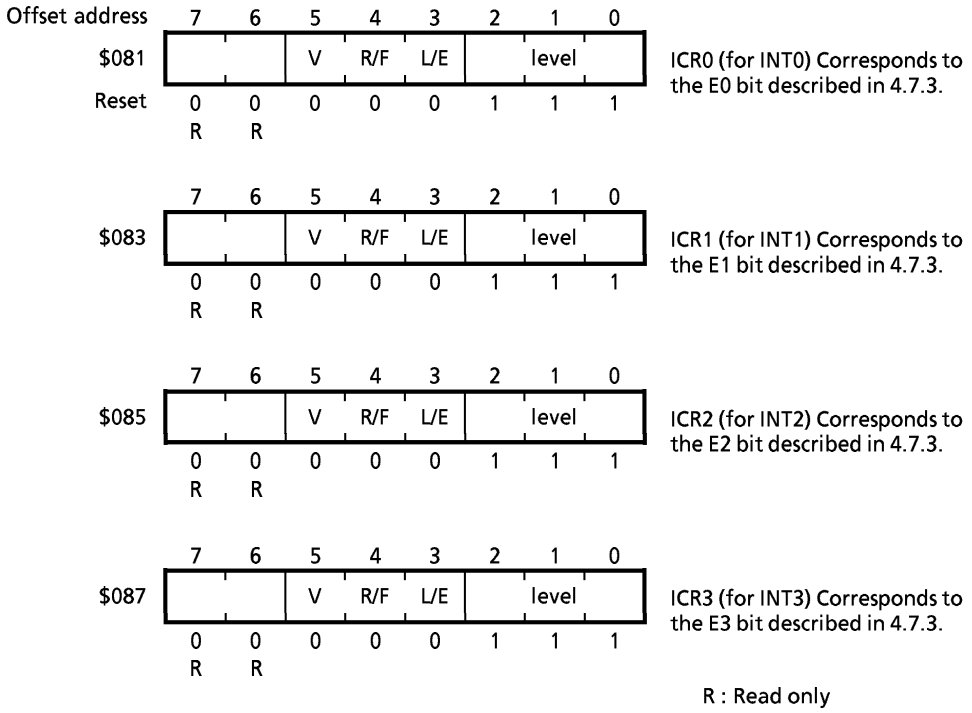
Table 4.3 Interrupt States

4.7 Register Configuration

4.7.1 Interrupt Controller Registers 0 to 3 (ICR0 to 3)

These registers control the external interrupt inputs (INT0, 1, 2, and 3). The registers set input mode and the interrupt level. They also select external vector input or automatic generation of the vector number.

After a hardware reset, ICR0 to 3 are all initialized to $07 (vector number from external source falling-edge mode, interrupt request level 7). These registers can only be written to when the interrupt is masked by the interrupt mask register (IMR).



V : Vector number automatic generation control

0 : Reads vector number from an external source instead of automatically generating vector number.

1 : Automatic vector number generation

R/F, L/E: External interrupt request input mode

R/F	L/E	Interrupt request input mode
0	0	Falling edge
1	0	Rising edge
0	1	Low level
1	1	High level

For example, $\overline{IPL2}=1$, $\overline{IPL1}=0$, and $\overline{IPL0}=0$ indicates request level 3. The following table shows the correspondence between IPLx and interrupt levels.

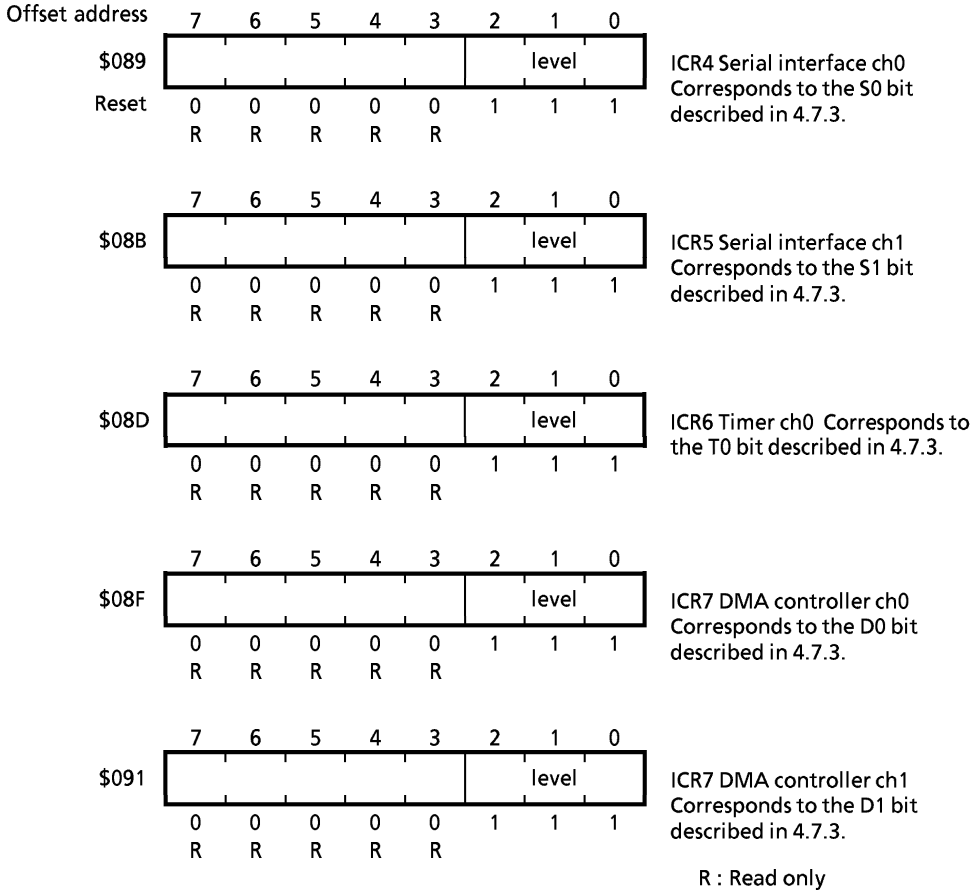
0 to 7 : Indicate the interrupt request level corresponding to $\overline{IPL0}$, $\overline{1}$, and $\overline{2}$ input to the core processor.

Interrupt level	$\overline{IPL2}$	$\overline{IPL1}$	$\overline{IPL0}$
0	1	1	1
1	1	1	0
2	1	0	1
3	1	0	0
4	0	1	1
5	0	1	0
6	0	0	1
7	0	0	0

4.7.2 Interrupt Control Registers 3 to 8 (ICR3 to 8)

These registers control interrupts from internal peripheral circuits.

After a hardware reset, ICR3 - 8 are all initialized to $07 (interrupt request level 7). These registers can be written to only when the interrupt is masked by the interrupt mask register (IMR).



Level : Interrupt request level

0 to 7 : Indicate the interrupt request level corresponding to IPL0, 1, and 2 input to the core processor.

4.7.3 Interrupt Mask Register (IMR)

This register sets the interrupt mask for each channel. 1 masks the interrupt (ignore interrupt). 0 unmasks the interrupt (enable interrupt). When masking an interrupt during operation, mask in the state whereby, even if an interrupt is generated for a channel, it will not be accepted by the core processor. (For example, set the interrupt level in the core processor status register to 7). This is necessary because, if an interrupt occurs during masking, the interrupt to be masked may be generated due to the timing mismatch.

After a hardware reset, this register is initialized to $313F (all interrupt channels masked).

Offset address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$094			D1	D0				T0			S1	S0	E3	E2	E1	E0	IMR
Reset	0	0	1	1	0	0	0	1	0	0	1	1	1	1	1	1	
	R	R			R	R	R		R	R							

R : Read only

D0 : DMA controller channel 1
D1 : DMA controller channel 0
T0 : Timer channel 0
S1 : Serial interface channel 1
S0 : Serial interface channel 0
E3 : External interrupt channel 3
E2 : External interrupt channel 2
E1 : External interrupt channel 1
E0 : External interrupt channel 0

4.7.4 Interrupt Pending Register (IPR)

This register indicates whether there is an interrupt request and that the interrupt request is not yet accepted by the core processor. 1 indicates that there is an interrupt request that has not yet been accepted by the core processor. 0 indicates that there is no interrupt request.

Each bit is automatically cleared when the request is accepted by the core processor. Clearing a bit by software cancels the interrupt request. However, the interrupt request must be cleared at the interrupt request source if subsequent interrupts are to be acknowledged.

After a hardware reset, this register is initialized to $0000 (no interrupt requests).

Offset address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$096			D1	D0				T0			S1	S0	E3	E2	E1	E0	IPR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	R	R			R	R	R		R	R							

R : Read only

4.7.5 Interrupt In-Service Register (IISR)

This register indicates whether or not an interrupt request has been accepted by the core processor. 1 indicates that a request has been accepted. 0 indicates that no request has been accepted.

While operating with the register set to 1 does not affect operation, clear each bit during the interrupt processing routine. (These bits are not cleared automatically).

After a hardware reset, the register is initialized to $0000 (no interrupt requests accepted).

Offset addres	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$098			D1	D0				T0			S1	S0	E3	E2	E1	E0	IISR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	R	R			R	R	R		R	R							

R : Read only

4.7.6 Interrupt Vector Number Register (IVNR)

This register specifies the upper three bits of the vector numbers. The lower five bits of the vector numbers are determined by the interrupt channel and the interrupt cause.

After a hardware reset, the register is initialized to $00.

Note : After releasing a reset, the 68HC000 core interrupt vectors overlap the internal peripheral circuit interrupt vectors. Therefore, set a vector number of $40 or more by software.

Offset addres	7	6	5	4	3	2	1	0	
$09B	Vector								IVNR
Reset	0	0	0	0	0	0	0	0	
				R	R	R	R	R	

R : Read only

Vector : Upper three bits of the vector number

Chapter 5 Serial Interface

5.1 Outline

The serial interface consists of two independent channels, which support full duplex universal asynchronous receiver / transmitter (UART). Both the receive and transmit modules use double buffers and the same data format and baud rate. However, the receive and transmit modules are functionally independent. The data format used is a standard mark/space format. The data format consists of one start bit, between 5 and 8 data bits, and one or two stop bits. The serial interface also supports parity bit processing. The serial interface includes only one prescaler, and one baud rate generator for each channel to generate the baud rate clocks. Thus, independent baud rate clocks are available for each channel. The system clock (CLK) or external clock (input from the BCLK pin) can be selected as the divider clock. Error detect (parity error, overrun error, framing error) is supported. Break detect and break transmit is also supported. If an interrupt cause (receive complete, transmit ready, error occurred, break detected) is generated in any channel, an interrupt request is forwarded to the core processor via the interrupt controller. A separate vector number can be generated in the interrupt acknowledge cycle specifically for the channel and interrupt cause. Channel 1 supports modem control signals.

Figure 5.1 is the serial interface block diagram. Figure 5.2 is a detailed channel diagram.

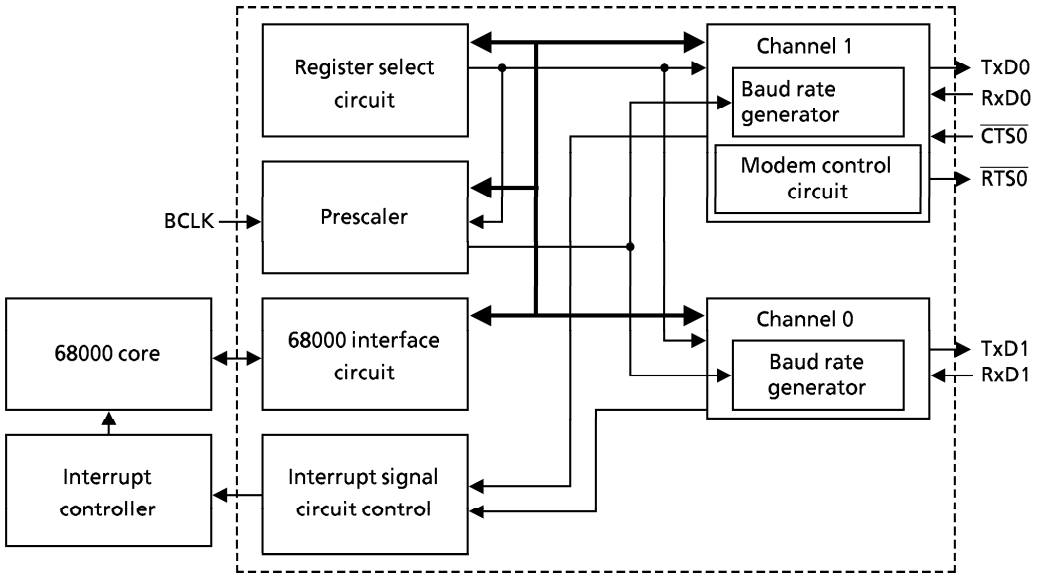


Figure 5.1 Serial Interface Block Diagram

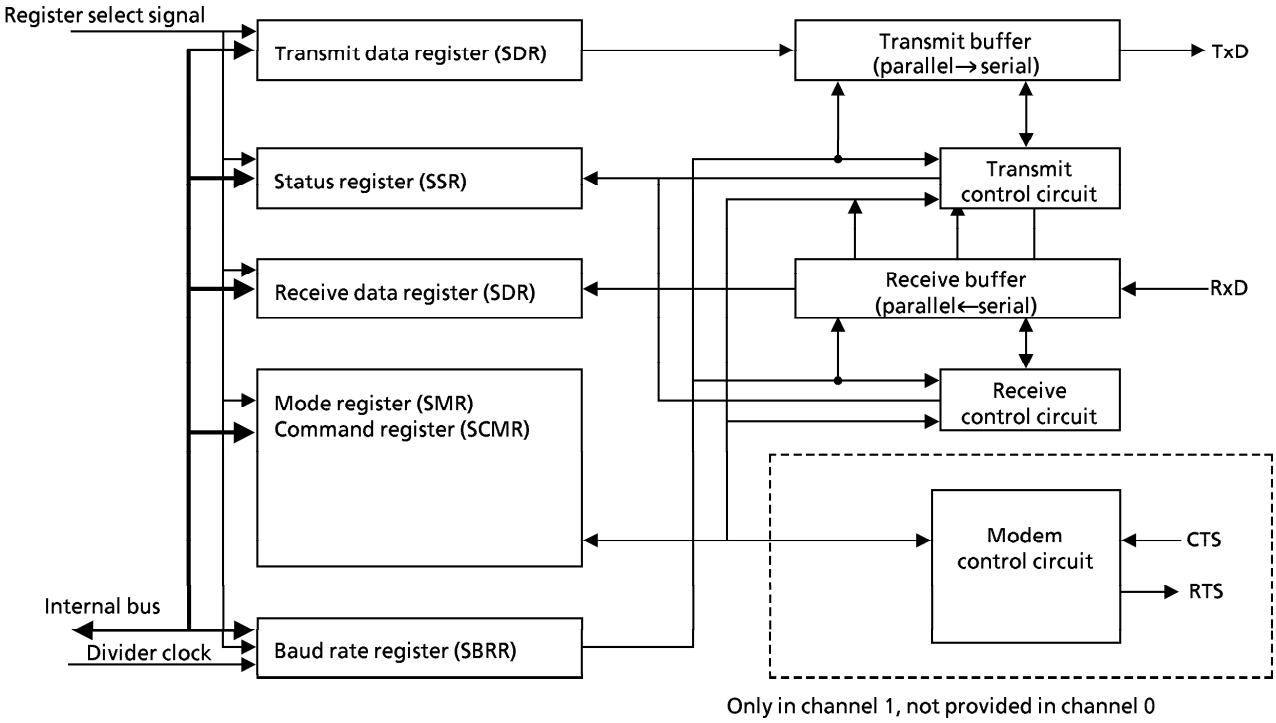


Figure 5.2 Channel Detail Diagram

Note : In the following description, the data register (SDR) is called the transmit data register at transmission, and the receive data register at reception. Although the transmit data register and the receive data register share the same register address, they are independent of each other. At read, the receive data register is accessed. At write, the transmit data register is accessed.

5.2 Communications

5.2.1 Data Format

Figure 5.3 shows the data frame formats input through the serial interface from the receive data input pins (RxD0, RxD1), and output to the transmit data output pins (TxD0, TxD1).

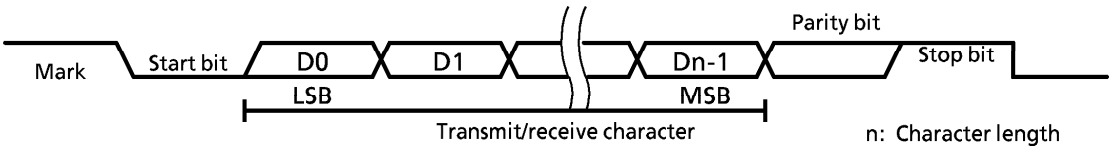


Figure 5.3 Data Frame

① When no data are transmitted or received, the data lines are at level 1 (mark state).

② The transmit (when sending) of the start bit (level 0) or detect (when receiving) of the start bit indicates the start of the data frame.

③ The transmit/receive character immediately follow the start bit, beginning with the least significant bit. The serial mode register (SMR) setting determines the character length n (from five to eight bits).

④ Setting the parity bit in the serial mode register (SMR) sends the parity bit at transmit, and checks the parity after receiving the parity bit at receive.

⑤ The stop bit (level 1) indicates the end of the data frame. At transmit, one or two stop bits are sent in accordance with the serial mode register setting. At receive, regardless of the serial mode register setting, only one bit is identified as the stop bit. The stop bit is the bit following the most significant data bit (last data bit received; when parity bit receive is not set) or the bit following the parity bit (when parity bit receive is set). If the stop bit is at 0 level, a framing error is generated.

⑥ A break consists of two consecutive frames where all data bits, the parity bit, and the stop bits, are at 0 level.

5.2.2 Baud Rate Clock Generation

The serial interface uses an 8-bit prescaler and an 8-bit baud rate generator to divide the system clock (CLK) or external input clock (BCLK) and thereby generate a baud rate clock. Figure 5.4 is the baud rate clock generation circuit block diagram.

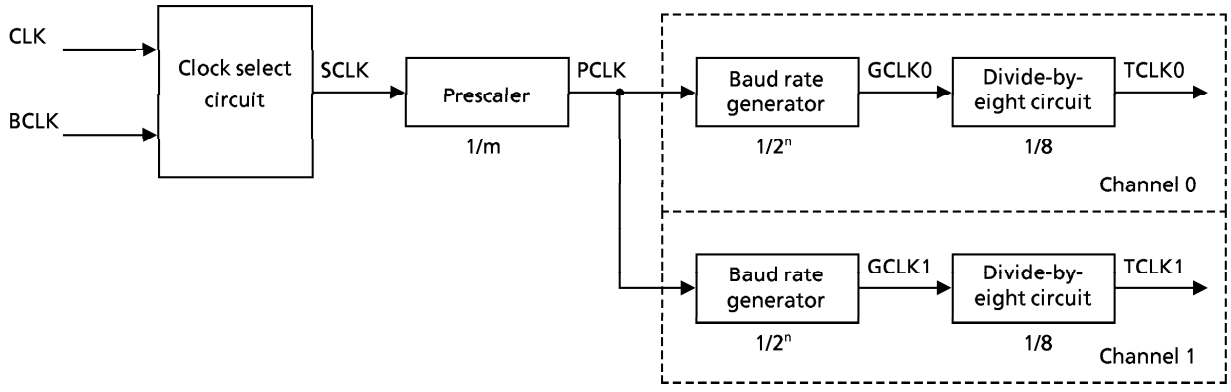


Figure 5.4 Baud Rate Clock Generation Circuit Block Diagram

The input clock select circuit selects either the system clock (CLK) or the external input clock (BCLK). The selected clock (SCLK) is divided by the prescaler by from 1/1 to 1/256 (PCLK), and the PCLK is divided by the baud rate generator by $1/2^n$ (n = 0 to 7) times (GCLK). The GCLK is used as the fundamental clock for serial interface transmit/receive. The GCLK is further divided by eight times (TCLK) for transmitting data frames. One prescaler is built into the serial interface. One baud rate generator is built into each channel.

5.2.3 Data Transmit

When a transmit character is written into the transmit data register (SDR), the serial interface resets the TxE (transmit data register buffer empty) bit of the status register (SSR) to 0 and checks whether transmit with the TxEN (transmit enable) bit of the command register (SCMR) is in progress. When CTS1 is used in channel 1, the serial interface also checks pin CTS1. When CTS1 is not used, if TxEN = 1 and transmit is not in progress, transmit begins with the transmit character sent from the transmit data register to the transmit buffer. When CTS1 is used, if TxEN = 1, CTS1 = 0, and transmit is not in progress, transmit begins in the same way. The transmit is synchronous with TCLK, which is generated by the baud rate generator. Figure 5.5 shows TLCK and the data frame detection timing.

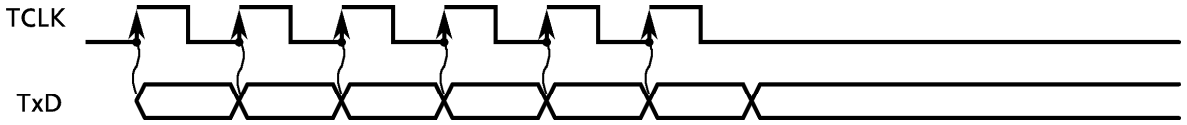


Figure 5.5 Data Frame Transmit Timing

When the serial interface starts transmit, the interface transmits the start bit and sets the TxRDY (transmit ready) bit of the status register (SSR) to 1. As the transmit character is sent from the transmit data register to the transmit buffer at that point, the next transmit character can be written in the transmit data register in advance. When the next transmit character is written in the transmit data register, it is retained in that register until transmission of the current transmit character is complete. After transmitting the start bit, the serial interface transmits the number of data bits specified by the CL1 - CL0 (character length) bits of the mode register (SMR), transmits the parity bit when the mode register PE (parity enable) bit is set to 1, and transmits from the transmit output pin (TxD) the number of stop bits specified by the mode register ST (stop bit length) bit.

If the transmit data register contains a next transmit character, the serial interface transmits the start bit and the next character after transmitting the stop bit of the current data frame. Once the serial interface has begun transmission, it continues transmission of the current data character even if the status of the CTS1 pin or the TxEN bit change and ends transmission with the stop bit.

If the SBRK (break transmit) bit of the mode register is set to 1, the serial interface sends a break until SBRK is reset to 0 regardless of TxEN, CTS1, or whether there is a transmit currently in progress.

5.2.4 Data Receive

The receive data input pin (RxD) is normally set to 1 (mark status). The serial interface samples the input at the GCLK generated by the baud rate generator. When it detects 0 level, it regards the 0 level as the beginning of the start bit. If the serial interface detects the 0 level three more consecutive times, it determines that the first 0 level is the beginning of a valid start bit. Then the serial interface determines the center point of subsequent receive characters, and samples the receive characters. Figure 5.6 shows how the start bit and the center point of receive characters are determined.

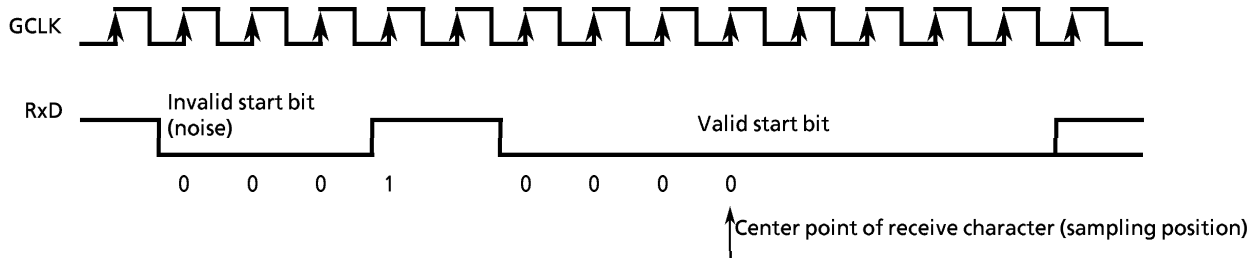


Figure 5.6 How to Determine Start Bits and Center Point of Receive Characters

If the mode register PEN (parity enable) bit is set to 1 and parity bit receive is specified, the serial interface samples the parity bit and compares the parity with that generated by the receive character. If the compared parities do not match, the serial interface sets the PE (parity error) bit of the status register to 1.

For stop bits, the serial interface samples the first stop bit irrespective of the stop bit length specified by the mode register ST (stop bit length) bit. If the result is not 1, it sets the FE bit (framing error) bit in the status register to 1.

The receive data bits are sampled sequentially and saved in the receive buffer. When the programmed number of data bits specified by CL1 - CL0 (character length) of the mode register have been received, the data bits are transferred from the receive buffer to the receive data register (SDR). When the character length is 5, 6 or 7, the upper bits in the receive data register (SDR) are unused. However, these bits are reset to 0 at transfer to the receive data register.

The RxRDY (receive ready) bit of the status register is set to 1 when a character is transferred from the receive buffer to the receive data register (SDR). At this time, the receive data register (SDR) value can be read, even while the next character is being received. If receive of the next character completes before the receive character is read, the new character is transferred from the receive buffer to the receive data register (SDR), overwriting the previous character. With this, the previous character is lost, and the status register OE (overrun error) bit is set to 1.

When two or more data frames are received where the receive characters and, if specified, the parity bit and stop bits, are all 0, the serial interface assumes a break and sets the RBRK (break detect) bit of the status register to 1.

If all error bits (status register bits OE, FE, PE, and RBRK) are set to 1, these settings are retained. Setting the control register ERS (error reset) bit to 1 clears the error bits to 0. Since the error bits remain cleared while the ERS bit is set to 1, subsequent errors cannot be detected. Clearing the ERS bit to 0 sets the error bits in accordance with the status of the receive data frame.

Parity errors and overrun errors do not affect receive, which continues as if no error had occurred. If a break is not detected and a framing error is generated, the serial interface continues receive as if a stop bit was actually present and detects the next start bit. If a break is detected, the start bit detection is suspended until the receive data input pin (RxD) is set once to 1.

5.3 Interrupt Control

5.3.1 Interrupt Causes

The serial interface supports the following four interrupt causes.

a) Transmit ready interrupt

At data transmit, the transmit data register can be loaded with the next transmit character.
TxRDY (transmit ready) = 1

b) Receive complete interrupt

At data receive, the receive character is sent from the receive buffer to the receive data register (SDR); therefore the receive data register is ready to be read.
RxRDY (receive ready) = 1 and RxEN (receive enable) = 1

c) Error occurred interrupt

A parity error (PE), overrun error (OE), or framing error (FE) is generated
(PE = 1, or OE or FE = 1) and RxEN = 1

d) Break detected interrupt

A break is detected.
RBRK (break detected) = 1 and RxEN = 1

5.3.2 Interrupt Mask

Interrupts in the entire serial interface can be masked by the control register INTM (interrupt mask) bit. Each interrupt cause can be masked by the mode register RxINTM (receive complete interrupt mask) bit, the ERINTM (error occurred interrupt mask) bit, or the TxINTM (transmit interrupt mask) bit. Note that break detected interrupts are masked by the RxINTM bit. Moreover, bits S0 and S1 (serial interface channel 0, 1 interrupt mask) of the interrupt control interrupt mask register (IMT) can mask interrupts for each channel.

5.3.3 Interrupt Generation Conditions

The interrupt generation conditions determined by the interrupt cause and mask are as follows.

Note n : Indicates the channel number n = 0, 1
+ : Indicates logical OR
x : Indicates logical AND

Channel n interrupts = (Sn = 0) x (INTM = 0) x{(transmit ready interrupt) + (receive complete interrupt) + (error occurred interrupt) + (break detected interrupt)}
Transmit ready interrupt = (TxINTM = 0) × (TxRDY = 1)
Receive complete interrupt = (RxINTM = 0) × (RxEN = 1) × (RxRDY = 1)
Error occurred interrupt = (RxEN = 1) × [(ERINTM = 0) × {(PE = 1) + (OE = 1) + (FE = 1)}]
Break detected interrupt = (RxINTM = 0) × (RBRK = 1)

5.3.4 Vector Number Generation

TMP68305 automatically generates vector numbers in the acknowledge cycle depending on the interrupt channels and causes. For details of vector numbers, see Table 4.2 in Chapter 4 Interrupt Controller.

The vector number register (IVNR) determines the upper three bits (7 to 5) of vector numbers generated in the acknowledge cycle. Bits 4 to 2 are determined by the channel number and generated by the interrupt controller. Bits 1 to 0 are determined by the interrupt cause and generated by the serial interface. (Figure 5.7)

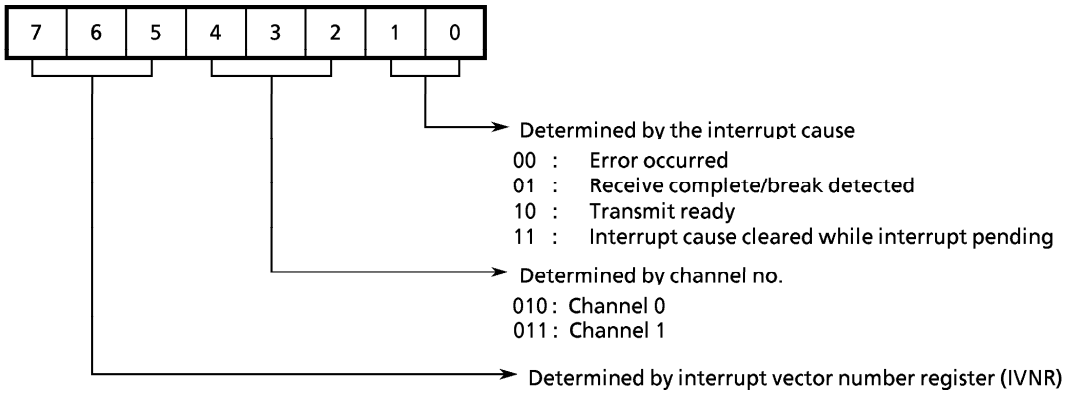


Figure 5.7 Serial Interface Vector Number Generation Method

When an interrupt is sent from the serial interface to the interrupt controller, the interrupt controller holds the interrupt and generates an interrupt request to the core processor. Interrupts pending (corresponding channel bits of the interrupt controller pending register are set to 1) due to masking by bits S2 to S0 (serial interface channel 2 to 0 interrupt mask) of the interrupt mask register (IMR) are accepted after the mask is released; interrupts pending due to execution of another interrupt with a higher priority are accepted after the higher-priority interrupt is fully processed.

When the interrupt is masked by the serial control register INTM bit, or the serial mode register RxINTM bit, ERINTM bit, or TxINTM bit, interrupt generation conditions are not satisfied and no interrupt request is sent to the interrupt controller. The serial interface does not retain masked interrupts. Therefore, if the interrupt cause is lost before releasing the masks of the control register INTM bit, or the mode register RxINTM bit, ERINTM bit, or TxINTM bit, the serial interface acts as if no interrupt is generated.

If conditions for the interrupt are no longer satisfied, the serial interface stops outputting an interrupt request to the interrupt controller. However, the interrupt controller holds the interrupt request and therefore continues to output the interrupt request to the core processor. If the pending bit of the channel corresponding to the interrupt pending register is not cleared (interrupt cancelled) before the core processor receives the interrupt request, the serial interface generates an interrupt acknowledge. When vector numbers are generated in the interrupt acknowledge cycle, numbers 7 to 2 are generated because the interrupt controller holds the interrupt request. However, as the interrupt generation conditions are no longer satisfied, the serial interface generates a vector number indicating "interrupt cause cleared while interrupt pending" using bits 1 to 0 of the vector number used to specify the interrupt cause. Figure 5.8 shows the relationship between the interrupt control circuit in the serial interface and the interrupt controller.

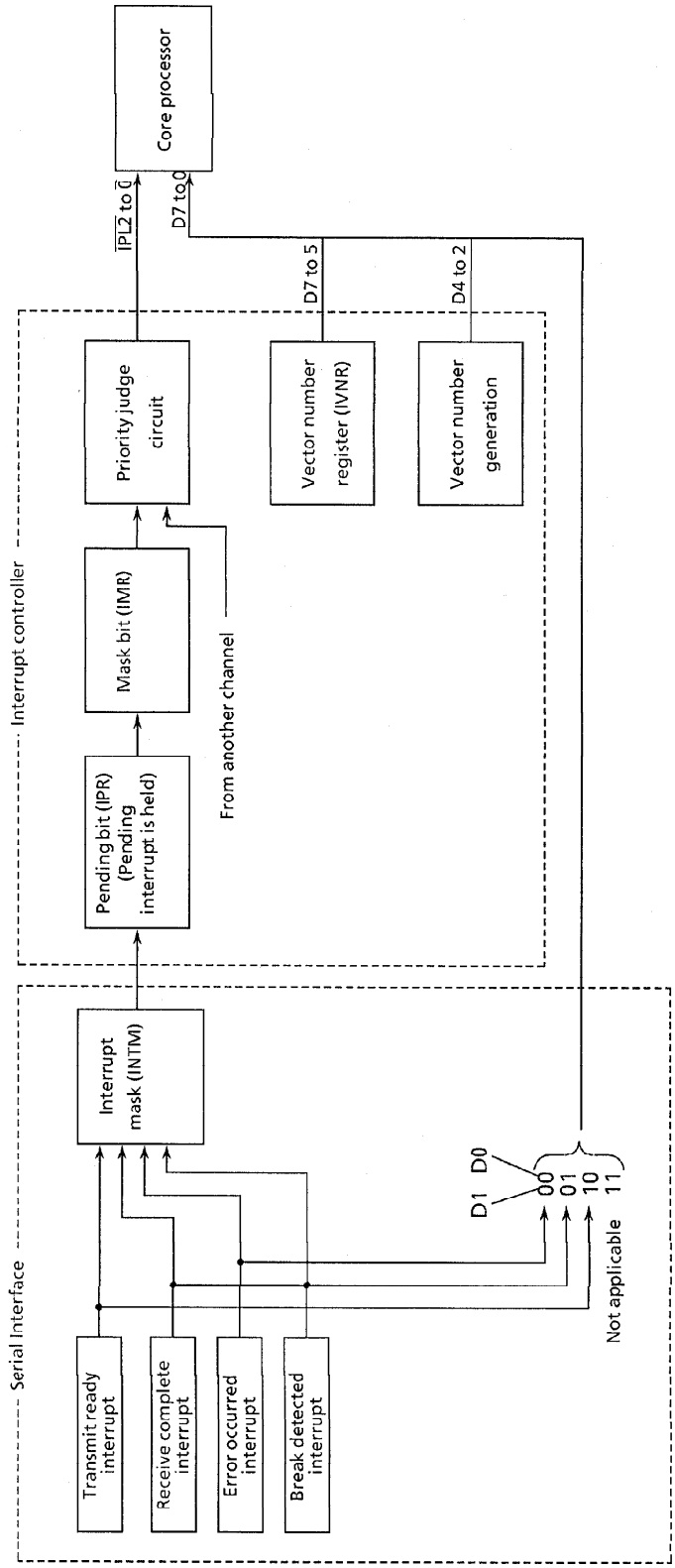


Figure 5.8 Relationship Between Interrupt Control Circuit and Interrupt Controller

5.3.5 Cautions

a) Break detected interrupts are masked by the RxINTM (receive interrupt mask) bit, but create an error occurred interrupt vector in the interrupt acknowledge cycle.

b) The TxRDY bit is still held in the transmit ready interrupt acknowledge cycle.

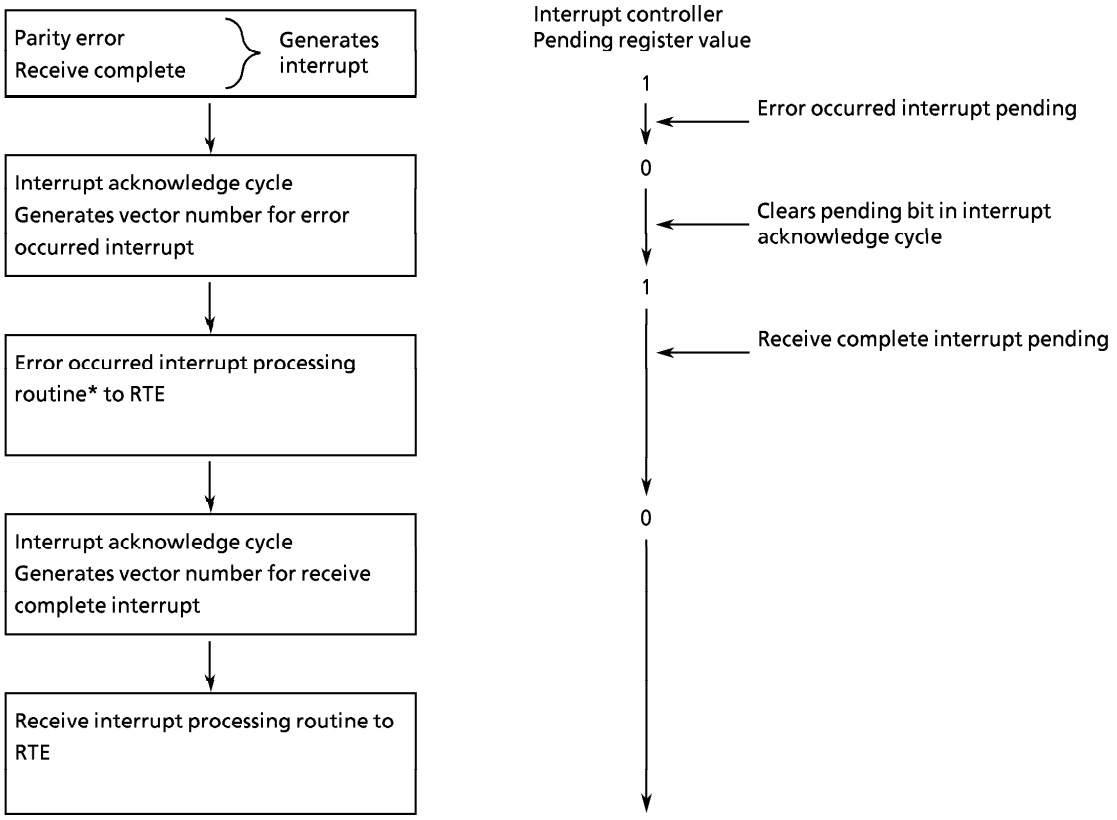
c) The TxRDY bit is initialized by a hardware reset (the $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ pins are simultaneously asserted). Therefore, the transmit ready interrupt can be used for the first data transmit after reset.

d) When interrupt request conditions are satisfied once, an interrupt processing is generated only once. That is, even if the transmit ready interrupt processing routine ends without the transmit data being written to the transmit data register (SDR), an interrupt will not be regenerated to jump to the transmit ready interrupt processing routine. If more than one error occurs simultaneously, the routine jumps only once to the error occurred interrupt processing routine.

e) The following is the interrupt processing priority when more than one interrupt request is generated simultaneously.

Error occurred interrupt -- High
Receive complete interrupt ↓
Transmit ready interrupt -- Low

Figure 5.9 shows an interrupt processing routine example when an error occurred interrupt and a receive complete interrupt are generated simultaneously.



Note* : When a receive error occurs, an error occurred interrupt and a receive complete interrupt are generated simultaneously. Reading the receive data register (SDR) in the error occurred interrupt processing routine resets RxRDY to 0 and prevents the receive complete interrupt generation conditions from being satisfied. Therefore, a vector number indicating "interrupt cause cleared while interrupt pending" is generated in the next interrupt acknowledge cycle.

Figure 5.9 Multiple Interrupt Processing Example

5.4 Initialization of Serial Interface

To initialize serial interface, set the RES (software reset) bit of the control register to 1. The serial interface is also initialized if a hardware reset is executed (the RESET and HALT pins are simultaneously asserted) because the RES bit is set to 1. The initialized status is held until the RES bit is reset to 0. In the initial state, transmit, receive, and interrupts are all disabled. The generation of superfluous operations can thus be avoided until the serial interface setup is complete.

Table 5.1 lists the initial register values.

Register	7	6	5	4	3	2	1	0
Serial control register (SCR)	CKSE		RES					INTM
	1	–	1	–	–	–	–	1
Serial command register (SCMR0 to 2)			RTS *1	ERS	SBRK	R_XEN	CTSEN *1	T_XEN
	–	–	0	1	0	0	0	0
Serial status register (SSR0 to 2)	–	RBRK	FE	OE	PE	T_XE	R_XRDY	T_XRDY
	–	0	0	0	0	1	0	1

– : Undefined
*1 : channel 1 only

Table 5.1 Serial Interface Register Initial Values

5.5 Register Description

5.5.1 Serial Control Register (SCR)

This register is used to make the serial interface basic settings.
Set the registers after resetting the RES bit to 0.

Offset address	7	6	5	4	3	2	1	0	
$18F	CKSE		RES					INTM	SCR
Reset	1	0	1	0	0	0	0	1	
		R		R	R	R	R		R : Read only

Note : Set bits which are reserved for future expansion and undefined to 0.

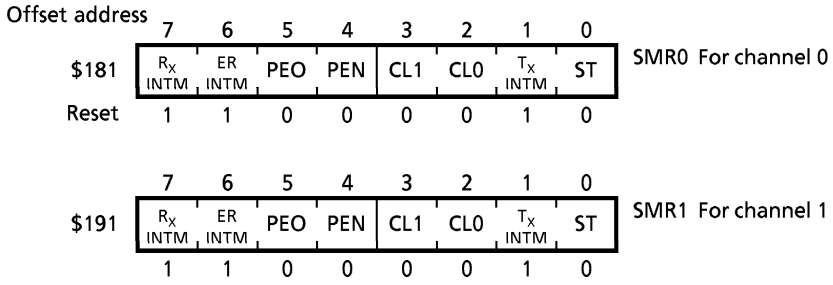
CKSE: Divider clock select
- 0 : External input clock (BCLK)
- 1 : System clock (CLK)

RES : Software reset
- 0 : Releases reset.
- 1 : Reset

INTM: Interrupt mask
- 0 : Enables interrupt signal generation.
- 1 : Disables interrupt signal generation.

5.5.2 Serial Mode Registers 0, 1 (SMR0, 1)

These registers, one per channel, specify the character structure and the interrupt generation masks.



RxINTM : Receive complete interrupt mask and break detect interrupt mask *2
- 0 : Interrupt valid
- 1 : Interrupt masked

ERINTM : Error generated interrupt mask *2
- 0 : Interrupt valid
- 1 : Interrupt masked

PEO : Parity select *1
- 0 : Even parity
- 1 : Odd parity

PEN : Parity control
- 0 : No parity
- 1 : Parity

CL1 to CL0 : Character length

CL1	CL0	Character length
0	0	5-bit character
0	1	6-bit character
1	0	7-bit character
1	1	8-bit character

TxINTM : Transmit interrupt mask
- 0 : Interrupt valid
- 1 : Interrupt masked

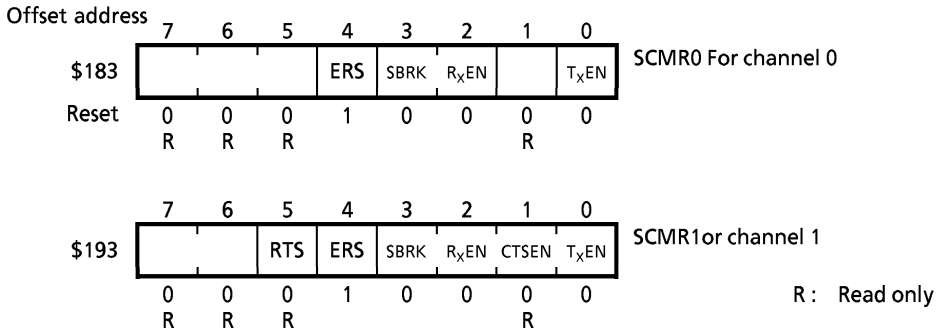
ST : Stop bit length
- 0 : One stop bit
- 1 : Two stop bits

*1 : Shows parity generation;
Even parity means that the number of 1's in the transmit / receive character including the parity bit even.
Odd parity means that the number of 1's in the transmit/receive character including the parity bit is odd.

*2 : ERINTM is the interrupt mask for framing errors, overrun errors, and parity errors. RxINTM is the interrupt mask for break detect.

5.5.3 Serial Command Registers 0, 1 (SCMR0, 1)

These registers, one per channel, are for transmit/receive control.



Note : Set bits which are reserved for future expansion and undefined to 0.

RTS : $\overline{\text{RTS1}}$ pin output control (channel 1 only)
- 0 : High output from $\overline{\text{RTS1}}$ pin
- 1 : Low output from $\overline{\text{RTS1}}$ pin

ERS : Error reset *1
- 0 : No operation
- 1 : Resets PE, OE, FE, and RBRK bits.

SBRK : Break transmit
- 0 : No break transmit
- 1 : Break transmit

RxEN : Receive enable
- 0 : Receive disable
- 1 : Receive enable

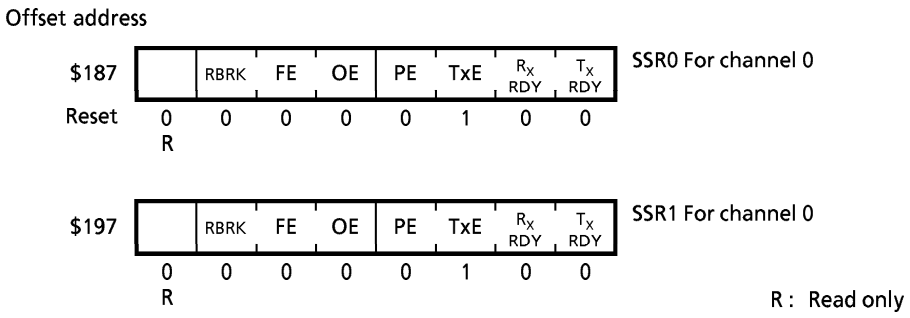
CTSEN : $\overline{\text{CTS1}}$ pin input control (channel 1 only)
- 0 : $\overline{\text{CTS1}}$ disable
- 1 : $\overline{\text{CTS1}}$ enable

TxEN : Transmit enable
- 0 : Transmit disable
- 1 : Transmit enable

*1 : While bit ERS is set to 1, the error bits (PE, OE, FE, RBRK) are continuously reset. Resetting ERS to 0 sets the error bits according to the status of the receive data frame. Therefore, when resuming error detection after resetting error bits, reset the ERS bit to 0 again.

5.5.4 Serial Status Registers 0, 1 (SSR0, 1)

These registers, one per channel, indicate the channel status.



RBRK : Break detect
- 0 : Does not detect break.
- 1 : Detects break.

FE : Framing error
- 0 : Does not generate framing error.
- 1 : Generates framing error.

OE : Overrun error
- 0 : Does not generate overrun error.
- 1 : Generates overrun error.

PE : Parity error
- 0 : Does not generate parity error.
- 1 : Generates parity error.

TxE : Transmit data empty
- 0 : The transmit data register (SDR) contains a transmit character, or a transmit is in progress (transmit character is in transmit buffer).
- 1 : The transmit data register (SDR) is empty and no transmit is in progress (no transmit character in transmit buffer).

RxRDY : Receive ready
- 0 : Receive not ready
- 1 : Receive ready (the receive data register (SDR) contains a receive character, the receive data register can be read)

TxRDY : Transmit ready *1
- 0 : Transmit not ready
- 1 : Transmit ready (transmit character can be written to transmit data register (SDR))

*1 : TxRDY is set to 1 when [{transmit data register empty} × (CTS = 0) × (TxEN = 1)]. When channel 1 does not use $\overline{CTS1}$, $\overline{CTS1}$ is treated as $\overline{CTS1} = 0$ without checking $\overline{CTS1}$ pin input.

5.5.5 Serial Prescaler Register (SPR)

The prescaler divides the input clock (system clock or BCLK pin input) by the divider ratio specified in this register.

The values to be set in this register are from 0 to 255.

Offset address	7	6	5	4	3	2	1	0	
$18D	P7	P6	P5	P4	P3	P2	P1	P0	SPR
Reset	–	–	–	–	–	–	–	–	

Set value	Divider ratio
0	× 1/256
1	× 1
2 to 255	× 1/2 to × 1/255

5.5.6 Baud Rate Register 0, 1 (SBRR0, 1)

The baud rate generator further divides, by the divider ratio specified in these registers, the clock (PCLK) divided by the prescaler. Each channel has a baud rate register. More than one bit cannot be set simultaneously. The clock obtained (GCLK) by dividing PCLK with the baud rate register setting is used as the serial interface transmit/receive fundamental clock. The clock actually used to shift out data bits and perform sampling (TCLK) is obtained by dividing the GCLK by eight. Table 5.2 shows the baud rate register settings and the divider ratios with the baud rate generator (GCLK/PCLK).

Offset address	7	6	5	4	3	2	1	0	
$185					B3	B2	B1	B0	SBRR0 For channel 0
	–	–	–	–	–	–	–	–	
	7	6	5	4	3	2	1	0	
$195					B3	B2	B1	B0	SBRR1 For channel 1
	–	–	–	–	–	–	–	–	

Baud rate register values				Divider ratio	
B3	B2	B1	B0	Decimal	
1	0	0	0	8	1/256
0	1	1	1	7	1/128
0	1	1	0	6	1/64
0	1	0	1	5	1/32
0	1	0	0	4	1/16
0	0	1	1	3	1/8
0	0	1	0	2	1/4
0	0	0	1	1	1/2
0	0	0	0	0	1

Table 5.2 Divider Ratios by Baud Rate Generator

Channel n baud rate set value is defined by the following equation:

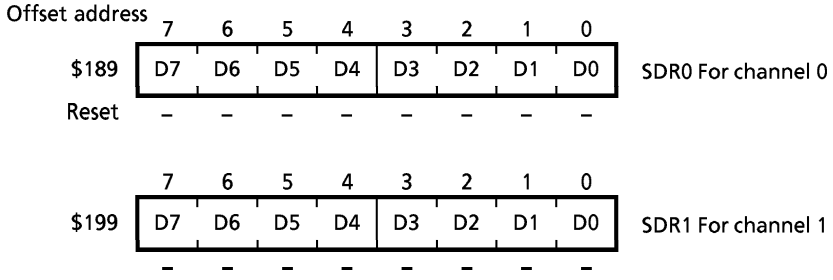
$$TCLKn = CLK \div PR \div 2BRR \div 8$$

n : Channel number (n = 0, 1)
TCLKn : Channel n baud rate (bps)
CLK : Input clock (system clock or BCLK pin input) frequency (Hz)
PR : Prescaler (SPR) set value (PR = 1 to 256 However, PR = 256 means value is 0)
BRRn : Channel n baud rate register (SBRRn) set value (BRRn = 0 to 8)

Table 5.3 shows baud rate setting examples.

5.5.7 Serial Data Registers 0, 1 (SDR0, 1)

These registers, one per channel, store transmit data and receive data. Although the transmit data register and the receive data register share the same address, the two registers are independent. The receive data register is accessed at read. The transmit data register is accessed at write.



SCLK = 16.67 MHz

Baud Rate (bps)	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error
38.4 K	0 ($00)	54 ($36)	+ 0.48 %	1 ($01)	27 ($1B)	+ 0.48 %
19.2 K	0 ($00)	109 ($6D)	– 0.44 %	2 ($02)	27 ($1B)	+ 0.48 %
14.4 K	0 ($00)	145 ($91)	– 0.20 %	4 ($04)	9 ($09)	+ 0.48 %
9600	0 ($00)	217 ($D9)	+ 0.02 %	3 ($03)	27 ($1B)	+ 0.48 %
4800	1 ($01)	217 ($D9)	+ 0.02 %	4 ($04)	27 ($1B)	+ 0.48 %
2400	2 ($02)	217 ($D9)	+ 0.02 %	5 ($05)	27 ($1B)	+ 0.48 %
1200	3 ($03)	217 ($D9)	+ 0.02 %	6 ($06)	27 ($1B)	+ 0.48 %
600	4 ($04)	217 ($D9)	+ 0.02 %	7 ($07)	27 ($1B)	+ 0.48 %

SCLK = 16.0 MHz

Baud Rate (bps)	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error
38.4 K	0 ($00)	52 ($34)	+ 0.16 %	1 ($01)	26 ($1A)	+ 0.16 %
19.2 K	0 ($00)	104 ($68)	+ 0.16 %	2 ($02)	26 ($1A)	+ 0.16 %
14.4 K	0 ($00)	139 ($8B)	– 0.08 %	2 ($02)	35 ($23)	– 0.79 %
9600	0 ($00)	208 ($D0)	+ 0.16 %	3 ($03)	26 ($1A)	+ 0.16 %
4800	1 ($01)	208 ($D0)	+ 0.16 %	4 ($04)	26 ($1A)	+ 0.16 %
2400	2 ($02)	208 ($D0)	+ 0.16 %	5 ($05)	26 ($1A)	+ 0.16 %
1200	3 ($03)	208 ($D0)	+ 0.16 %	6 ($06)	26 ($1A)	+ 0.16 %
600	4 ($04)	208 ($D0)	+ 0.16 %	7 ($07)	26 ($1A)	+ 0.16 %

SCLK = 12.5 MHz

Baud Rate (bps)	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error
38.4 K	0 ($00)	41 ($29)	– 0.76 %	1 ($01)	20 ($14)	+ 1.73 %
19.2 K	0 ($00)	81 ($51)	+ 0.47 %	2 ($02)	20 ($14)	+ 1.73 %
14.4 K	0 ($00)	109 ($6D)	– 0.45 %	2 ($02)	27 ($1B)	+ 0.47 %
9600	0 ($00)	163 ($A3)	– 0.15 %	3 ($03)	20 ($14)	+ 1.73 %
4800	1 ($01)	163 ($A3)	– 0.15 %	4 ($04)	20 ($14)	+ 1.73 %
2400	2 ($02)	163 ($A3)	– 0.15 %	5 ($05)	20 ($14)	+ 1.73 %
1200	3 ($03)	163 ($A3)	– 0.15 %	6 ($06)	20 ($14)	+ 1.73 %
600	4 ($04)	163 ($A3)	– 0.15 %	7 ($07)	20 ($14)	+ 1.73 %

Figure 5.3 Baud Rate Setting Examples (1)

SCLK = 8.0 MHz

Baud Rate (bps)	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error
38.4 K	0 ($00)	26 ($1A)	+ 0.16 %	1 ($01)	13 ($0D)	+ 0.16 %
19.2 K	0 ($00)	52 ($34)	+ 0.16 %	2 ($02)	13 ($0D)	+ 0.16 %
14.4 K	0 ($00)	69 ($45)	+ 0.64 %	2 ($02)	17 ($11)	+ 2.12 %
9600	0 ($00)	104 ($68)	+ 0.16 %	3 ($03)	13 ($0D)	+ 0.16 %
4800	1 ($01)	104 ($68)	+ 0.16 %	4 ($04)	13 ($0D)	+ 0.16 %
2400	2 ($02)	104 ($68)	+ 0.16 %	5 ($05)	13 ($0D)	+ 0.16 %
1200	3 ($03)	104 ($68)	+ 0.16 %	6 ($06)	13 ($0D)	+ 0.16 %
600	4 ($04)	104 ($68)	+ 0.16 %	7 ($07)	13 ($0D)	+ 0.16 %

SCLK = 7.3728 MHz

Baud Rate (bps)	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error
38.4 K	0 ($00)	24 ($18)	0 %	1 ($01)	12 ($0C)	0 %
19.2 K	0 ($00)	48 ($30)	0 %	2 ($02)	12 ($0C)	0 %
14.4 K	0 ($00)	64 ($40)	0 %	2 ($02)	16 ($10)	0 %
9600	0 ($00)	96 ($60)	0 %	3 ($03)	12 ($0C)	0 %
4800	0 ($00)	192 ($C0)	0 %	4 ($04)	12 ($0C)	0 %
2400	1 ($01)	192 ($C0)	0 %	5 ($05)	12 ($0C)	0 %
1200	2 ($02)	192 ($C0)	0 %	6 ($06)	12 ($0C)	0 %
600	3 ($03)	192 ($C0)	0 %	7 ($07)	12 ($0C)	0 %

SCLK = 1.8432 MHz

Baud Rate (bps)	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error
38.4 K	0 ($00)	6 ($06)	0 %	1 ($01)	3 ($03)	0 %
19.2 K	0 ($00)	12 ($0C)	0 %	2 ($02)	3 ($03)	0 %
14.4 K	0 ($00)	16 ($10)	0 %	2 ($02)	4 ($04)	0 %
9600	0 ($00)	24 ($18)	0 %	3 ($03)	3 ($03)	0 %
4800	0 ($00)	48 ($30)	0 %	4 ($04)	3 ($03)	0 %
2400	0 ($00)	96 ($60)	0 %	5 ($05)	3 ($03)	0 %
1200	0 ($00)	192 ($C0)	0 %	6 ($06)	3 ($03)	0 %
600	1 ($01)	192 ($C0)	0 %	7 ($07)	3 ($03)	0 %

Figure 5.3 Baud Rate Setting Examples (2)

Chapter 6 DMA Controller

6.1 Outline

The DMA controller (DMAC) has two independent channels, a 24-bit memory address counter, and a 16-bit data transfer counter. The DMAC can transfer data at high speed without passing through the CPU.

Each channel is provided with the $\overline{DREQ0}$ and $\overline{DREQ1}$ external pins for receiving DMA requests, and $\overline{DACK0}$ and $\overline{DACK1}$ external pins for responding to requests. In addition, a channel is provided with shared pin $\overline{DONE}$, which can suspend DMA operation or notify end of DMA service.

Two independent DMA channels
Maximum 16M bytes of address space
Maximum of 64K - 1 byte/word for block transfer
Transfer operands can be either bytes or words
Maximum transfer speed of 8M bytes per second
Byte or demand bus mode for external requests
Single- or dual-address mode
Burst or cycle-steal transfer mode
Transfer direction: memory↔memory or memory↔I/O device
An interrupt can be generated at DMA service completion or when an error occurs.
Supports I/O devices with an ACK · READY function

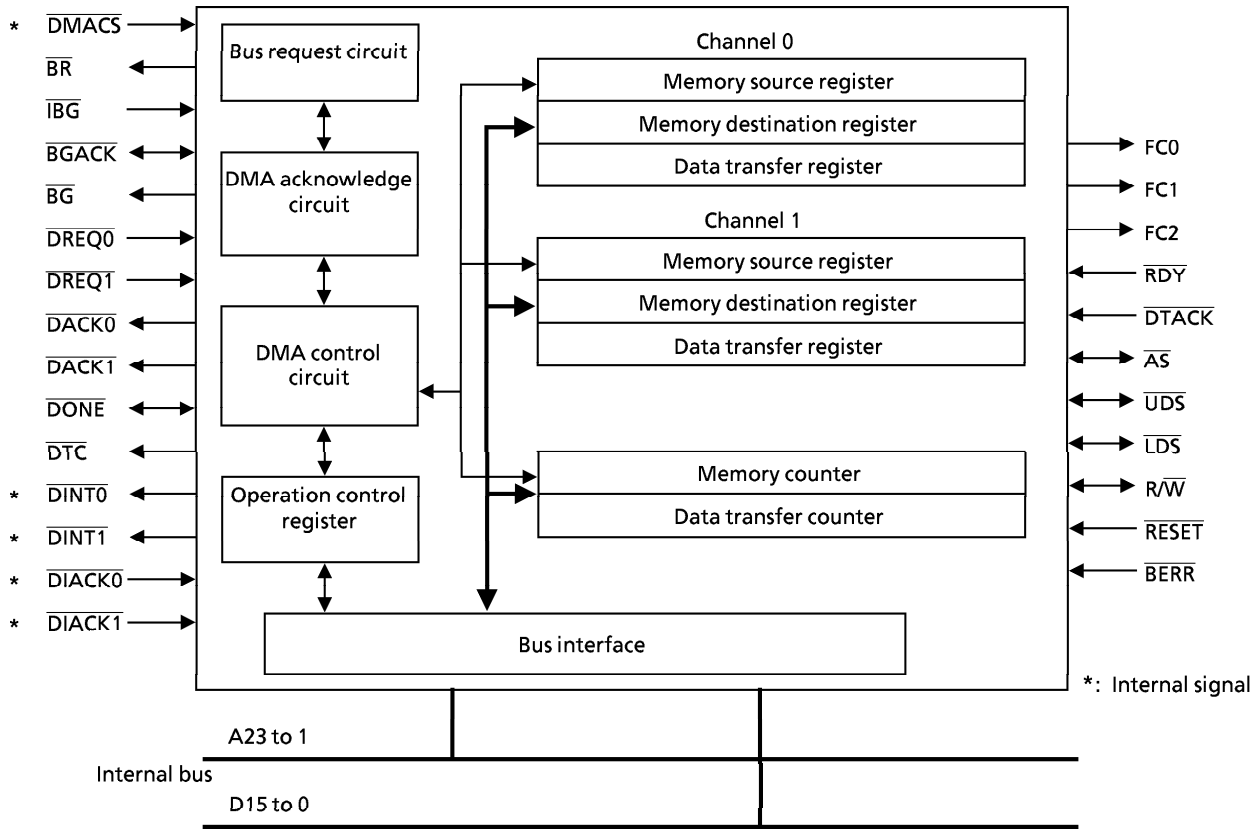


Figure 6.1 DMA Controller Block Diagram

6.2 Operational Description

6.2.1 Bus Arbitration

The DMAC asserts the $\overline{BR}$ (bus request) signal to obtain bus mastership. The core detects the BR signal and asserts the $\overline{BG}$ (bus grant) signal. Receiving the $\overline{BG}$ signal, the DMAC monitors the bus status, checks that the bus is released, then asserts the $\overline{BGACK}$ (bus acknowledge) signal. The DMAC asserts the $\overline{OWN}$ signal half a clock before assertion of the $\overline{BGACK}$ signal.

The $\overline{BG}$ signal from the core processor is output externally via the DMAC.

6.2.2 Transfer Request Generation

There are two types of DMA transfer requests. One type is used between memory and a peripheral I/O device. In this case, the peripheral I/O device asserts the $\overline{DREQ}$ (DMA request) signal to generate the request. (This type can also be used between memory and memory-mapped I/O.) The other type is used between memory areas (memory-mapped I/O devices). In this case, the request is generated by setting the DST bit of the channel control register (CHCR).

6.2.2.1 External Requests

For DMA between memory and a peripheral I/O device (including DMA between memory and memory-mapped I/O), after the initial settings are made, the peripheral I/O device asserts the $\overline{DREQ}$ (DMA request) signal to generate the request.

Next, the DMAC performs bus arbitration with the core processor then, after obtaining bus mastership, starts the DMA transfer.

In the above case, the DMAE bit in the channel control register (CHCR) and the operation control register (OPCR) must be set.

The OPC bit of the channel status register (CHSR) is set when the data transfer count register (DTCR) is set to $0000 at completion of the DMA transfer. As long as the OPC bit is set, $\overline{DREQ}$ is ignored. Writing again to DTCR clears the OPC bit.

When edge mode is set, the $\overline{DREQ}$ signal is sampled at the rising edge of the system clock (SCLK). Therefore, the signal is received provided DREQ remains low for at least one clock cycle. Then, when S4 falls, the DMAC acknowledges $\overline{DREQ}$.

6.2.2.2 Auto Request

Auto request is supported for memory-to-memory (or memory-mapped I/O device) transfers only. The request is generated by software. Setting the DMAE bit of OPCR (operation control register) and CHCR (channel control register) generates a DMA request. After obtaining bus mastership, the DMAC continuously transfers data until the value in the data transfer count register (DTCR) reaches $0000. In this mode, $\overline{DONE}$ or $\overline{DACK}$ signals are not asserted.

When the DTCR value reaches $0000, the DST bit is cleared.

6.2.3 Ending and Suspending Transfer

6.2.3.1 Data Transfer Complete

When the DMA bus cycle ends, $\overline{DTC}$ is asserted. $\overline{DTC}$ is asserted half a clock before $\overline{USD}$ and $\overline{LDS}$ are negated, and negated half a clock later.

6.2.3.2 Service Complete

When the DTCR value reaches $0000, $\overline{DONE}$ is asserted and block transfer ends for that channel.

The OPC bit of CHSR is set, and a service complete signal is output to the internal interrupt controller (provided the INTE bit of CHCR and OPCR is set). While the OPC bit is set, the DMAC ignores external requests ($\overline{DREQ}$) and does not transfer data. The OPC bit can only be cleared by writing data in DTCR.

6.2.3.3 Suspension

An external device asserting $\overline{\text{DONE}}$ to the external input pin sets the DIT bit of CHSR, suspends the current channel DMA transfer after completion of the transfer bus cycle, and cancels the request. Asserting $\overline{\text{DONE}}$ to the external input pin in other than a DMA cycle ignores the $\overline{\text{DREQ}}$ input.

This is particularly useful for suspending the DMA transfer during an auto request burst transfer. The remainder of the transfer is restarted with the next request. However, a falling edge is always required for external requests.

6.2.4 Channel Priority

The DMA request channel priority is fixed at ch0 > ch1. Channels where requests are made before the SCLK rising edge one clock before the $\overline{\text{BGACK}}$ signal output by the DMAC are selected in this order.

If a request is made from ch0 while ch1 is performing burst transfer, the request is held until the current DMA transfer is complete. However, if the request is generated during a cycle-steal transfer, the transfer is temporarily interrupted and the ch0 transfer is performed. At completion of the high priority channel transfer, transfer is resumed on ch1.

6.2.5 Priority with External Bus Master

The internal DMAC priority is higher than the external bus master priority because the internal/external priority is fixed internally using the daisy-chain method.

Accordingly, while the internal DMAC is operating, external bus master requests are held and the BG signal is not asserted until the $\overline{\text{BGACK}}$ signal is negated.

Even if an external bus master request is generated, the $\overline{\text{BG}}$ signal is not output externally if an internal DMAC bus request is generated before the $\overline{\text{BG}}$ signal is output by the core processor.

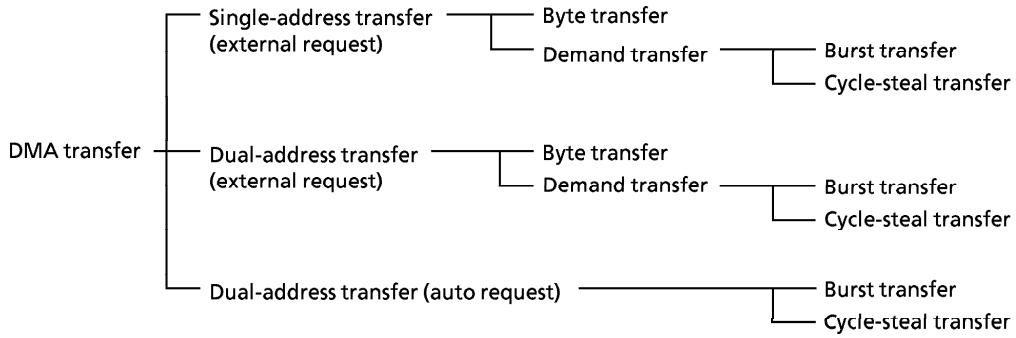
If the $\overline{\text{DONE}}$ signal is continuously asserted while the DMAC is not operating, all internal DMAC transfer requests are ignored enabling the external bus master to have the highest priority.

6.2.6 Address Transfer Mode

There are two DMAC address transfer modes.

One is single-address transfer mode, in which the DMA transfer is performed in one bus cycle. This mode is used only for transfers between memory and I/O devices with an ACK function or with an ACK · READY function.

The other mode is dual-address transfer mode, in which the DMA transfer is performed in two bus cycles. While this mode is used only for memory-to-memory (or memory-mapped I/O device) transfer for auto requests, the mode can also be used for transfer between memory and I/O devices with an ACK function for external requests. In this case, the data read from the memory specified by the source address are stored in the DMAC temporary register.



6.2.6.1 Single-Address Transfer Mode

This mode is useful for transfers between memory and I/O devices with an acknowledge function. The transfer uses handshaking between the $\overline{DREQ}$ and $\overline{DACK}$ signals. Memory is addressed; the port is selected for the I/O device by the $\overline{DACK}$ signal output by the DMAC.

The transfer procedure is as follows. After initialization, the $\overline{DREQ}$ signal is asserted from the I/O device and bus arbitration starts. After bus mastership is obtained, DMA operation starts. The $\overline{DACK}$ signal is asserted and, at the same time, the memory read/write cycle is executed. A $\overline{DTACK}$ signal is asserted for the memory. The DMAC asserts DTC signal after acknowledging DTACK signal and ends transfer. However, as the read/write (R/$\overline{W}$) signal control is for memory, the R/$\overline{W}$ signal for I/O devices must be inverted by an external circuit. Only this mode supports transfer to I/O devices with an acknowledge and ready function. For transfer, the RDY bit in CHCR must be set for each channel. The transfer starts with handshaking between the $\overline{DACK}$ and $\overline{RDY}$ signals. The transfer procedure is as follows. After initialization, the $\overline{DREQ}$ signal is asserted from the I/O device and bus arbitration starts. After bus mastership is obtained, DMA operation starts, and the $\overline{DACK}$ signal is asserted. The DMAC waits for the $\overline{DTACK}$ signal from memory and the $\overline{RDY}$ signal from the I/O device with an ACK · READY function. When both are received, the DMAC asserts a $\overline{DTC}$ signal and ends transfer.

This mode can only be requested by external requests.

6.2.6.2 Dual-Address Transfer Mode

This mode is used for memory-to-memory (or memory-mapped I/O device) transfer where both source and destination have an address. Addressing is performed for both source and destination. The DMAC transfers data by dividing the bus cycle into read and write cycles. The $\overline{DACK}$ and $\overline{DONE}$ signals are asserted in the read or write cycle in which a memory-mapped I/O device is being accessed. The read data are temporarily stored in the DMAC temporary register.

Byte memory access is used for transfers between 16-bit memory and I/O devices with 8-bit ports. The bus is not released during switching between the two bus cycles (between read and write). The $\overline{DTC}$ signal is asserted after the $\overline{DTACK}$ signals are acknowledged for the read and write cycles.

This mode can be requested by either external or auto requests.

6.2.7 Transfer Mode

Byte and demand transfer modes are supported. Demand transfer mode has further two modes: burst and cycle-steal transfer modes.

Dual transfer mode by auto request also has two modes: burst and cycle-steal transfer modes.

6.2.7.1 Byte Transfer Mode

This mode is used only for transfers between memory and I/O devices.

When the $\overline{DREQ}$ signal is asserted, after bus arbitration, the DMAC obtains bus mastership, performs one data transfer, and releases the bus. Accordingly, the transfer ends after one bus cycle for single-address transfers and after two bus cycles for dual-address transfers. Note that before another transfer can be requested, the previous request must be cleared.

6.2.7.2 Demand Transfer Mode

This mode is used only for transfers between memory and I/O devices. When the $\overline{DREQ}$ signal is asserted for the DMAC, after bus arbitration, bus mastership is obtained, and the DMA cycle is generated. However, DMA cycles continue to be executed as long as the $\overline{DREQ}$ signal is asserted. Accordingly, if the $\overline{DREQ}$ signal is negated (prior to the S4 clock falling edge), the DMA cycle ends at that point and the bus is released. To request this mode externally, use level mode.

6.2.7.3 Cycle-Steal Transfer Mode

In this mode, when a transfer request (external or auto request) is generated for the DMAC, after bus arbitration, bus mastership is obtained. After the DMAC transfers one byte or word, it passes bus mastership to the core processor. After the core processor completes one bus cycle, it releases bus mastership back to the DMAC, which executes another byte or word transfer. In this mode, transfer operation repeatedly continuous. Cycle-steal transfer ends under one of the following four conditions:

① The DTCR value reaches $0000.
② The $\overline{\text{DONE}}$ signal is asserted (suspension).
③ A bus error ($\overline{\text{BERR}}$) occurs.
④ A $\overline{\text{DREQ}}$ signal is negated (in demand transfer mode).

Even if an external $\overline{\text{BR}}$ signal is asserted in this mode, the external request is held over.

6.2.7.4 Burst Transfer Mode

In this mode, when a transfer request is made to the DMAC, after obtaining bus mastership, the DMAC has exclusive possession of the bus, during which time it continuously executes DMA cycles, until transfer ends or until the completion conditions are satisfied. Burst transfer ends under one of the following four conditions:

① The DTCR value reaches $0000.
② The $\overline{\text{DONE}}$ signal is asserted (suspension).
③ A bus error ($\overline{\text{BERR}}$) occurs.
④ A $\overline{\text{DREQ}}$ signal is negated (in demand transfer mode).

6.2.8 Interrupt Generation

DMAC has the following two interrupt causes:

1) When the DTCR value reaches $0000, the DMA service completion interrupt signal is output.
2) Error interrupt This interrupt is generated under the following conditions.

① Bus error ($\overline{\text{BERR}}$ signal is asserted)
② Address error (word access to an odd-numbered address)
③ Count error (transfer request with DTCR set to $0000)

When a bus error occurs during dual-address transfer, the BERW bit of CHSR can be used to check whether the error occurred in the read or the write cycle.

If an address or count error occurs, the error interrupt is generated when the transfer is requested to the DMAC and therefore no DMA transfer is executed.

All interrupt signals are output to the internal interrupt controller.

6.3 I/O Device Support

In single-address transfer mode, I/O devices with an ACK function or with an ACK · READY function can transfer data to the DMAC. In dual-address transfer mode, only external memory-mapped I/O devices and I/O devices with an ACK function can transfer data to the DMAC. The internal peripheral I/O devices cannot perform DMA transfers.

The following lists external I/O devices that can support DMA.

	Address mode	Device type	Request		Operand size	Device size	
			Channel	Mode		8 bits	16 bits
DMA transfer	Single address	I/O device with ACK function I/O device with ACK · READY function	Ch0, Ch1	External ($\overline{DREQ}$)	Byte	○	–
					Word	–	○
	Dual address	I/O device with ACK function Memory-mapped I/O device	Ch0, Ch1	External ($\overline{DREQ}$) Auto (DST bit)	Byte	○	○
					Word	–	○

6.4 Addressing and Data Maps

Memory address updating due to DMAC data transfer is determined by the mode setting conditions.

Table 6.2

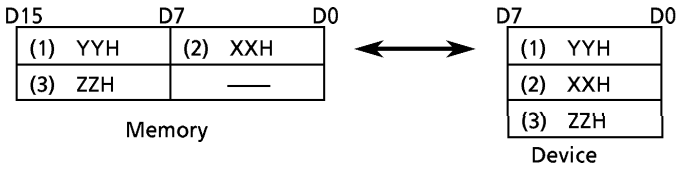
	Operand size	Device size	Memory address increment		I/O address increment	
			Single	Dual	Single	Dual
DMA transfer	Byte	8 bits	+ 1	+ 1	–	+ 2
		16 bits	–	+ 1	–	+ 1
	Word	16 bits	+ 2	+ 2	–	+ 2

The following are examples of mode setting conditions.

① Single-address transfer mode: operand→byte, device→8-bit

Under these conditions, the I/O device is connected to either the upper or lower data bus. Therefore, when reading from memory, control is required such that the read data are collected on the side of the data bus to which the device is connected. When writing to memory, the converse is required. Therefore, during the DMA cycle, detect the $\overline{DACK}$ signal and switch the bus in accordance with the $\overline{UDS}/\overline{LDS}$ signal.

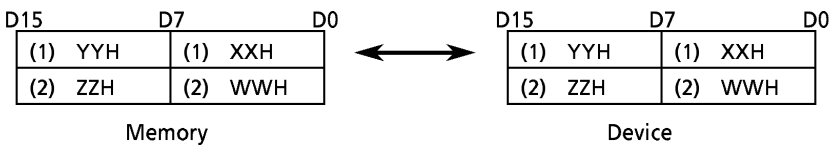
The start memory address can be either even-numbered or odd-numbered.



② Single-address transfer mode: operand→word, device→16-bit

Under these conditions, the above control is not necessary because both the I/O device and memory are connected to a 16-bit data bus.

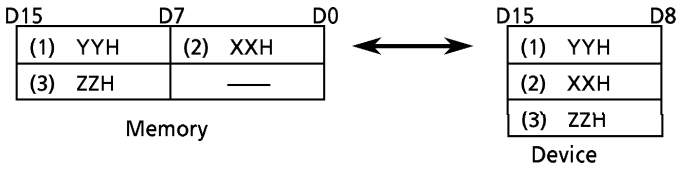
The start memory address must be even-numbered.



③ Dual-address transfer mode: operand→byte, device→8-bit

Under these conditions, the I/O device is connected to either the upper or lower data bus as in 1 above. However, for dual-address transfer, data bus switching by an external circuit is not required because the read data are stored in the DMAC temporary register. (Switching is performed by the DMAC.)

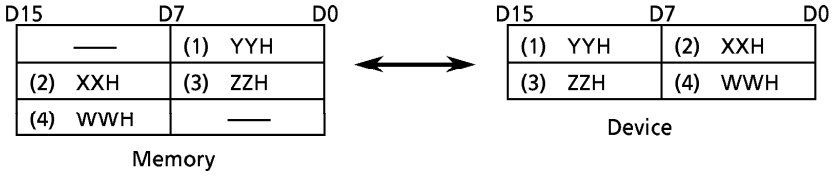
While the memory start address can be either even- or odd-numbered, the start device address must correspond to the side of the data bus to which the device is connected. For example, if the device is connected to the upper data bus, the address must be even-numbered.



④ Dual-address transfer mode: operand→byte, device→16-bit

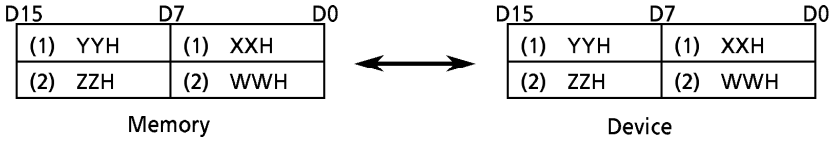
Under these conditions, no particular limitations apply because the I/O device and memory are both connected to a 16-bit data bus.

No limitations apply to the memory and device start addresses.



⑤ Dual-address transfer mode: operand→word, device→16-bit

Under these conditions, while the I/O device and memory are both connected to a 16-bit data bus, the memory and device start addresses must both be even-numbered.



6.4.1 Reset Operation

The following processes are performed when the DMAC is reset.

① The DMAC internal registers are initialized to the following values.

OPCR : $00
CHCR0, 1 : $0000
CHSR : $0000
SMAR0, 1 : Undefined
DMAR0, 1 : Undefined
DTCR0, 1 : Undefined

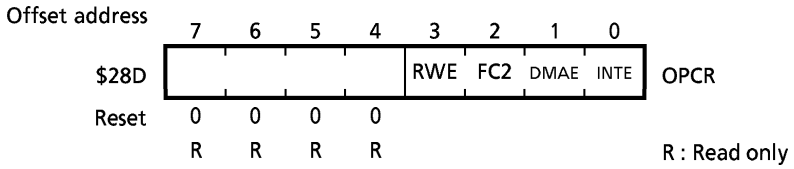
② If a reset occurs during DMA transfer, execution halts, even if in the middle of a bus cycle, and the bus is released. Therefore, the DMA transfer is not performed normally.

In addition, all control signals are negated and transfer stops.

6.5 Register Configuration

6.5.1 Operation Control Register (OPCR)

This register enables operation of the two DMAC channels.
A reset initializes the register to $00.



RWE : DMAC internal register read/write control
- 0 : Disables register read/write.
- 1 : Enables register read/write.

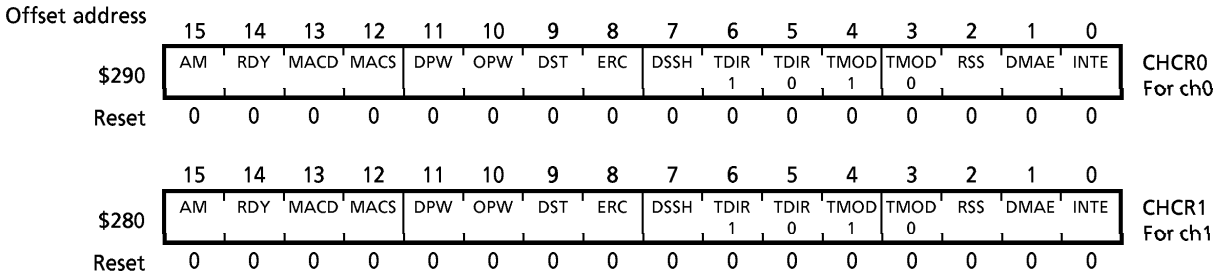
FC2 : DMA transfer address space setting
- 0 : User data space
- 1 : Supervisor data space

DMAE : DMA operation setting
- 0 : Disable DMA operation
- 1 : Enable DMA operation

INTE : Interrupt generation control
- 0 : No interrupt request signal
- 1 : Interrupt request signal

6.5.2 Channel Control Registers 0, 1 (CHCR0, 1)

These registers set the modes for the DMA channels.
A reset initializes the registers to $0000.



AM : Single/dual transfer select
- 0 : Single transfer
- 1 : Dual transfer

RDY : Device with/without $\overline{RDY}$ function select
- 0 : Disable $\overline{RDY}$
- 1 : Enable $\overline{RDY}$

MACD : DMAR countup function setting
- 0 : Disable DMAR countup function.
- 1 : Enable DMAR countup function.

MACS : SMAR countup function setting
0 : Disable SMAR countup function.
1 : Enable SMAR countup function.

DPW : Device port width select
0 : 8-bit width
1 : 16-bit width

OPW : Operand width select
0 : 8-bit width
1 : 16-bit width

DST : DMA request initiation control using software
(Automatically cleared after service completion)
0 : Does not initiate DMA request by software.
1 : Initiates DMA request by software.

ERC : Error flag clear
0 : No operation
1 : Clears ERR, ERRCD0, and ERRCD1 bits of CHSR.

DSSH : DMA request trigger cause select
0 : Hardware ($\overline{\text{DREQ}}$) cause (external request)
1 : Software (program) cause (auto request)

TDIR1, 0 : Data transfer direction

TDIR1	TDIR0	Data transfer direction
0	0	Memory→I/O
0	1	I/O→memory
1	1	Memory→memory (memory-mapped I/O→memory)
1	0	Memory→memory (memory→memory-mapped I/O)

TMOD1, 0 : Transfer mode setting

TMOD1	TMOD0	Transfer mode
0	0	Byte transfer (valid only at external request)
0	1	Cycle-steal transfer on demand transfer (at external request) Cycle-steal transfer (at auto request)
1	1	Burst transfer on demand transfer (at external request) Burst transfer (at auto request)

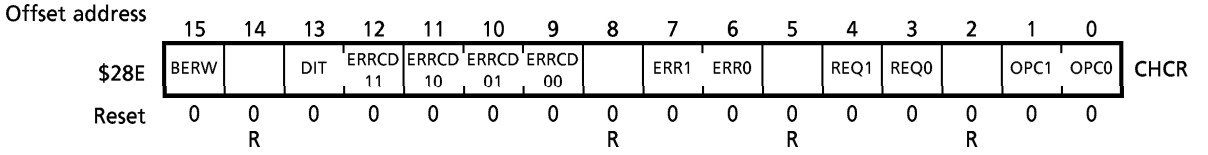
RSS : DMA request signal input mode select
0 : Edge mode
1 : Level mode

DMAE : Operation setting
0 : Disable DMA operation.
1 : Enable DMA operation.

INTE : Interrupt generation control
0 : No interrupt request signal
1 : Interrupt request signal

6.5.3 Channel Status Register (CHSR)

This register indicates the operation status of the two DMAC channels.
A reset initializes the register to $0000.



Offset address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$28E	BERW		DIT	ERRCD 11	ERRCD 10	ERRCD 01	ERRCD 00		ERR1	ERR0		REQ1	REQ0		OPC1	OPC0	CHCR
Reset	0	0 R	0	0	0	0	0	0 R	0	0	0 R	0	0	0 R	0	0	

R : Read only

BERW : Bus error at dual address transfer
- 0 : Generates in write cycle.
- 1 : Generates in read cycle.

DIT : DONE input generation
- 0 : No $\overline{\text{DONE}}$ input
- 1 : $\overline{\text{DONE}}$ input

ERRCDx1, 0 : Error code

ERRCD11	ERRCD10	Error Code
0	1	Channel 1 bus error
1	0	Channel 1 address error
1	1	Channel 1 count error

ERRCD01	ERRCD00	Error Code
0	1	Channel 0 bus error
1	0	Channel 0 address error
0	1	Channel 0 count error

ERR (0, 1) : DMA channel 0, 1 error
- 0 : No error
- 1 : Error

REQ (0, 1) : DMA channel 0, 1 request
- 0 : No DMA request
- 1 : DMA request received

OPC (0, 1) : DMA channel 0, 1 complete
- 0 : DMA service not complete
- 1 : DMA service complete

Note : The OPC bit is cleared when the data transfer count register for the relevant channel (DTCR) is rewritten.

6.5.4 Data Transfer Count Registers 0, 1 (DTCR0, 1)

These registers set the number of data bytes or words to transfer on the DMA channel.

After reset, the registers are undefined. The number of data transfers does not depend on the operand width (byte or word).

Offset address

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$292	DT15	DT14	DT13	DT12	DT11	DT10	DT9	DT8	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0	DTCR0 For ch0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$282	DT15	DT14	DT13	DT12	DT11	DT10	DT9	DT8	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0	DTCR1 For ch1
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

6.5.5 Source Memory Address Registers 0, 1 (SMAR0, 1)

These 24-bit registers store the DMA channel source memory address.

At single address transfer, the registers store memory addresses.

After reset, the registers are undefined.

Offset address

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$294									SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16	SMAR0 For ch0 (Upper)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$296	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	SMAR0 For ch0 (Lower)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$284									SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16	SMAR1 For ch1 (Upper)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$286	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	SMAR1 For ch1 (Lower)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

6.5.6 Destination Memory Address Registers 0, 1 (DMAR0, 1)

These 24-bit registers store the DMA channel destination memory address.

Offset address

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$294									DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16	DMAR0 For ch0 (Upper)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

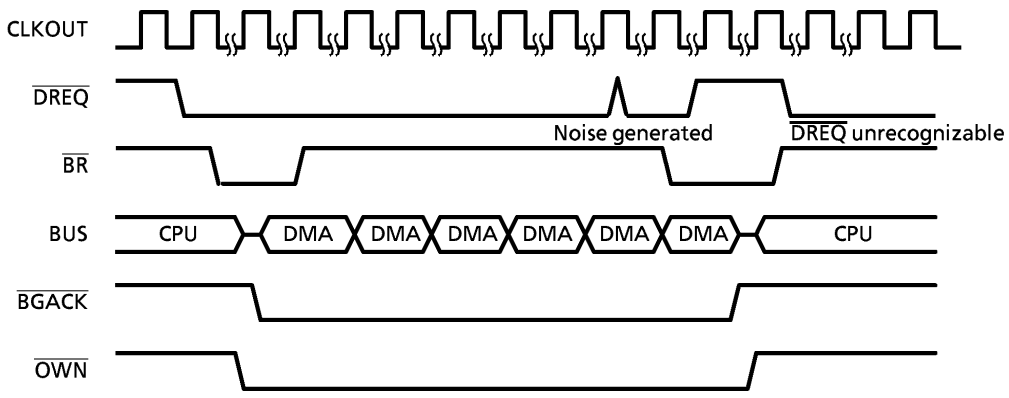
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$29A	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	DMAR0 For ch0 (Lower)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$288									DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16	DMAR1 For ch1 (Upper)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$28A	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	DMAR1 For ch1 (Lower)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

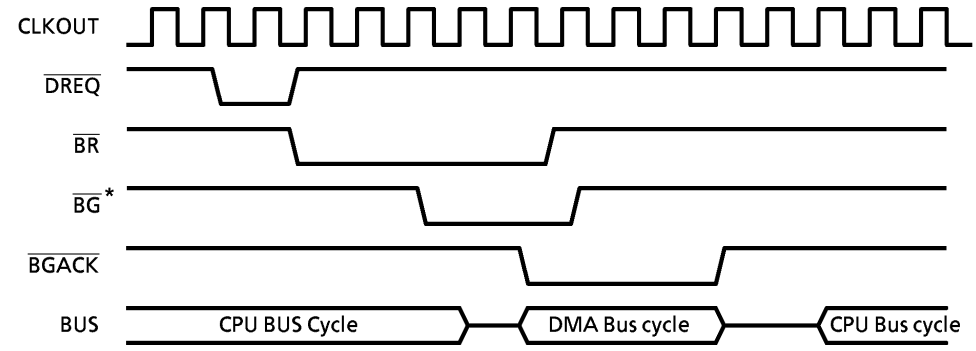
6.5.7 Cautions

1. At word access transfer, write even-numbered values in the source memory address registers and the destination memory address registers.
2. At single address transfer, use only the source memory address registers.
3. When writing to registers, first set the RWE bit of the operation control register. At DMA transfer, do not specify an assigned address in a DMAC register.
4. When an address error and a count error occur simultaneously, the count error in the status register is valid.
5. If noise occurs in the $\overline{\text{DREQ}}$ signal during a command burst transfer, the $\overline{\text{DREQ}}$ signal may be thereafter partially unrecognizable. Hence, care is required. (See the diagram below)
   To restore the signal (to recognize the $\overline{\text{DREQ}}$ signal and output the $\overline{\text{BR}}$ signal), first clear the DMAE bit of the channel control registers, then set the bit.



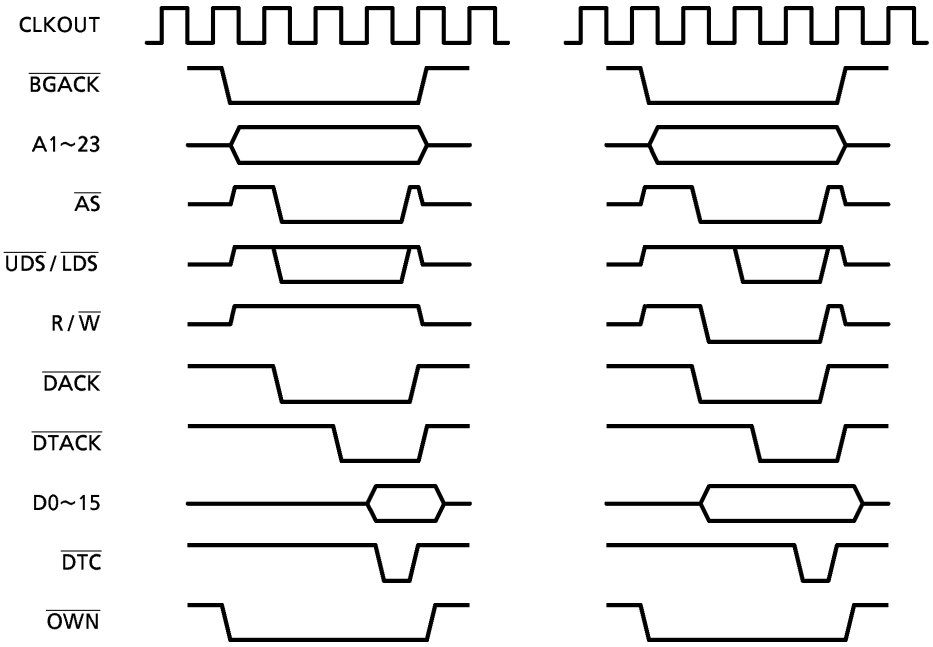
6.5.8 Bus Cycle Timing Chart

(1) DMAC bus arbitration



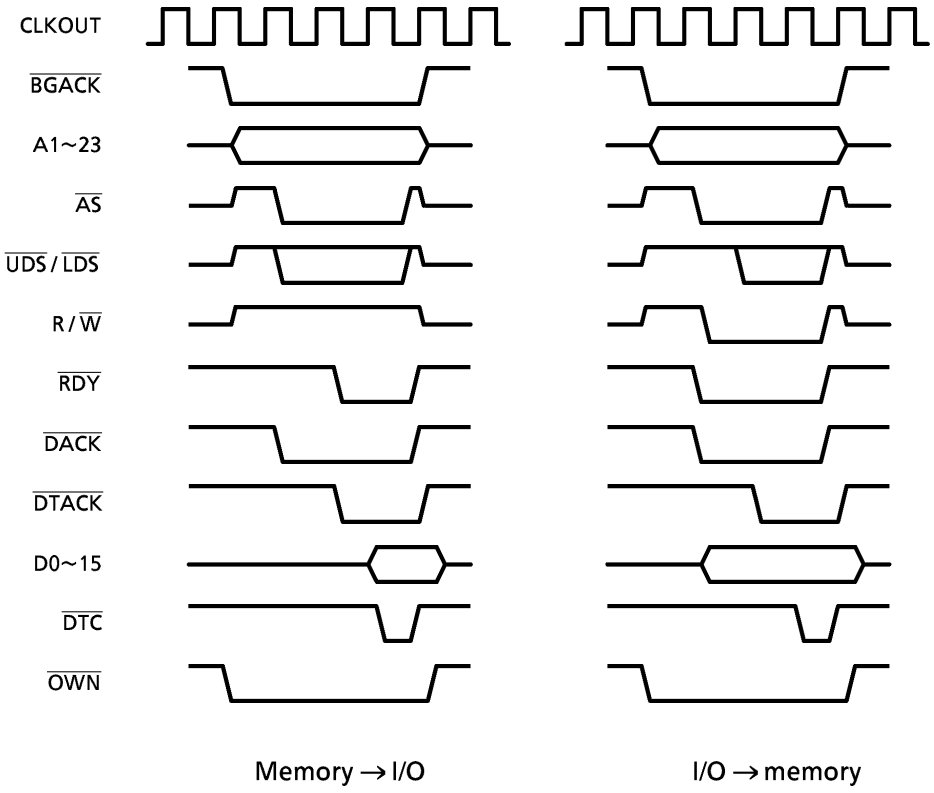
* : Core processor output $\overline{BG}$

(2) Single address transfer (without $\overline{RDY}$)

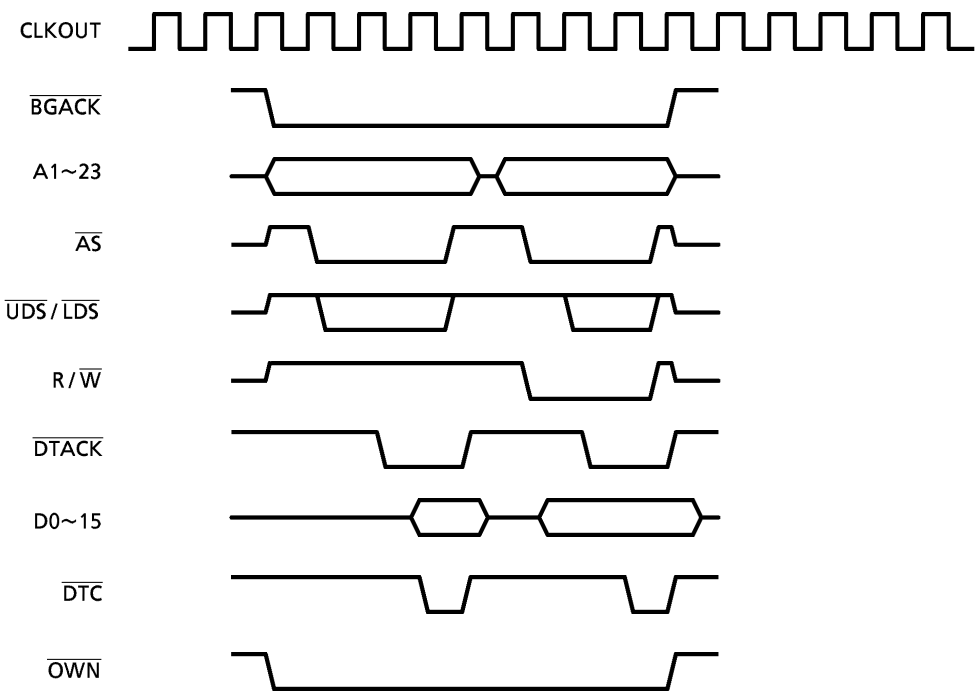


Memory → I/O I/O → memory

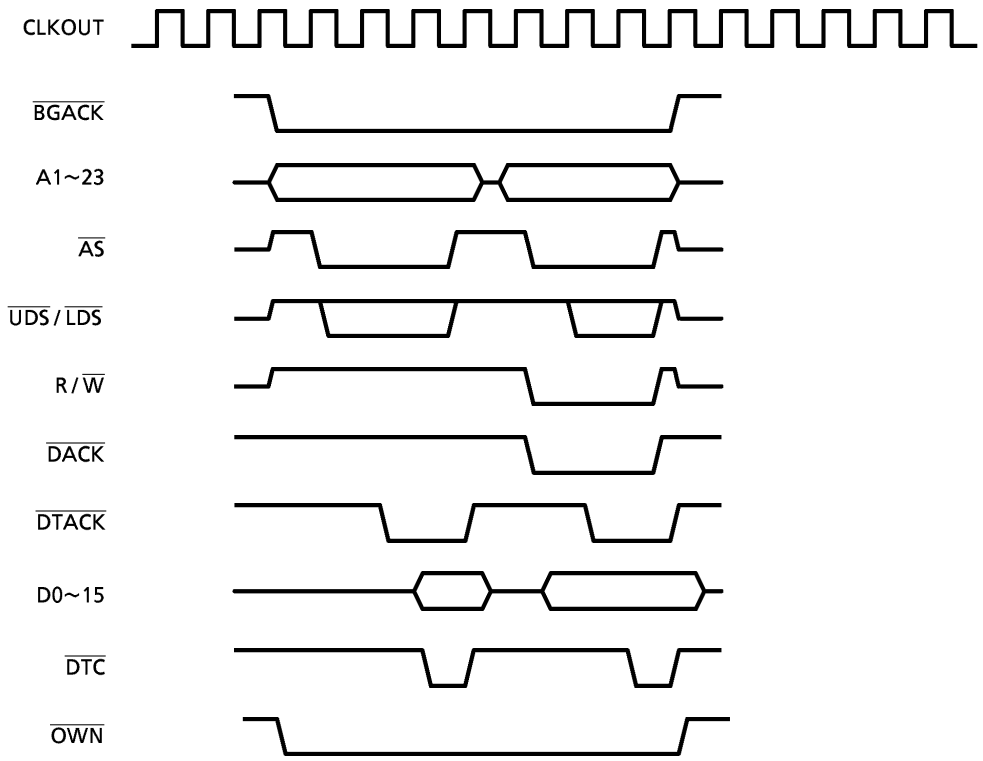
(3) Single address transfer (with $\overline{RDY}$)



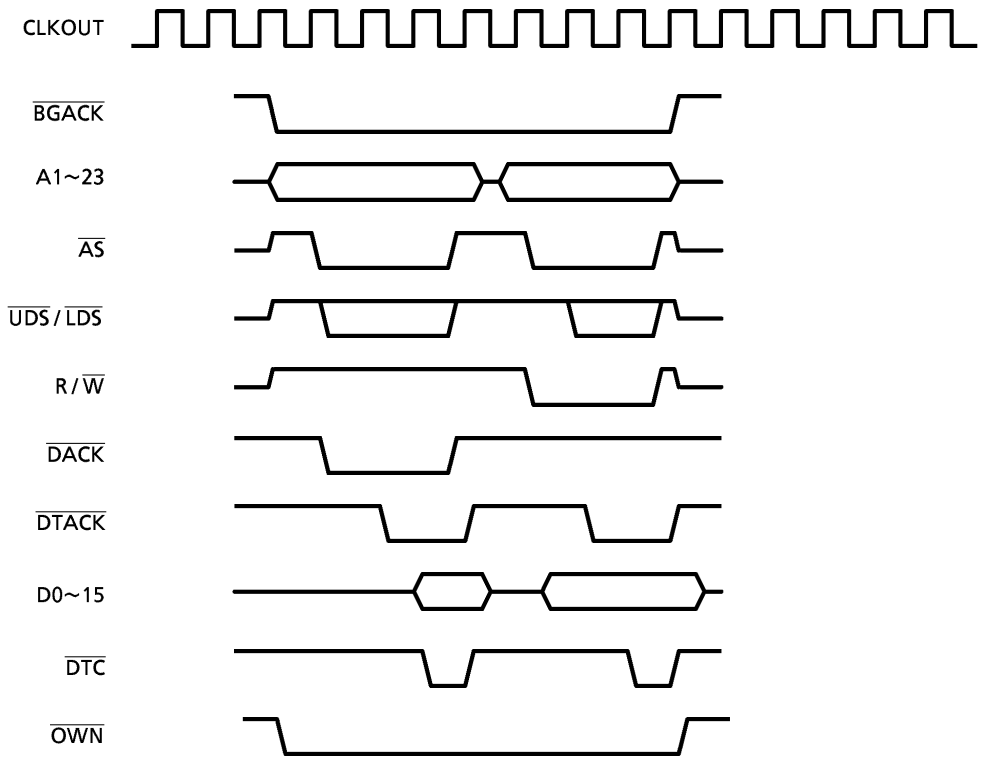
(4) Dual address transfer (Memory → memory)



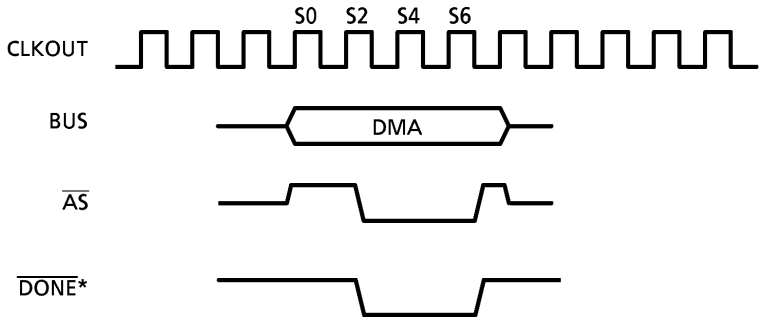
(5) Dual address transfer (Memory → memory-mapped I/O)



(6) Dual address transfer (Memory-mapped I/O → memory)



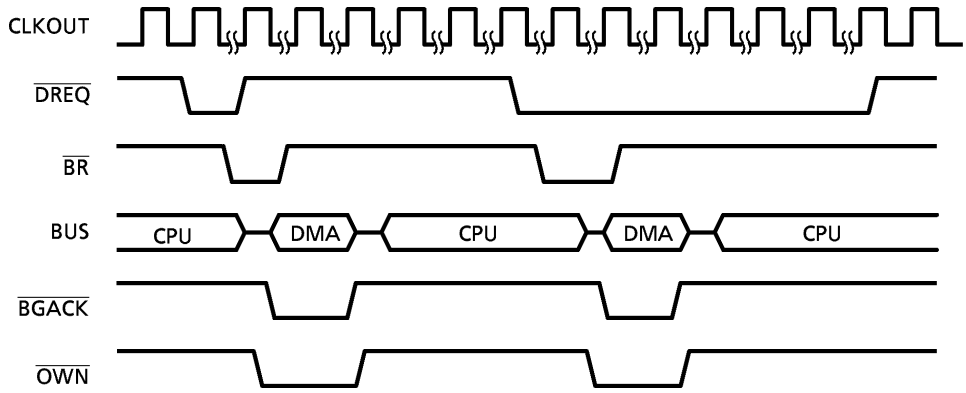
(7) Service complete



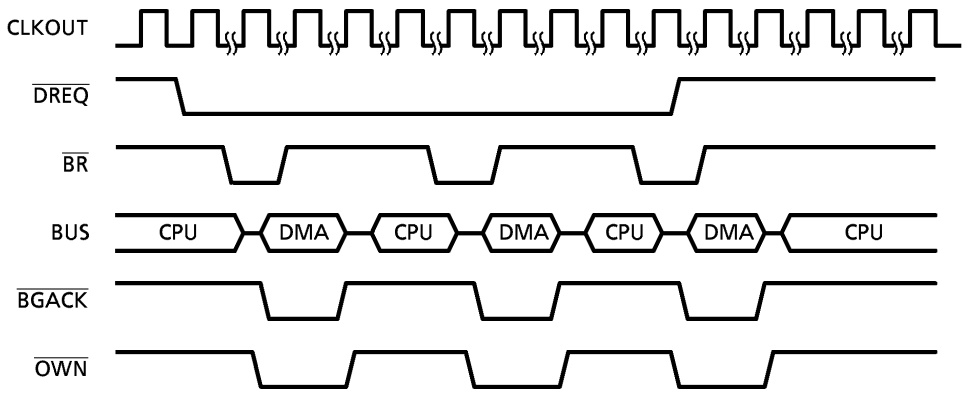
* Open drain output

6.5.8.1 Transfer Modes

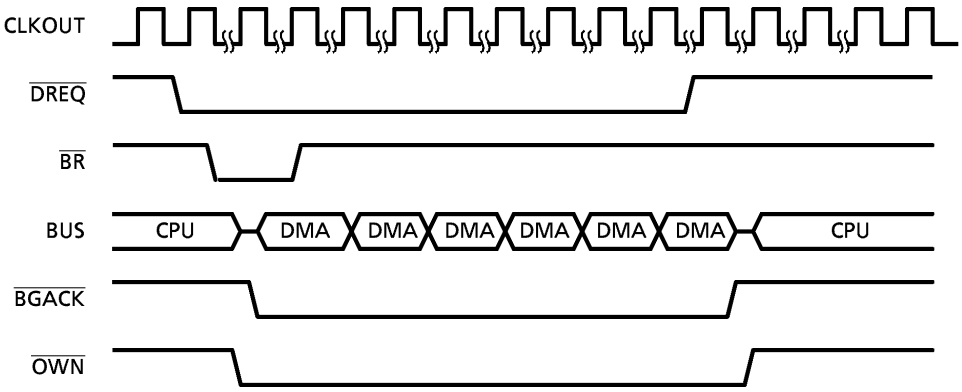
(1) Byte transfer



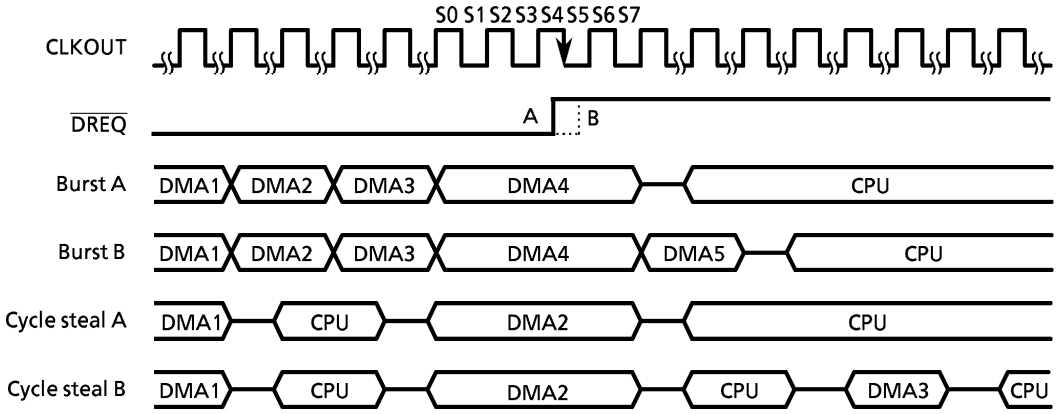
(2) Cycle steal transfer



(3) Burst transfer

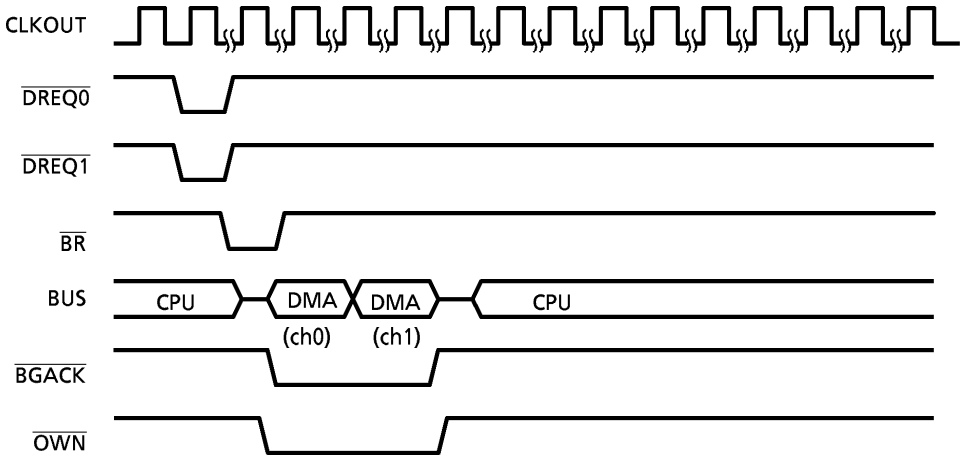


(4) Transfer suspended due to DREQ (at burst or cycle steal transfer)

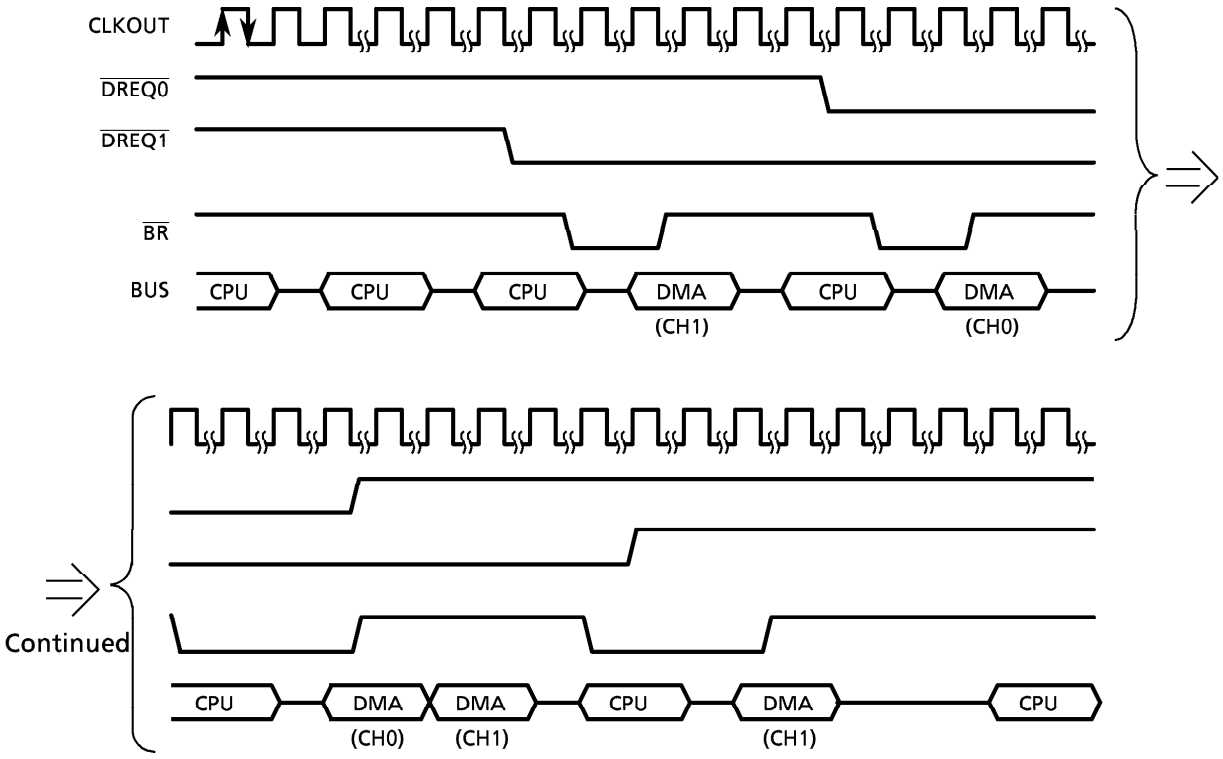


6.5.8.2 Multiple Requests and Priority

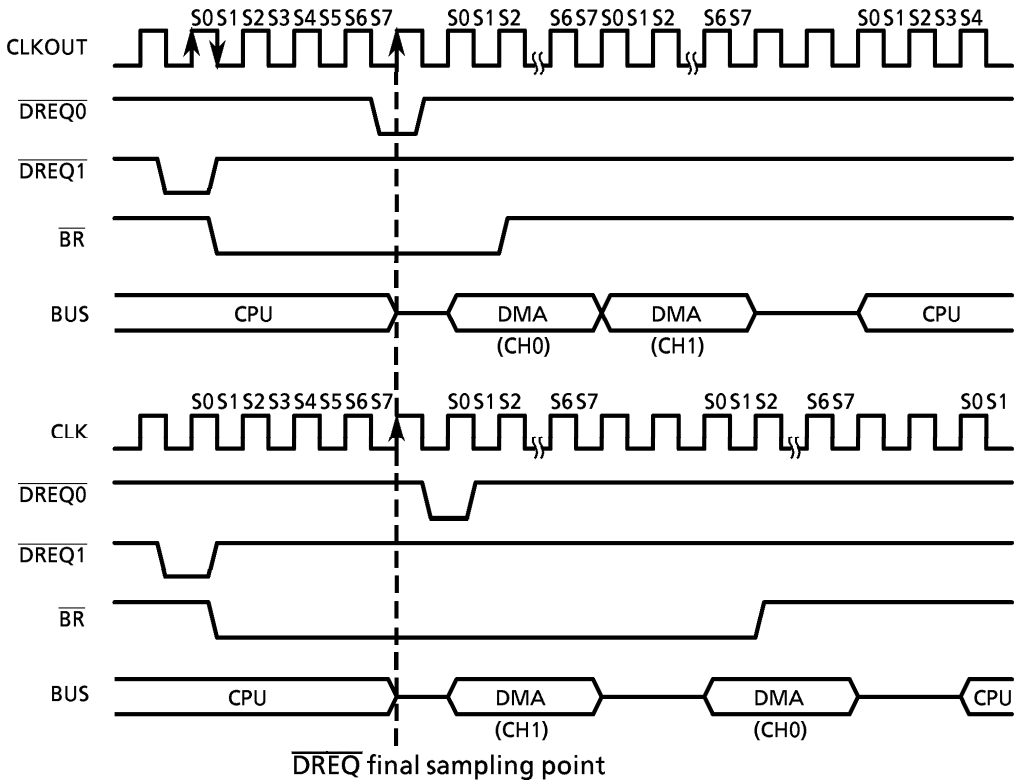
(1) Multiple requests



(2) Priority at cycle steal transfer

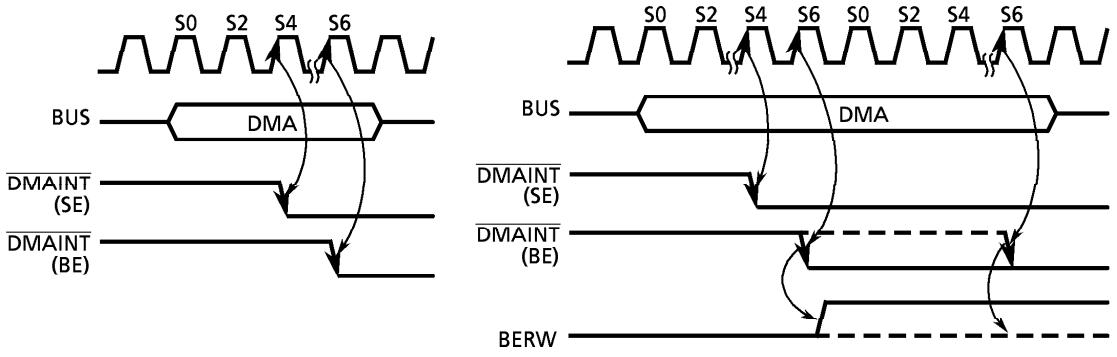


(3) Receive timing at multiple requests



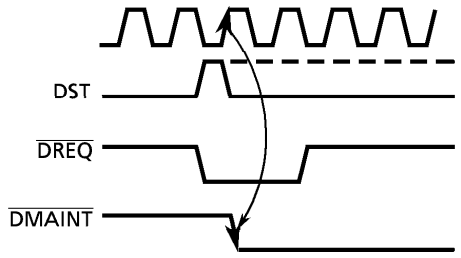
6.5.8.3 Interrupt Request Timings

(1) Service complete bus error



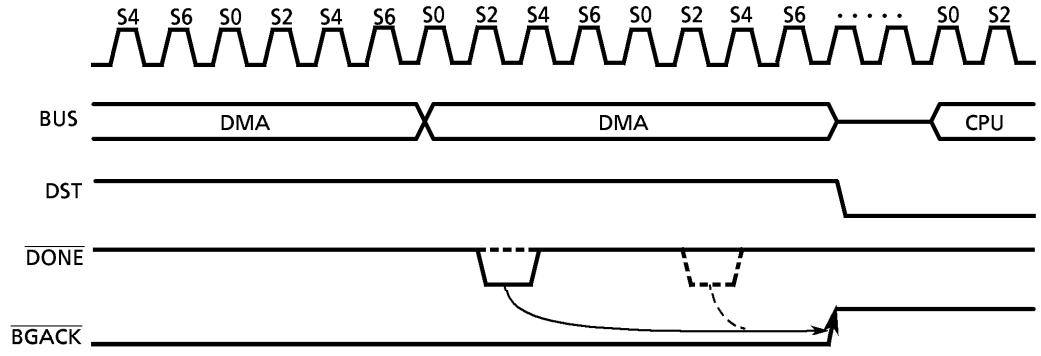
Single Dual

(2) Address error/count error



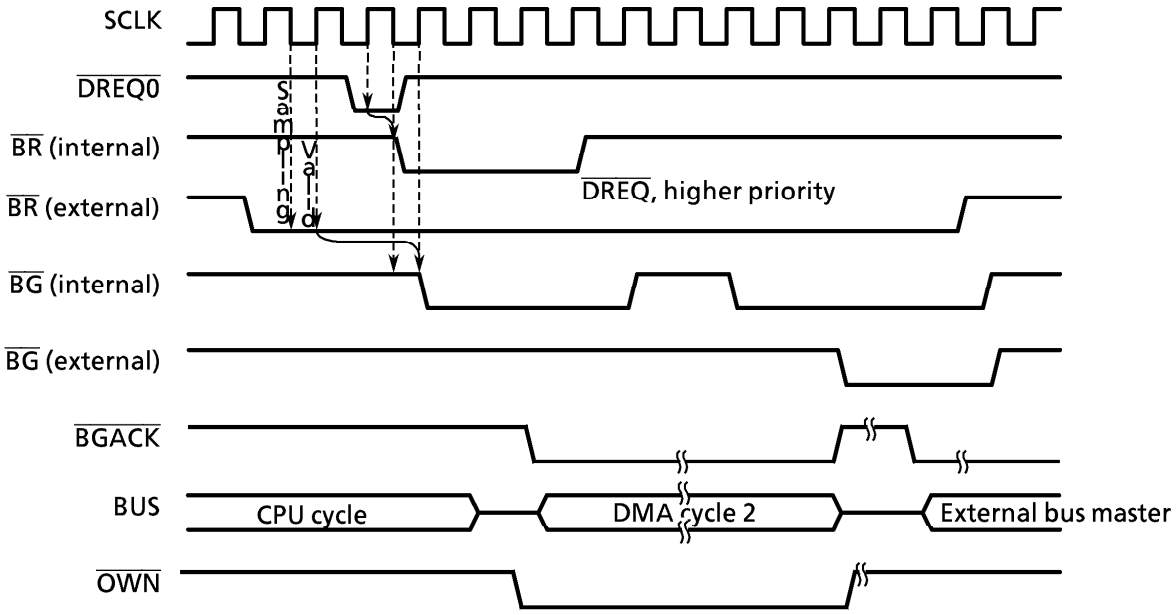
DST is cleared when error occurs.

(3) Suspension



Transfer suspended during dual address transfer

6.5.8.4 Priority for Internal and External $\overline{BR}$



* The priority between internal and external $\overline{BR}$ is determined according to the timings of internal $\overline{BR}$ output and core internal $\overline{BG}$ output.
Internal $\overline{BR}$ is output at the clock falling edge; external $\overline{BG}$, at the clock rising edge. Thus, if $\overline{DREQ}$ is sampled at the next clock, the external $\overline{BR}$ prevails over the internal $\overline{BR}$.

Chapter 7 Timer

7.1 Outline

The timer consists of one channel with a 16-bit counter. The timer has an 8-bit prescaler and generates interrupt requests.

Built-in 16-bit counter
Count data are readable
Generates interrupts.
Programmable operating modes
Max. 8 MHz count

Figure 7.1 shows the timer outline.

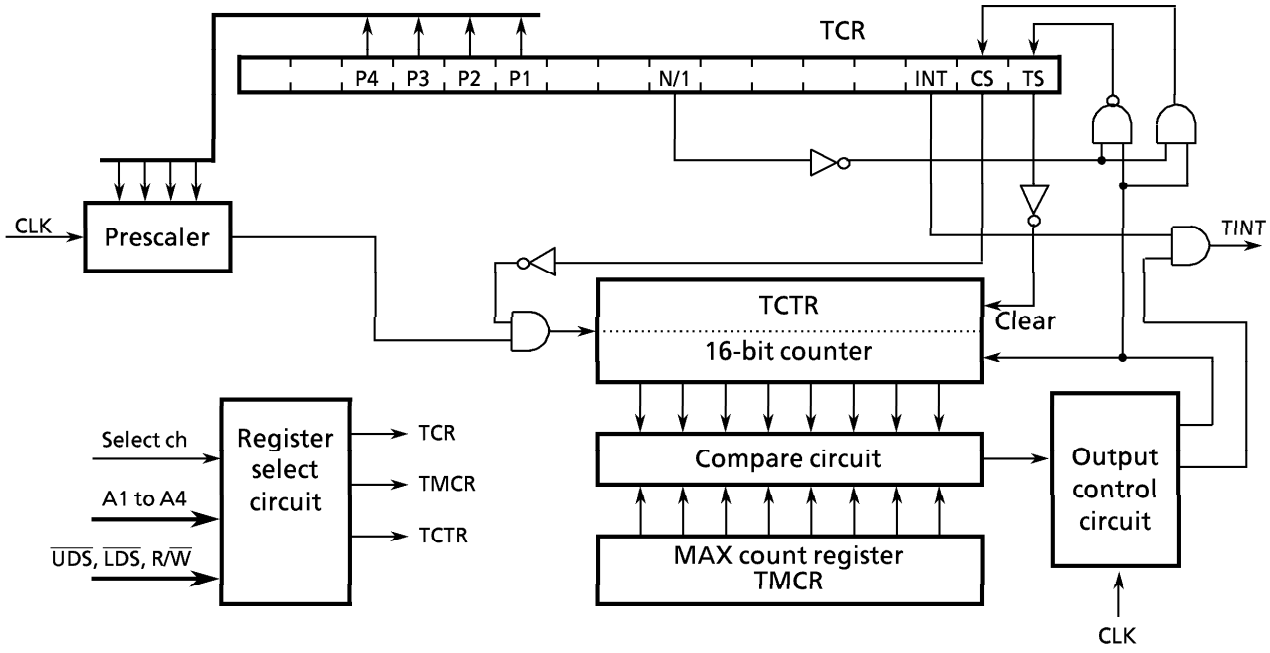


Figure 7.1 Timer Channel Outline

7.2 Operational Description

The internal counter is a 16-bit upcounter. Immediately the count value in the 16-bit counter reaches the value set in the MAX count register, the channel generates a count match signal that clears the count value. The channel can generate an interrupt request signal within the channel.

The channel has three 16-bit registers: control register, count register, and MAX count register.

7.2.1 Setting and Modifying Maximum Counts, Reading Counts

The value written in the MAX count register specifies the maximum count.

Changing the MAX count register sets and modifies the maximum count. (The register can be rewritten during operation.)

7.3 8-Bit Prescaler

The 8-bit prescaler divides the system clock by 2, 4, 8, 16, 32, 64, 128, or 256 to obtain a count clock. Use the control register to set the divider ratio.

7.4 Interrupt Generation

The control register controls interrupt generation mode.

The timer ignores and does not store the count match signal during modes in which interrupt request signals are not generated. Therefore, even though modes are switched, an interrupt is not generated.

Interrupt generation timing: an interrupt request signal is generated when a count match signal is generated.

7.5 Register Reading/Writing

Reading a register during a count operation does not affect the count operation or timer output signal.

During write to a register during counting, an interrupt or the count match signal (match signal for the count value and MAX count register value) is ignored.

7.6 Register Configuration

7.6.1 Timer Count Register

The count value is read from this register. Only the upper byte or the lower byte can be read. The register is normally read only.

Offset address $20C	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	TCTR
	C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	R	

R : Read only

7.6.2 MAX Count Register

The value written to the MAX count register specifies the maximum count.

Offset address $204	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	TMCR MAX For ch0
	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

7.6.3 Timer Control Register

The control register controls the count operation. The register consists of the following bits.

Offset address $200	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	TCR
			P4	P3	P2	P1			N/1					INT	CS	TS	
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	
	R	R					R	R		R	R	R	R				

P4, P3, P2, P1: Prescaler divider ratio setting

P4	P3	P2	P1	Divider ratio
0	0	0	1	1/2
0	0	1	0	1/4
0	0	1	1	1/8
0	1	0	0	1/16
0	1	0	1	1/32
0	1	1	0	1/64
0	1	1	1	1/128
1	X	X	X	1/256

X : Any value

N/I : Repeat specification
0 : Halts count operation after one cycle. (One shot)
When the MAX count value is reached, the CS bit is set to 1 and the TS bit to 0, halting the count operation.
1 : Repeats count operation. (Repeat)

INT : Interrupt request bit
0 : No interrupt request signal
1 : Interrupt request signal

CS : Count clock input control
0 : Inputs count clock to counter. (Starts count operation.)
1 : Does not input count clock to counter. (Stops count operation.)

TS : Timer operation setting
0 : Clears counter.
1 : Releases clear state and start counter.

Chapter 8 Clock Generator

The clock generator generates clocks supplied internally and externally. The frequency is 16.67MHz and the generated clocks are output to CLKOUT. As an alternative, the system clock can be directly input from the X2 pin. In this case, leave the X1 pin open.

8.1 Crystal Oscillator

TMP68305 incorporates a crystal oscillator. Connecting the oscillator to the external pins (X1, X2) obtains the required clock. Figure 8.1 is an example of oscillator connection.

Frequency	Model number	Equivalent series resistance (max.)	$C_1 = C_2$	Rd
16.67 MHz	CSA – 309	50 Ω	10pF	560 Ω

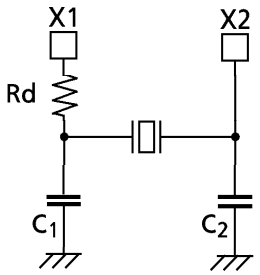


Figure 8.1 Crystal Oscillator Connecting Example

The oscillation frequency is determined depending on the crystal oscillator's load capacitance and external capacitance, C1 and C2. Since the crystal oscillator's equivalent resistance (Ra) and external capacitance C1 and C2 (Rd) greatly affect oscillation, to start and maintain stable oscillation, use the above recommended values.

8.2 Ceramic oscillator

Table 8.2 Ceramic Oscillator (by Murata Mfg.) and External Capacitance

Frequency	Model number	$C_1 = C_2$
16.67 MHz	CSA16.67M x 040	10pF

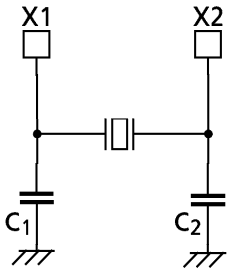
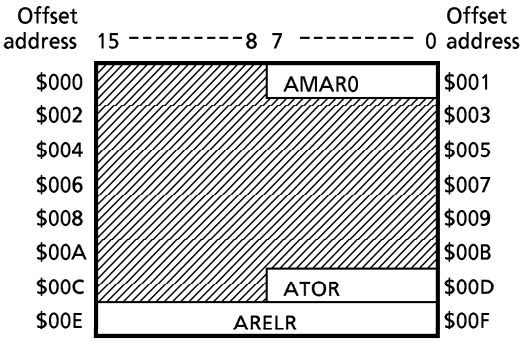


Figure 8.2 Ceramic Oscillator Connection Example

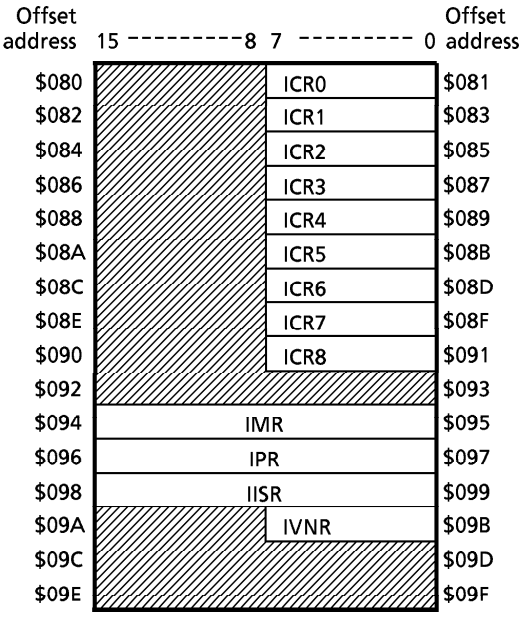
Chapter 9 Internal Peripheral Device Register Maps

9.1 Register map (1)

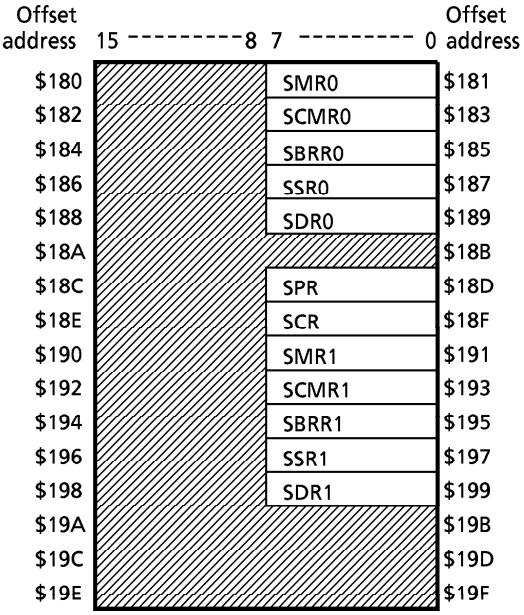
○ Address decoder



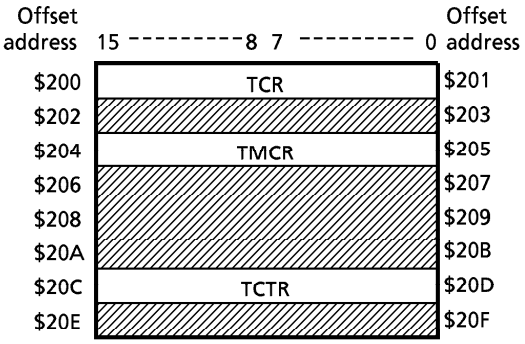
○ Interrupt controller



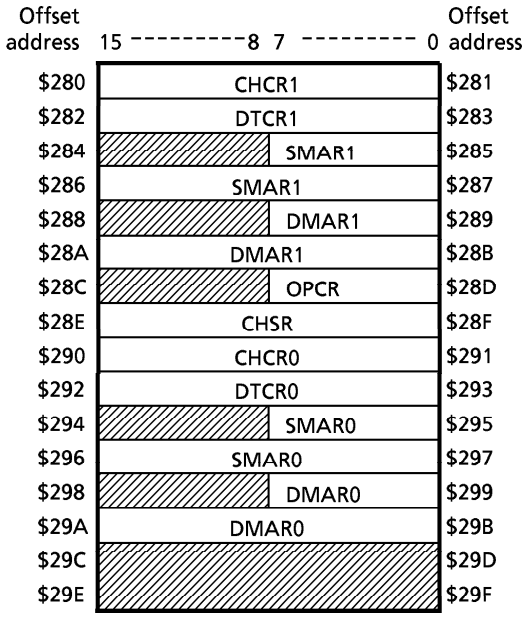
○ Serial interface



○ 16-bit timer



○ DMA controller



9.2 Register map (2)

○ Address decoder

Symbol	Name	Offset address	Data bus Upper bytes : 15 - 8 (even-numbered addresses) Lower bytes : 7 - 0 (odd-numbered addresses)							
			15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0
AMAR	Memory Address Register	$001	A23	A22	A21				EN	S
			0	0	0	0	0	0	1	0
			R/W			R			R/W	
ATOR	Time Out Register For $\overline{BERR}$ generation	$00D					256	128	64	32
			0	0	0	0	1	0	0	0
			R				R/W			
ARELR	Relocaton Register	$00E	A23	A22	A21	A20	A19	A18	A17	A16
			1	1	1	1	1	1	1	1
			R/W							
		$00F	A15	A14	A13	A12	A11	A10		
	For internal registers		1	1	1	1	1	1	0	0
			R/W						R	

○ Interrupt controller (1)

Symbol	Name	Offset address	Data bus Upper bytes : 15 - 8 (even-numbered addresses) Lower bytes : 7 - 0 (odd-numbered addresses)							
			15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0
ICR0	Interrupt Control Register 0 For external interrupt (INT0)	$081			V	R/F	L/E	Level		
			0	0	0	0	0	1	1	1
			R		R/W					
ICR1	Interrupt Control Register 1 For external interrupt (INT1)	$083			V	R/F	L/E	Level		
			0	0	0	0	0	1	1	1
			R		R/W					
ICR2	Interrupt Control Register 2 For external interrupt (INT2)	$085			V	R/F	L/E	Level		
			0	0	0	0	0	1	1	1
			R		R/W					
ICR3	Interrupt Control Register 3 For external interrupt (INT3)	$087			V	R/F	L/E	Level		
			0	0	0	0	0	1	1	1
			R		R/W					
ICR4	Interrupt Control Register 4 For serial ch0	$089						Level		
			0	0	0	0	0	1	1	1
			R					R/W		
ICR5	Interrupt Control Register 5 For serial ch1	$08B						Level		
			0	0	0	0	0	1	1	1
			R					R/W		
ICR6	Interrupt Control Register 6 For timer ch0	$08D						Level		
			0	0	0	0	0	1	1	1
			R					R/W		
ICR7	Interrupt Control Register 7 For DMAC ch0	$08F						Level		
			0	0	0	0	0	1	1	1
			R					R/W		
ICR8	Interrupt Control Register 8 For DMAC ch1	$091						Level		
			0	0	0	0	0	1	1	1
			R					R/W		

○ Interrupt controller (2)

Symbol	Name	Offset address	Data bus Upper bytes : 15 - 8 (even-numbered addresses) Lower bytes : 7 - 0 (odd-numbered addresses)							
			15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0
IMR	Interrupt Mask Register	$094			D1	D0				T0
			0	0	1	1	0	0	0	1
			R		R/W		R			R/W
		$095			S1	S0	E3	E2	E1	E0
			0	0	1	1	1	1	1	1
			R		R/W					
IPR	Interrupt Pending Register	$096			D1	D0				T0
			0	0	0	0	0	0	0	0
			R		R/W		R			R/W
		$097			S1	S0	E3	E2	E1	E0
			0	0	0	0	0	0	0	0
			R		R/W					
IISR	Interrupt In Service Register	$098			D1	D0				T0
			0	0	0	0	0	0	0	0
			R		R/W		R			R/W
		$099			S1	S0	E3	E2	E1	E0
			0	0	0	0	0	0	0	0
			R		R/W					
IVNR	Interrupt Vector Number Register	$09B	Vector							
			0	0	0	0	0	0	0	0
			R/W		R					

○ Serial interface (1)

Symbol	Name	Offset address	Data bus Upper bytes : 15 - 8 (even-numbered addresses) Lower bytes : 7 - 0 (odd-numbered addresses)							
			15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0
SMR0	Serial Mode Register 0	$181	RxINTM	ErINTM	PEO	PEN	CL1	CL0	TxINTM	ST
			1	1	–	–	–	–	1	–
	For ch0		R/W							
SCMR0	Serial Command Register 0	$183				ERS	SBRK	RxEN		TxEN
			0	0	0	1	0	0	0	0
	For ch0		R			R/W			R	R/W
SBRR0	Serial Baudrate Register 0	$185					B3	B2	B1	B0
			0	0	0	0	0	0	0	0
	For ch0		R				R/W			
SSR0	Serial Status Register 0	$187		RBRK	FE	OE	PE	TxE	RxRDY	TxRDY
			0	0	0	0	0	1	0	0
	For ch0		R							
SDR0	Serial Data Register 0	$189	D7	D6	D5	D4	D3	D2	D1	D0
			–	–	–	–	–	–	–	–
	For ch0		R/W							
SPR	Serial Prescaler Register	$18D	P7	P6	P5	P4	P3	P2	P1	P0
			–	–	–	–	–	–	–	–
			R/W							
SCR	Serial Control Register	$18F	CKSE		RES					INTM
			1	0	1	0	0	0	0	1
			R/W	R	R/W	R				R/W

○ Serial interface (2)

Symbol	Name	Offset address	Data bus Upper bytes : 15 - 8 (even-numbered addresses) Lower bytes : 7 - 0 (odd-numbered addresses)							
			15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0
SMR1	Serial Mode Register 1	$191	RxINTM	ErINTM	PE0	PEN	CL1	CL0	TxINTM	ST
			1	1	–	–	–	–	1	–
	For ch1		R/W							
SCMR1	Serial Command Register 1	$193			RTS	ERS	SBRK	RxEN	CTSEN	TxEN
			0	0	0	1	0	0	0	0
	For ch1		R		R/W					
SBRR1	Serial Baudrate Register 1	$195					B3	B2	B1	B0
			0	0	0	0	0	0	0	0
	For ch1		R				R/W			
SSR1	Serial Status Register 1	$197		PBRK	FE	OE	PE	TxE	RxRDY	TxRDY
			0	0	0	0	0	1	0	0
	For ch1		R							
SDR1	Serial Data Register 1	$199	D7	D6	D5	D4	D3	D2	D1	D0
			–	–	–	–	–	–	–	–
	For ch1		R/W							

○ Timer

Symbol	Name	Offset address	Data bus Upper bytes : 15 - 8 (even-numbered addresses) Lower bytes : 7 - 0 (odd-numbered addresses)							
			15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0
TCR	Timer Control Register	$200			P4	P3	P2	P1		
			0	0	0	0	0	0	0	0
			R		R/W				R	
		$201	N/1					INT	CS	TS
			0	0	0	1	0	0	1	0
			R/W	R				R/W		
TMCR	Timer MAX Count Register	$204	M15	M14	M13	M12	M11	M10	M9	M8
			1	0	0	0	0	0	0	0
			R/W							
		$205	M7	M6	M5	M4	M3	M2	M1	M0
			0	0	0	0	0	0	0	0
			R/W							
TCTR	Timer Count Register	$20C	C15	C14	C13	C12	C11	C10	C9	C8
			–	–	–	–	–	–	–	–
			R							
		$20D	C7	C6	C5	C4	C3	C2	C1	C0
			–	–	–	–	–	–	–	–
			R							

○ DMA controller (1)

Symbol	Name	Offset address	Data bus Upper bytes : 15 - 8 (even-numbered addresses) Lower bytes : 7 - 0 (odd-numbered addresses)							
			15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0
CHCR1	Channel Control Register 1	$280	AM1	RDY1	MACD1	MACS1	DPW1	OPW1	DST1	ERC1
			0	0	0	0	0	0	0	0
			R/W							
		$281	DSSH1	TDIR11	TDIR10	TMOD11	TMOD10	RSS1	DMAE1	INTE1
			0	0	0	0	0	0	0	0
	For ch1		R/W							
DTCR1	Data Transfer Count Register 1	$282	DT15	DT14	DT13	DT12	DT11	DT10	DT9	DT8
			–	–	–	–	–	–	–	–
			R/W							
		$283	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0
			–	–	–	–	–	–	–	–
	For ch1		R/W							
SMAR1	Source Memory Address Register1	$285	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16
			–	–	–	–	–	–	–	–
			R/W							
		$286	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8
			–	–	–	–	–	–	–	–
			R/W							
		$287	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0
	For ch1		–	–	–	–	–	–	–	–
			R/W							
DMAR1	Distination Memory Address Register 1	$289	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16
			–	–	–	–	–	–	–	–
			R/W							
		$28A	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8
			–	–	–	–	–	–	–	–
			R/W							
		$28B	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
	For ch1		–	–	–	–	–	–	–	–
			R/W							
OPCR	Operation Control Register	$28D					RWE	FC2	DMAE	INTE
			0	0	0	0	0	0	0	0
			R				R/W			

○ DMA controller (2)

Symbol	Name	Offset address	Data bus Upper bytes : 15 - 8 (even-numbered addresses) Lower bytes : 7 - 0 (odd-numbered addresses)							
			15/7	14/6	13/5	12/4	11/3	10/2	9/1	8/0
CHSR	Channel Status Register	$28E	BERW		DIT	ERRCD11	ERRCD10	ERRCD01	ERRCD00	
			0	0	0	0	0	0	0	0
			R/W	R	R/W					R
		$28F	ERR1	ERR0		REQ1	REQ0		OPC1	OPC0
			0	0	0	0	0	0	0	0
			R/W		R	R/W		R	R/W	
CHCR0	Channel Control Register 0	$290	AM0	RDY0	MACD0	MACS0	DPW0	OPW0	DST1	ERC0
			0	0	0	0	0	0	0	0
			R/W							
		$291	DSSH0	TDIR01	TDIR00	TMOD01	TMOD00	RSS0	DMAE0	INTE0
	For ch0		0	0	0	0	0	0	0	0
			R/W							
DTCR0	Data Transfer Count Register 0	$292	DT15	DT14	DT13	DT12	DT11	DT10	DT9	DT8
			–	–	–	–	–	–	–	–
			R/W							
		$293	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0
	For ch0		–	–	–	–	–	–	–	–
			R/W							
SMAR0	Source Memory Address Register0	$295	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16
			–	–	–	–	–	–	–	–
			R/W							
		$296	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8
			–	–	–	–	–	–	–	–
			R/W							
		$297	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0
	For ch0		–	–	–	–	–	–	–	–
			R/W							
DMAR0	Distination Memory Address Register0	$299	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16
			–	–	–	–	–	–	–	–
			R/W							
		$29A	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8
			–	–	–	–	–	–	–	–
			R/W							
		$29B	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0
	For ch0		–	–	–	–	–	–	–	–
			R/W							

Chapter 10 Electrical Characteristics

This section describes the electrical characteristics and timings of TMP68305.

10.1 Maximum Ratings

Table 10.1

Parameter	Symbol	Rating	Unit
		TMP68305	
Power supply voltage	Vcc	– 0.3~ + 6.5	V
Input voltage	Vin	– 0.3~ + 6.5	V
Operating temperature	Ta	0~ + 70	°C
Storage temperature	Tstg	– 55~ + 150	°C

251194

While this device includes input protection circuits against high-voltage static electricity or damage from electric fields, avoid using a voltage higher than the maximum rating. Connect input pins not in use to GND or VCC.

10.2 DC Characteristics

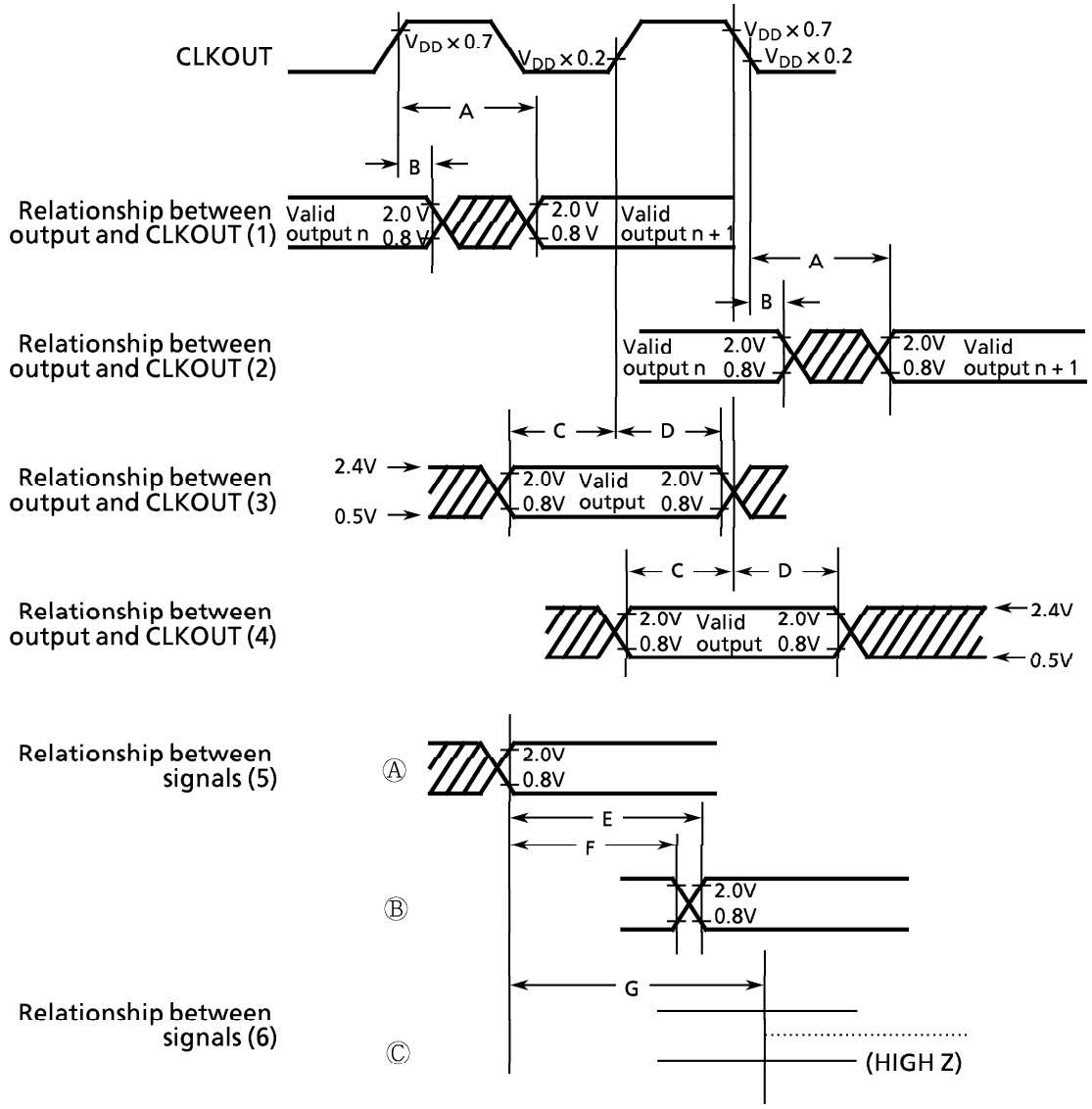
Table 10.2
(V_{CC} = 5.0 V ± 5 %, GND = 0 V, Ta = 0 to 70 °C)

Parameter		Symbol	Min	Max	Unit
High-level input voltage	Except ×2, $\overline{DREQ0}$, and $\overline{DREQ1}$	V_{IH}	2.0	V_{CC}	V
	×2		0.7V_{CC}	V_{CC}	
	$\overline{DREQ0}$, $\overline{DREQ1}$		2.2	V_{CC}	
Low-level input voltage	Except ×2	V_{IL}	GND−0.3	0.8	V
	×2		−0.3	0.3V_{CC}	
Input leakage current	(5.25V)	I_{IN}			μA
	$\overline{BERR}$, BCLK, $\overline{BR}$, $\overline{EBG}$, $\overline{BGACK}$		−	2.5	
	$\overline{DTACK}$,		−	2.5	
	$\overline{HALT}$, $\overline{RESET}$, NOR / $\overline{EMU}$,		−	20	
	$\overline{INT0}$ to $\overline{INT3}$, RxD0, RxD1,		−	20	
	$\overline{DREQ0}$, $\overline{DREQ1}$, $\overline{CTS1}$		−	20	
	$\overline{DONE}$, ×2, $\overline{RDY}$		−	20	
Tri-state (off state)	(2.4V/0.4V)	I_{TS}			μA
input current	$\overline{AS}$, A1 to A23, D0 to D15,FC0 to		−	20	
	FC2, $\overline{UDS}$, $\overline{LDS}$, R / $\overline{W}$		−	20	
			−	20	
High-level output voltage	(IOH = −400μA)	V_{OH}			V
	$\overline{AS}$, A1 to A23, D0 to D15, $\overline{BG}$,		V_{CC}−0.75	−	
	FC0 to FC2, $\overline{UDS}$, $\overline{LDS}$, R / $\overline{W}$,		V_{CC}−0.75	−	
	$\overline{CS0}$ to $\overline{CS3}$, $\overline{RTS1}$, $\overline{IACK0}$ to $\overline{IACK3}$		V_{CC}−0.75	−	
	TxD0, TxD1, CLKOUT,		V_{CC}−0.75	−	
	$\overline{EIPL0}$ to $\overline{EIPL2}$, $\overline{DACK0}$, $\overline{DACK1}$		V_{CC}−0.75	−	
	$\overline{DTC}$, $\overline{OWN}$		V_{CC}−0.75	−	
Low-level output voltage		V_{OL}			V
(IOL = 1.6mA)	$\overline{HALT}$, $\overline{RESET}$		−	0.5	
(IOL = 3.2mA)	A1 to A23, $\overline{BG}$, FC0 to FC2		−	0.5	
(IOL = 5.3mA)	$\overline{AS}$, D0 to D15, $\overline{UDS}$, $\overline{LDS}$, R / $\overline{W}$,		−	0.5	
	$\overline{CS0}$ to $\overline{CS3}$, $\overline{RTS1}$, $\overline{DTACK}$, $\overline{BERR}$,		−	0.5	
	$\overline{IACK0}$ to $\overline{IACK3}$, TxD0, TxD1,		−	0.5	
	CLKOUT, $\overline{BR}$, $\overline{BGACK}$		−	0.5	
(IOL = 8.9mA)	$\overline{DONE}$		−	0.5	
Current consumption		I_D			mA
	f = 16.67 MHz		−	100	
Power consumption		P_D			W
	f = 16.67 MHz		−	0.525	
Input capacitance		C_{IN}			pF
(Vin = 0 V, Ta = 25 °C : Frequency = 1 MHz)			−	20.0	
Load capacitance		C_L			pF
	$\overline{HALT}$		−	70	
	Others		−	130	

251194

10.3 AC Electrical Characteristics

10.3.1 Details of AC Electrical Characteristics



A : Maximum output delay

B : Minimum output hold time

C : Minimum input setup time

D : Minimum input hold time

E : Signal (A) valid - signal (B) valid time

F : Signal (A) valid - signal (B) invalid time

G : Signal (A) valid - signal (C) high impedance time

(Note) CLKOUT is based on the waveform when load capacitance is 100pF.

251194

Figure 10.1

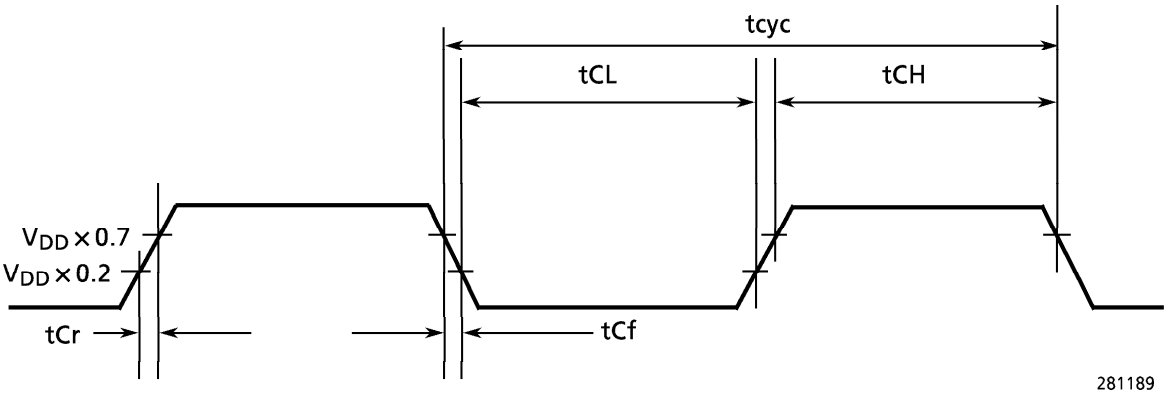
10.3.2 AC Electrical Characteristics - Clock Timings

Table 10.3

(VCC = 5.0 V ± 5 %, GND = 0 V, Ta = 0 to 70 °C; See Figure 10.2.)

Parameter	Symbol	16.67 MHz		Unit
		Min.	Max.	
Operating frequency	f	8.0	16.67	MHz
Cycle time	tcyc	60	125	ns
CLKOUT pulse width	tCL tCH	25 25	62.5 62.5	ns
Rise and fall times	tCr tCf	– –	10 10	ns

251194



Note : Unless otherwise specified, timings are measured between a low voltage of VDD x0.2V and a high voltage of VDD x 0.7V. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8V and 2.0V.

Figure 10.2 CLKOUT Output Timing

10.3.3 AC Electrical Characteristics - Read and Write Timings (1/4)

Table 10.4

(VCC = 5.0 V ± 5 %, GND = 0 V, Ta = 0 to 70 °C; See figures 10.3 and 10.4.)

Number	Parameter	Symbol	16.67 MHz		Unit
			Min.	Max.	
1	CLKOUT cycle	tCYC	60	125	ns
2	CLKOUT width low	tCL	25	62.5	ns
3	CLKOUT width high	tCH	25	62.5	ns
4	CLKOUT fall time	tCf	–	10	ns
5	CLKOUT rise time	tCr	–	10	ns
6	CLKOUT low to address valid	tCLAV	–	50	ns
6A	CLKOUT high to FC valid	tCHFCV	–	45	ns
7	CLKOUT high to address, data bus high impedance (maximum)	tCHADZ	–	50	ns
8	CLKOUT high to address, FC invalid (minimum)	tCHAFI	0	–	ns
9[1]	CLKOUT high to $\overline{AS}$, $\overline{DS}$ low	tCHSL	3	40	ns
11[2]	Address valid to $\overline{AS}$, $\overline{DS}$ low (read)/Address valid to $\overline{AS}$ low (write)	tAVSL	15	–	ns
11A[2]	FC valid to $\overline{AS}$, $\overline{DS}$ low (read)/FC valid to $\overline{AS}$ low (write)	tFCVSL	30	–	ns
12[1]	CLKOUT low to $\overline{AS}$, $\overline{DS}$ high	tCLSH	3	40	ns
13[2]	$\overline{AS}$, $\overline{DS}$ high to address/FC invalid	tSHAFI	10	–	ns
14[2]	$\overline{AS}$, $\overline{DS}$ width low (read)/$\overline{AS}$ width low (write)	tSL	120	–	ns
14A[2]	$\overline{DS}$ width low (write)	tDSL	60	–	ns
15[2]	$\overline{AS}$, $\overline{DS}$ width high	tSH	60	–	ns

251194

AC Electrical Characteristics - Read and Write Timings (2/4)

(V_{CC} = 5.0 V ± 5 %, GND = 0 V, Ta = 0 to 70 °C; See figures 10.3 and 10.4)

Number	Parameter	Symbol	16.67 MHz		Unit
			Min.	Max.	
16	CLKOUT high to control bus high impedance	tCHCZ	–	50	ns
17[2]	$\overline{AS}$, $\overline{DS}$ high to R/$\overline{W}$ high (read)	tSHRH	10	–	ns
18[1]	CLKOUT high to R/$\overline{W}$ high	tCHRH	0	40	ns
20[1]	CLKOUT to R/$\overline{W}$ low (write)	tCHRL	0	40	ns
20A[2,6]	$\overline{AS}$ low to R/$\overline{W}$ valid (write)	tASRV	–	10	ns
21[2]	Address valid to R/$\overline{W}$ low (write), FC valid to R/$\overline{W}$ low (write)	tAVRL	0	–	ns
21A[2]	FC valid to R/W low (write)	tFCVRL	20	–	ns
22[2]	R/$\overline{W}$ low to $\overline{DS}$ low (write)	tRLSL	20	–	ns
23	CLKOUT low to data out valid (write)	tCLDO	–	50	ns
25[2]	$\overline{AS}$, $\overline{DS}$ high to data out invalid (write)	tSHDOI	15	–	ns
26[2]	Data out valid to $\overline{DS}$ low (write)	tDOSL	15	–	ns
27[5]	Data in to CLKOUT low (setup time on read)	tDICL	7	–	ns
28[2]	$\overline{AS}$, $\overline{DS}$ high to $\overline{DTACK}$ high	tSHDAH	0	110	ns
29	$\overline{AS}$, $\overline{DS}$ negate to data invalid (hold time on read)	tSHDII	0	–	ns

251194

AC Electrical Characteristics - Read and Write Cycles (3/4)

(V_{CC} = 5.0 V ± 5 %, GND = 0 V, Ta = 0 to 70 °C; See figures 10.3 and 10.4.)

Number	Parameter	Symbol	16.67 MHz		Unit
			Min.	Max.	
30	$\overline{AS}$, $\overline{DS}$ high to $\overline{BERR}$ high	tSHBEH	0	–	ns
31[2.5]	$\overline{DTACK}$ low setup time (input data)	tDALDI	–	40	ns
32	$\overline{HALT}$ and $\overline{RESET}$ input transition	tRHr, f	0	150	ns
33	CLKOUT high to $\overline{BG}$ low	tCHGL	–	40	ns
34	CLKOUT high to $\overline{BG}$ high	tCHGH	–	40	ns
35	$\overline{BR}$ low to $\overline{BG}$ low	tBRLGL	1.5	3.5	Clk. Per.
36[7]	$\overline{BR}$ high to $\overline{BG}$ high	tBRHGH	1.5	3.5	Clk. Per.
37	$\overline{BGACK}$ low to $\overline{BG}$ high	tGALGH	1.5	3.5	Clk. Per.
37A[8]	$\overline{BGACK}$ low to BR high	tGALBRH	10	1.5 Clocks	ns
38	$\overline{BG}$ low to control, address, data bus, high impedance ($\overline{AS}$ high)	tGLZ	–	50	ns
39	$\overline{BG}$ width high	tGH	1.5	–	Clk. Per.

251194

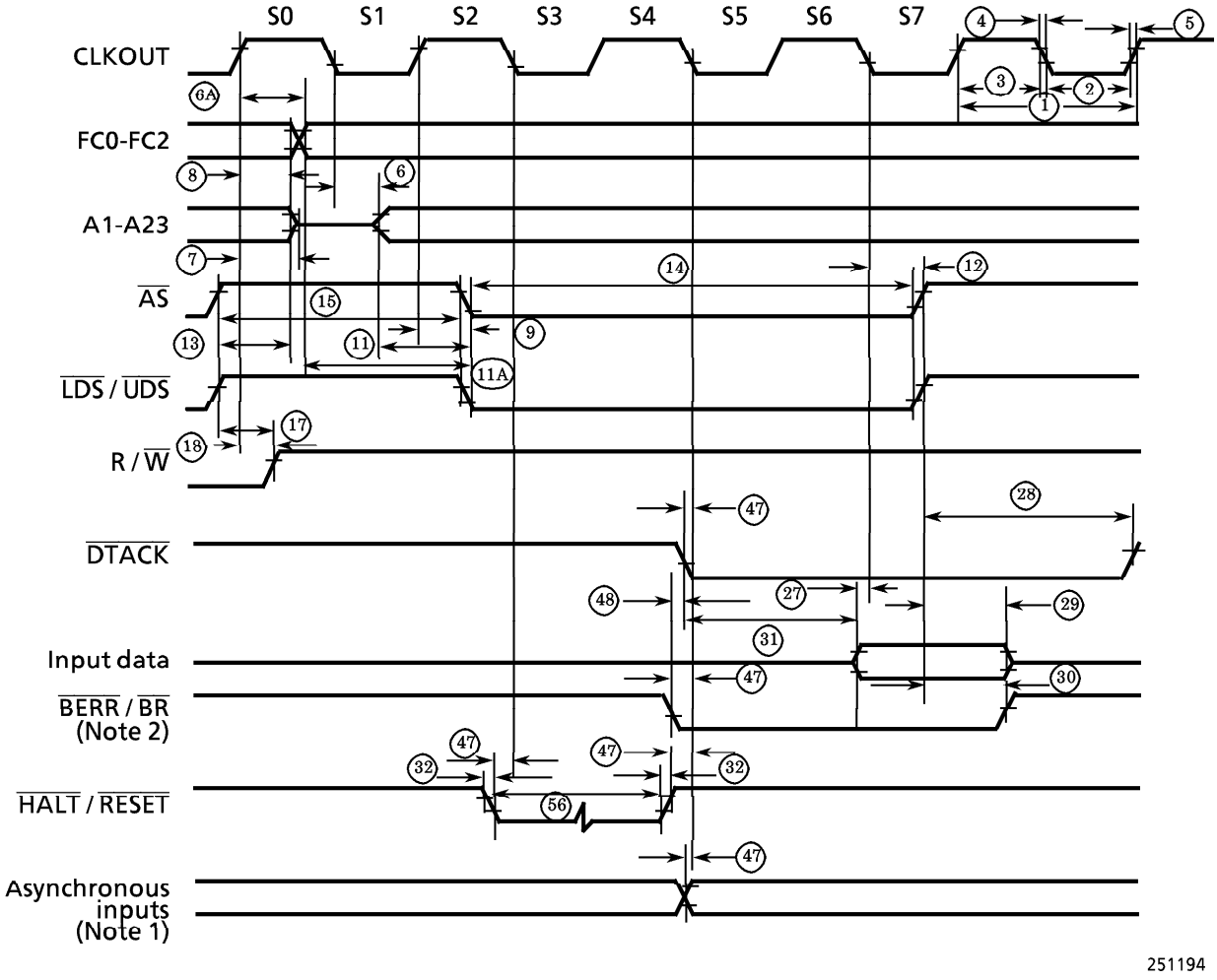
AC Electrical Characteristics - Read and Write Cycles (4/4)

(V_{CC} = 5.0 V ± 5 %, GND = 0 V, Ta = 0 to 70 °C; See figures 10.3 and 10.4.)

Number	Parameter	Symbol	16.67 MHz		Unit
			Min.	Max.	
46	$\overline{BGACK}$ width low	tGAL	1.5	–	Clk. Per.
47[5]	Asynchronous input setup time	tASI	10	–	ns
48[2,3]	$\overline{BERR}$ low to $\overline{DTACK}$ low	tBELDAL	10	–	ns
53	CLKOUT high to data out invalid	tCHDOI	0	–	ns
55	R/$\overline{W}$ low to data bus drive	tRLDBD	0	–	ns
56[4]	$\overline{HALT}$/$\overline{RESET}$ pulse width	tHRPW	10	–	Clk. Per.
57	$\overline{BGACK}$ high to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ drive	tGASD	1.5	–	Clk. Per.
58[7]	$\overline{BR}$ high to control bus drive	tGHSD	1.5	–	Clk. Per.

251194

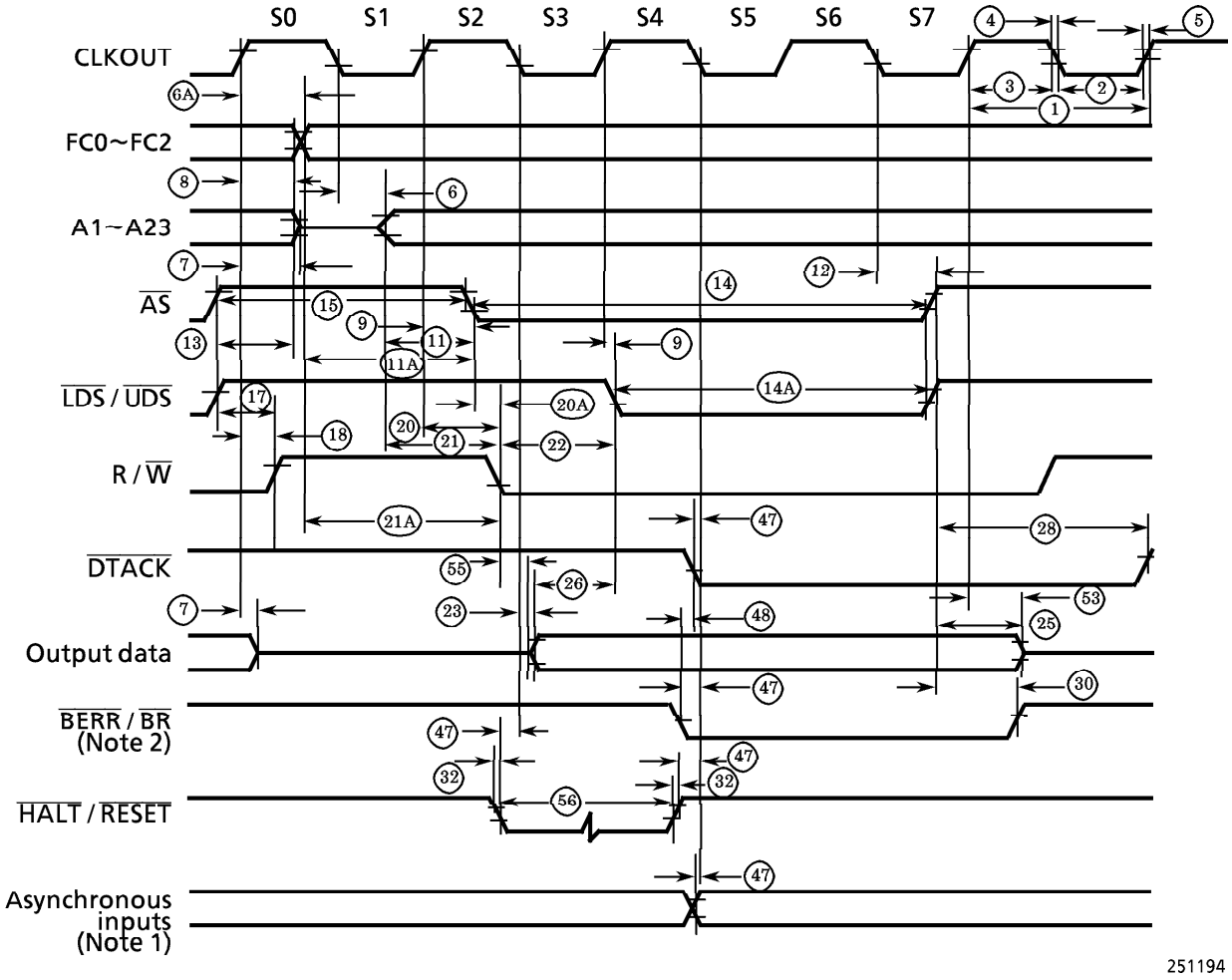
Notes:
1. For a loading capacitance of less than or equal to 50 pF, subtract 5 ns from the value given in the maximum columns.
2. Actual value depends on CLKOUT cycle.
3. If #47 is satisfied for both $\overline{DTACK}$ and $\overline{BERR}$, #48 may be 0 nanoseconds.
4. At power on, hold reset status for 1 to 100 μs to stabilize MPU circuits. See #56 for the minimum reset times following power on.
5. If the asynchronous setup time (#47) requirements are satisfied, the $\overline{DTACK}$ low-to-data setup time (#31) requirement can be ignored. The data must only satisfy the data-in to CLKOUT-low setup time (#27) requirement for the following cycle.
6. When $\overline{AS}$ and R/$\overline{W}$ are equally loaded (±20%), subtract 5 ns from the tASRV maximum value.
7. The processor will negate $\overline{BG}$ and begin driving the bus again if external arbitration logic negates $\overline{BR}$ before asserting $\overline{BGACK}$.
8. To guarantee proper operation, the minimum value must be met. If the maximum value is exceeded, $\overline{BG}$ may be reasserted.



Notes:

1. Asynchronous input $\overline{\text{BGACK}}$ is detected at the CLKOUT's falling edge.
2. It is necessary to assert at this timing only if $\overline{\text{BR}}$ is recognized at the end of this bus cycle.
3. Unless otherwise specified, timings are measured between a low voltage of 0.8V and a high voltage of 2.0V. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8 V and 2.0 V.

Figure 10.3 Read Cycle Timing



Notes :

1. Unless otherwise specified, timings are measured between a low voltage of 0.8V and a high voltage of 2.0V. The voltage swing through this range should start outside and pass through the range such that the rise or fall will be linear between 0.8V and 2.0V.
2. Because of loading variations, R/$\overline{W}$ may become valid after $\overline{AS}$ even though both are asserted at the rising edge of S2 (#20A).

Figure 10.4 Write Cycle Timing

10.3.4 AC Electrical Characteristics - Bus Arbitration (1/2)

Table 10.5

(VCC = 5.0 V ± 5 %, GND = 0 V, Ta = 0 to 70 °C; See figures 10.5 - 7)

Number	Parameter	Symbol	16.67 MHz		Unit
			Min.	Max.	
7	CLKOUT high to address, data bus high impedance	tCHADZ	–	50	ns
16	CLKOUT high to control bus high impedance	tCHCZ	–	50	ns
33	CLKOUT high to $\overline{BG}$ low	tCHGL	–	40	ns
34	CLKOUT high to $\overline{BG}$ high	tCHGH	–	40	ns
35	$\overline{BR}$ low to $\overline{BG}$ low	tBRLGL	1.5	3.5	Clk. Per.
36[1]	$\overline{BR}$ high to $\overline{BG}$ high	tBKHGH	1.5	3.5	Clk. Per.
37	$\overline{BGACK}$ low to $\overline{BG}$ high	tGALGH	1.5	3.5	Clk. Per.
37A[2]	$\overline{BGACK}$ low to $\overline{BR}$ high	tGALBRH	10	1.5 Clocks	ns
38	$\overline{BG}$ low to control, address, data bus high impedance ($\overline{AS}$ high)	tGLZ	–	50	ns
39	$\overline{BG}$ width high	tGH	1.5	–	Clk. Per.
46	$\overline{BGACK}$ width low	tGAL	1.5	–	Clk. Per.
47	Asynchronous input setup time	tASI	10	–	ns
57	$\overline{BGACK}$ high to control bus drive	tGABD	1.5	–	Clk. Per.
58[1]	$\overline{BG}$ high to control bus drive	tGHBD	1.5	–	Clk. Per.

251194

Notes :

1. The processor will negate $\overline{BG}$ and begin driving the bus again if external arbitration logic negates $\overline{BR}$ before asserting $\overline{BGACK}$.
2. To guarantee proper operation, the minimum value must be met. If the maximum value is exceeded, $\overline{BG}$ may be reasserted.

AC Electrical Characteristics - Bus Arbitration (2/2)

(VCC = 5.0 V ± 5 %, GND = 0 V, Ta = 0 to 70 °C; See figures 10.5 to 7)

Number	Parameter	Symbol	16.67 MHz		Unit
			Min.	Max.	
110	$\overline{DREQ}$ setup time (CLKOUT high)	tCHDRL	15	–	ns
111	$\overline{DREQ}$ hold time (CLKOUT high)	tCHDRH	30	–	ns
112	$\overline{BGACK}$ low to $\overline{DREQ}$ high	tBLDRH	20	–	ns
113[9]	CLKOUT to $\overline{BR}$ low	tCBRL	–	30	ns
114	Control bus invalid to $\overline{BGACK}$ low	tASZBL	1.0	2.0	Clk.
115	CLKOUT high to $\overline{BGACK}$ low	tCHBL	–	35	ns
116	CLKOUT high to $\overline{BGACK}$ high impedance	tCHBZ	–	35	ns
133	$\overline{BGACK}$ high impedance to $\overline{BG}$ low	tBGZGL	–	30	ns

251194

9. The second and subsequent cycle steal transfers start at CLKOUT high; otherwise, CLKOUT low.

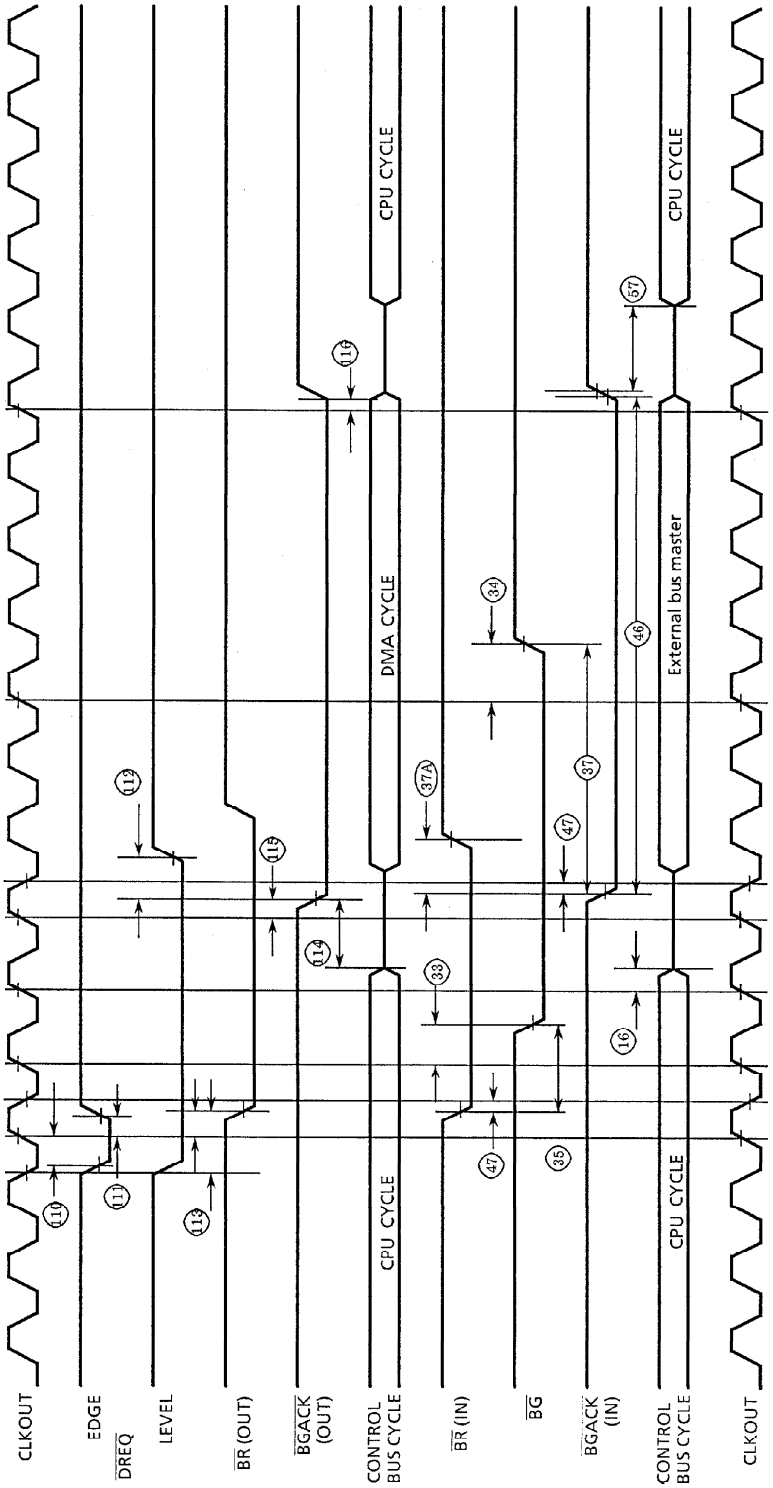


Figure 10.5 Bus Arbitration Timing (active case)

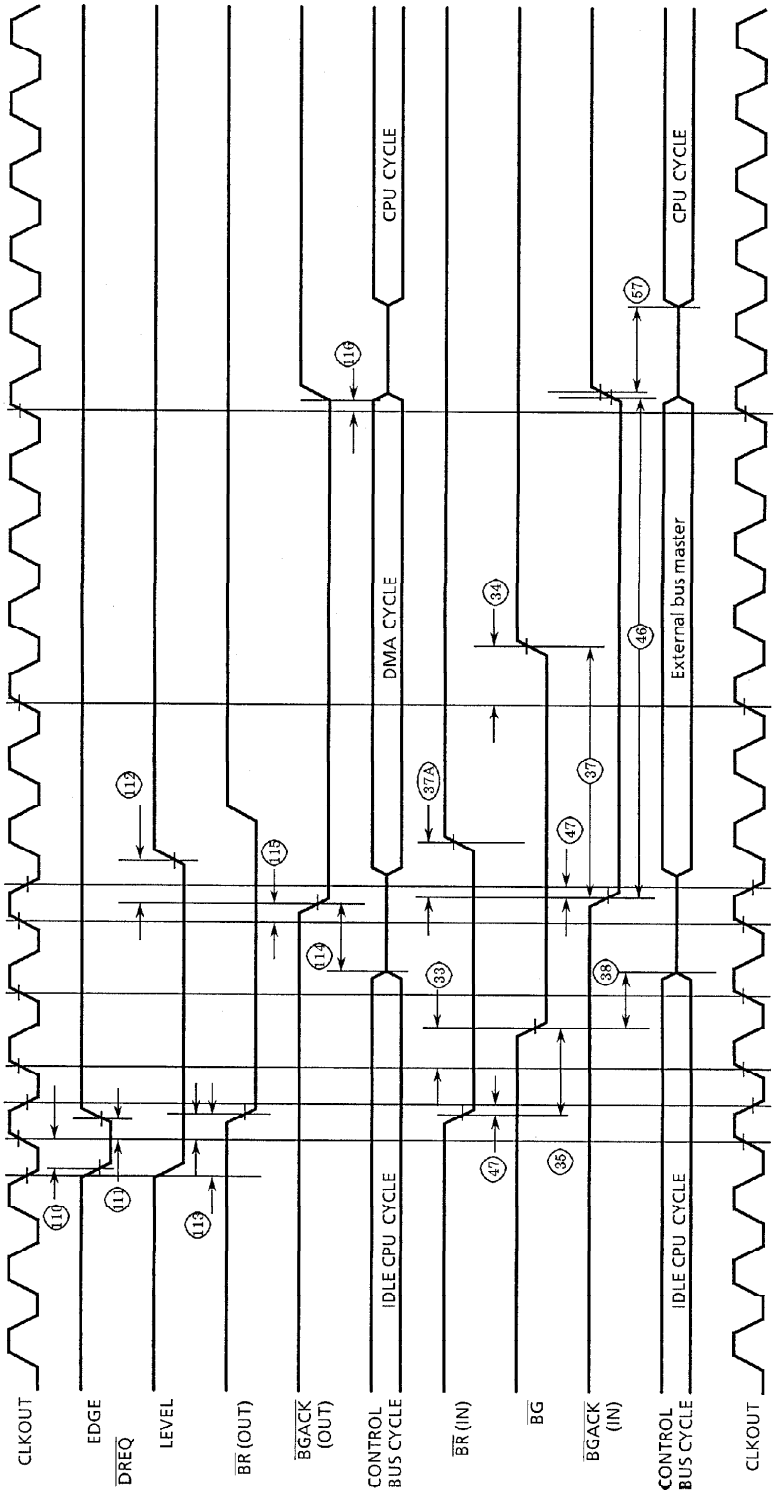
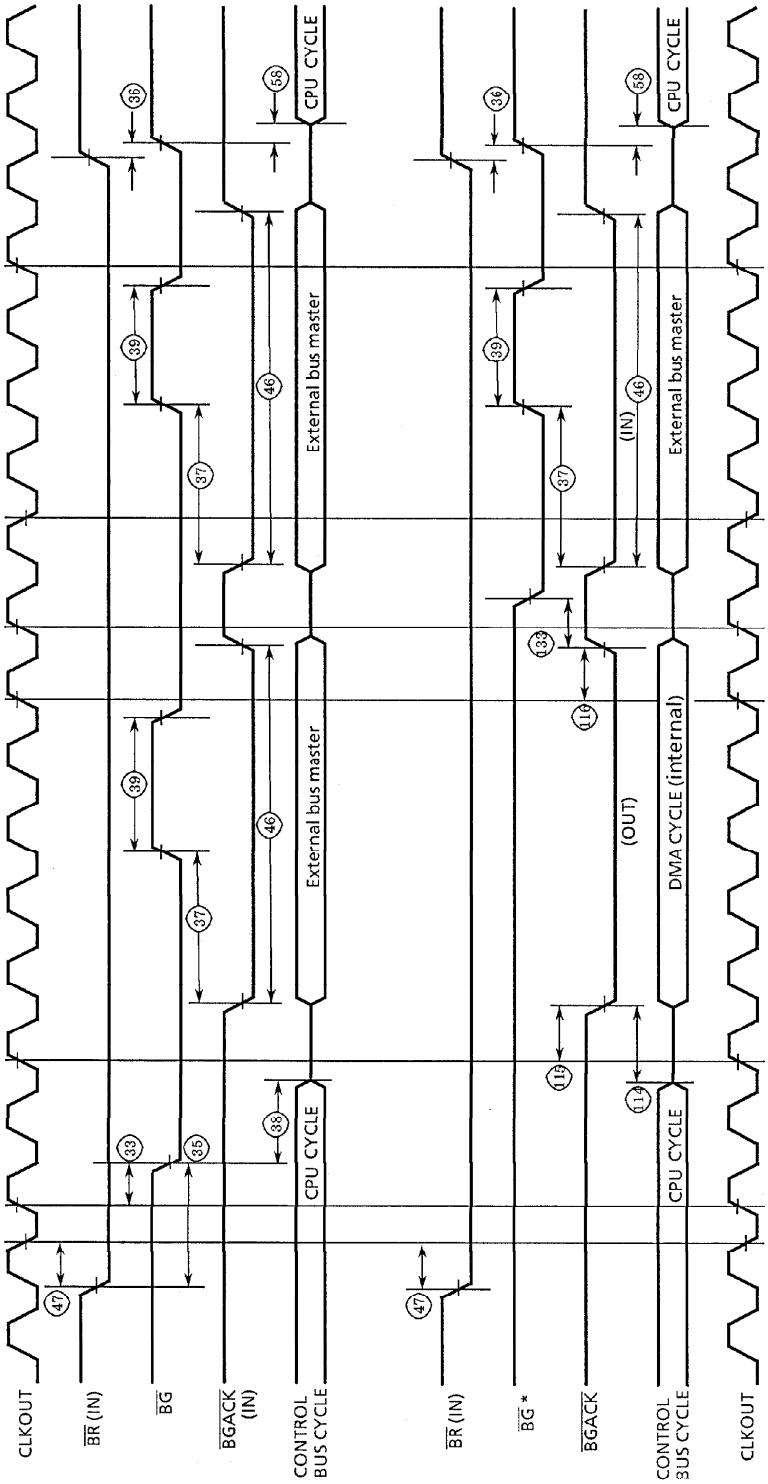


Figure 10.6 Bus Arbitration Timing (idle bus case)



*: When the internal DMAC and the external bus master assert BR simultaneously, after the internal DMA cycle ends, BG is asserted at the timing according to specification #133.

Figure 10.7 Bus Arbitration Timing (multiple bus requests)

10.3.5 AC Electrical Characteristics - Peripherals

Table 10.6
(VCC = 5.0 ± 5%, GND = 0V, Ta = 0 to 70°C; See figures 10.8 to 10.17)

Number	Parameter	Symbol	16.67 MHz		Unit
			Min.	Max.	
47	Asynchronous input setup time	tASI	10	—	ns
101	CLKOUT to $\overline{CS}$, $\overline{IACK}$	tCDS	—	50	ns
103	BCLK cycle	tBCYC	125	—	ns
104	BCLK width low	tBCL	55	—	ns
105	BCLK width high	tBCH	55	—	ns
106	BCLK rise time	tBCr	—	10	ns
107	BCLK fall time	tBCf	—	10	ns
117	$\overline{BGACK}$ low to address valid	tBLAV	—	60	ns
118	$\overline{BGACK}$ high impedance to address FC invalid	tBZAZ	0	60	ns
119	$\overline{BGACK}$ low to $\overline{AS}$ drive	tBLASV	—	40	ns
120	$\overline{BGACK}$ high impedance to $\overline{AS}$ high impedance	tBZASZ	—	50	ns
121	CLKOUT high to $\overline{AS}$, $\overline{DS}$, R/$\overline{W}$ low	tCHASL	—	50	ns
122	CLKOUT high to $\overline{DACK}$ high	tCHDAH	—	50	ns
123	CLKOUT high to $\overline{DACK}$ low	tCHDAL	0	50	ns
124	$\overline{DACK}$ high to $\overline{DONE}$ high	tCHDOH	—	30	ns
125	$\overline{DACK}$ low to $\overline{DONE}$ low	tCLDOL	—	30	ns
126	CLKOUT high to $\overline{DTC}$ high	tCHDTH	—	30	ns
127	CLKOUT high to $\overline{DTC}$ low	tCHDTL	—	45	ns
128	$\overline{DONE}$ input setup time ($\overline{AS}$ low)	tASLDOL	20	—	ns
129	$\overline{DONE}$ input width	tDOW	1.5	—	Clk.
130	$\overline{OWN}$ low to $\overline{BGACK}$ low	tOLBL	20	—	ns
131	$\overline{BGACK}$ low to FC valid	tBLFV	—	25	ns
132	$\overline{BGACK}$ high impedance to $\overline{OWN}$ high	tBZOH	20	—	ns
134	$\overline{DTC}$ high to $\overline{DACK}$ high	tDTHDAH	0	—	ns
135	$\overline{RDY}$ low to $\overline{DS}$ low (on write)	tRLDL	100	—	ns
136	$\overline{DTC}$ width low	tDTW	35	—	ns
137	$\overline{DTACK}$ input setup time (DMA cycle)	tDDAL	20	—	ns

251194

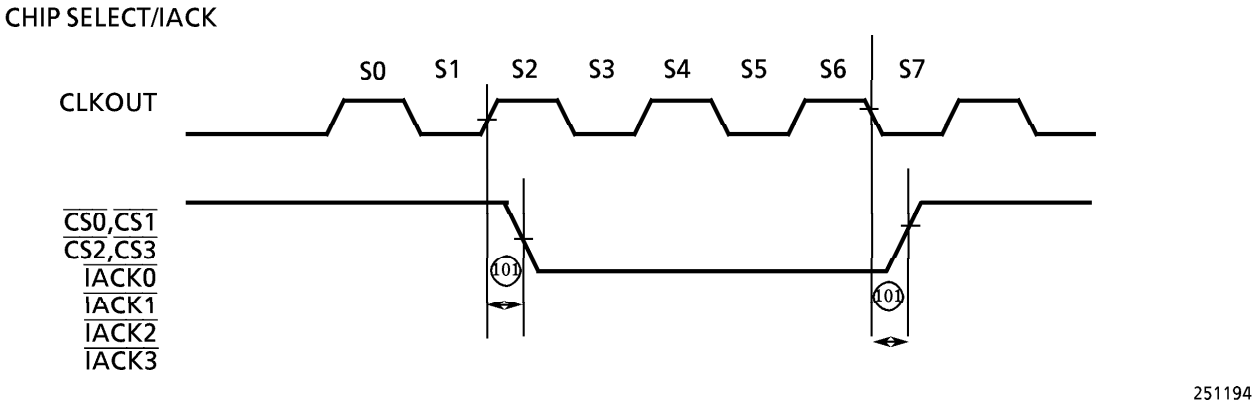


Figure 10.8 $\overline{CS}$, $\overline{IACK}$ Timing

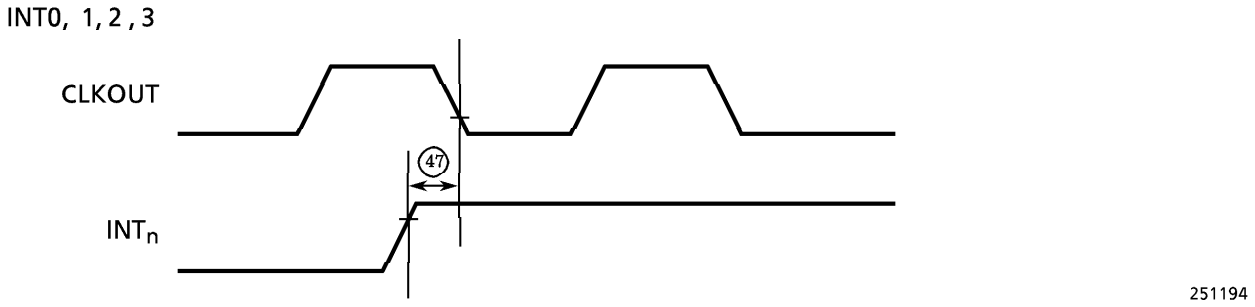


Figure 10.9 Interrupt Request Timing

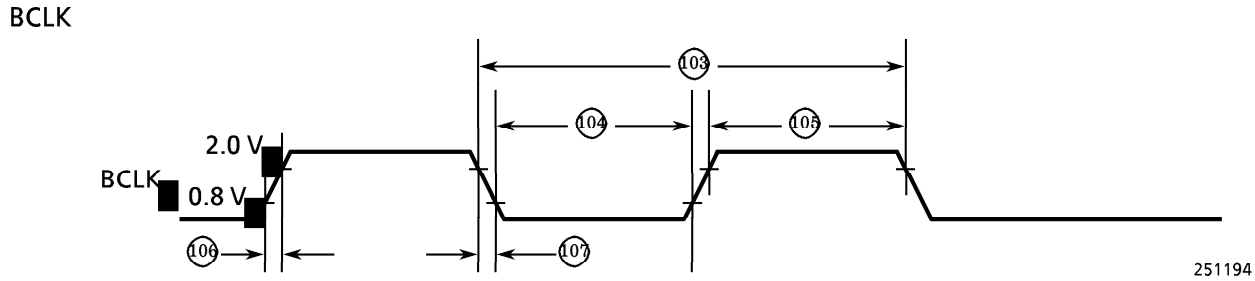


Figure 10.10 Baud Rate Clock Timing

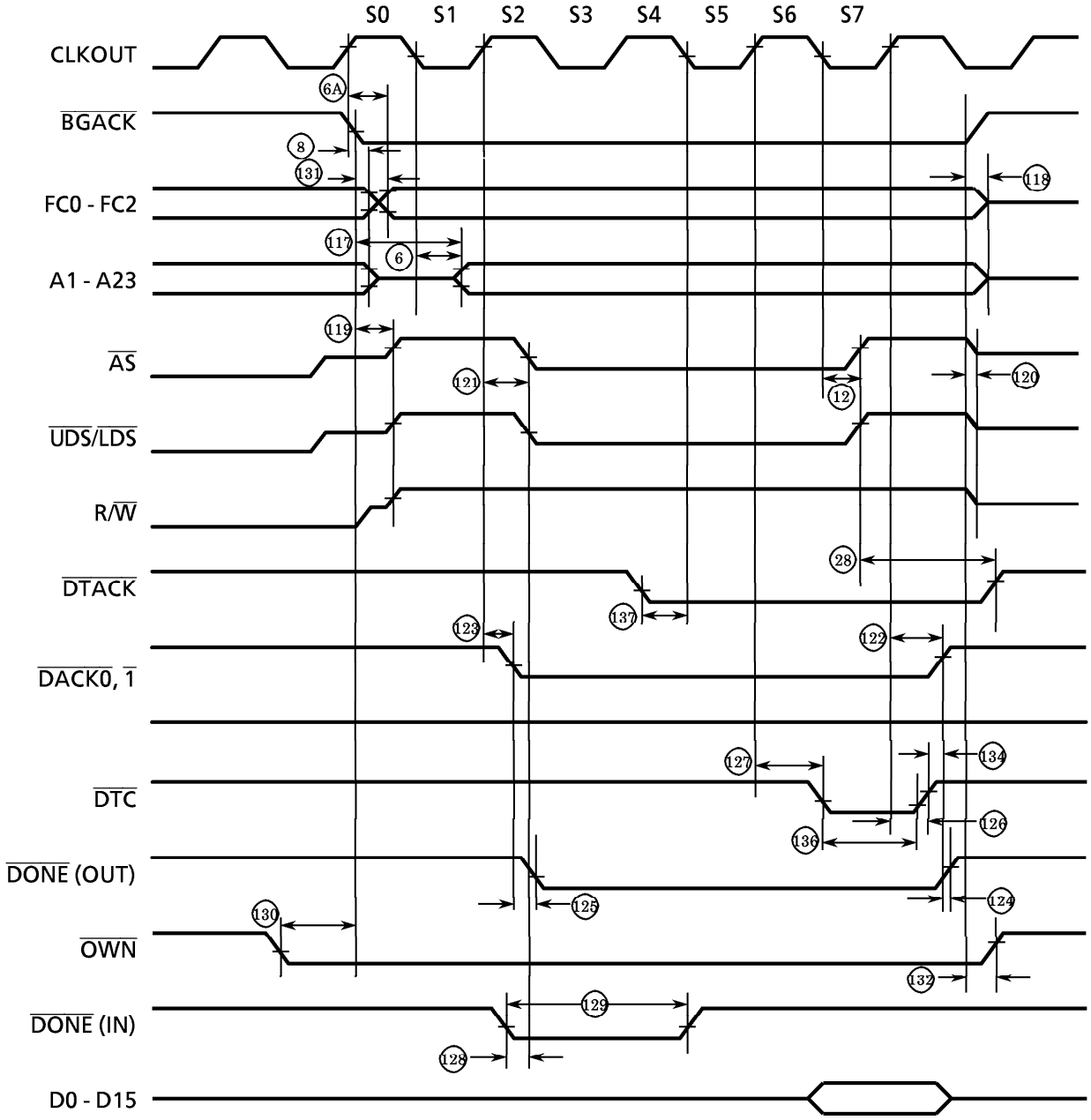
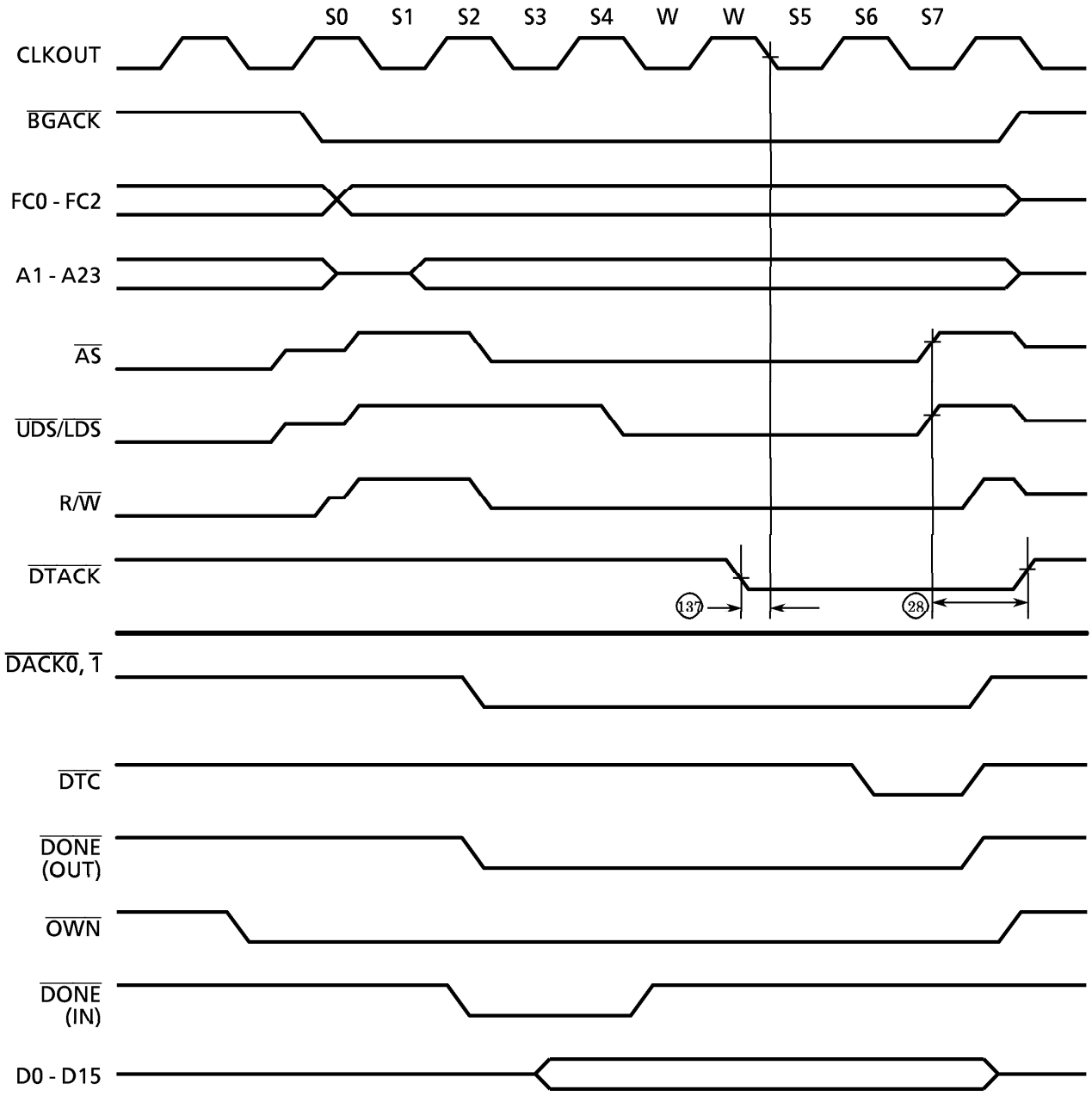


Figure 10.11 Single Address Mode (without $\overline{RDY}$, on read)



251194

Figure 10.12 Single Address Mode (without $\overline{RDY}$, on write)

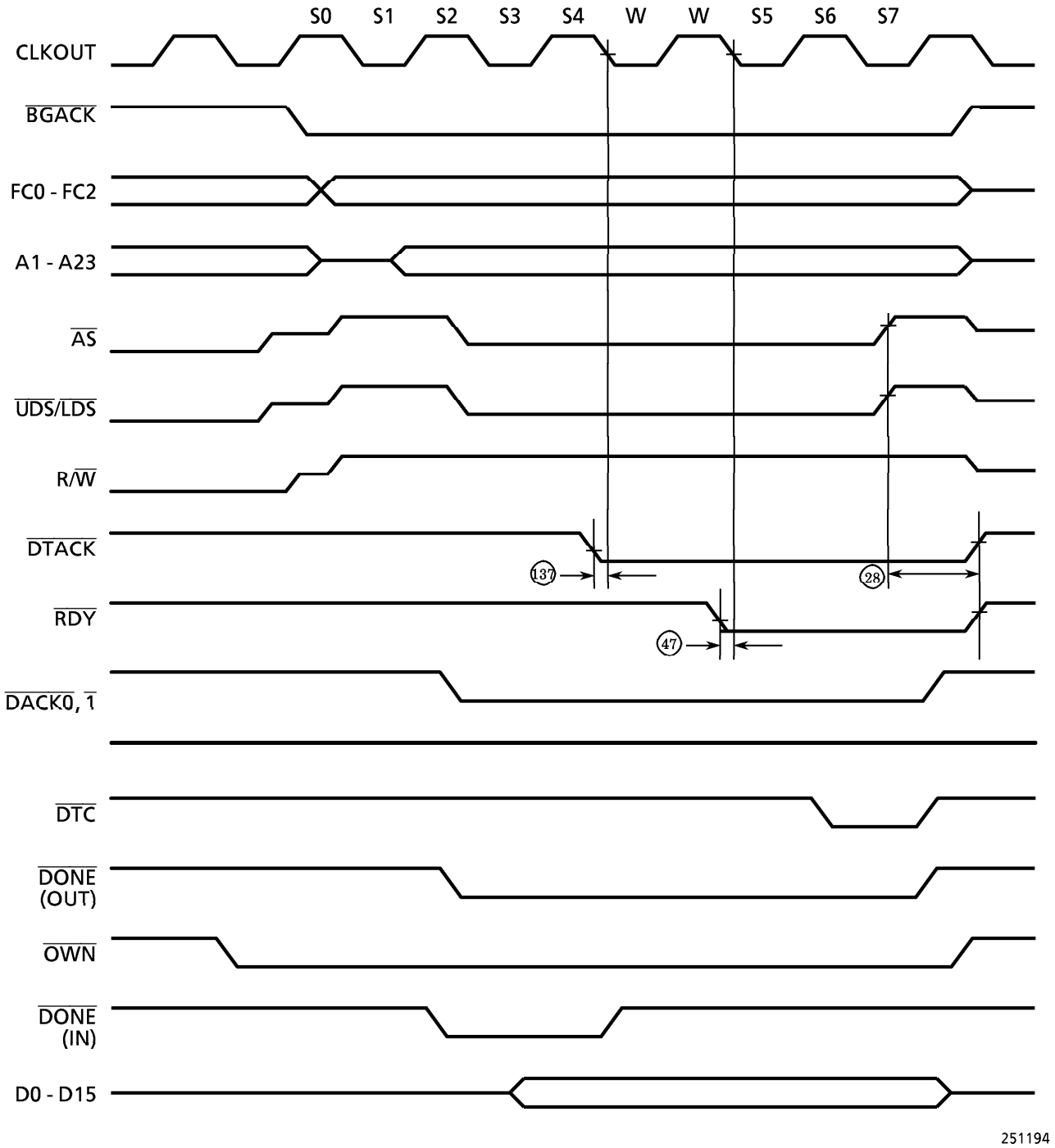


Figure 10.13 Single Address Mode (with $\overline{RDY}$, on read)

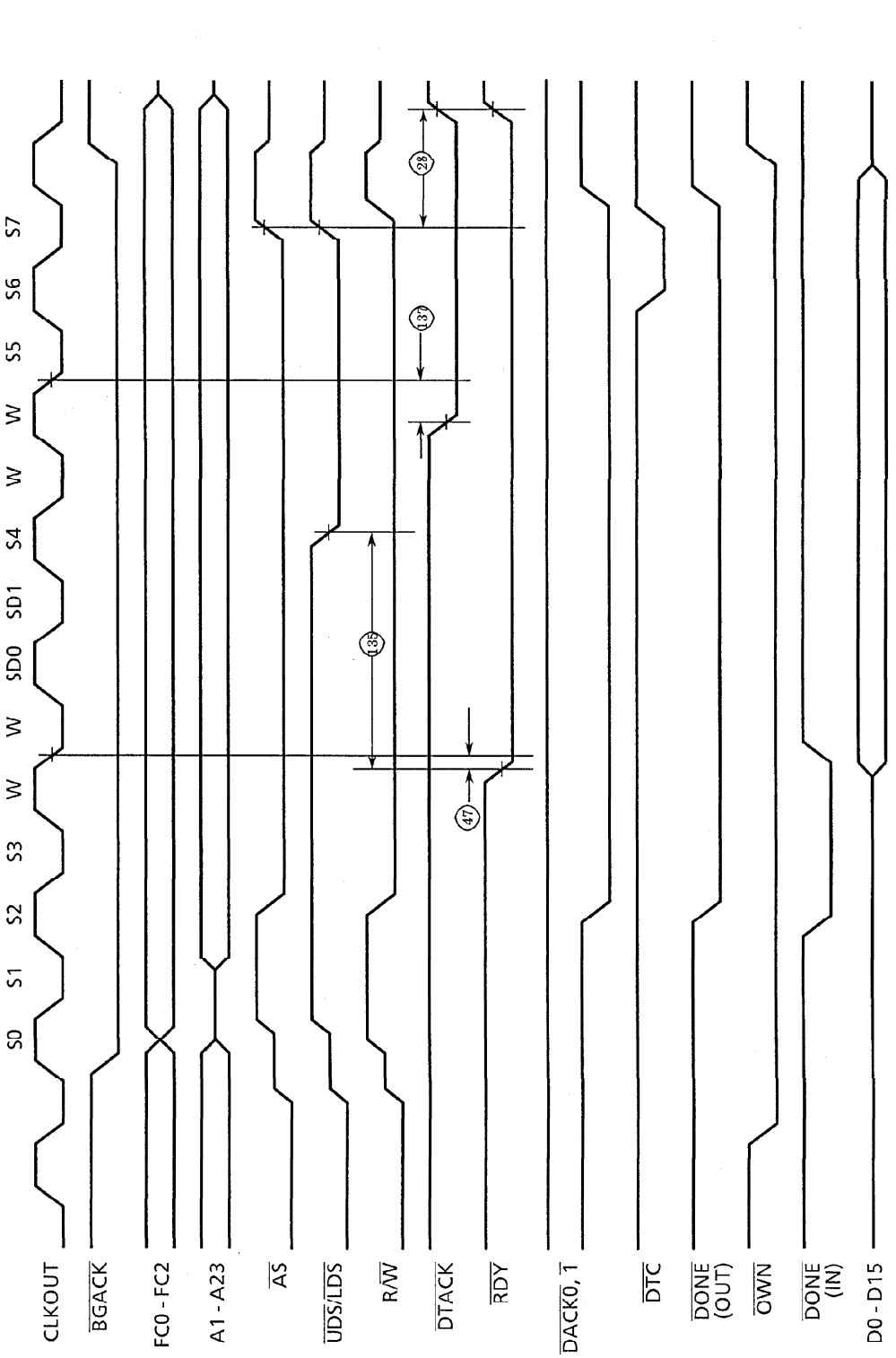


Figure 10.14 Single Address Mode (with $\overline{RDY}$, on write)

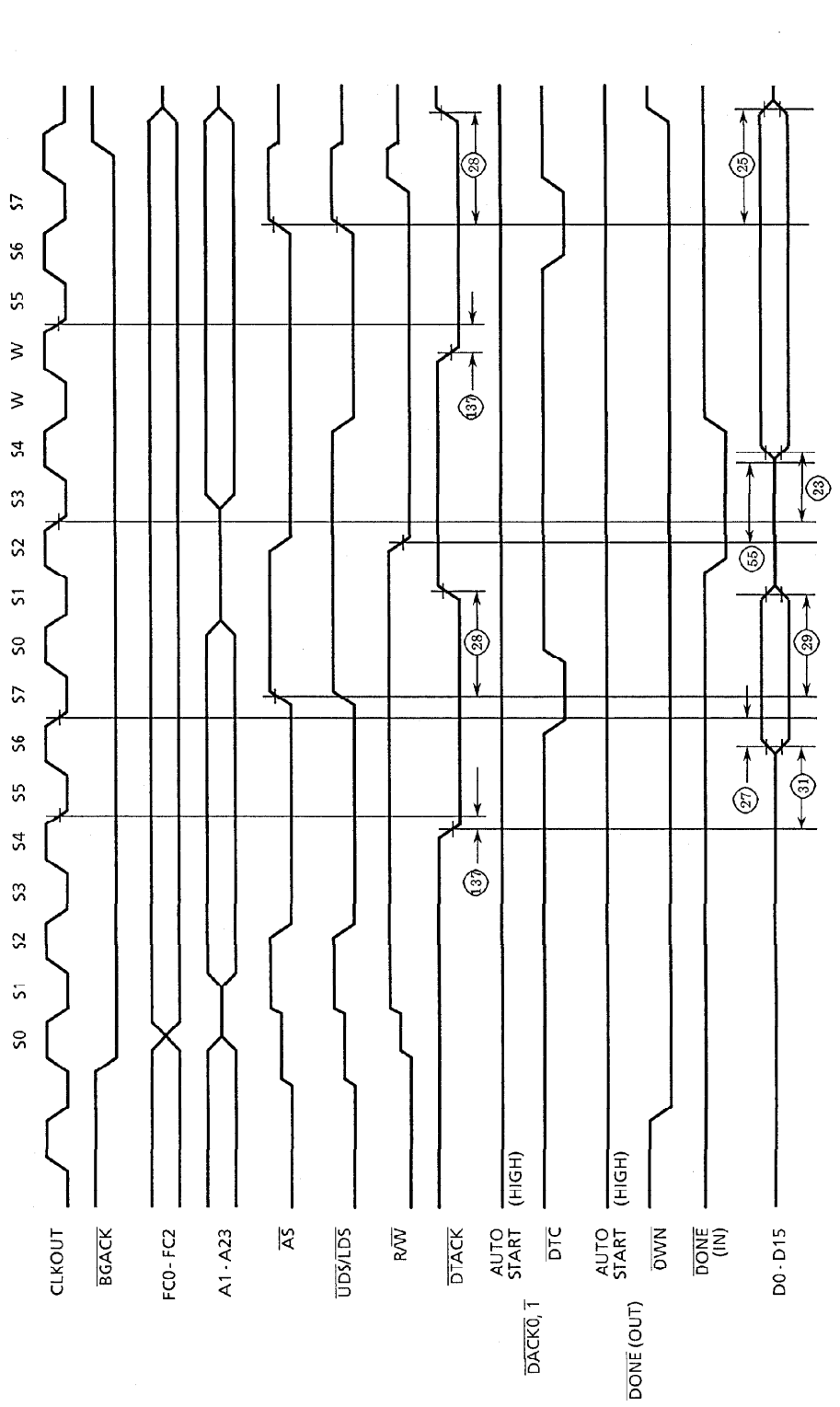


Figure 10.15 Dual Address Mode (memory↔memory)

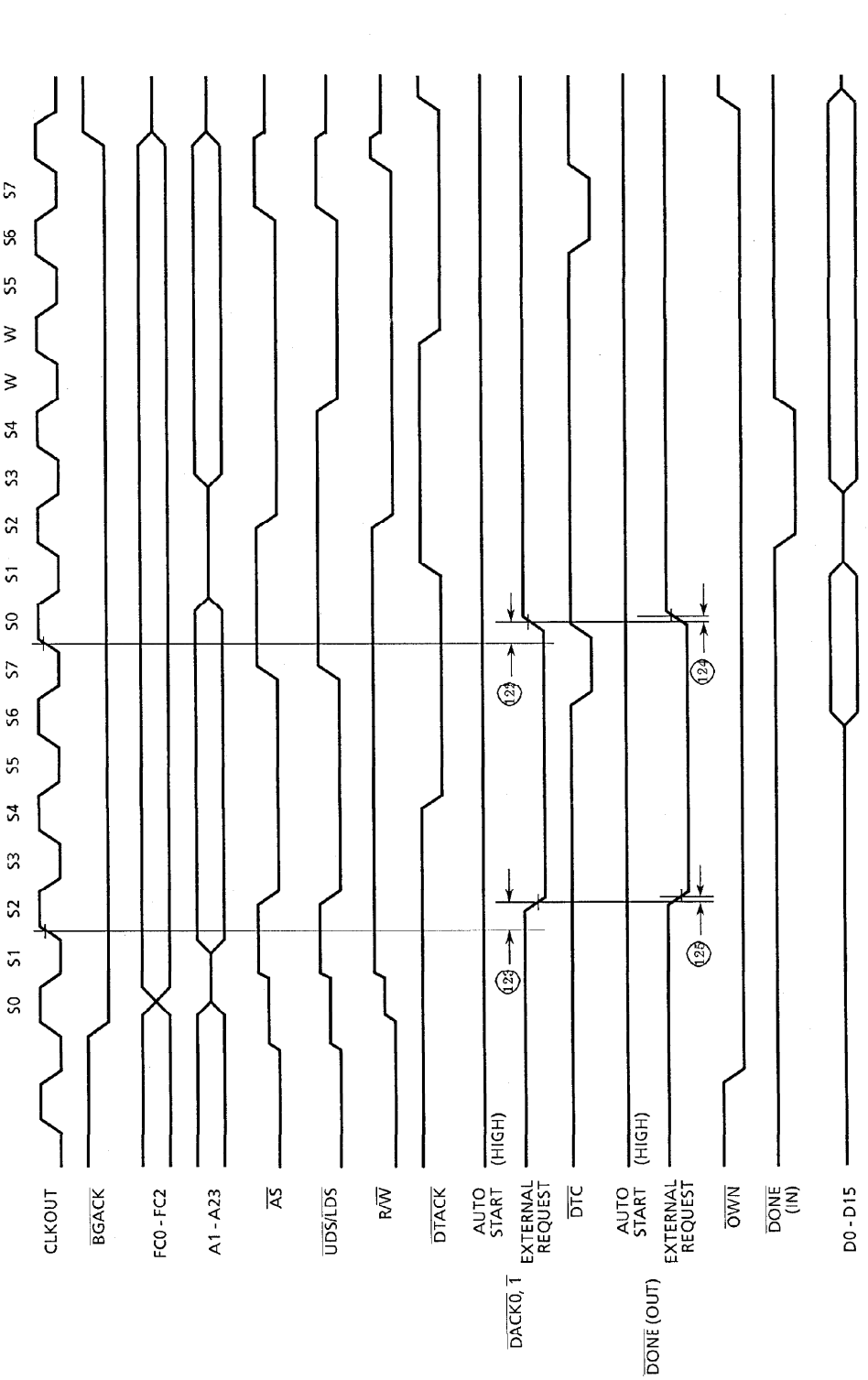


Figure 10.16 Dual Address Mode (memory-mapped I/O↔memory)

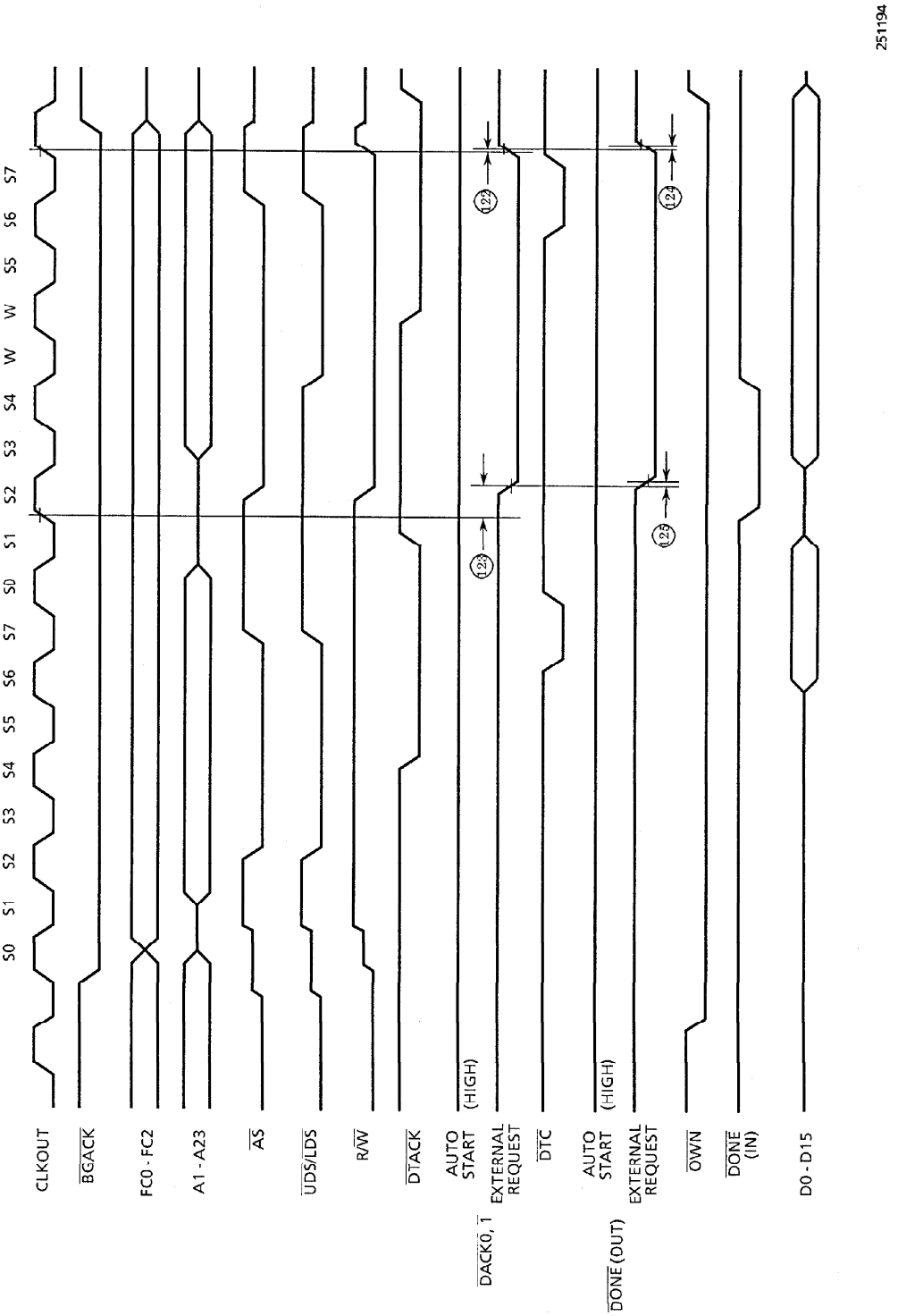


Figure 10.17 Dual Address Mode (memory↔memory-mapped I/O)

Chapter 11 Development Environment

11.1 Emulation mode

To put TMP68305 in emulation mode (henceforth, EMU mode), set the NOR/$\overline{\text{EMU}}$ pin to low. In EMU mode, the TMP68305 CPU is completely disconnected and only peripheral devices are active. In EMU mode, the emulator controls the peripheral devices.

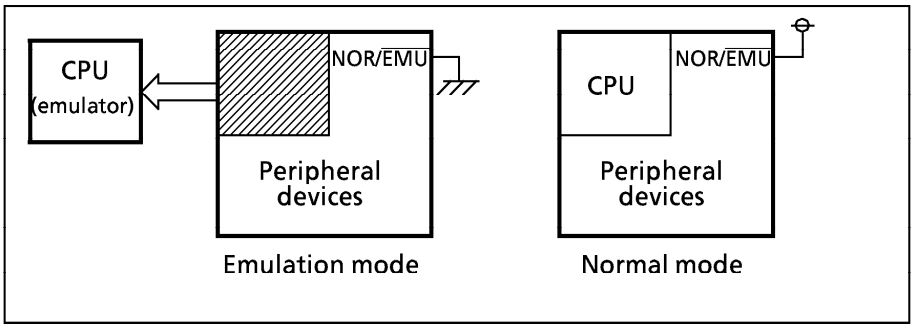


Figure 11.1 Outline of Normal and Emulation Modes

11.1.1 Pin Functions

(1) NOR/$\overline{\text{EMU}}$: Fixed to low in EMU mode. Fixed to high in normal mode.

(2) $\overline{\text{EIPL0}}$~$\overline{2}$: In EMU mode, $\overline{\text{EIPL0}}$ to $\overline{\text{IPL2}}$ are interrupt request (IPL) signals output from the TMP68305 internal peripheral circuits to the emulator. In normal mode, these signals are input signals. As they have no significance, fix $\overline{\text{EIPL0}}$, $\overline{\text{EIPL1}}$to low level and $\overline{\text{EIPL2}}$ in Open. (See *1 in the figure)

(3) $\overline{\text{EBG}}$: In EMU mode, this pin receives $\overline{\text{BG}}$ signals from the emulator. If there is an external bus master, connect the signal output from the TMP68305 $\overline{\text{BG}}$ pin to the external bus master.
As these signals have no significance in normal mode, fix to high level. (See *1 in the figures)

(4) TM0, TM1 : As these pins are not significant in either normal or EMU mode, make Open.

11.2 Connecting to General-Purpose 68000 Emulator

In principle, TMP68305 emulation can be performed using any commercially available general-purpose 68000 emulator. Note, however, the following points :

(1) As TMP68305F does not have E, $\overline{VPA}$, and $\overline{VMA}$ signals for controlling 8-bit peripheral devices, the $\overline{VPA}$ input to the emulator must be pulled-up.

(2) The 68000 interrupt vector automatic generation function cannot be used because TMP68305 does not have a $\overline{VPA}$ signal. However, TMP68305F has an equivalent function in its internal interrupt controller. For interrupt requests from external devices, use the interrupt input pins INT0 to INT3 connected to the interrupt controller instead of $\overline{IPL}$.

(3) The $\overline{DTACK}$ and $\overline{BERR}$ signals are open drain outputs in -EMU mode. Therefore, the DTACK and $\overline{BERR}$ signals generated on the target board must also be open drain outputs and pull-up resistors must be provided. At *2 in the figures, high resistance can result in slow rise times and cause operational problems. Therefore, use resistances of about 1 kΩ.

(4) The following problems can occur due to a slow rise time for the AS signal output from the general-purpose 68000 emulator.
- If another interrupt occurs while an interrupt is pending, the pending bit is cleared and subsequent interrupts are not accepted.
- If the $\overline{AS}$ signal setup time (*1 in Figure 11.4) does not satisfy the specification, the operation of the $\overline{DTACK}$ automatic generation function may be impaired and $\overline{DTACK}$ may not be asserted. (If $\overline{DTACK}$ is not asserted, the bus cycle does not complete and a bus error is generated.) (Figure 11.4)

For some 68000 emulators, the $\overline{AS}$ signal is always too slow. This problem can be solved, for example, by changing the relative phases of the clock externally input from the X2 pin to the emulator and to 68305, as shown in Figure 11.5.

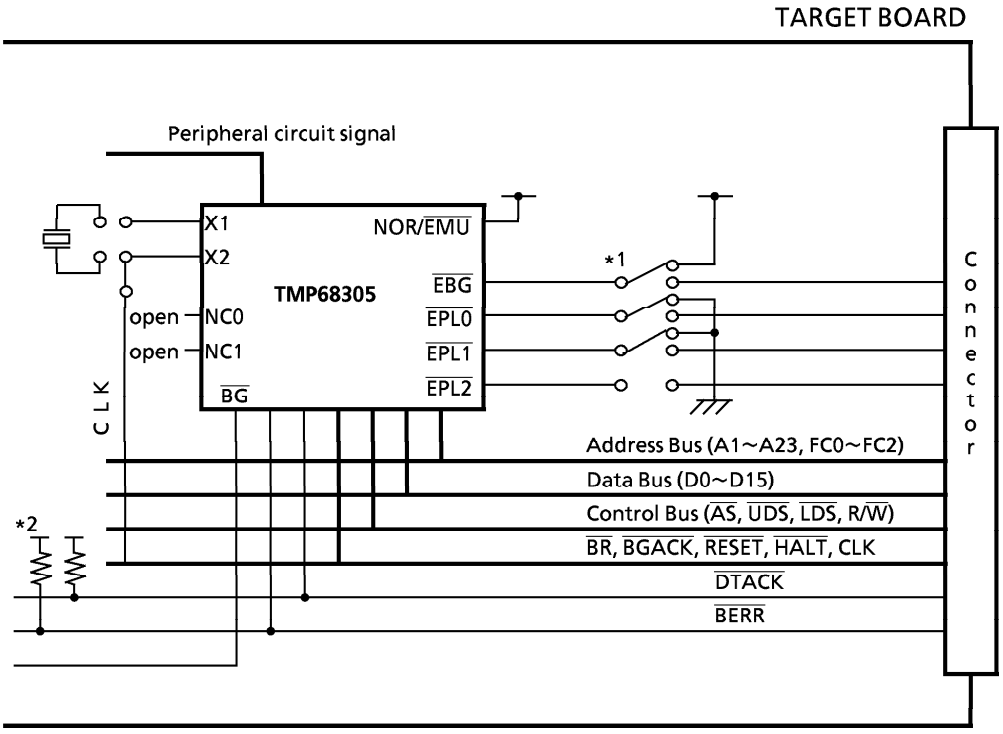


Figure 11.2 Wiring in Normal Mode

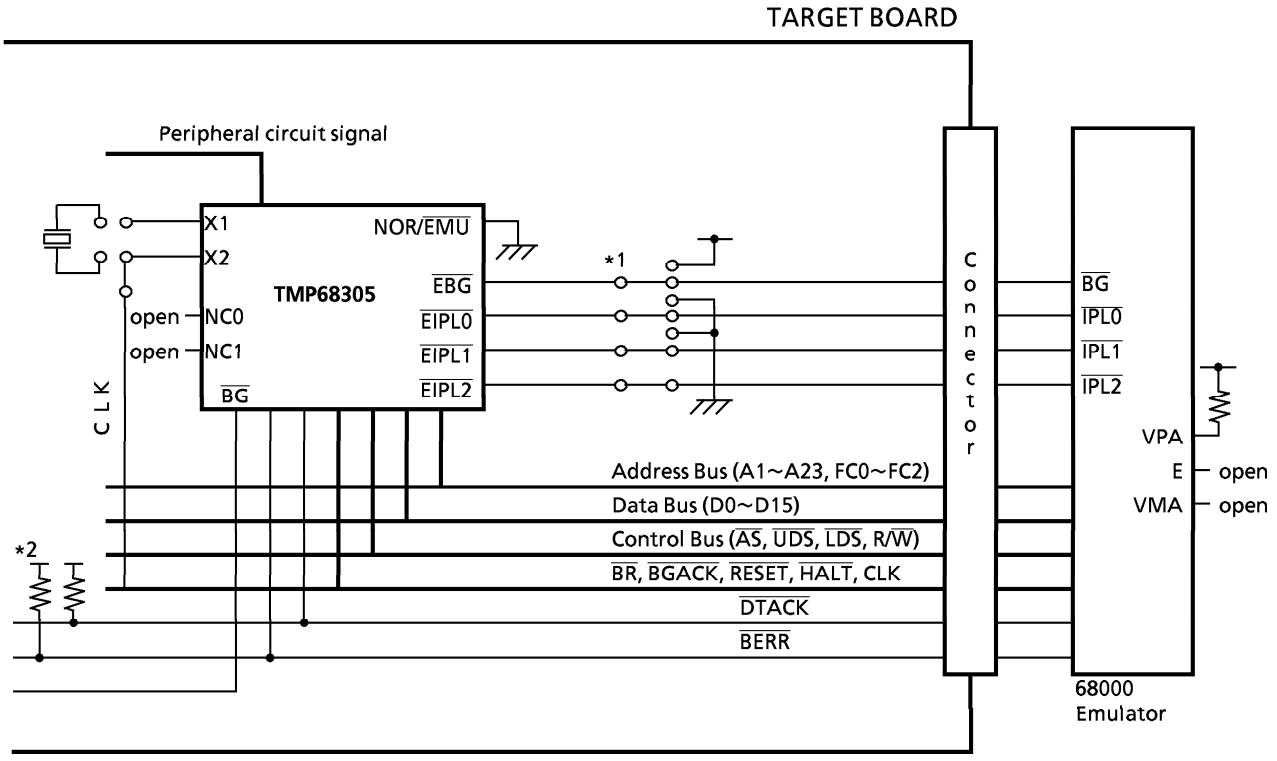


Figure 11.3 Connection to Emulator in EMU Mode

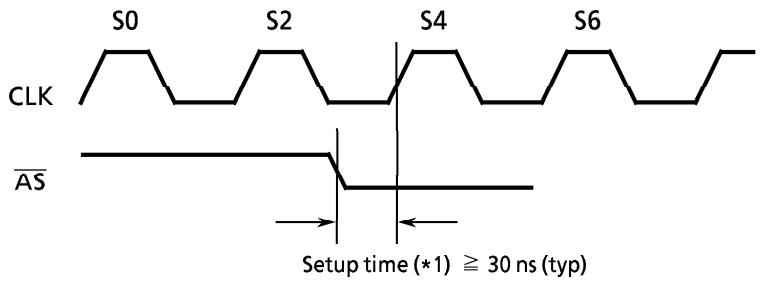


Figure 11.4 AS Signal Setup Time during Emulation

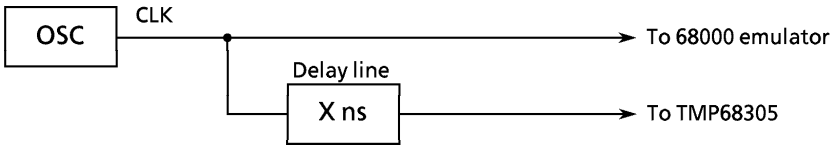


Figure 11.5 Clock Adjustments for Emulation