Part 4

TMP68303F-16

1. Introduction

TMP68303 is a high-speed, high-function 16-bit microprocessor developed for applications in control devices.

TMP68303 uses the 68HC000, the CMOS version of the 68000, as its core processor. It also includes peripheral circuits such as a serial interface, parallel interface, timer, interrupt controller, DMA controller, DRAM controller, stepping motor controller, and address decoder.

In addition, TMP68303 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
- 10-bit parallel I/O interface
- 3-channel, 16-bit timer/counter (with built-in watchdog timer)
- 2-channel, 8-bit timer/counter
- 12-channel interrupt controller (3 external channels, 9 internal channels)
- 3-channel DMA controller (max. 8 MB/s)
- DRAM controller (RAS/CAS, multiplexed address output)
- 2-channel stepping motor controller (4-phase output)
- 2-channel chip-select signal output (CS0, CS1)
- Automatic wait insertion
- Bus monitor function
- Low power consumption (CMOS)

TMP68303 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 TMP68303 is disconnected from the bus, and the internal peripheral circuits are controlled by address, data, and control signals from the development tool.

Figure 1.1 is the TMP68303 block diagram.

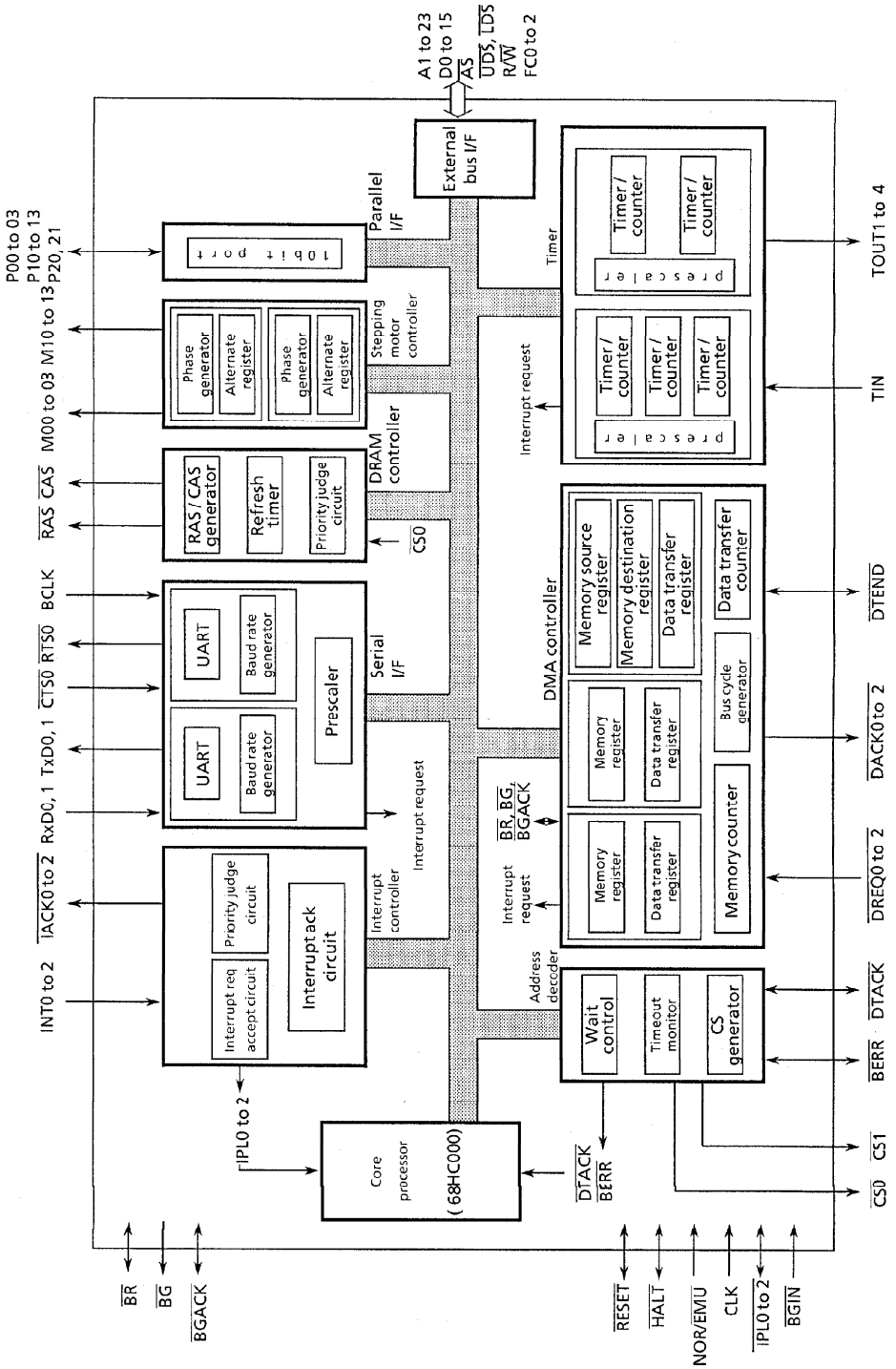


Figure 1.1 TMP68303 Block Diagram

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" is used to show that signals are active or true, irrespective of whether the signal is electrically high or low. "Negate" is used to show that signals are inactive or false.

2.1 Pin Assignments

Figure 2.1 shows the pin assignments.

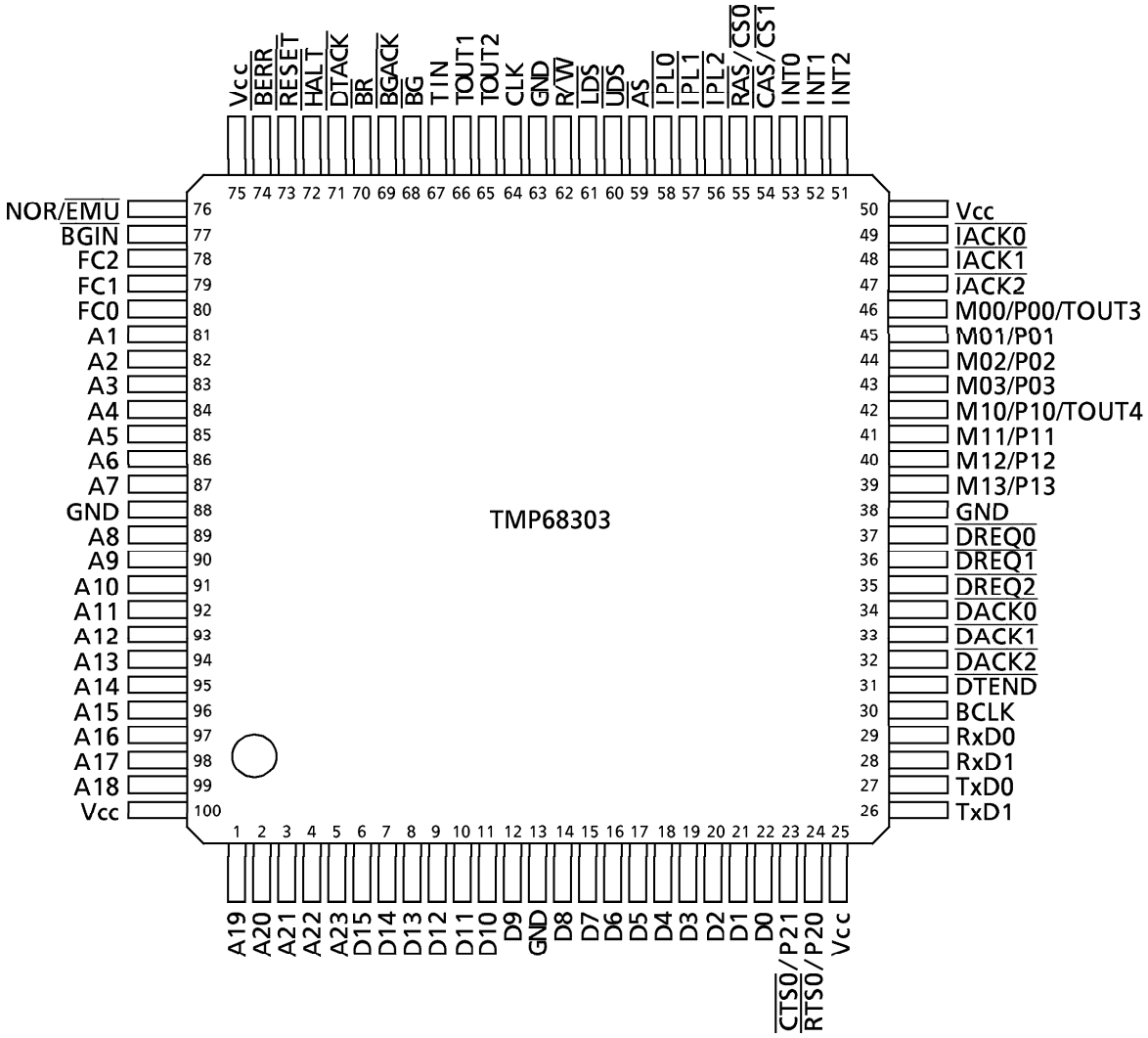


Figure 2.1 Pin Assignments (top view)

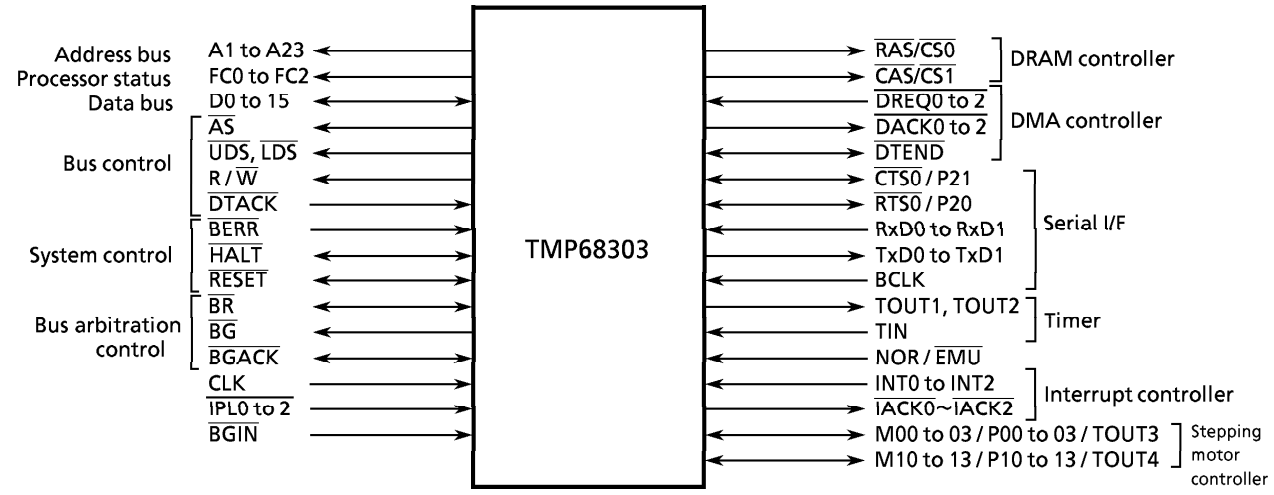
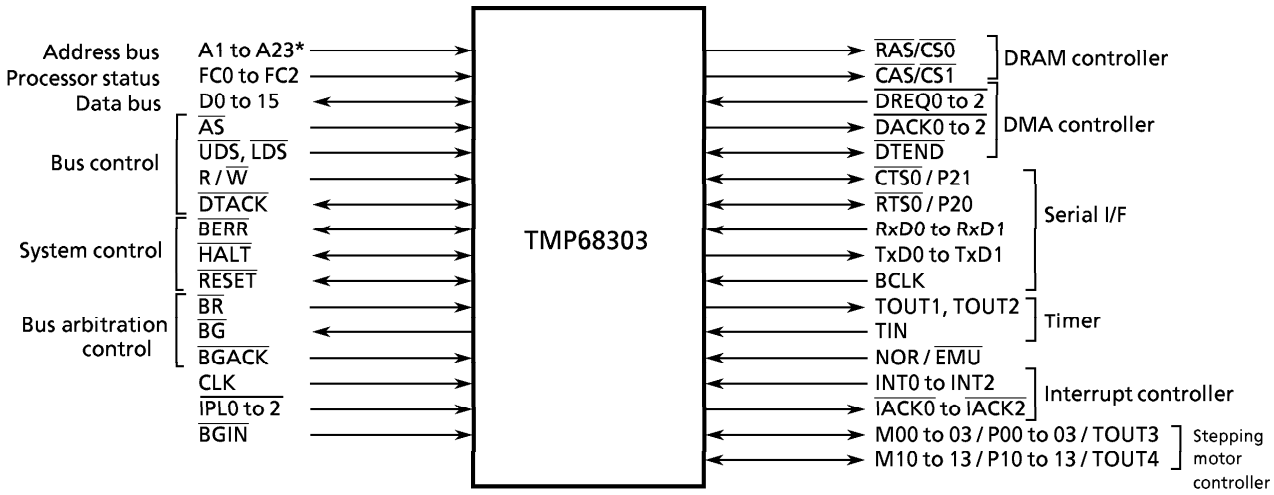


Figure 2.2 Normal Mode Input/Output Signals When Configured as Bus Master (including DMA controller active)



* Address (A1~A11) pins are input/output when the built-in DRAM controller starts.

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

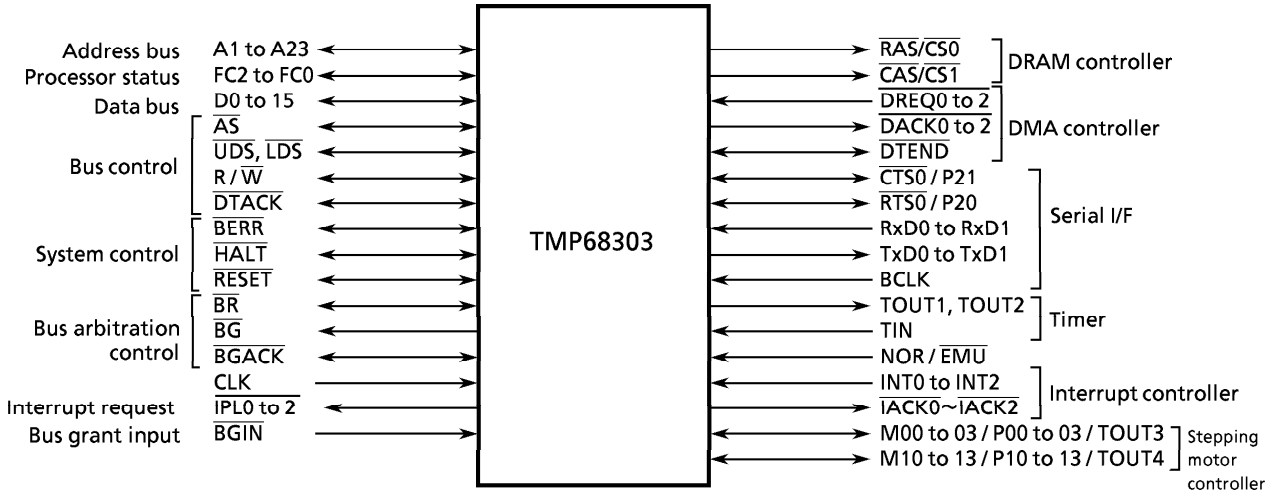


Figure 2.4 Emulation Mode Input/Output Signals

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 16 M 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 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.
CLK	I	I	Clock input pin.
INT0, INT1, INT2	I	I	External interrupt request input.
$\overline{IACK0}$, $\overline{IACK1}$ $\overline{IACK2}$	O	O	These signals indicate the interrupt acknowledge cycles corresponding to INT0, INT1, and INT2.
NOR / $\overline{EMU}$	I	I	Normal mode/emulation mode switch signal.
$\overline{IPL0}$, $\overline{IPL1}$ $\overline{IPL2}$	I	O	These signals are interrupt requests output from the internal interrupt controller in emulation mode only. In normal mode, the signal is input but does not have any function. Thus, fix them to low.
$\overline{BGIN}$	I	I	This signal is a CPU output $\overline{BG}$ signal to the internal DMA controller in emulation mode. 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
	NOR	EMU	
P00 / M00 / TOUT3 P01 / M01 P02 / M02	I/O	I/O	General-purpose I/O port or stepping motor control port 0 output signals. P00 is also used for timer 3 output.
P10 / M10/TOUT4 P11 / M11 P12 / M12	I/O	I/O	General-purpose I/O port or stepping motor control port 1 output signals. P10 is also used for timer 4 output.
$\overline{RAS}$ / $\overline{CS0}$	O	O	Raw address strobe signal to the DRAM or signal used to access the memory area specified by the address decoder.
$\overline{CAS}$ / $\overline{CS1}$	O	O	Column address strobe signal to the DRAM or signal used to access the memory area specified by the address decoder.
$\overline{DREQ0}$, $\overline{DREQ1}$, $\overline{DREQ2}$	I	I	DMA request signals to the DMA controller
$\overline{DACK0}$, $\overline{DACK1}$, $\overline{DACK2}$	O	O	Acknowledge signals to peripheral devices from the DMA controller
$\overline{DTEND}$	I/O.D	I/O.D	Disable signal used to ignore DMA requests, end signal used to suspend DMA operation, or output signal used to indicate transfer complete.
TOUT1, TOUT2	O	O	Timers 1 and 2 output signals
TIN	I	I	Signal input to timer channel 0, 1 and 2
P21 / $\overline{CTS0}$	I/O	I/O	General-purpose I/O port or CTS signal for serial interface channel 0
P20 / $\overline{RTS0}$	I/O	I/O	General-purpose I/O port or RTS signal for serial interface channel 0
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.
Vcc	–	–	Power input pin (+ 5 V)
GND	–	–	GND pin (0 V)

Table 2.1 Data Bus Control By Data Strobe

$\overline{UDS}$	$\overline{LDS}$	R/$\overline{W}$	D8~D15	D0~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 all 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

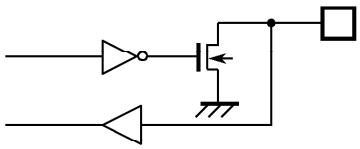
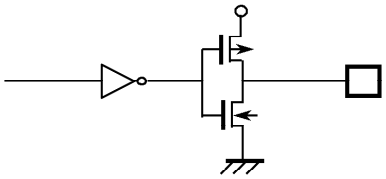
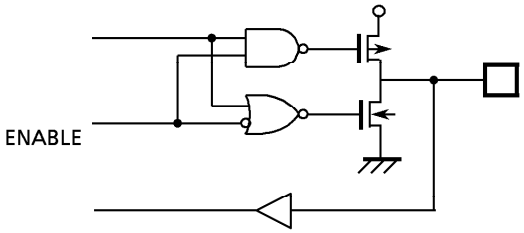
Table 2.3 Signal Summary

Signal name	Mnemonic	Input/output	Active state	Tri-state output	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.	—	—	—
Interrupt priority	$\overline{IPL0}$, $\overline{IPL1}$, $\overline{IPL2}$	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
Clock	CLK	In	H	—	—	—	—
Power input	V_{CC}	In	—	—	—	—	—
Ground	GND	In	—	—	—	—	—
Chip select	$\overline{CS0}$, $\overline{CS1}$	Out	L	N	—	—	H
Raw address strobe	$\overline{RAS}$	Out	L	N	—	—	H
Column address strobe	$\overline{CAS}$	Out	L	N	—	—	H
DMA request	$\overline{DREQ0 \text{ to } 2}$	In	L	—	—	—	—
DMA acknowledge	$\overline{DACK0 \text{ to } 2}$	Out	L	N	—	—	H
DMA transfer end/suspend	$\overline{DTEND}$	In/out	L	O.D.	—	—	—
Clear to send	$\overline{CTS0}$	In/out	L	—	—	—	—
Request to send	$\overline{RTS0}$	In/out	L	—	—	—	—
Receive data	RxD0 to 1	In	H	—	—	—	—
Transmit data	TxD0 to 1	Out	H	N	—	—	H
Baud rate clock	BCLK	In	H	—	—	—	—
I/O port	P00 to 03,10 to 13,20,21	In/out	H	—	—	—	—
Timer output	Tout 1~4	Out	H	N	—	—	L
Timer input	TIN	In	L	—	—	—	—
Mode switch	NOR / $\overline{EMU}$	In	—	—	—	—	—
Interrupt request	INT0~2	In	H	—	—	—	—
Interrupt acknowledge	$\overline{IACK0 \sim 2}$	Out	L	N	—	—	H
Stepping motor output	M00~03,10~13	Out	H	—	—	—	L
Bus grant input	$\overline{BGIN}$	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
$\overline{\text{RESET}}$, $\overline{\text{HALT}}$, $\overline{\text{DTACK}}$, $\overline{\text{BERR}}$ $\overline{\text{BR}}$, $\overline{\text{BGACK}}$ $\overline{\text{DTEND}}$		Sink open drain output
$\overline{\text{BG}}$ TOUT1, TOUT2 TxD0, TxD1 $\overline{\text{IACK0}}$, $\overline{\text{IACK1}}$, $\overline{\text{IACK2}}$ $\overline{\text{DACK0}}$, $\overline{\text{DACK1}}$, $\overline{\text{DACK2}}$ $\overline{\text{RAS}}$ / $\overline{\text{CS0}}$, $\overline{\text{CAS}}$ / $\overline{\text{CS1}}$		Push/pull output
CLK TIN RxD0, RxD1 BCLK INT0, INT1, INT2 NOR / $\overline{\text{EMU}}$		
A1~A23, D0~D15 FC0~FC2, $\overline{\text{AS}}$, $\overline{\text{UDS}}$, $\overline{\text{LDS}}$ R/$\overline{\text{W}}$ $\overline{\text{IPL0}}$, $\overline{\text{IPL1}}$, $\overline{\text{IPL2}}$ $\overline{\text{BGIN}}$ P00~P03, P10~P13 P20/$\overline{\text{RTS0}}$ P21/$\overline{\text{CTS0}}$		Tri-state output

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. A $\overline{DTACK}$ signal can be internally generated to allow the insertion of between 0 and 7 wait cycles in each memory area.

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 ($\overline{CSn}$), inserts the pre-set wait cycle, and outputs the internal $\overline{DTACK}$ signal, which is generated automatically, to the core processor. Depending on circumstances, it may be possible to control the wait cycle inserted using the externally generated $\overline{DTACK}$ signal.

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 (area selected by $\overline{CS0}$)	|
Memory area 1 (area selected by $\overline{CS1}$)	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 $\overline{CSn}$ signal is asserted.)
This prevents inadvertently accessing areas during the IACK cycle.

When the vector number is input from an external source, the address decoder can set the number of wait cycles for a IACK cycle independently of other areas.

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

(Note) The $\overline{CSn}$ signal and the DRAM control signal ($\overline{RAS}/\overline{CAS}$) are Pin multiplexed. Even if the DRAM controller accesses the specified memory area when the DRAM controller is enabled, the $\overline{CSn}$ signal is not output; nor does the wait function operate. The $\overline{CS0}$ area is valid for the DRAM controller.
After reset is released, the $\overline{CSn}$ signal output is disabled.

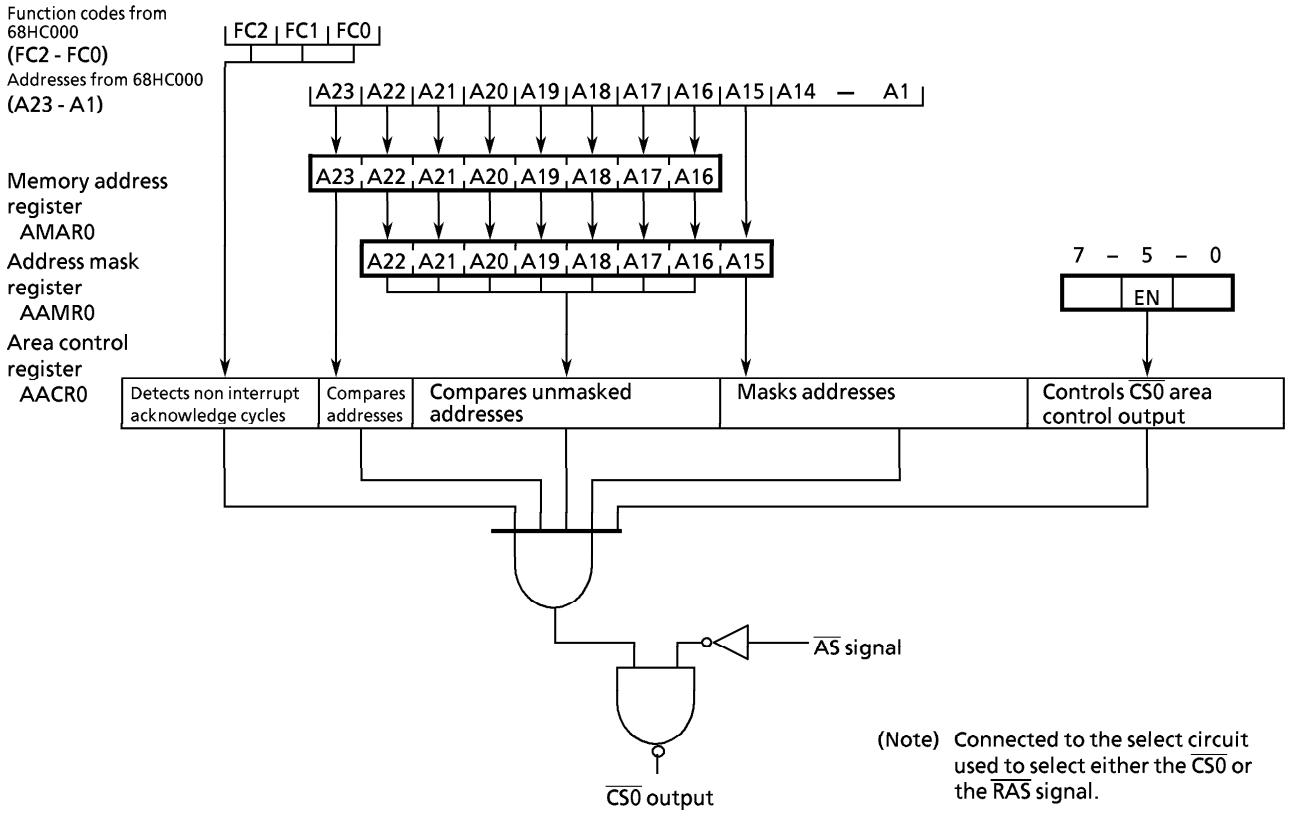


Figure 3.1 Chip Select Signal $\overline{CS0}$ Generation

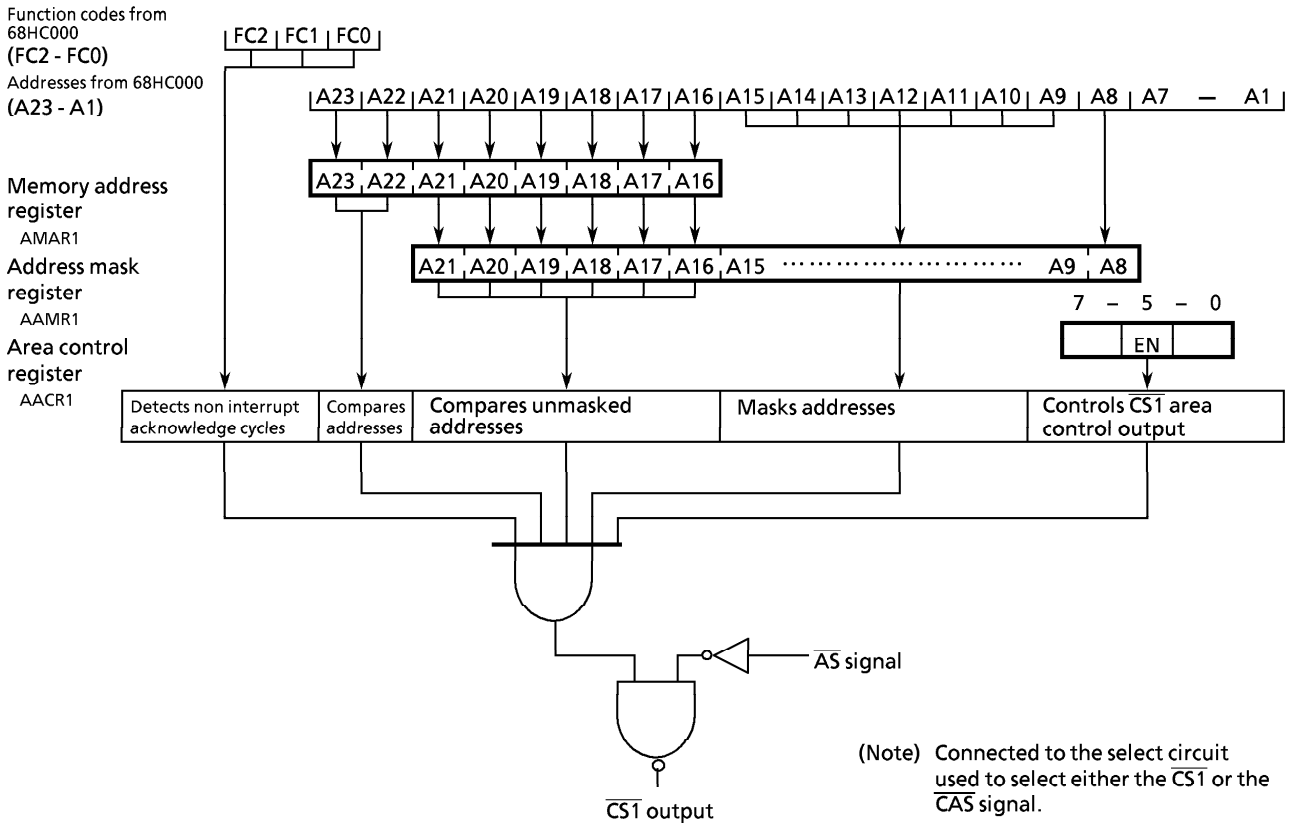


Figure 3.2 Chip Select Signal $\overline{CS1}$ Generation

Figure 3.3 shows the output status of area selection signals for overlapping areas.

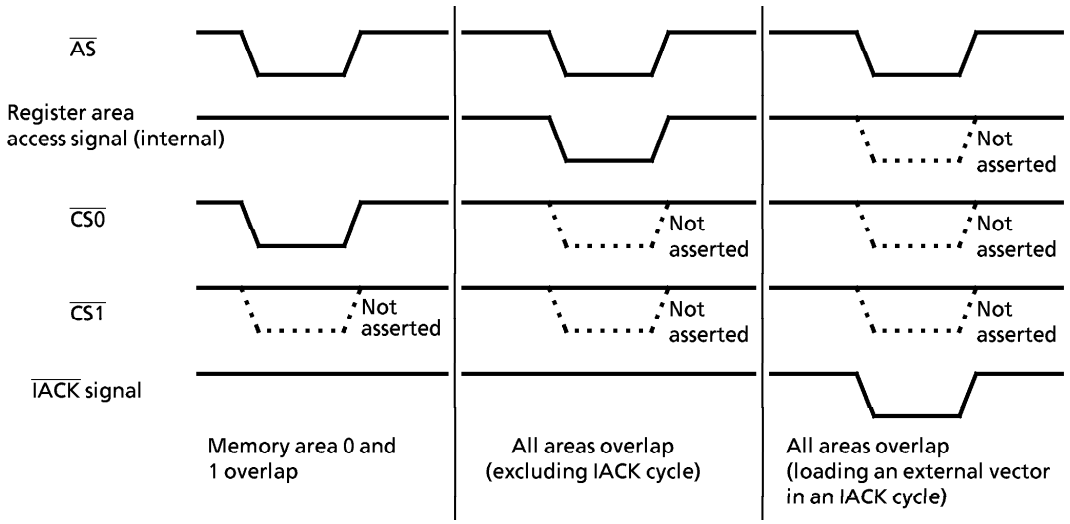


Figure 3.3 Output Status of Area Selection Signals

Figure 3.4 shows the relationship between the $\overline{CSn}$ signal and the $\overline{AS}$ signal.

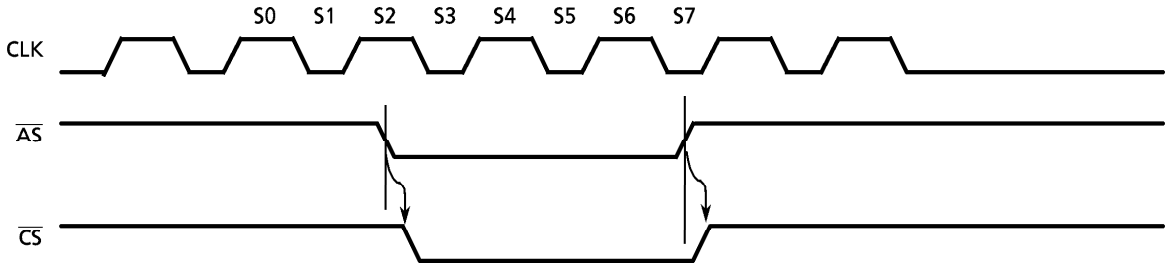


Figure 3.4 $\overline{CS}$ Signal and $\overline{AS}$ Signal Relationship

3.2.1 Memory Area

The memory area register (AMAR) specifies the start address in the memory area. Two areas can be specified in the 16M-byte address space. 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 address mask register determines the size of the memory area. Figure 3.2 shows the relationship between the size of the area and the value in the address mask register for contiguous address space.

Table 3.2 Relationship Between Memory Area Size and Address Mask Register

Memory Size / Mask Register	256	512	32K	64K	128K	256K	512K	1M	2M	4M	8M
For $\overline{CS0}$ (AAMR0)	—	—	00	01	03	07	0F	1F	3F	7F	FF
For $\overline{CS1}$ (AAMR1)	00	01	—	03	07	0F	1F	3F	7F	FF	—

The numeric values in the Table are hexadecimal

The address mask register addresses for masking are determined in units of bits. Therefore, depending on the setting, non-contiguous memory areas can be specified. For example, setting the memory address register to $00 and the address mask register to $5F for $\overline{CS1}$ specifies two memory areas, $00000 - $07FFFF and $100000 - $17FFFF.

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 3.5 shows actual address generation methods for the registers. The register area after reset release is allocated to $FFFC00 - $FFFFF (last part of address space). (Figure 3.6.)

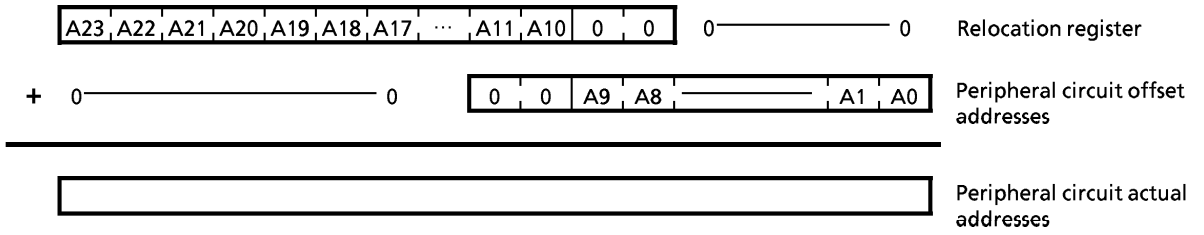


Figure 3.5 Real Register Address Generation

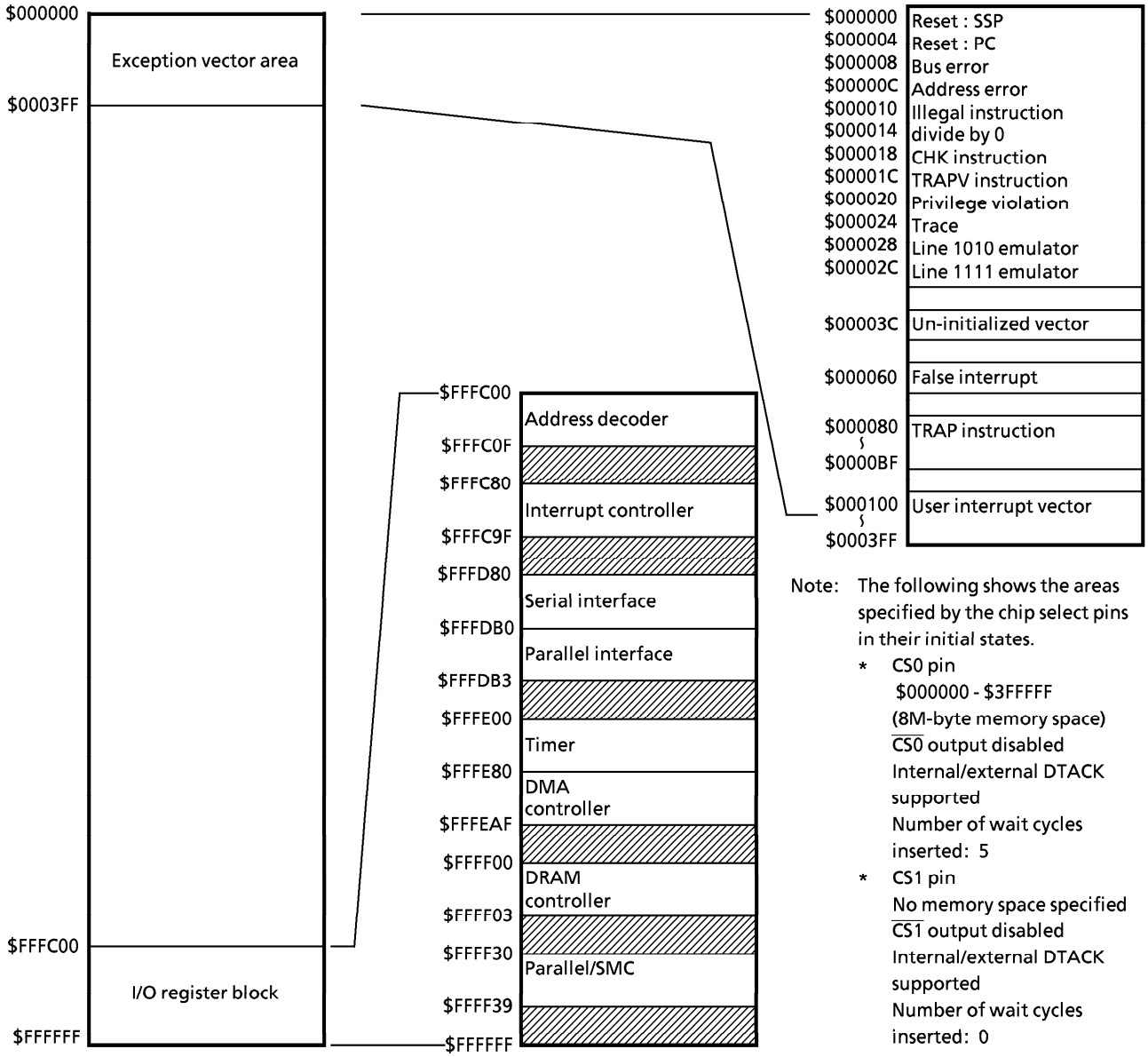
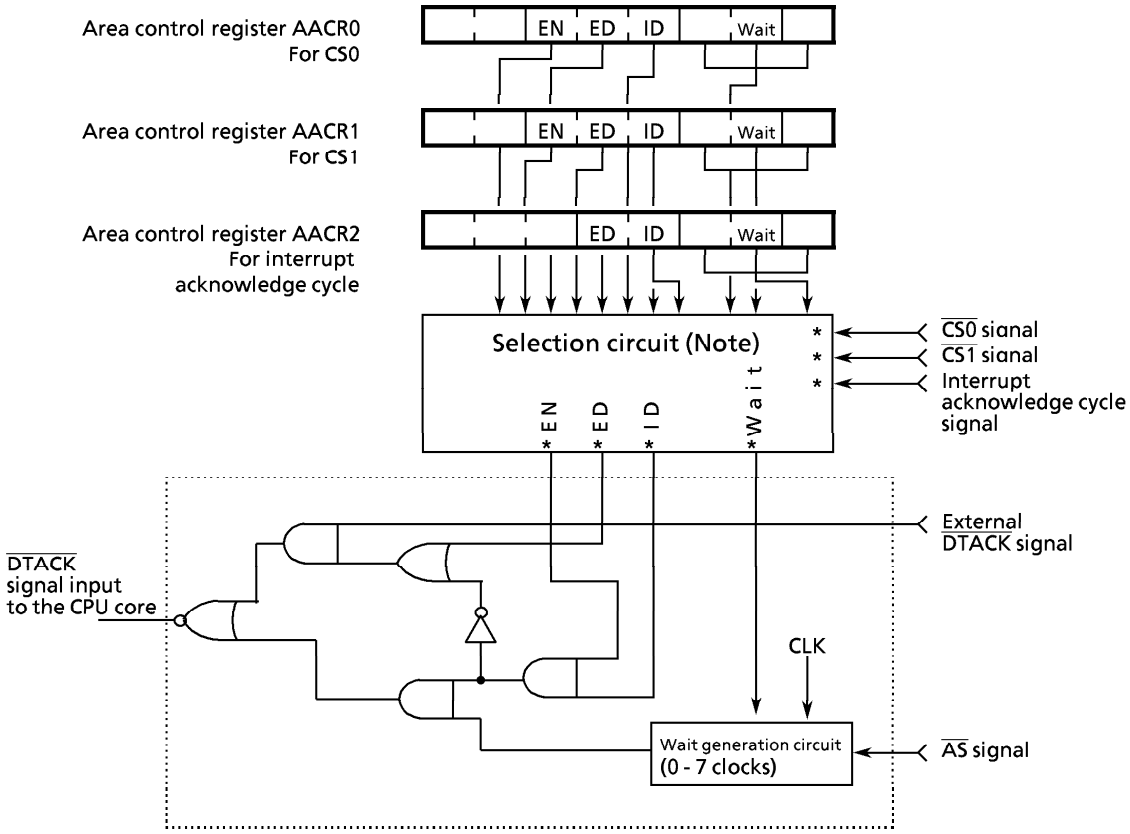


Figure 3.6 Memory Map After Reset Release

3.3 Automatic $\overline{\text{DTACK}}$ Generation

The address decoder can generate a $\overline{\text{DTACK}}$ signal for each memory area. The register area automatically generates $\overline{\text{DTACK}}$ to operate the bus with a zero wait. The register area can select whether $\overline{\text{DTACK}}$ signal generated in an external circuit or that generated in the address decoder is valid, or whether both are valid for the memory area. When both are valid, the first signal asserted is the $\overline{\text{DTACK}}$ signal output to the core processor. The address decoder can control the $\overline{\text{DTACK}}$ signal output to the core processor by inserting 0 to 7 wait cycles. To control the $\overline{\text{DTACK}}$ signal, the address decoder can also insert wait cycles in the IACK cycle (which loads external exception vectors) the same as for the memory area. Figure 3.7 illustrates the $\overline{\text{DTACK}}$ signal generation circuit.

The internal peripheral circuit registers of TMP68303 automatically generate the $\overline{\text{DTACK}}$ signal to enable access with no wait. However, note that to access registers related to the 8-bit timers, port 0 and 1, or stepping motor controller, 0 to 3 waits are inserted depending on the register.



Note: During assertion of signals marked with an asterisk (*), the selection circuit inputs the set register values to the circuit in the dotted-line box.
During the interrupt acknowledge cycle, *EN is 1.
When internal peripheral circuit registers are accessed, *EN and *ID are set to 1 and *wait is set to 0. (Excluding the registers related to the 8-bit timers, port 0 and 1, and stepping motor controller.)

Figure 3.7 $\overline{\text{DTACK}}$ Signal Generation Circuit

For DMA single-address transfer between memory and I/O devices, even if only internal $\overline{\text{DTACK}}$ (ED = 0, ID = 1) is selected, $\overline{\text{DTACK}}$ must be input externally, because internal and external $\overline{\text{DTACK}}$ are compared and the slower one is effective.

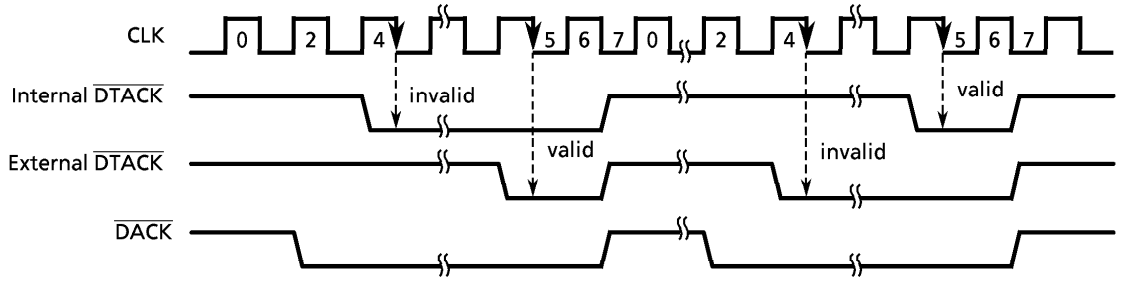


Figure 3.8 $\overline{\text{DTACK}}$ for DMA Single Address Transfer Mode

3.4 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 3.9 illustrates the $\overline{\text{BERR}}$ signal generation circuit. Figure 3.10 shows the $\overline{\text{BERR}}$ signal generation timing.

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

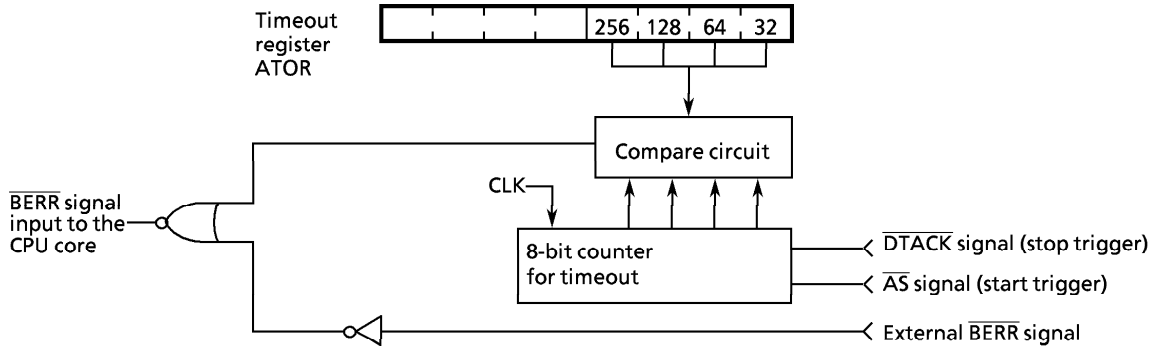


Figure 3.9 $\overline{\text{BERR}}$ Signal Generation Circuit

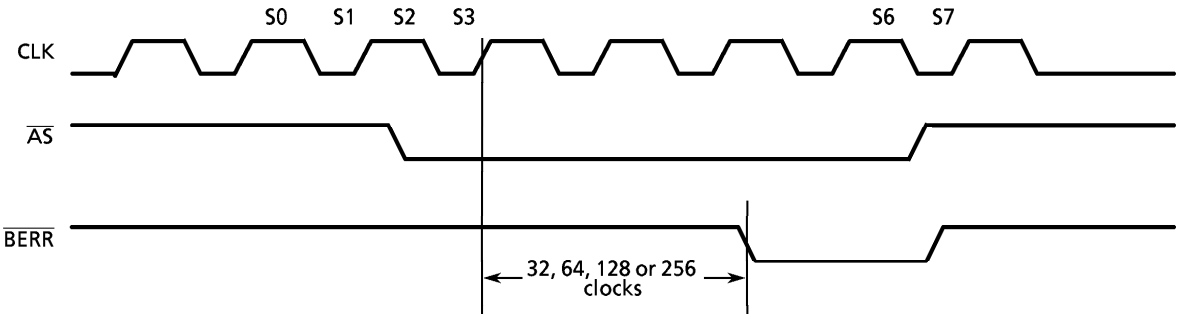


Figure 3.10 $\overline{\text{BERR}}$ Signal

3.5 Register Configuration

3.5.1 Memory Address Registers (AMAR0, 1)

These registers specify the start address of the memory area. The start address of the memory area can be set only within the boundary of the area determined by the address mask register. For example, if the size of the area is 4M bytes for $\overline{CS1}$, only bits A23 and A22 of the memory address register are compared with the address, and four start addresses are available: $000000, $400000, $800000, and $C00000. (If the area is 256 or 512 bytes, the start address has a 64K-byte boundary because of the relationship with masked addresses.)

After reset, AMAR0 (for $\overline{CS0}$) is $00, and AMAR1 (for $\overline{CS1}$) is undefined.

Offset address	15	14	13	12	11	10	9	8	
$000	A23	A22	A21	A20	A19	A18	A17	A16	AMAR0 (for $\overline{CS0}$)
Reset	0	0	0	0	0	0	0	0	

	15	14	13	12	11	10	9	8	
$004	A23	A22	A21	A20	A19	A18	A17	A16	AMAR1 (for CS1)
	–	–	–	–	–	–	–	–	

3.5.2 Address Mask Registers (AAMR0, 1)

These registers set the size of the memory area by masking the address output from the core processor. Addresses A21 to A8 can be masked.

The following table lists address mask register bits and their corresponding addresses to be masked.

Mask register bit	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Address to be masked for $\overline{CS0}$	A22	A21	A20	A19	A18	A17	A16	A15
Address to be masked for $\overline{CS1}$	A21	A20	A19	A18	A17	A16	A15 - A9	A8

Setting the bits to 1 masks the corresponding addresses; to 0, does not mask.

After reset, AAMR0 (for $\overline{CS0}$) is $FF, and AAMR1 (for $\overline{CS1}$) is undefined.

Offset address	7	6	5	4	3	2	1	0	
$001	M22	M21	M20	M19	M18	M17	M16	M15	AAMR0 (for $\overline{CS0}$)
Reset	1	1	1	1	1	1	1	1	

	7	6	5	4	3	2	1	0	
$005	M21	M20	M19	M18	M17	M16	*1	M8	AAMR1 (for CS1)
	–	–	–	–	–	–	–	–	

*1 : M15 - M9

3.5.3 Area Control Registers (AACR0, 1, 2)

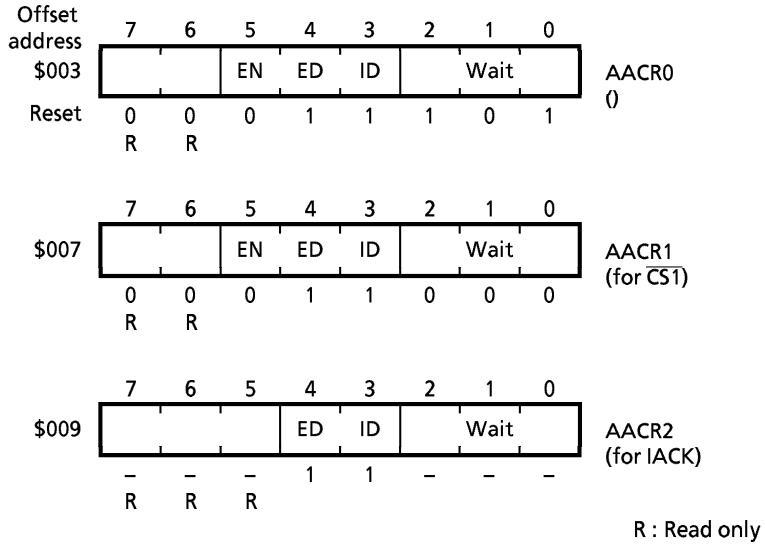
Registers AACR0 and AACR1 enable or disable memory areas in response to $\overline{CS0}$ and $\overline{CS1}$ signals, specify $\overline{DTACK}$ signal generation mode, and the number of waits to insert.

When an area is disabled, the $\overline{CS}$ signal is not asserted although the specified memory area is accessed, and the address decoder does not generate a $\overline{DTACK}$ signal. Register AACR2 specifies $\overline{DTACK}$ generation mode in the interrupt acknowledge cycle, and the number of waits to insert.

The setting in register AACR2 determines the $\overline{DTACK}$ generation mode in the IACK cycle when no vector number is automatically generated in an external interrupt. When a vector number is automatically generated due to an interrupt from an internal device or an external interrupt, a zero wait $\overline{DTACK}$ is generated regardless of the setting of register AACR2.

After a hardware reset, AACR0 is set to $1D (area disable, both external and internal $\overline{DTACK}$ enable, up to five waits) and AACR1 is set to $18 (area disable, both external and internal $\overline{DTACK}$ enable, no waits).

AACR2 is set to $18 (both external and internal $\overline{DTACK}$ enable, no waits).



EN : Area control

0 : Area disable ($\overline{CSn}$ signal not output, no internal $\overline{DTACK}$ generated)

1 : Area enable ($\overline{CSn}$ signal output, internal $\overline{DTACK}$ generated)

ED, ID : $\overline{DTACK}$ selection

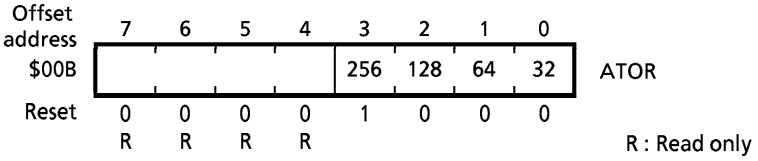
ED	ID	$\overline{DTACK}$ Used
0	0	Use prohibited (even when set, ED = 0, ID = 1)
0	1	Use internal $\overline{DTACK}$ (even when $\overline{DTACK}$ pin asserted, external $\overline{DTACK}$ ignored)
1	0	Use external $\overline{DTACK}$ (no $\overline{DTACK}$ generated by address decoder)
1	1	Use both internal and external $\overline{DTACK}$ (the first asserted is valid)

Wait : Number of waits to be inserted

When using internal $\overline{DTACK}$, set the number of waits to be inserted. Settings are from 0 to 7 clocks.

3.5.4 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).

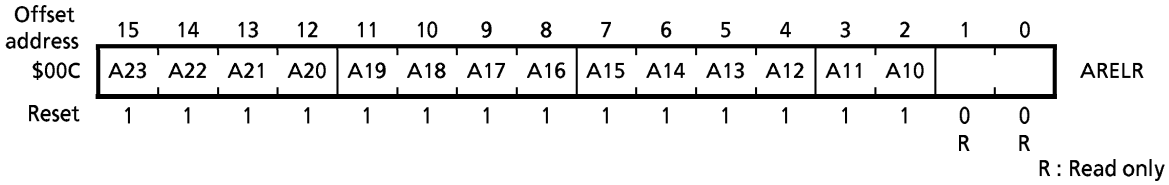


Set value				Time until $\overline{BERR}$ generation
256	128	64	32	
0	0	0	0	No $\overline{BERR}$ generation
0	0	0	1	Generated after 32 clocks
0	0	1	X	Generated after 64 clocks
0	1	X	X	Generated after 128 clocks
1	X	X	X	Generated after 256 clocks

X : Any value

3.5.5 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.



3.6 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.

3.7 Cautions

1. While area control register 2 is for the interrupt acknowledge cycle, the settings here are valid only for reading external vector numbers by external interrupts.
2. When both the internal and external DTACK (ED=1, ID=1) are selected in the area control register, accessing an area under the conditions specified below prevents normal execution of the next cycle. This occurs with either CS0 or CS1. For example, if the conditions are satisfied in a CS1 cycle followed by a CS0 cycle, CS0 cycle becomes an abnormal cycle. In the reverse case, even if the CS0 or CS1 access is repeated and the conditions are satisfied, then the next cycle becomes an abnormal cycle.

Internal wait set value	Actual number of waits							
	0	1	2	3	4	5	6	7
0	○	–	–	–	–	–	–	–
1	○	○	–	–	–	–	–	–
2	×a	○	○	–	–	–	–	–
3	○	×	○	○	–	–	–	–
4	○	○	×	○	○	–	–	–
5	○	○	○	×	○	○	–	–
6	○	○	○	○	×	○	○	–
7	○	○	○	○	○	×	○	○

○ : If the bus cycle ends at this condition, the next bus cycle receives the set number of waits and operates normally (no problem).

× : If the bus cycle ends at this condition, the next bus cycle generates the following:

- When the set number of waits for the next bus cycle is one or two and the cycle is the read cycle, this read cycle has no-wait access.
  If the read cycle is repeated, all subsequent cycles have no-wait access.
  (Because the Xa status shown in the table above continues.)
- When the set number of waits for the next bus cycle is other than two and the cycle is a read cycle, this cycle has no-wait access.
  If the cycle is repeated, only the first cycle has no-wait access.
  (Because when the actual number of waits is 0, the status is O.)
- When the next bus cycle is an internal peripheral circuit register read or write cycle, ends normally.

For example, if the wait setting for CS0 is 2, the wait setting for CS1 is 4, the actual cycle for CS1 ends at wait and the immediate next cycle is CS0, then there is no wait. If, at this time, CS0 uses only the internal DTACK (ED = 0, ID = 1), there is still no wait. If an interrupt vector is written from an external source, the number of waits can be set in the interrupt acknowledge cycle. However, if prior to the acknowledge cycle, the bus cycle ends in an X state as shown in the table above, there is no wait.

If the above conditions are satisfied, the next cycle may not operate normally. Therefore, do not make settings which result in combinations marked with "x" in the table.

3. $\overline{CS0}$ / $\overline{CS1}$ output is disabled until area enable. Therefore, the $\overline{CS0}$ / $\overline{CS1}$ pin cannot be used for the program ROM select signal after reset.

4. Interrupter Controller

4.1 Overview

The interrupt controller provides 10 interrupt channels. Seven of the channels are for internal peripheral circuits, while the other three are for external interrupts from interrupt request input pins INT0, 1, and 2. Interrupt request levels input to the core processor (the pattern of "0"s and "1"s input to the core processor IPL0, 1, and 2) can be set for each channel. Priorities can be set independently. Interrupt request input mode (input level, rising / falling edge) for external interrupts can also be set independently. 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, or 2) corresponding to an external interrupt channel is asserted in accordance with the interrupt acknowledge cycle.

The auto-vectored interrupt function supported by the TMP68HC000 is not available because the TMP68301AK does not have 68000 interface signals.

Figure 4.1 shows a block diagram of the interrupt controller.

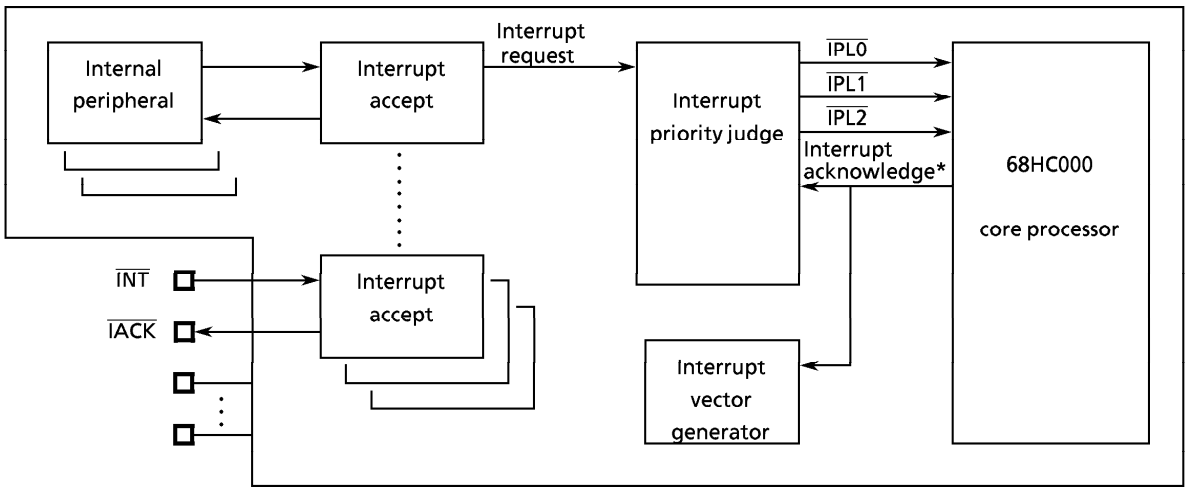


Figure 4.1 Interrupt Controller Block Diagram

4.2 Interrupt Request

When an interrupt request is present on 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 are generated on 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, or 2).

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

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

Note : When switching from level mode to edge mode, the 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
3. Clear the pending bits
4. Release the interrupt request mask

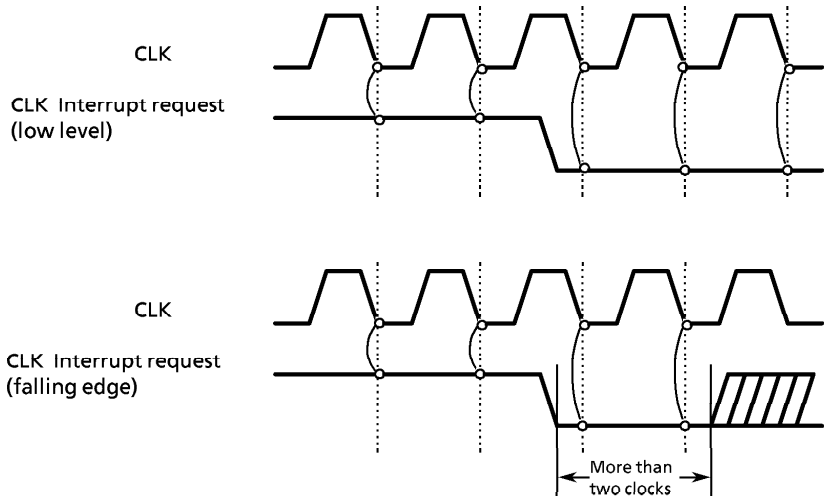


Figure 4.2 External Interrupt Request

4.3 Priority Between Channels

The interrupt controller can set the 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
	Timer	Channel 0
	Serial interface	Channel 0
	External interrupt request	Channel 1
	Timer	Channel 1
	Serial interface	Channel 1
	Serial interface	Channel 2
	Timer	Channel 2
Low	External interrupt request	Channel 2

Table 4.1 Priority of Channels Set to 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 performed. The interrupt request level is released on addresses A1 to A3 is output and the corresponding interrupt vector number is read from data bus D0 to D7.

The 68301A interrupt controller has a function to automatically generate the vector number to be read by the core processor during the $\overline{\text{IACK}}$ 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 IACKn signal corresponding to the interrupt and obtain the vector externally.

If more than one channel issues requests at the same level, an interrupt acknowledge signal is asserted for the channel with the highest priority in Table 4.1. The interrupt acknowledge signal asserted here only applies internally to the 68301A and is not output externally. However, depending on the register setting, external interrupts can be output externally as $\overline{\text{IACK}}$ signals. $\overline{\text{IACK}}$ signals are asserted using the same timing as that for $\overline{\text{AS}}$ signals. Area control register 2 in the address decoder can be set to insert a WAIT into IACK cycles. For details, see the address decoder section.

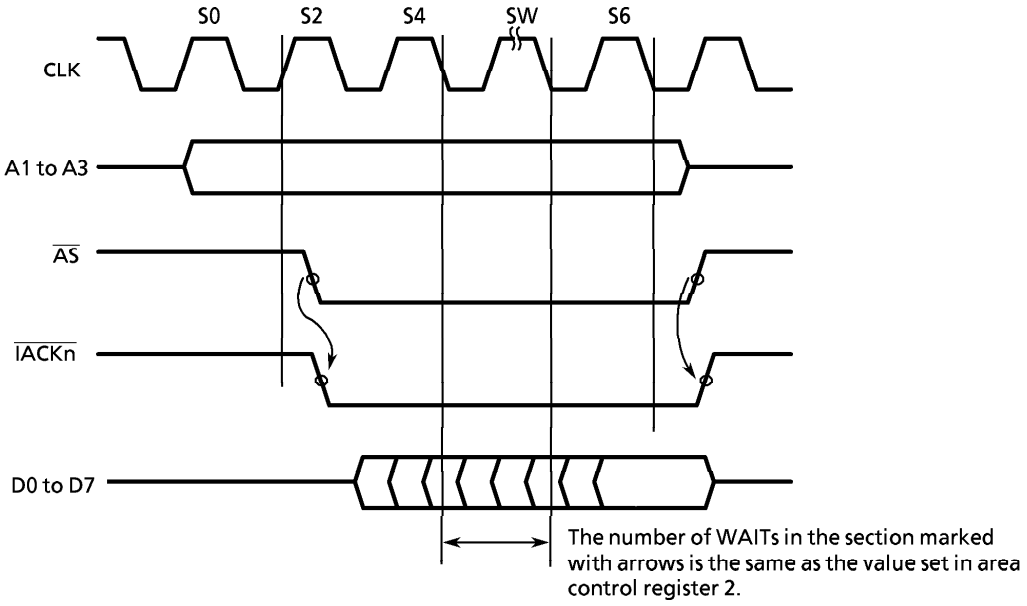


Figure 4.3 IACK Signals for External Vector Fetch by External Interrupt

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 five lower bits of the vector are determined depending on the interrupt channel or request. The three upper bits are set using the interrupt vector number register (IVNR). See Table 4.2 for a list of vector numbers.

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

Serial interface and DMA controller interrupt requests generate vector numbers not only corresponding to the channel generating the interrupt, but also in accordance with the cause of the interrupt.

Channel	Cause	Vector Number
External interrupt Channel 0		XXX00000
External interrupt Channel 1		XXX00001
External interrupt Channel 2		XXX00010
Timer 0 Channel 0		XXX00100
Timer 1 Channel 1		XXX00101
Timer 2 Channel 2		XXX00110
Serial interface Channel 0	Receive error, break detection	XXX01000
	Receive complete	XXX01001
	Transmit ready	XXX01010
	Interrupt source cleared while interrupt pending (Note 1)	XXX01011
Serial interface Channel 1	Receive error, break detection	XXX01100
	Receive complete	XXX01101
	Transmit ready	XXX01110
	Interrupt source cleared while interrupt pending (Note 1)	XXX01111
Serial interface Channel 2	Receive error, break detection	XXX10000
	Receive complete	XXX10001
	Transmit ready	XXX10010
	Interrupt source cleared while interrupt pending (Note1)	XXX10011
Other	Default vector (Note 2)	XXX11111

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

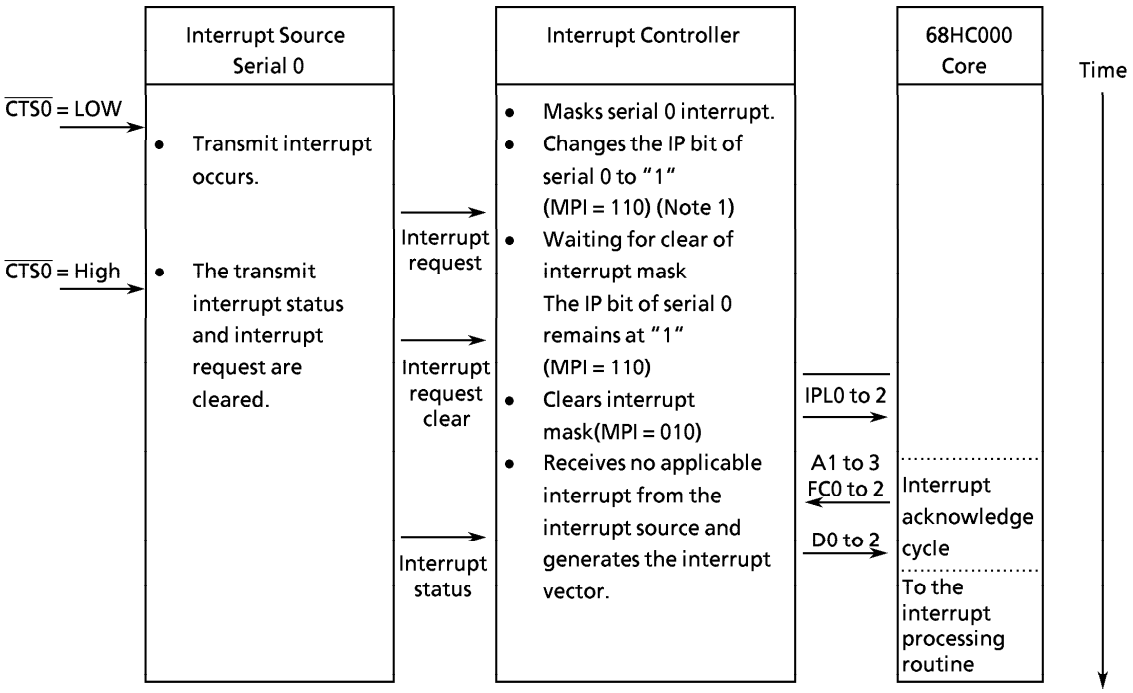
Table 4.2 Vector Number List

Note1: This vector number is generated when the cause of the interrupt becomes unknown if the interrupt source is cleared before the interrupt acknowledge is returned for a pending interrupt.

Note2: If the CPU accepts an interrupt, the IACK cycle starts after 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 by the IACK cycle.

4.5.1 Interrupt Source Cleared While Interrupt Pending

This interrupt is generated under the following conditions. The following description is based on the generation of a serial 0 interrupt.



When an interrupt is generated by the interrupt source, an interrupt request is passed 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 controller waits for the interrupt request to be issued to the 68HC000 core. (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. (The above example shows 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 control is still pending, the interrupt controller sends an interrupt request to the 68HC000 core when the interrupt mask is cleared. 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:

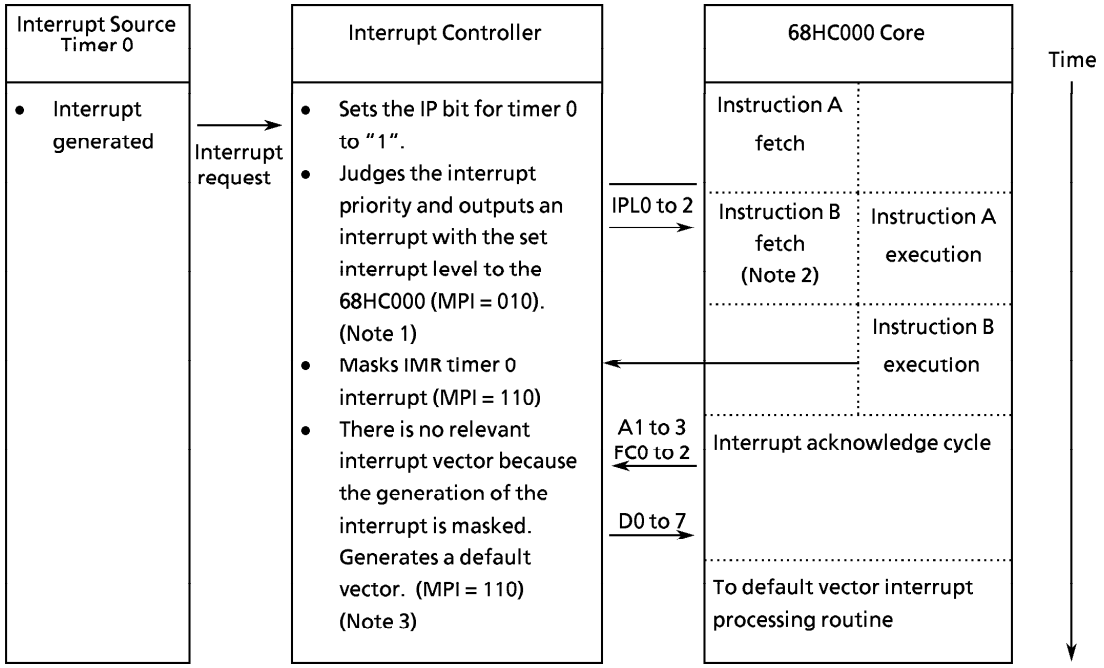
(a) The interrupt is masked by the serial mode register (SMRn) while it is pending.
(b) The $\overline{CTS0}$ input changes to high while a transmit ready interrupt is pending. (TxRDY is set to "0" when $\overline{CTS0}$ changes to high, clearing the transmit interrupt cause.)
(c) In the serial command 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 the error interrupt processing routine.

(If both ERINTM and RxINTM of the serial mode register [SMRn] are "0" and both interrupt masks are cleared, both error interrupt and receive interrupt are generated when an error occurs. Since 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, check your software and make necessary modification so that these interrupts will not occur.

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, an interrupt request is passed to the interrupt controller. As a result, the interrupt controller sets the relevant IP bit to "1", checks that the interrupt is the highest priority interrupt, and outputs an 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 = "1" (M=1), 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 $\overline{IPL0}$ to $\overline{2}$ interrupts, it attempts to jump to the interrupt sequence operations. However the interrupt sequence is delayed until instruction B fetched by the 68HC000 core is executed. If instruction B masks the generated interrupt (Note 2), the instruction is executed and the interrupt request by the interrupt controller is cancelled. Subsequently, even if the 68HC000 core starts the interrupt acknowledge cycle, the interrupt controller does not have a corresponding interrupt and so generates a default vector indicating that there is no relevant interrupt and ends the interrupt acknowledge cycle.

Even if a default vector interrupt is generated, the interrupt generated remains in the interrupt controller in the pending state (Note 3). Accordingly, after clearing the interrupt mask, a normal interrupt sequence can be performed.

If you simply wish to return to the main processing and disable vector generation when this default interrupt occurs, insert an instruction before the interrupt mask instruction (instruction B), to set the 68HC000 core interrupt level to the highest level. 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 level of the interrupt generated is 7, the 68HC000 core cannot disable the interrupt request and the default vector is generated. In this case, perform default vector processing.

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 these bits are described below. The set (setting the bit to "1") and reset (setting the bit to "0") methods vary according to a bit.

Mask Bit (M)	Control Bit for masking interrupt requests
1	Masks the interrupt request.
0	Unmasks the interrupt request.
set	Set by hardware reset or by writing "1" by software.
reset	Set to "0" by software.

Pending bit (P)	Indicates an interrupt request occurred and is pending (waiting for interrupt processing).
1	An interrupt request occurred and is pending.
0	No interrupt request occurred
set	An interrupt request occurred (this bit cannot be set to "1" by software).
reset	Reset by hardware reset or by writing "0" by software when the interrupt request is accepted by the core processor. (See Notes)

In-service bit (I)	Indicates that an interrupt request has been accepted by the core processor.
1	Indicates that an interrupt request has been accepted.
0	Indicates that no interrupt request has been accepted.
set	An interrupt request is accepted by the core processor (this bit cannot be set to "1" by software).
reset	Hardware reset Set to "0" by software

Note1 : The function to clear pending bits by software is used when initializing the whole system or when pending bits are set by unnecessary interrupts. However, if pending bits set according to interrupts generated by internal peripheral circuits are cleared to "0", interrupts will no longer occur. 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 operated 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, an interrupt request is present due to an interrupt source in a channel, the corresponding IP bit is set immediately after the INTM bit is cleared to "0". To avoid this, disable interrupt requests for the 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

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

Note2 : For level mode 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 for the bit to be cleared. Writing "1" to the pending bit will not affect operation but enables interrupts to be generated even though the pending bit is mistakenly cleared.

The interrupt states for the mask bit (M), pending bit (P), and in-service bit (I) values are shown in Table 4.4.

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	Another 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 generated while interrupt is being masked
1	1	1	Interrupt request generated after masking interrupt during an interrupt processing routine.

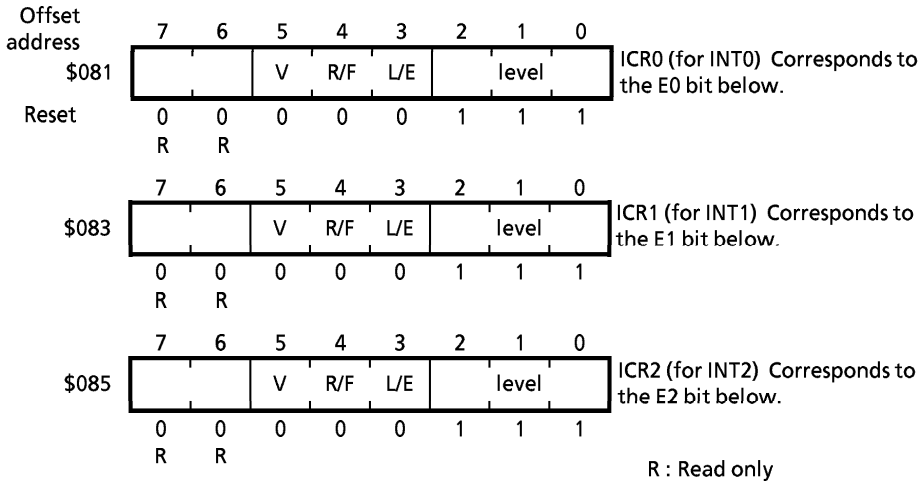
Table 4.3 Interrupt Status

4.7 Register Configuration

4.7.1 Interrupt Controller Registers 0, 1, and 2 (ICR0, 1, and 2)

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

After a hardware reset, ICR0, 1, and 2 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 automatic generation of vector number

1 : Vector number automatic generation

R/F, L/E : Request input mode for external interrupts

R/F	L/E	Interrupt Request Input Mode
0	0	Falling edge
1	0	Rising edge
0	1	Low level
1	1	High level

Level : Interrupt request leve

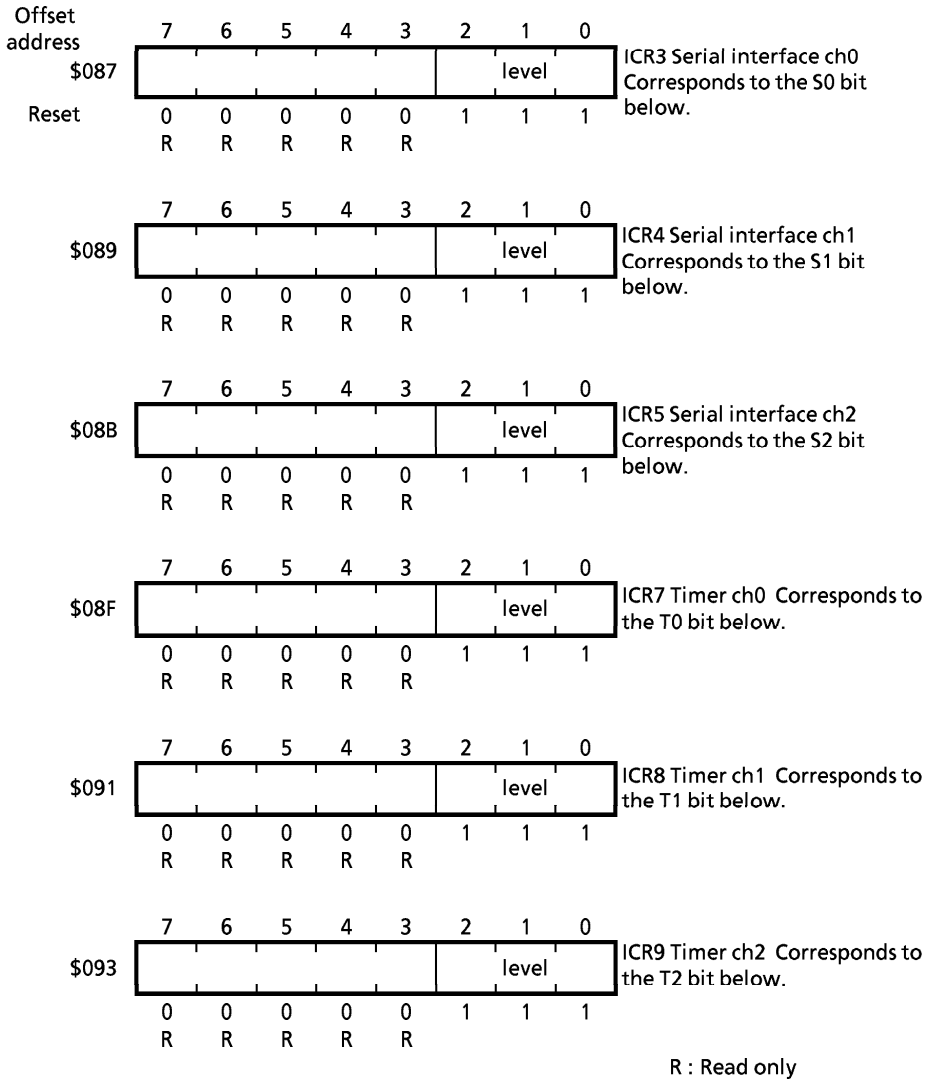
0 to 7 : Indicate the interrupt request level corresponding to $\overline{IPL0}$, $\overline{1}$, and $\overline{2}$ to be input to the core processor. For example, $\overline{IPL2} = \overline{1}$, $\overline{IPL1} = \overline{0}$, and $\overline{IPL0} = \overline{0}$ indicate request level 3. The following table shows the correspondence between IPLx and interrupt levels.

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 5, 7 to 9 (ICR3 to 5, 7 to 9)

These are the interrupt control registers for interrupts from internal peripheral circuits.

After a hardware reset, ICR3 to 5 and 7 to 9 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 to be input to the core processor.

4.7.3 Interrupt Mask Register (IMR)

This register sets the interrupt mask for a channel. A "1" masks the interrupt (ignore interrupt). A "0" unmasks the interrupt (allow interrupt). If masking an interrupt during operation, set the channels corresponding to the bits to be modified to a state whereby, even if an interrupt is generated, 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 $07F7 (all interrupt channels masked).

Offset address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$094						T2	T1	T0	P	S2	S1	S0		E2	E1	E0	IMR
Reset	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	
	R	R	R	R	R								R				

T2 : Timer channel 2
T1 : Timer channel 1
T0 : Timer channel 0
S2 : Serial interface channel 2
S1 : Serial interface channel 1
S0 : Serial interface channel 0
E2 : External interrupt channel 2
E1 : External interrupt channel 1
E0 : External interrupt channel 0
P : Parallel interface (only the case that IO12 is used for $\overline{INT6}$ (External interrupt))

R : Read only

Note : Bit 7 is normally set to "1".

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. A "1" indicates that there is an interrupt request that has not yet been accepted by the core processor. A "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 using software cancels the interrupt request. However, to enable subsequent interrupts, the interrupt source must be cleared.

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						T2	T1	T0	P	S2	S1	S0		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 : 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. A "1" indicates that a request has been accepted. A "0" indicates that a request has not 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 address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$098						T2	T1	T0	P	S2	S1	S0		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 : Read only

4.7.6 Interrupt Vector Number Register (IVNR)

This register specifies the three upper bits of a vector number. The five lower bits of a vector number are determined by the interrupt channel or the interrupt source.

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

Note : After a reset is released, the 68HC000 core interrupt vectors overlap the internal peripheral circuit interrupt vectors. Therefore, set the value of IVNR to $40 or more by software.

Offset address	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 : Three upper bits of the vector number

4.8 Interrupt Expansion Function

In the TMP68301A, interrupt channels assigned to unused peripheral circuits can be used as external interrupt inputs by setting the expansion interrupt register. Thus, a maximum of seven channels can be used for external interrupt inputs in addition to the three standard external interrupt channels.

The interrupt input pins are assigned as follows:

Peripheral circuit interrupt channel	Interrupt input pin	Interrupt input name
Serial interface ch0	RxD0	$\overline{INT3}$
Serial interface ch1	RxD1	$\overline{INT4}$
Serial interface ch2	RxD2	$\overline{INT5}$
	IO12	$\overline{INT6}$ (Note)
Timer ch0	TIN	$\overline{INT7}$
Timer ch1	IO14 / $\overline{DSR0}$	$\overline{INT8}$ (Note)
Timer ch2	IO15 / $\overline{DTR0}$	$\overline{INT9}$ (Note)

Note : When used as interrupt inputs, the pins must be first set to input in the parallel direction register.

Only falling-edge input mode is available for expanded external interrupts. However, like the standard external interrupt inputs, the state must be held for at least two clocks after the falling edge.

Multiple falling edges may occur before the processor accepts an interrupt request. The processor treats these edges as if they were the same interrupt request.

There is no IACK output signal corresponding to the expanded external interrupts. Therefore, the vector number is generated automatically by the interrupt controller because the vector number cannot be input from an external source during the interrupt acknowledge cycle. Vector numbers generated at this time are as follows.

Interrupt channel	Interrupt input name	Vector number
Serial interface ch0	$\overline{INT3}$	XXX010**
Serial interface ch1	$\overline{INT4}$	XXX011**
Serial interface ch2	$\overline{INT5}$	XXX100**
	$\overline{INT6}$	XXX101**
Timer ch0	$\overline{INT7}$	XXX00100
Timer ch1	$\overline{INT8}$	XXX00101
Timer ch2	$\overline{INT9}$	XXX00110

XXX : The three upper bits specified by the vector number register

** : The two lower bits of the interrupt vector number assigned to the serial interface depend on the status of the standard interrupt channels. Accordingly, when using these interrupts, set the same destination address in all four. The same applies to $\overline{INT6}$.

4.8.1 Expansion Interrupt Register (IEIR)

This register controls the switching of interrupt channels between internal peripheral circuits and external interrupt inputs. A "1" indicates external interrupt input. A "0" indicates interrupts from the internal peripheral circuits (external interrupt input disabled).

After a hardware reset, this register is initialized to $00 (all channels set to internal peripheral circuit interrupts).

Offset address	7	6	5	4	3	2	1	0	
$09D		T2	T1	T0	P	S2	S1	S0	IEIR
Reset	0	0	0	0	0	0	0	0	
	R								R : Read only

T2 : Timer ch2

T1 : Timer ch1

T0 : Timer ch0

P : (Note 2)

S2 : Serial interface ch2

S1 : Serial interface ch1

S0 : Serial interface ch0

Note1: Even if external interrupt input is enabled, the internal peripheral circuits can perform their original functions. However, the pins assigned as interrupt inputs (for example, the serial interface RxD) cannot be used for their original functions.

The peripheral circuits can function but cannot generate interrupts if the interrupt channels are used for interrupt inputs.

Note2: When using IO12 for $\overline{INT6}$, set this register to "1".

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 0 supports modem control signals.

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

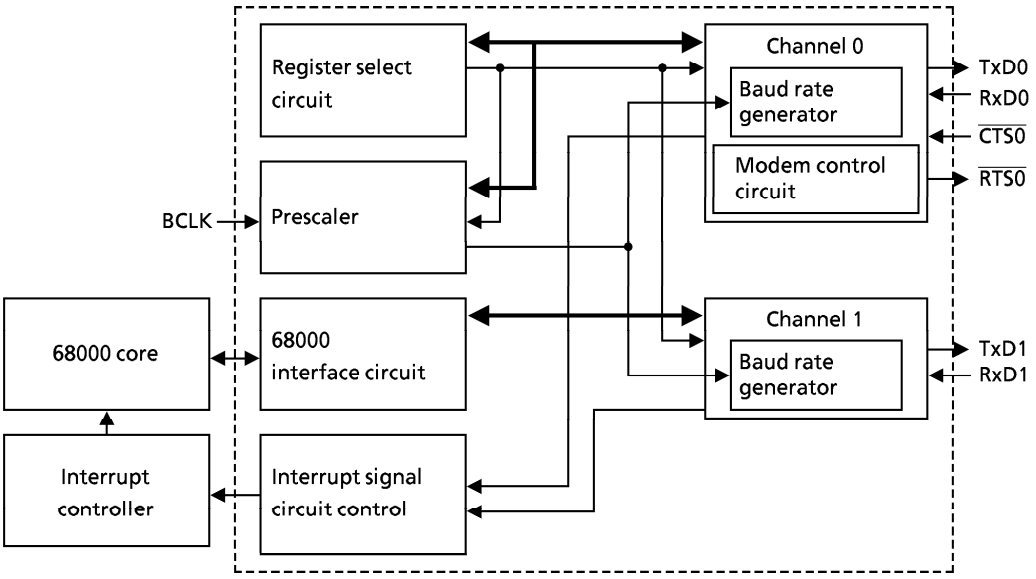


Figure 5.1 Serial Interface Block Diagram

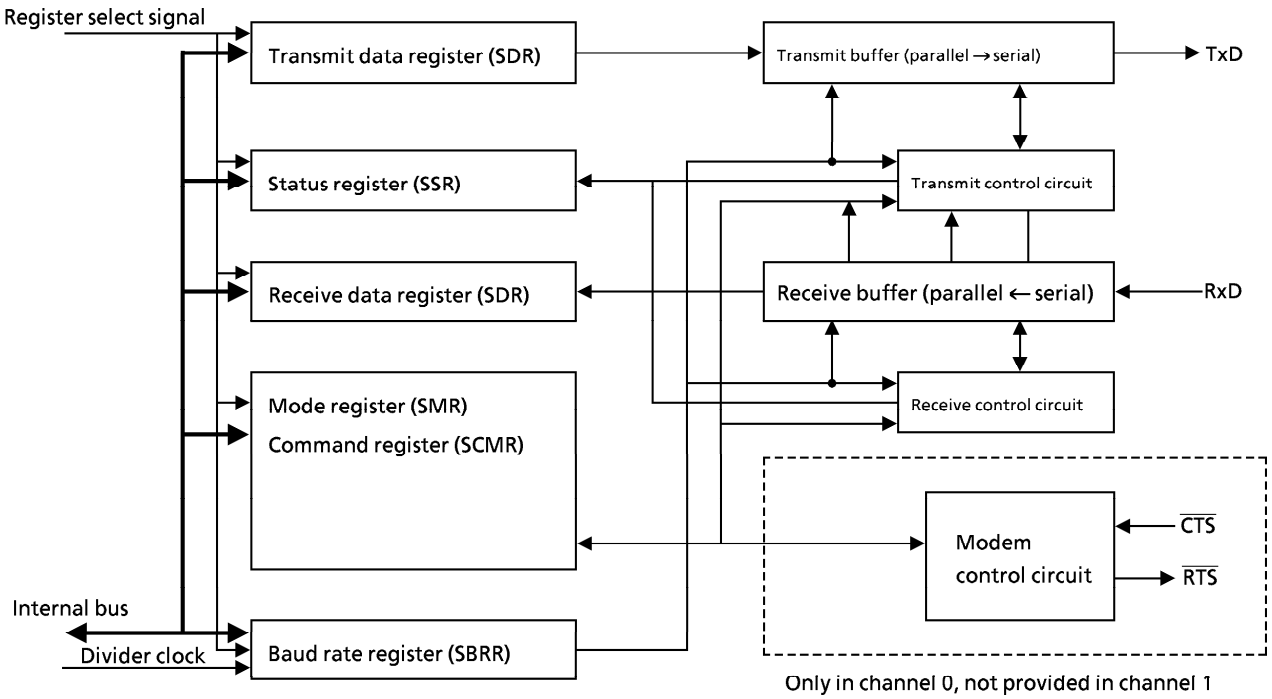


Figure 5.2 Channel Detail Diagram

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 and RxD1), and output to the transmit data output pins (TxD0 and TxD1).

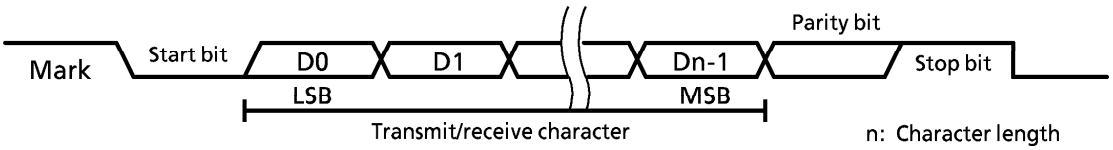


Figure 5.3 Data Frame

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.

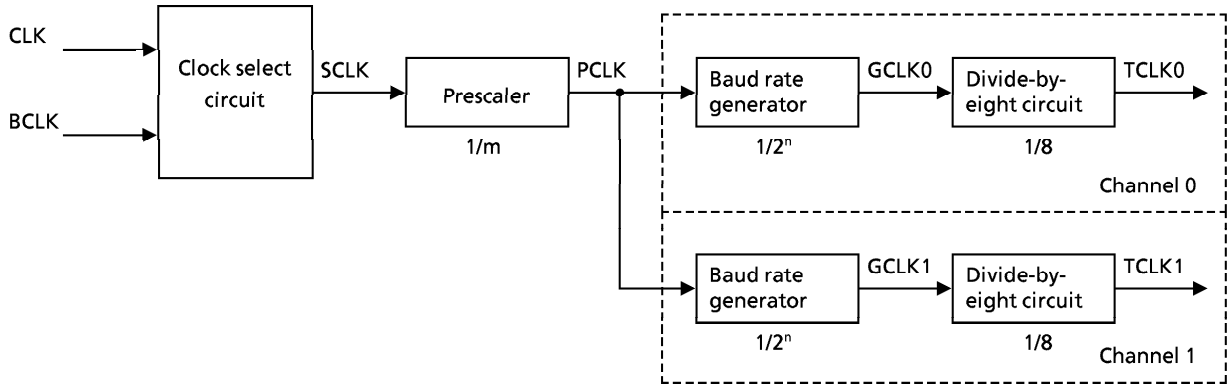


Figure 5.4 Baud Rate Clock Generation Circuit Block Diagram

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 - 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 $\overline{CTS0}$ is used in channel 0, the serial interface also checks pin P21/$\overline{CTS0}$. When $\overline{CTS0}$ is not used, if TxEN = 1 and transmit is not in progress, transmit begins with the transmit character loaded from the transmit data register to the transmit buffer. When $\overline{CTS0}$ is used, if TxEN = 1, P21/$\overline{CTS0}$ = 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.

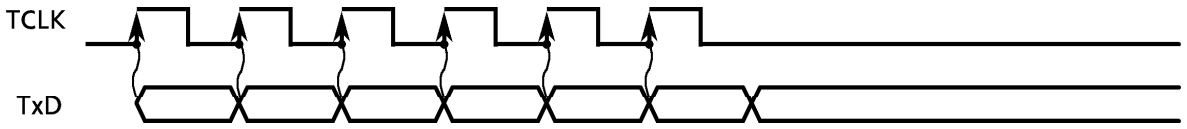


Figure 5.5 Data Frame Transmit Timing

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 loaded from the transmit data register to the transmit buffer at that point, the next transmit character can be loaded 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 transmit, it continues transmit of the current data character even if the status of the P21/$\overline{CTS0}$ pin or the TxEN bit change and ends transmit 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, $\overline{CTS0}$, 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.

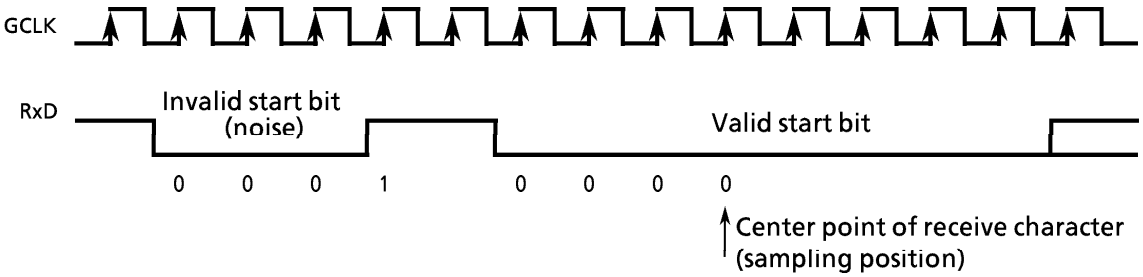


Figure 5.6 How to Determine Start Bits and Center Point of Receive Characters

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 transmit 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 enabled) = 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 and 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

× : Indicates logical AND

Channel n interrupts = (Sn = 0) × (INTM = 0) ×
{(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

TMP68303 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)

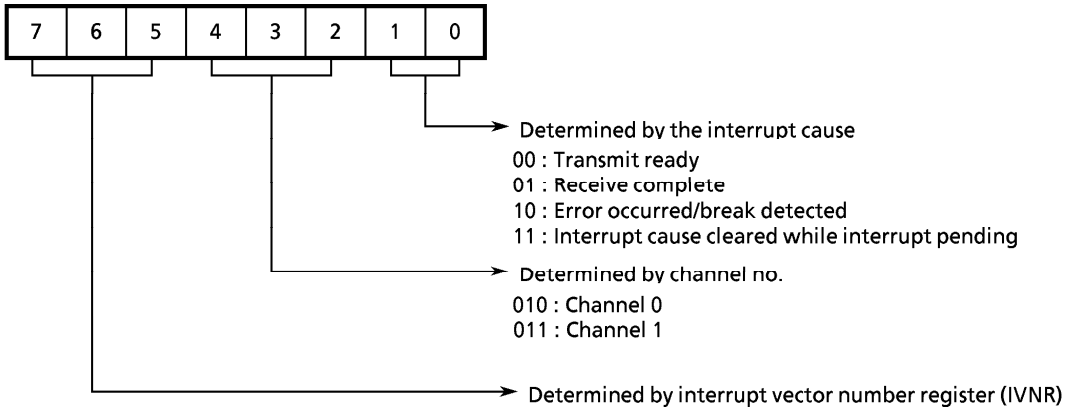


Figure 5.7 Serial Interface Vector Number Generation Method

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 S0 and S1 (serial interface channel 0 and 1 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.

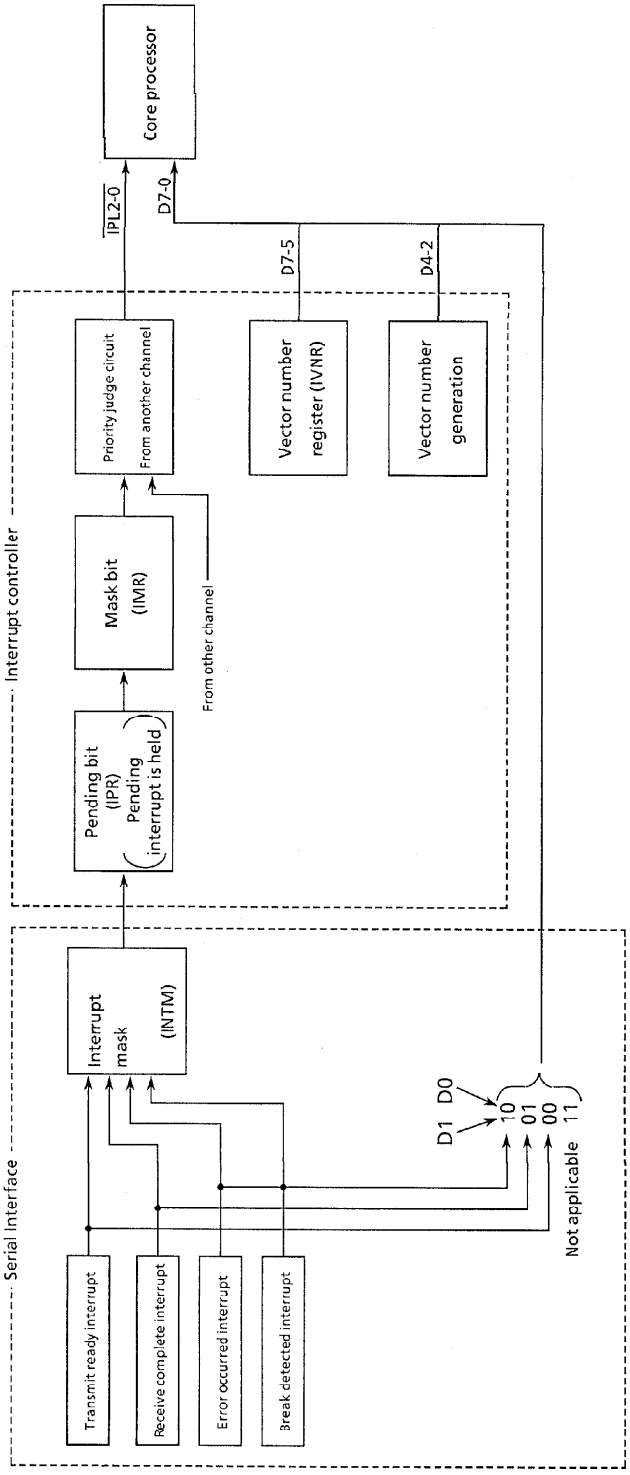


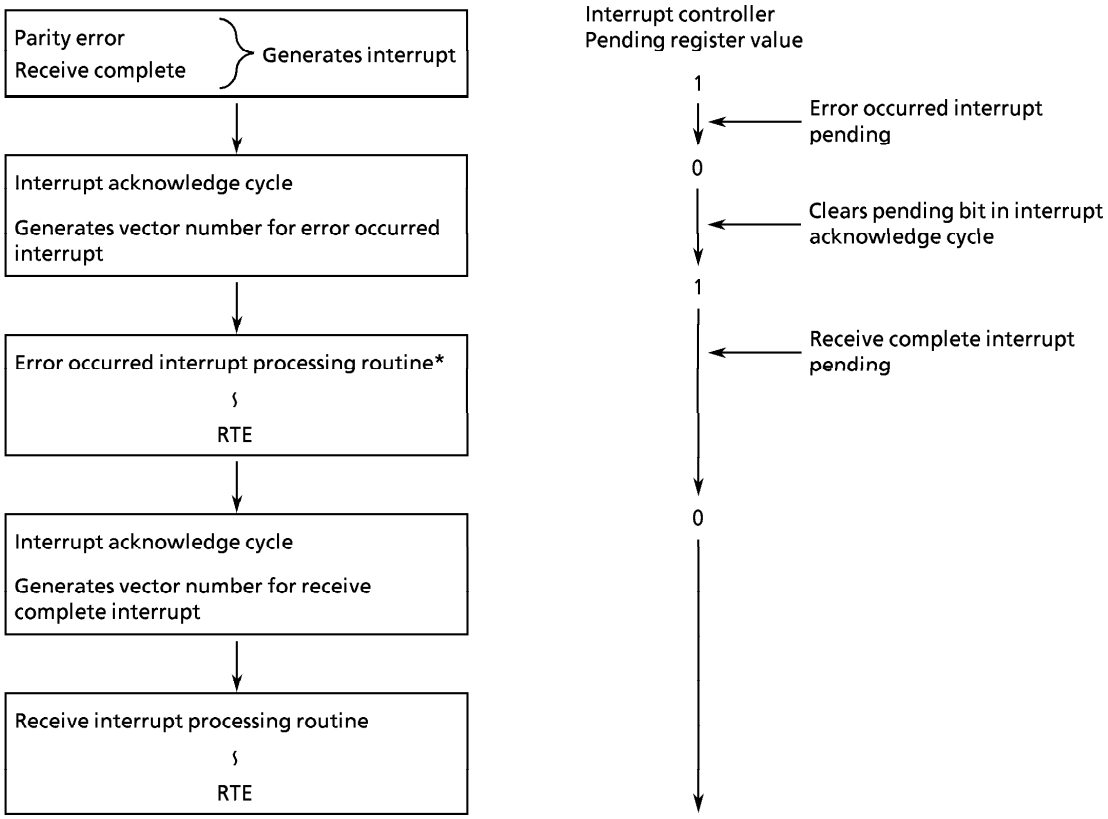
Figure 5.8 Relationship Between Interrupt Control Circuit and Interrupt Controller

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 RESET and 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 $\overline{\text{RESET}}$ and $\overline{\text{HALT}}$ 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, 1)			RTS *1	ERS	SBRK	R_XEN		T_XEN
	–	–	0	1	0	0	–	0
Serial status register (SSR0, 1)		RBRK	FE	OE	PE	T_XE	R_XRDY	T_XRDY
	–	0	0	0	0	1	0	0

– : Undefined
*1 : channel 0 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 $18F	7	6	5	4	3	2	1	0	
	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 : Reset cansel
- 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.

Offset address	7	6	5	4	3	2	1	0	
$181	R_X INTM	ER INTM	PEO	PEN	CL1	CL0	T_X INTM	ST	SMR0 For channel 0
Reset	1	1	0	0	0	0	1	0	
	7	6	5	4	3	2	1	0	
$191	R_X INTM	ER INTM	PEO	PEN	CL1	CL0	T_X INTM	ST	SMR1 For channel 1
	1	1	0	0	0	0	1	0	

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-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

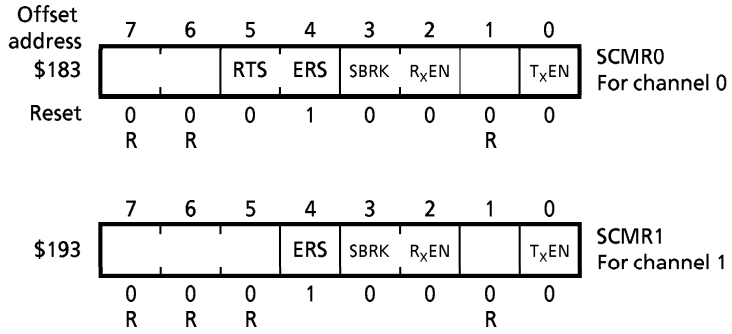
ST : Stop bit length
- 0 : One stop bit
- 1 : Two stop bits

*1 : Shows parity generation;
Even parity means that the number of 1's in the transmit/receive character including the parity bit is 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 : RTS0 pin output control (channel 0 only)
- 0 : High output from $\overline{RTS0}$ pin
- 1 : Low output from $\overline{RTS0}$ 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

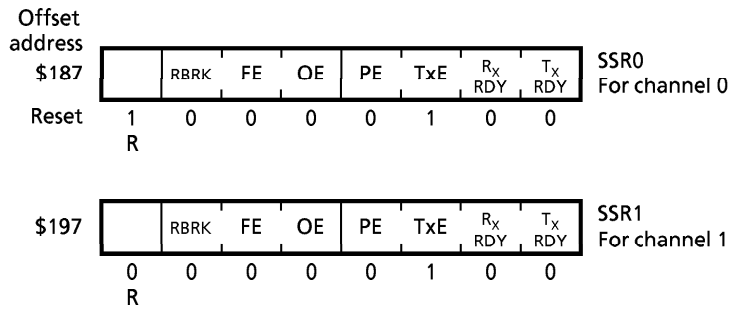
TxEN : Transmit enable
- 0 : Transmit disable
- 1 : Transmit enable

*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.

*2 : RTS0 multiplexes with the parallel interface I/O port: P20/$\overline{RTS0}$. Thus, the pin is used for $\overline{RTS0}$ output only when bit EMSIO of the port2 control register P2CR is set to 1.

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} x (CTS = 0) x (TxEN = 1)].
When channels 1 and 0 do not use CTS0, CTS0 is treated as $\overline{CTS0} = 0$ without checking CTS0 pin input. CTS0 multiplexes with the parallel interface I/O port: P21/$\overline{CTS0}$. Thus, the pin is used for CTS0 input only when bit EMSIO of the Port2 control register (P2CR) is set to 1.

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
2~255	× 1/2~ × 1/255

5.5.6 Serial Baud Rate Registers 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	B7	B6	B5	B4	B3	B2	B1	B0	SBRR0 For channel 0
	–	–	–	–	–	–	–	–	
	7	6	5	4	3	2	1	0	
$195	B7	B6	B5	B4	B3	B2	B1	B0	SBRR1 For channel 1
	–	–	–	–	–	–	–	–	

Baud rate register values								Divider ratio	
B7	B6	B5	B4	B3	B2	B1	B0	10 Decimal	
1	0	0	0	0	0	0	0	128	1/128
0	1	0	0	0	0	0	0	64	1/64
0	0	1	0	0	0	0	0	32	1/32
0	0	0	1	0	0	0	0	16	1/16
0	0	0	0	1	0	0	0	8	1/8
0	0	0	0	0	1	0	0	4	1/4
0	0	0	0	0	0	1	0	2	1/2
0	0	0	0	0	0	0	1	1	1

Table 5.2 Divider Ratios by Baud Rate Generator

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

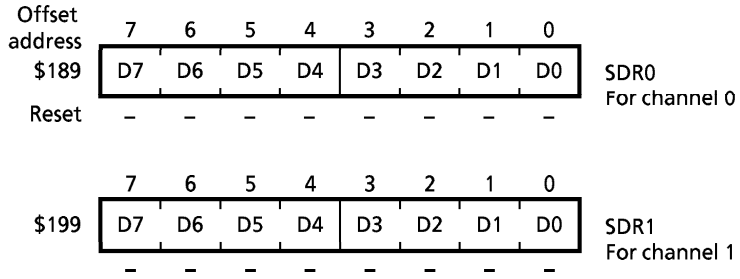
$$TCLKn = CLK \div PR \div BRRn \div 8$$

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 - 256 However, PR = 256 means value is 0)
BRRn : Channel n baud rate register (SBRRn) set value (BRRn = 1 - 128)

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 = 16.67MHz

Baud Rate (bps)	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error
38.4K	1 ($01)	54 ($36)	+ 0.48%	2 ($02)	27 ($1B)	+ 0.48%
19.2K	1 ($01)	109 ($6D)	– 0.44%	4 ($04)	27 ($1B)	+ 0.48%
14.4K	1 ($01)	145 ($91)	– 0.20%	16 ($10)	9 ($09)	+ 0.48%
9600	1 ($01)	217 ($D9)	+ 0.02%	8 ($08)	27 ($1B)	+ 0.48%
4800	2 ($02)	217 ($D9)	+ 0.02%	16 ($10)	27 ($1B)	+ 0.48%
2400	4 ($04)	217 ($D9)	+ 0.02%	32 ($20)	27 ($1B)	+ 0.48%
1200	8 ($08)	217 ($D9)	+ 0.02%	64 ($40)	27 ($1B)	+ 0.48%
600	16 ($10)	217 ($D9)	+ 0.02%	128 ($80)	27 ($1B)	+ 0.48%

SCLK = 16.0MHz

Baud Rate (bps)	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error
38.4K	1 ($01)	52 ($34)	+ 0.16%	2 ($02)	26 ($1A)	+ 0.16%
19.2K	1 ($01)	104 ($68)	+ 0.16%	4 ($04)	26 ($1A)	+ 0.16%
14.4K	1 ($01)	139 ($8B)	– 0.08%	4 ($04)	35 ($23)	– 0.79%
9600	1 ($01)	208 ($D0)	+ 0.16%	8 ($08)	26 ($1A)	+ 0.16%
4800	2 ($02)	208 ($D0)	+ 0.16%	16 ($10)	26 ($1A)	+ 0.16%
2400	4 ($04)	208 ($D0)	+ 0.16%	32 ($20)	26 ($1A)	+ 0.16%
1200	8 ($08)	208 ($D0)	+ 0.16%	64 ($40)	26 ($1A)	+ 0.16%
600	16 ($10)	208 ($D0)	+ 0.16%	128 ($80)	26 ($1A)	+ 0.16%

SCLK = 12.5MHz

Baud Rate (bps)	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error
38.4K	1 ($01)	41 ($29)	– 0.76%	2 ($02)	20 ($14)	+ 1.73%
19.2K	1 ($01)	81 ($51)	+ 0.47%	4 ($04)	20 ($14)	+ 1.73%
14.4K	1 ($01)	109 ($6D)	– 0.45%	4 ($04)	27 ($1B)	+ 0.47%
9600	1 ($01)	163 ($A3)	– 0.15%	8 ($08)	20 ($14)	+ 1.73%
4800	2 ($02)	163 ($A3)	– 0.15%	16 ($10)	20 ($14)	+ 1.73%
2400	4 ($04)	163 ($A3)	– 0.15%	32 ($20)	20 ($14)	+ 1.73%
1200	8 ($08)	163 ($A3)	– 0.15%	64 ($40)	20 ($14)	+ 1.73%
600	16 ($10)	163 ($A3)	– 0.15%	128 ($80)	20 ($14)	+ 1.73%

Figure 5.3 Baud Rate Setting Examples (1)

SCLK = 8.0MHz

Baud Rate (bps)	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error
38.4K	1 ($01)	26 ($1A)	+0.16%	2 ($02)	13 ($0D)	+0.16%
19.2K	1 ($01)	52 ($34)	+0.16%	4 ($04)	13 ($0D)	+0.16%
14.4K	1 ($01)	69 ($45)	+0.64%	4 ($04)	17 ($11)	+2.12%
9600	1 ($01)	104 ($68)	+0.16%	8 ($08)	13 ($0D)	+0.16%
4800	2 ($02)	104 ($68)	+0.16%	16 ($10)	13 ($0D)	+0.16%
2400	4 ($04)	104 ($68)	+0.16%	32 ($20)	13 ($0D)	+0.16%
1200	8 ($08)	104 ($68)	+0.16%	64 ($40)	13 ($0D)	+0.16%
600	16 ($10)	104 ($68)	+0.16%	128 ($80)	13 ($0D)	+0.16%

SCLK = 7.3728MHz

Baud Rate (bps)	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error
38.4K	1 ($01)	24 ($18)	0%	2 ($02)	12 ($0C)	0%
19.2K	1 ($01)	48 ($30)	0%	4 ($04)	12 ($0C)	0%
14.4K	1 ($01)	64 ($40)	0%	4 ($04)	16 ($10)	0%
9600	1 ($01)	96 ($60)	0%	8 ($08)	12 ($0C)	0%
4800	1 ($01)	192 ($C0)	0%	16 ($10)	12 ($0C)	0%
2400	2 ($02)	192 ($C0)	0%	32 ($20)	12 ($0C)	0%
1200	4 ($04)	192 ($C0)	0%	64 ($40)	12 ($0C)	0%
600	8 ($08)	192 ($C0)	0%	128 ($80)	12 ($0C)	0%

SCLK = 1.8432MHz

Baud Rate (bps)	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error	BRR ( ) Hexadecimal	PR ( ) Hexadecimal	Error
38.4K	1 ($01)	6 ($06)	0%	2 ($02)	3 ($03)	0%
19.2K	1 ($01)	12 ($0C)	0%	4 ($04)	3 ($03)	0%
14.4K	1 ($01)	16 ($10)	0%	4 ($04)	4 ($04)	0%
9600	1 ($01)	24 ($18)	0%	8 ($08)	3 ($03)	0%
4800	1 ($01)	48 ($30)	0%	16 ($10)	3 ($03)	0%
2400	1 ($01)	96 ($60)	0%	32 ($20)	3 ($03)	0%
1200	1 ($01)	192 ($C0)	0%	64 ($40)	3 ($03)	0%
600	2 ($01)	192 ($C0)	0%	128 ($80)	3 ($03)	0%

Figure 5.3 Baud Rate Setting Examples (2)

6. Parallel Interface

This parallel interface incorporates a general-purpose 10-bit I/O port. I/O ports consist of two 4-bit ports (ports P0, P1) and one 2-bit port (port P2).

Ports P0 and P1 also function as stepping motor control ports (M1, M2). Port P2 is also used as a control pin ($\overline{CTS0}$, $\overline{RTS0}$) for the internal serial interface.

Port control registers are allocated to the stepping motor controller (ports P0, P1) and serial interface (port P2) register areas.

General I/O bit ports (also used as other pins)
I/O specifiable in units of bits

6.1 Ports P0, P1 (P00 - P03, P10 - P13)

Ports P0 and P1 are 4-bit general-purpose I/O ports. Input or output can be specified in units of bits using the control register (SMCMR). A reset resets the control register to 0 and sets ports P0 or P1 to input mode.

P00 to P03 and P10 to P13 also function as stepping motor control ports. Either general-purpose I/O ports or stepping motor control ports is selected using SMCMR. At reset, general-purpose I/O port is set.

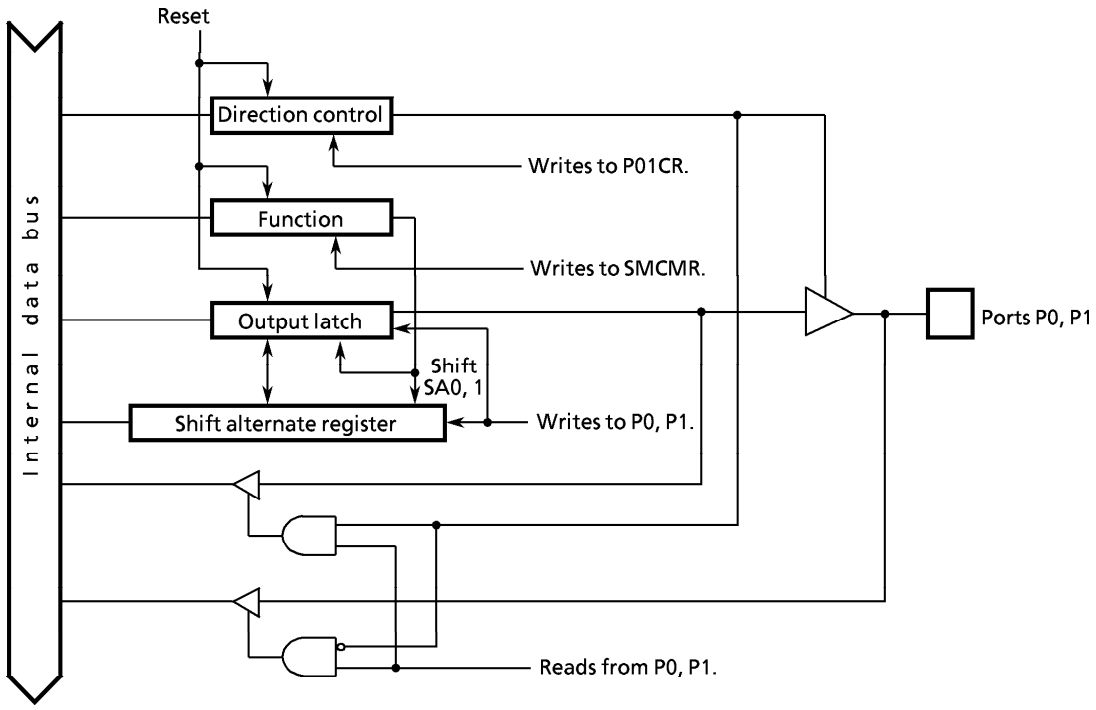
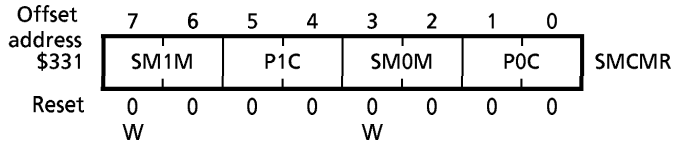


Figure 6.1 Ports P0, P1

Port P0/1 register is configured as shown below:



W : Write only

SM1M, SM0M : Stepping motor control ports P0, P1
When parallel ports are used, these bits are not used.

SM1M		SM0M		Function
bit7	bit6	bit3	bit2	
0	0	0	0	—
0	1	0	1	—
1	0	1	0	4-phase 1-step excitation or 4-phase 2-step excitation (full step)
1	1	1	1	4-phase 1-/2-step excitation

Note: After bits 7 and 3 are read, they are always set to 1.

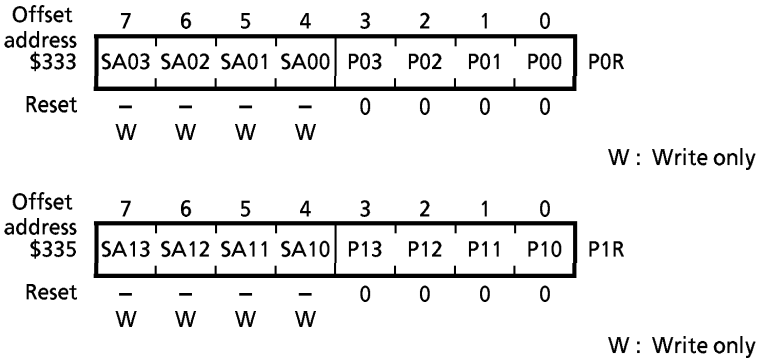
P1C : Port P1 function setting

P1C		P13	P12	P11	P10	SMC trigger signal
bit5	bit4					
0	0	IN/OUT	IN/OUT	IN/OUT	IN/OUT	—
0	1	IN/OUT	IN/OUT	IN/OUT	IN/Tout4	—
1	0	IN/M13	IN/M12	IN/M11	IN/M10	Timer 4
1	1					

P0C : Port P0 function setting

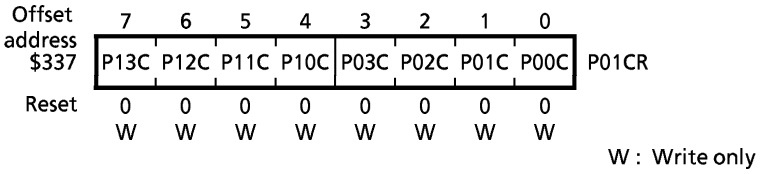
P0C		P03	P02	P01	P00	SMC trigger signal
bit1	bit0					
0	0	IN/OUT	IN/OUT	IN/OUT	IN/OUT	—
0	1	IN/OUT	IN/OUT	IN/OUT	IN/Tout3	—
1	0	IN/M03	IN/M02	IN/M01	IN/M00	Timer 3
1	1					

IN : Input port OUT : Output port Tout3/Tout4 : Timer output port
M03-00 : Stepping motor control port P0
M13-10 : Stepping motor control port P1



SA13-10: Stepping motor control shift alternate register 1
SA03-00: Stepping motor control shift alternate register 0
P13-10 : Port P1
P03-00 : Port P0

As the bit 3~0 of this register has input buffer and output buffer assigned same offset address, it is impossible to read the contents of these bits at input mode. (The input port data are red in this case)



P13C-P10C : Specify port P1 I/O.
P03C-P00C : Specify port P0 I/O.

0 : Input
1 : Output

Note : After this register is read, it is always set to $FF.

6.2 Port P2 (P20, P21)

Port P2 is a 2-bit general-purpose I/O port. Input or output can be specified in units of bits using control register bits P20CR and P21CR.

In addition to functioning as a general-purpose I/O port, P2 also functions as a control pin for the internal serial interface. This function is selected using the EMSIO bit in the control register.

P21 is also used as the $\overline{CTS0}$ input pin for the internal serial interface; P20, as the $\overline{RTS0}$ output pin. Setting the EMSIO bit to 1 sets P20 to $\overline{RTS0}$ output pin and P21 to $\overline{CTS0}$ input pin. The EMSIO bit value has a higher priority than the P20CR and P21CR bit values.

A reset resets the output latch and control register $00 and sets ports P2 to I/O mode.

Offset address	7	6	5	4	3	2	1	0	
$1B1	0	0	0	0	0	0	P21	P20	P2R
Reset	0 R	0 R	0 R	0 R	0 R	0 R	0	0	

R : Read only

Offset address	7	6	5	4	3	2	1	0	
$1B3	EMSIO	0	0	0	0	0	P21CR	P20CR	P2CR
Reset	0	0 R	0 R	0 R	0 R	0 R	0 W	0 W	

R : Read only
W : Write only

EMSIO : Port P2 function select
- 0 : General-purpose I/O port
- 1 : Internal serial interface control pin (Both CTS0 and RTS0 can be used.)

P21CR : P21 I/O mode control (write only)
- 0 : Input mode
- 1 : Output mode

P20CR : P20 I/O mode control (write only)
- 0 : Input mode
- 1 : Output mode

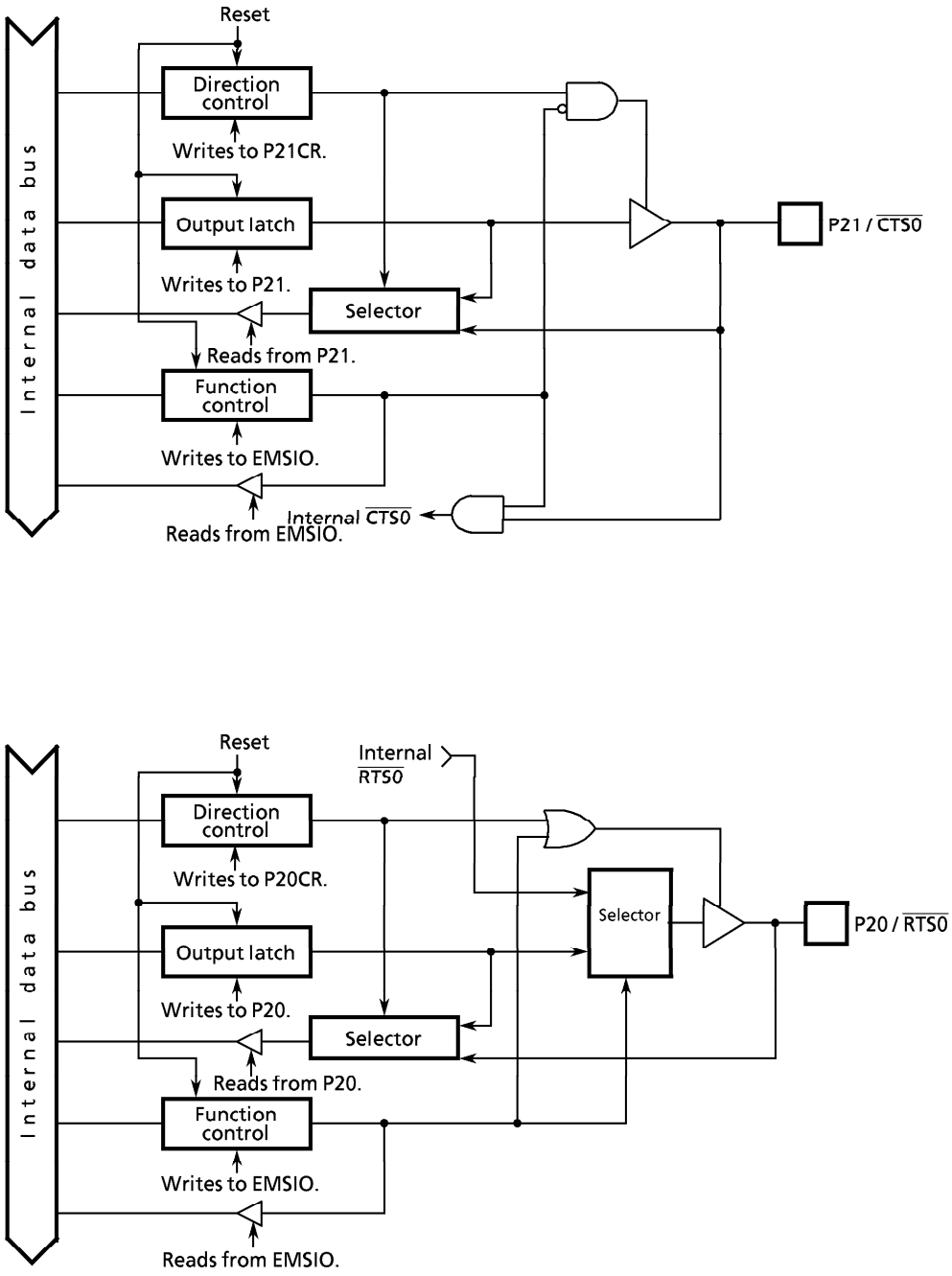


Figure 6.2 Ports P2

7. 16-Bit Timer

7.1 Outline

The timer consists of three independent channels, each with a 16-bit counter. Each channel has an 8-bit prescaler (valid only when the system clock is used). Channels 1 and 2 generate interrupt requests. The timer can cascade-connect the two channels for long counts. Channel 0 is used only as a runaway detect circuit (watchdog timer).

Three independent channels each with a built-in 16-bit counter
Count data are readable
Generates interrupts (channels 1 and 2 only)
Software/hardware trigger
Programmable operating modes
Outputs any duty comparison waveform (channels 1 and 2 only)
Outputs one-shot pulse (channels 1 and 2 only)
Event count
Max. 8MHz count (at 16 MHz)
Watchdog timer (channel 0 reset output)

Figures 7.1 and 7.2 show the timer outline.

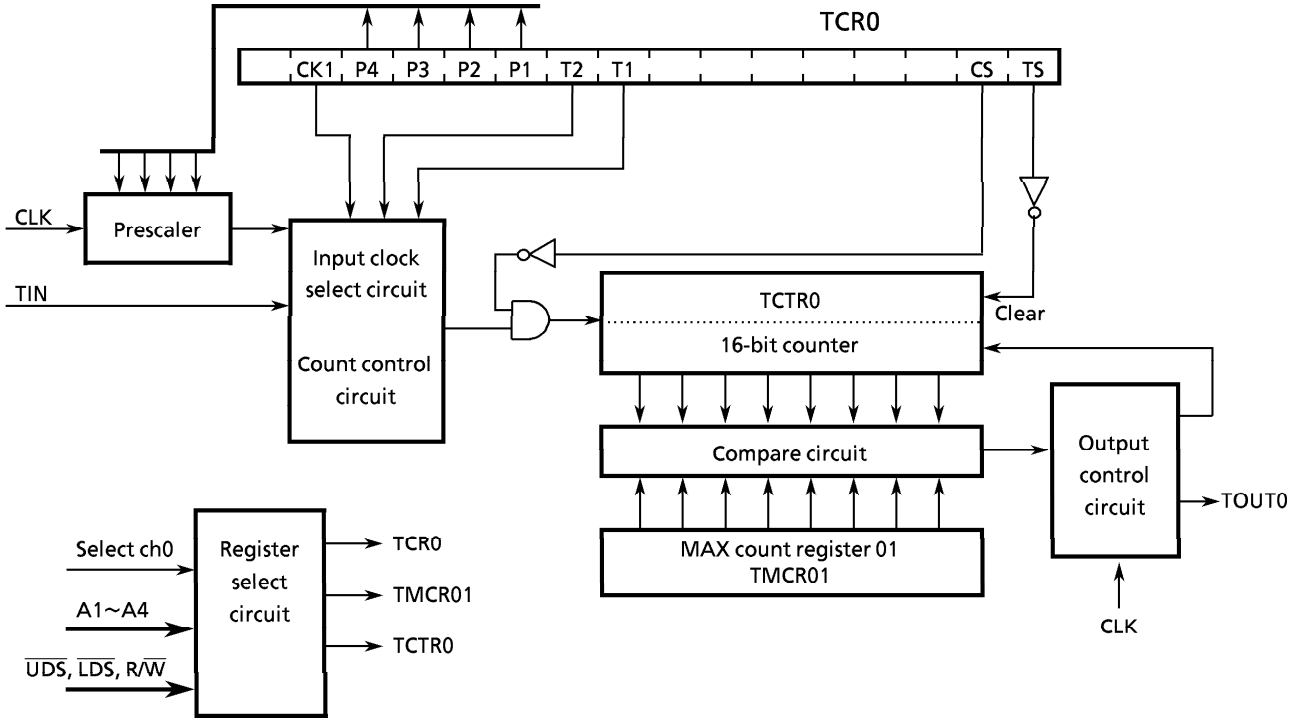


Figure 7.1 Timer Channel 0 Outline

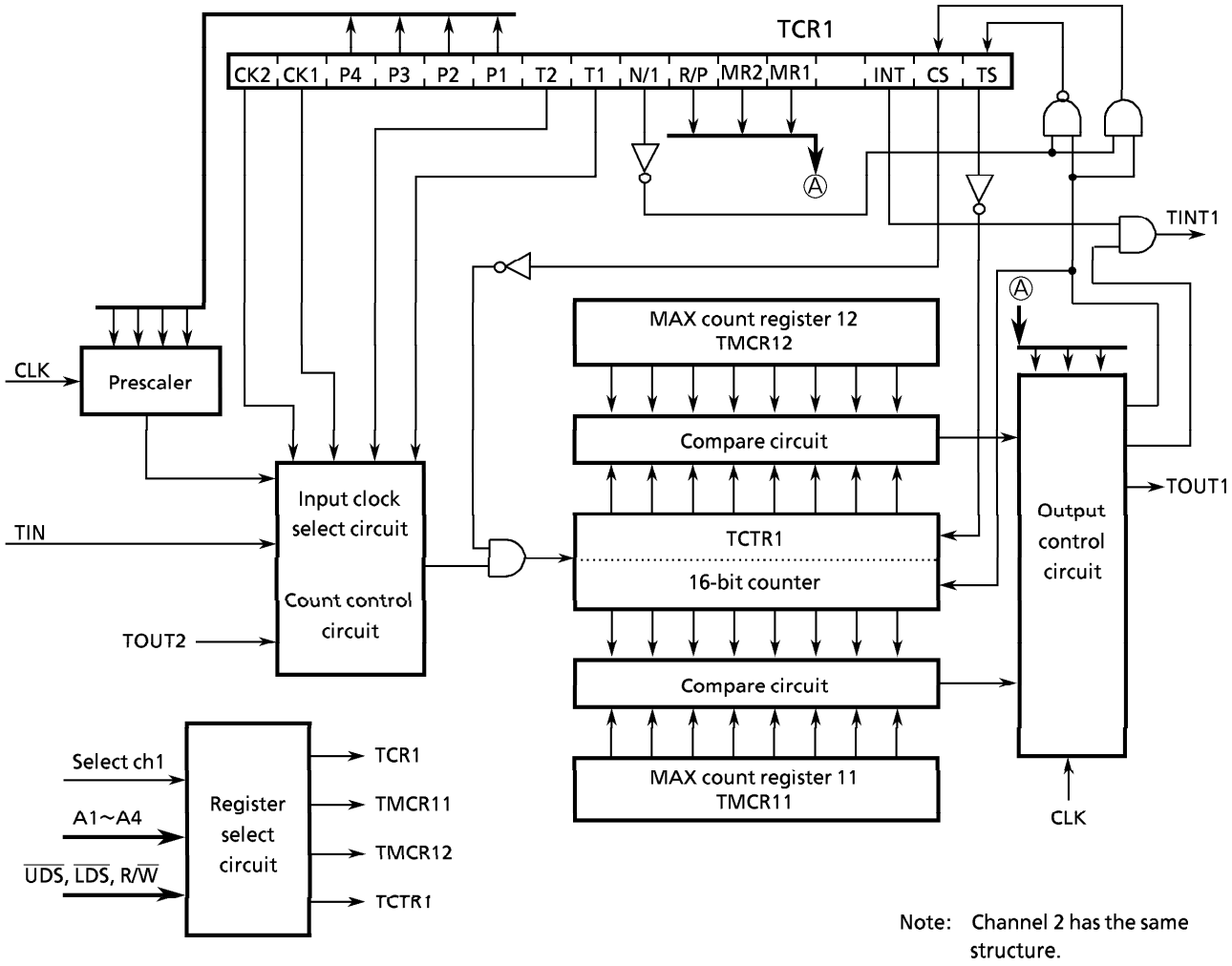


Figure 7.2 Timer Channel 1 Outline

7.2 Operational Description

Each channel internal counter is a 16-bit upcounter. Immediately the count value set 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.

7.2.1 Channel 0

Channel 0 has three registers: a (16-bit) control register, a (16-bit) count register, and a (16-bit) MAX count register. The internal counter is a 16-bit upcounter. Immediately the count value set 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. This channel can receive external clocks or external signals from external input pins (shared by channels 0 - 2). Channel 0 is exclusively used as a watchdog timer. It can output resets to the core processor and peripheral circuits.

7.2.2 Channels 1 and 2

Channels 1 and 2 have four registers: MAX count register 1 and 2, a control register, and a count register. The internal counter is a 16-bit upcounter. Immediately the count value set 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. These channels have external input pins (shared by channels 0 - 2) and external output pins, and can count up external events and output any waveform.

7.2.3 Setting and Modifying Maximum Counts, Reading and Initializing 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.)

The count value can be read from the count register during a count.

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.

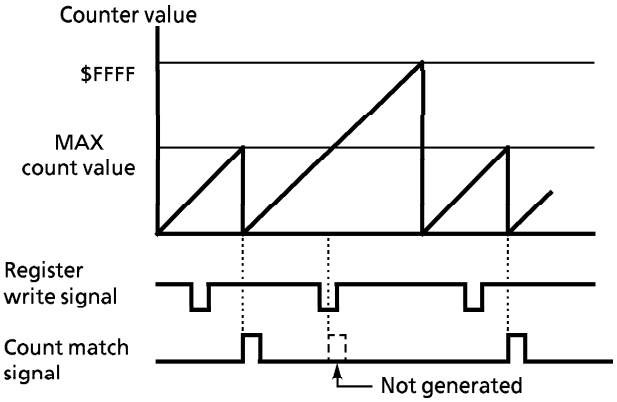
The timer ignores and does not store the count match signal during modes in which interrupt request signals are not generated. Therefore, even though modes are switched, an interrupt is not generated.

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 or interrupts. Note the following when writing to a register during a count operation.

When the timer count a clock obtained by dividing the system clock by 2 using the 8-bit prescaler is used for timer count, or when a TIN input clock half the frequency of the system clock is used for timer count, the count match signal may not be generated and the counter value may reach $FFFF if the register write signal and the count match signal (timer count value and MAX count register value comparison match signal) are generated simultaneously in the same channel.



These conditions occur when a match signal and write signal occur simultaneously in the same channel. For example, the above situation occurs when a channel 0-related register is written to at the same time as a channel 0 match signal is generated. However, this situation does not occur when a channel 1 register is written to at the same time a channel 0 match signal is generated.

To avoid this occurring, set the 8-bit prescaler divider ratio to 4 or higher. When using the TIN input clock, set the clock to 1/4 the system clock frequency or lower. When using a clock obtained by dividing the system clock by two, increase the interrupt priority level by program so that data are written to the register immediately after the count match signal is generated rather than at the time of the next match signal generation.

7.6 Timer Operating Mode

7.6.1 TIN Pin Function

The external input pin (TIN) can be used for inputting the external clock, and trigger and control signals.

Signals input from TIN are all sampled at the rising edge of the system clock. Clocks and pulses input from TIN must have a pulse width of one system clock cycle or more.

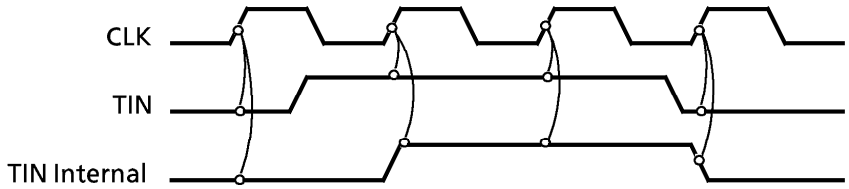


Figure 7.3 TIN Timing

7.6.1.1 External Clock

The external clock (TIN) can be used as the count clock.

7.6.1.2 Count Control Signal

Count control functions include the count start function and count wait function. These are selected by the count control register.

Count start function:

This function inputs a trigger signal from the external input pin (TIN). The count starts when the trigger signal is set to low (falling edge). (Any input after the second input is ignored.)

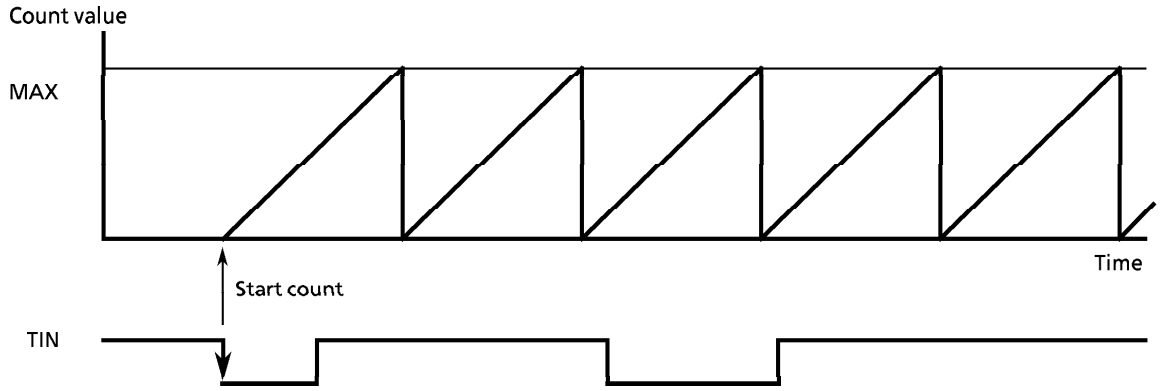


Figure 7.4 Count Start Function

Count Wait Function

This function inputs the control signal from the external input pin (TIN). The count is performed while the control signal is low. When the control signal is high, the count is stopped. When the control signal goes low again, the count resumes. (If the counter is not cleared, the count continues from the previous value.)

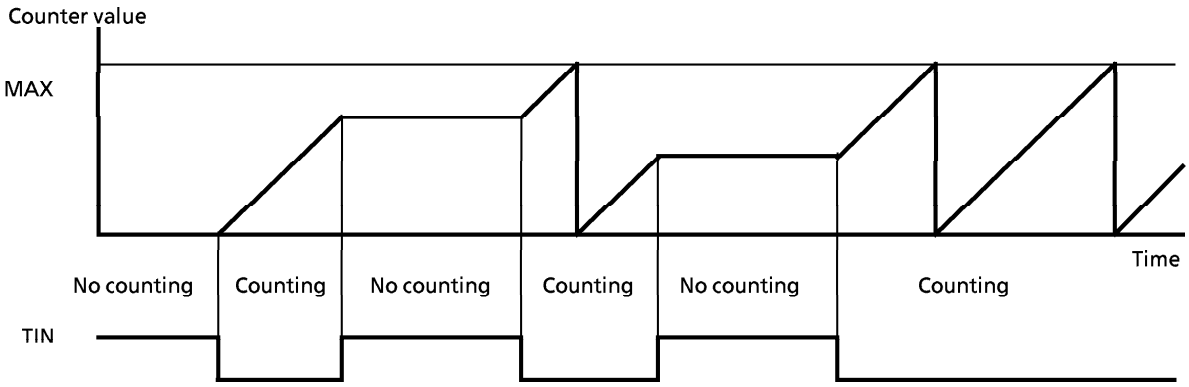


Figure 7.5 Count Wait Function

7.6.2 Count Clock Selection

The maximum count frequency is 8MHz. Therefore, use an external clock of 8MHz or less. As the prescaler divides the frequency by two or more, the maximum frequency is never exceeded when the system clock is used.

The count clock can be either the system clock, an external clock, or the output of another counter. Select the count clock when the counter is stopped.

7.6.3 TOUT Pin Function (Channels 1, 2)

The external output pin (TOUT) is used to output a pulse or square wave. The device can be set to either generate a pulse or invert the output level at a count match. By using two MAX counter registers, any square wave can be generated.

The signals output from TOUT (pulse generation or output level inversion timing) are synchronized with the rising edge of the system clock. If a pulse is generated, the pin, which is normally low, goes high for the duration of one clock at a count match. TOUT is low after a reset.

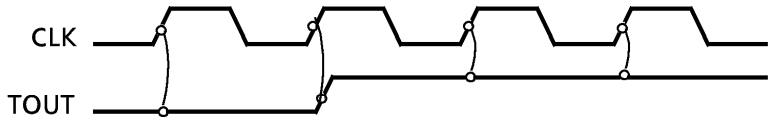


Figure 7.4 TOUT Timing

7.6.4 MAX Count Register Setting

Each channel has two MAX registers (henceforth, MAX1 and MAX2). Either register can be used individually or both registers can be used alternately. Control register settings specify how the MAX count registers are used.

7.7 Register Configuration

7.7.1 Timer Count Registers

The count values are read from these registers (TCTR0, TCTR1, and TCTR2). Only the upper byte or the lower byte can be read, too. The timer count registers are read-only.

Offset address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$20C	C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0	TCTR0
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	
$22C	C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0	TCTR1
	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	
$24C	C15	C14	C13	C12	C11	C10	C9	C8	C7	C6	C5	C4	C3	C2	C1	C0	TCTR2
	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

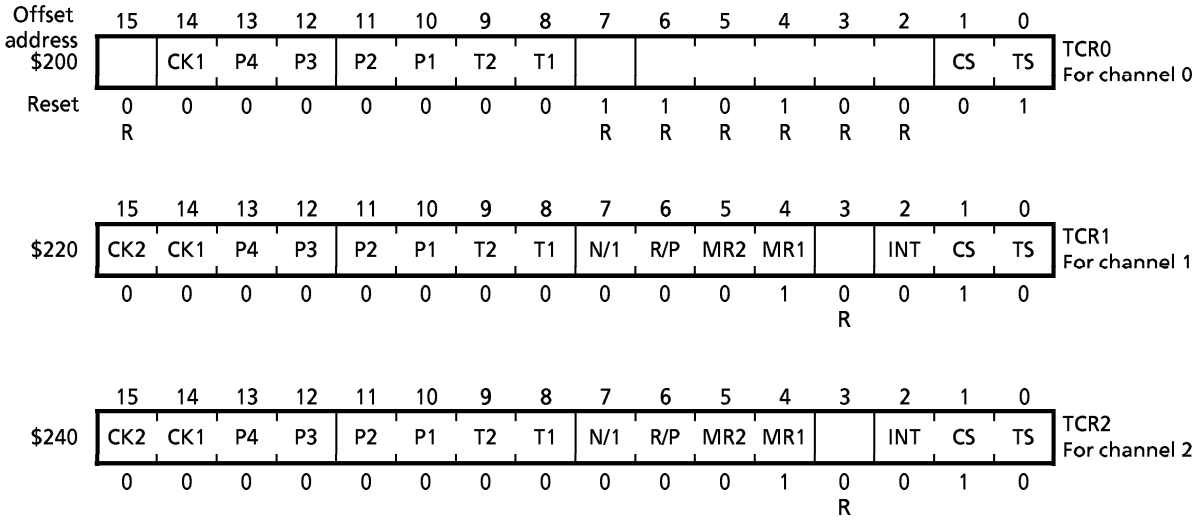
7.7.2 MAX Count Registers

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

Offset address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$204	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	TMCR01 MAX1
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	For ch0
$224	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	TMCR11 MAX1
	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	For ch1
$244	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	TMCR21 MAX1
	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	For ch2
$228	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	TMCR12 MAX2
	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	For ch1
$248	M15	M14	M13	M12	M11	M10	M9	M8	M7	M6	M5	M4	M3	M2	M1	M0	TMCR22 MAX2
	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	For ch2

7.7.3 Timer Control Registers

The control registers control the count operation. The registers consist of the following bits.



CK2, CK1 : Set the input clock for the counter.

CK2	CK1	Input clock
0	0	System clock (CLK)
0	1	External clock (TIN pin)
1	0	Prohibited
1	1	For cascade connection Channel 1 : Channel 2 counter output Channel 2 : Channel 1 counter output

Note: Channel 0 is used only when CK2 = 0.

Note: Other timer outputs 1 and 2 are for cascade connection.
P4, P3, P2, P1: Set prescaler divider ratio.
Valid only when the system clock (CLK) is used as the input clock. (Not required if other input clock is used.)

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

T2, T1: Count mode selection

T2	T1	Function select
0	0	When an external clock is used or when count control is not used
1	0	Uses the count start function. (When repeatedly using this function, first change the setting to another value then return to this setting.)
1	1	Uses the count wait function.

N/1 : 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, ending the count operation.
1 : Repeats count operation. (Repeat)

R/P : Output signal control (channels 1 and 2 only)
0 : Generates a pulse as the output signal. (pulse)
1 : Inverts the output level. (level inversion)

MR2, MR1 : MAX count register setting

MR2	MR1	MAX count register select
0	1	MAX1
1	0	MAX2
1	1	MAX1 and MAX2 alternately

INT : Interrupt request bit
0 : No interrupt request
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.

Note: As channel 0 does not have a waveform output function, the channel 0 control register has no functions based on R/P, MR2, or MR1.

8. 8-Bit Timers

TMP68303 includes two internal 8-bit interval timers (timer 3 and 4), which can be operated independently.

Each 8-bit timer consists of an 8-bit main timer (timers 31 and 41) and an 8-bit sub-timer (timer 30 and 40). Cascade-connecting these timers generates two 16-bit timers. The 8-bit timers can operate in the following four modes:

Two 8-bit interval timer modes
Two 16-bit interval timer modes
Two 8-bit PPG (programmable pulse generator) output mode
Two 8-bit PWM output mode

Figure 8.1 is a block diagram of 8-bit timer 3 (timer 30/31).

Timer 4 has the same circuit configuration as timer 3.

Each timer consists of an 8-bit up-counter, an 8-bit comparator, an 8-bit timer register, and a timer flip-flop.

Internal clocks ϕT1, ϕT16, and ϕT256, three input clock sources for the timers, are obtained using the 9-bit prescaler shown in Figure 8.2.

The 8-bit timer operating modes and the timer flip-flops are controlled by five control registers (TMDR, TCCR, TRCR, TIRCR, and TIMR).

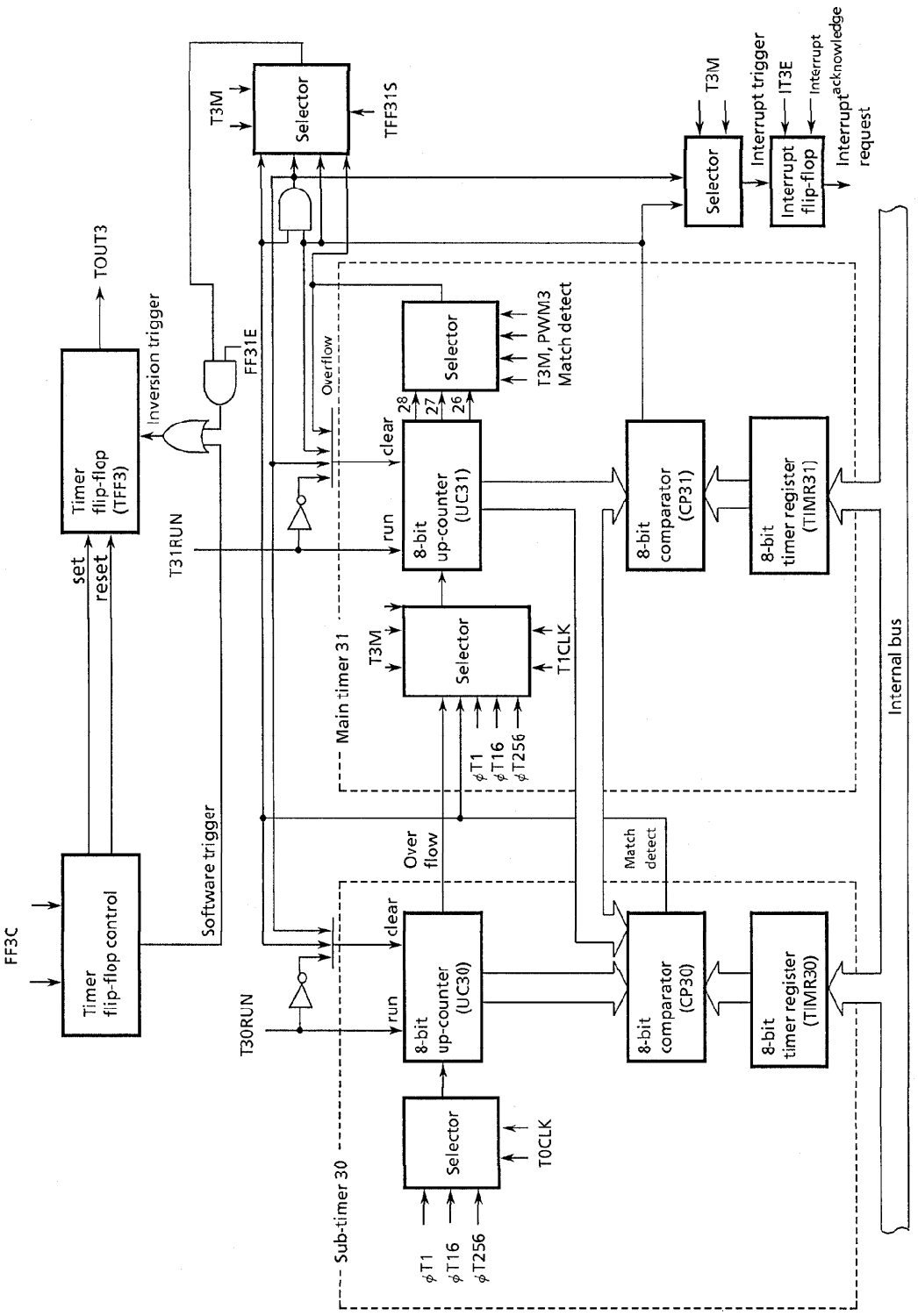


Figure 8.1 8-Bit Timer Block Diagram (Timer 3)

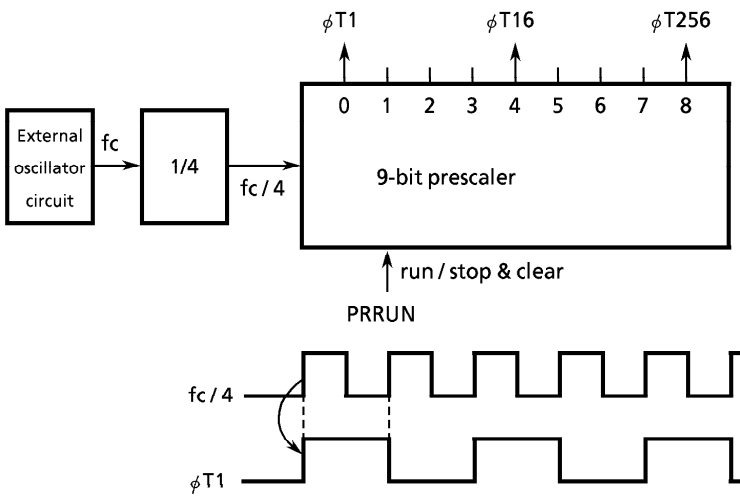
8.1 Functional Description

8.1.1 Prescaler

The 9-bit prescaler further divides the clock obtained by dividing the system clock (fc) by four (fc/4) to generate the input clock for the 8-bit timers.

The 8-bit timers use three clocks: ϕT1, ϕT16, and ϕT256.

Bit 5 (PRRUN) of the timer control register (TRCR) controls start and stop of the prescaler. Setting PRRUN = 1 starts counting. Setting TRUN5 = 0 stops counting and clears the value to zero. PRRUN is cleared to 0 at reset, thus clearing and stopping the prescaler.



Frequency

Input clock \ fc	8MHz	16MHz
ϕT1	1.0μs	0.5μs
ϕT16	16μs	8μs
ϕT256	256μs	128μs

Figure 8.2 Prescaler

8.1.2 Up-counter

The up-counter is an 8-bit binary counter that counts using the input clock specified by the 8-bit timer clock control register (TCCR) and 8-bit timer mode register (TMDR).

TCCR selects the input clocks used for sub-timer 30 and 40 from the three internal clocks: ϕT1, ϕT16, and ϕT256.

However, when TMPR is in 16-bit timer mode, ϕT256 cannot be selected as the input clock for sub-timers 30 or 40.

Input clock selections for main timer 31 and 41 vary according to the operating mode. When 16-bit timer mode is set, the overflow output of sub-timer 30 and 40 becomes the input clock, regardless of the TCCR setting. When other than 16-bit timer mode is set, the TCCR setting determines whether the comparator output (match detect) of sub-timer 30 or 40, or one of the internal clocks ϕT1, ϕT16, or ϕT256 is used as the input clock.

The TMDR register specifies the operating mode. At reset, 8-bit timer mode is set.

The up-counter independently controls the count start operation and count stop and clear operations for each interval timer based on the timer control register (TRCR) setting. At reset, all up-counters are cleared and the timers stopped.

8.1.3 Timer Register

The 8-bit timer register sets the interval time. The match detect signal becomes active whenever the up-counter value match the setting of the timer register. Setting the timer register to $00 causes the match detect signal from the comparator to become active at up-counter overflow.

8.1.4 Comparator

The comparator compares the up-counter value with the timer register setting. If the two values match, the comparator zero-clears the up-counter and generates an interrupt trigger signal. If timer flip-flop inversion is enabled, the comparator simultaneously inverts the timer flip-flop value. However, with the sub-timer, the comparator does not generate an interrupt trigger signal or a timer flip-flop inversion signal.

8.1.5 Timer Flip-Flop (Timer F-F)

The timer flip-flop is inverted by the interval timer match detect signal (output by the comparator). The timer flip-flop value is output from timer output pins TOUT3 (multiplexed with P00) or TOUT4 (multiplexed with P10).

One timer flip-flop is assigned to timer 3 and the other to timer 4, called TFF3 and TFF4, respectively. TFF3 is output to TOUT3. TFF4 is output to TOUT4.

The timer flip-flop control register (TFFCR) controls the timer flip-flops.

8.1.6 Interrupt Request Flip-Flop (Interrupt F-F)

The interrupt request flip-flops generate an interrupt request signal to the interrupt controller using interval timer interrupt trigger signals (output by the comparator).

The timer interrupt request control register (TIRCR) controls the interrupt flip-flops. Interrupt request signals are cleared by an acknowledge signal from the interrupt controller or by writing 0 to IT3E.

8.2 Register Configuration

8.2.1 8-Bit Timer Registers (TIMR30, 31, 40, 41)

These 8-bit registers set the interval times.

Offset address	7	6	5	4	3	2	1	0	
$261	T307	T306	T305	T304	T303	T302	T301	T300	TIMR30 Sub-timer 30
Reset	-	-	-	-	-	-	-	-	
$263	T317	T316	T315	T314	T313	T312	T311	T310	TIMR31 Main timer 31
Reset	-	-	-	-	-	-	-	-	
$265	T407	T406	T405	T404	T403	T402	T401	T400	TIMR40 Sub-timer 40
Reset	-	-	-	-	-	-	-	-	
$267	T417	T416	T415	T414	T413	T412	T411	T410	TIMR41 Main timer 41
Reset	-	-	-	-	-	-	-	-	

8.2.2 8-Bit Timer Clock Control Register (TCCR)

This register controls the clocks input to the timers.

Offset address $269 Reset

7	6	5	4	3	2	1	0	
T3CLK		T2CLK		T1CLK		T0CLK		TCCR
0	0	0	0	0	0	0	0	

T3CLK : Sets the input clock for main timer 41.

T3CLK		TMDR	
bit7	bit6	T4M ≠ 01	T4M = 01
0	0	Sub-timer 40 comparator output	Sub-timer 40 overflow output
0	1	Internal clock ϕT1	
1	0	Internal clock ϕT16	
1	1	Internal clock ϕT256	

T2CLK : Sets the input clock for sub-timer 40.

T2CLK		TMDR	
bit5	bit4	T4M ≠ 01	T4M = 01
0	0	—	—
0	1	Internal clock ϕT1	Internal clock ϕT1
1	0	Internal clock ϕT16	Internal clock ϕT16
1	1	Internal clock ϕT256	Not available

T1CLK : Sets the input clock for main timer 31.

T1CLK		TMDR	
bit3	bit2	T3M ≠ 01	T3M = 01
0	0	Sub-timer 30 comparator output	Sub-timer 30 overflow output
0	1	Internal clock ϕT1	
1	0	Internal clock ϕT16	
1	1	Internal clock ϕT256	

T0CLK : Sets the input clock for sub-timer 30.

T0CLK		TMDR	
bit1	bit0	T3M ≠ 01	T3M = 01
0	0	—	—
0	1	Internal clock ϕT1	Internal clock ϕT1
1	0	Internal clock ϕT16	Internal clock ϕT16
1	1	Internal clock ϕT256	Not available

8.2.3 8-Bit Timer Flip-Flop Control Register (TFFCR)

This register controls the timer flip-flops for each timer (TFF3 and TFF4).

Offset address $26B

7	6	5	4	3	2	1	0	
FF4C		FF41E	TFF4IS	FF3C		FF31E	TFF3IS	TFFCR
–	–	0	0	–	–	0	0	Reset
W	W			W	W			W : Write only

FF4C : Timer flip-flop TFF4 control (Write-only. The result of reading is always 11.)

bit7	bit6	Function
0	0	Clears TFF4 to 0.
0	1	Sets TFF4 to 1.
1	0	Inverts TFF4 value (software inversion).
1	1	Don't care

FF41E : Timer flip-flop TFF4 inversion control

0 : Invert disable

1 : Invert enable

TFF4IS : Sets the inversion signal for timer flip-flop TFF4.

bit4	TMDR bit T4M (bits 7, 6)			
	00	01	10	11
0	8-bit timer mode	TFF4 non-invertible		
1	8-bit timer mode	16-bit timer mode	PPG mode	PWM + 8-bit timer mode

FF3C : Timer flip-flop TFF3 control (Write-only. The result of reading is always 11.)

bit3	bit2	Function
0	0	Clears TFF3 to 0.
0	1	Sets TFF3 to 1
1	0	Inverts TFF3 value (software inversion).
1	1	Don't care

FF31E : Timer flip-flop TFF3 inversion control

0 : Invert disable

1 : Invert enable

TFF3IS : Sets the inversion signal for timer flip-flop TFF3.

bit0	TMDR bit T3M (bits 3, 2)			
	00	01	10	11
0	8-bit timer mode	TFF3 non-invertible		
1	8-bit timer mode	16-bit timer mode	PPG mode	PWM + 8-bit timer mode

8.2.4 8-Bit Timer Mode Register (TMDR)

This register sets the operating mode for each timer.

Offset address $26D Reset

7	6	5	4	3	2	1	0	
T4M		PWM4		T3M		PWM3		TMDR
0	0	0	0	0	0	0	0	

T4M : Sets operating mode for timer 4.

bit7	bit6	Operating Mode
0	0	8-bit timer
0	1	16-bit timer
1	0	8-bit programmable pulse generator output
1	1	8-bit PWM output + 8-bit timer

PWM4 : Sets interval for timer 4 in PWM mode (other than PWM mode, don't care).

bit5	bit4	Period
0	0	Unused
0	1	2^6–1 clock
1	0	2^7–1 clock
1	1	2^8–1 clock

T3M : Timer 3 operating mode

bit3	bit2	Operating Mode
0	0	8-bit timer
0	1	16-bit timer
1	0	8-bit programmable pulse generator output
1	1	8-bit PWM output + 8-bit timer

PWM3 : Sets interval for timer 3 in PWM mode (other than PWM mode, don't care).

bit1	bit0	Period
0	0	Unused
0	1	2^6–1 clock
1	0	2^7–1 clock
1	1	2^8–1 clock

8.2.5 8-Bit Timer operation Control Register (TRCR)

This register controls start and stop of each timer.

Offset address $26F Reset

7	6	5	4	3	2	1	0	
		PR RUN		T41 RUN	T40 RUN	T31 RUN	T30 RUN	TRCR
0	0	0	0	0	0	0	0	
R	R		R					R : Read only

PRRUN : Prescaler operation control
T41RUN : Main timer 41 operation control
T40RUN : Sub-timer 40 operation control
T31RUN : Main timer 31 operation control
T30RUN : Sub-timer 30 operation control

0 : Stop and clear
1 : Count

8.2.6 8-Bit Timer Interrupt Request Control Register (TIRCR)

This register controls interrupt requests for each timer.

Offset address $271 Reset

7	6	5	4	3	2	1	0	
						IT4E	IT3E	TIRCR
1	1	1	1	1	1	0	0	
R	R	R	R	R	R			R : Read only

IT4E : Timer 4 interrupt request control
0 : Disables interrupt.
1 : Enables interrupt generation.

IT3E : Timer 3 interrupt request control
0 : Disables interrupt.
1 : Enables interrupt generation.

8.3 8-Bit Timer Mode

The two interval timers 3 and 4 can be used independently as 8-bit interval timers. As operation of both timers is the same, the following explanation uses timer 3 only.

(1) Generating a fixed-interval interrupt:

To generate a timer 3 interrupt at fixed intervals using timer 3 (main timer 31 only), first halt timer 3, then set the operating mode, input clock, and interval using registers TMDR, TCCR, and TIMR31 respectively. Next, enable the interrupt request signal and start timer 3 counting.

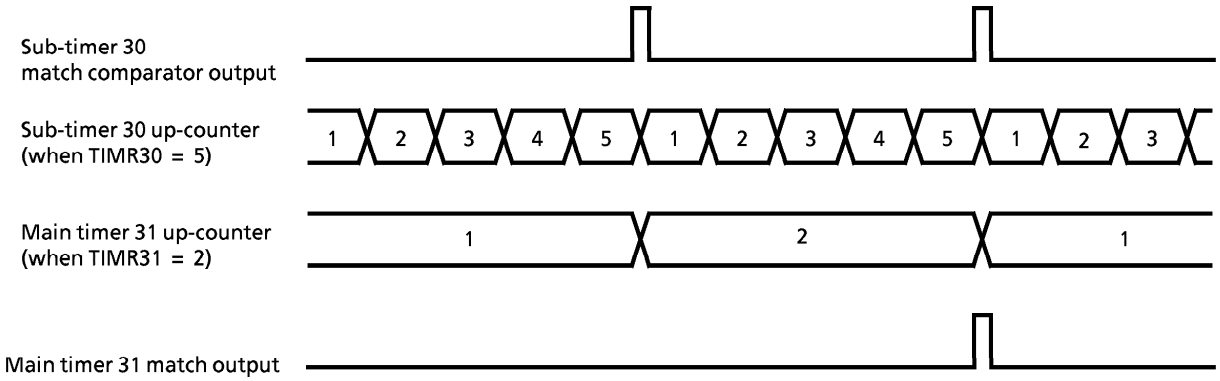
Select an input clock from the following table.

Table 8.1 Interrupt Interval and Input Clock Selection for 8-Bit Timers

Interrupt interval @ fc = 16 MHz	Resolution	Input clock
0.5μs ~ 127.5μs	0.5μs	ϕT1
8.0μs ~ 2.04ms	8.0μs	ϕT16
128.0μs ~ 32.64ms	128.0μs	ϕT256

(2) Counting up main timer 31 when sub-timer 30 match output:

Set to 8-bit timer mode and set the input clock for main timer 31 to the sub-timer 30 comparator output.



(3) Invert output by software:

The timer F/F value can be inverted independently of the timer operation.

Writing 10 to FF3C of TFFCR inverts the TFF3 value. Writing 10 to FF4C of TFFCR inverts the TFF4 value.

(4) Timer flip-flop initial setting

The timer flip-flop value can be initialized to 0 or to 1 independently of the timer operation.

For example, to set TFF3 to 0, write 00 to FF3C of TFFCR. To set TFF3 to 1, write 01 to FF3C of TFFCR.

8.4 16-Bit Timer Mode

The main timer and the sub-timer can be cascade-connected to create a 16-bit interval timer. As operation of timers 3 and 4 is the same, the following explanation uses timer 3 only.

To cascade-connect sub-timer 30 and main timer 31 to create a 16-bit interval timer, set bit T3M of TMDR to 01.

Setting the timer to 16-bit timer mode sets the overflow output of sub-timer 30 as the input clock of main timer 31, regardless of the TCCR setting. Use TCCR to set the sub-timer 30 input clock. However, note that ϕT256 cannot be used as a sub-timer 30 input clock in 16-bit timer mode. Table 3.3 shows the relationship between the timer and interrupt interval and the input clock selection.

Table 8.2 16-Bit Timer and Interrupt Interval vs Input Clock Selection

Timer (interrupt) interval @ fc = 16 MHz	Resolution	Resolution timer 0 input clock
0.5μs~32.77ms	0.5μs	ϕT1
8.0μs~524.28ms	8.0μs	ϕT16

To set the timer (interrupt) interval, set the lower eight bits in sub-timer register TIMR30 and the upper eight bits in main timer register TIMR31. In this case, always set TIMR30 first, as writing data to TIMR30 temporarily disables comparison, and writing to TIMR31 starts comparison.

Setting example: to generate an interrupt every 0.5s when fc=16MHz, set TIMR30 and TIMR31 as follows:

When ϕT16 (=8μs @16MHz) is selected as the input clock,

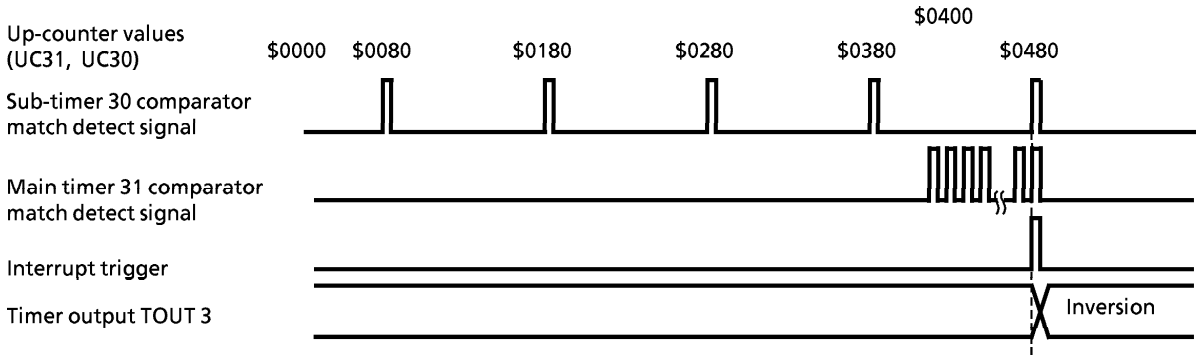
0.5s ÷ 8μs = 62500 = $F424

Accordingly, set TIMR31 = $F4 and TIMR30 = $24.

The sub-timer 30 comparator match is output whenever up-counter UC30 value matches the TIMR30 setting. However, up-counter UC30 is not cleared at this time.

A match signal is output whenever the main timer 31 comparator detects that up-counter UC31 value matches the main timer register TIMR31 setting. When the comparators of timers 30 and 31 both output match detect signals simultaneously, up-counters UC30 and UC31 are cleared to 0 and an interrupt trigger is generated. If inversion is enabled, the timer flip-flop TFF3 value is inverted.

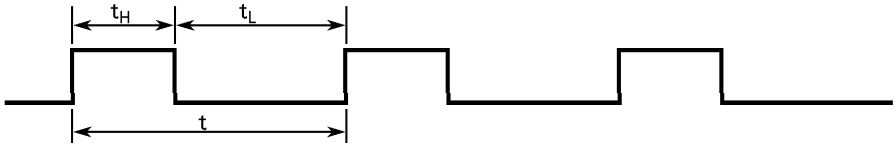
Example : When TIMR31 = $04 and TIMR30 = $80



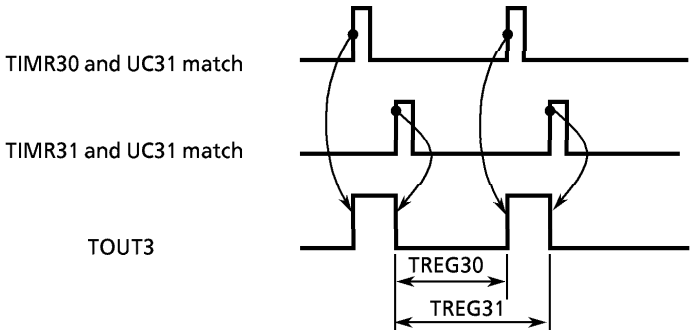
8.5 8-Bit PPG (Programmable Pulse Generator) Output Mode

A variable-frequency and variable-duty square wave can be output using main timer 31 or 41. The output pulse can be active high or active low.

With timer 3, the wave is output to TOUT 3 (multiplexed with P00). With timer 4, the wave is output to TOUT 4 (multiplexed with P10).



The following uses timer 3 as an example. (Timer 4 operates the same way.)



In this mode, a programmable square wave is output by inverting the timer output whenever an 8-bit up-counter 31 (UC31) value matches the setting in sub-timer register TIMR30 or main timer register TIMR31.

However, the following condition must be satisfied: (TIMR30 setting) < (TIMR31 setting).

The sub-timer 30 up-counter (UC30) cannot be used in this mode. However, set timer 30 to count by setting T30RUN of TRCR to 1.

Figure 8.3 illustrates this mode using a block diagram.

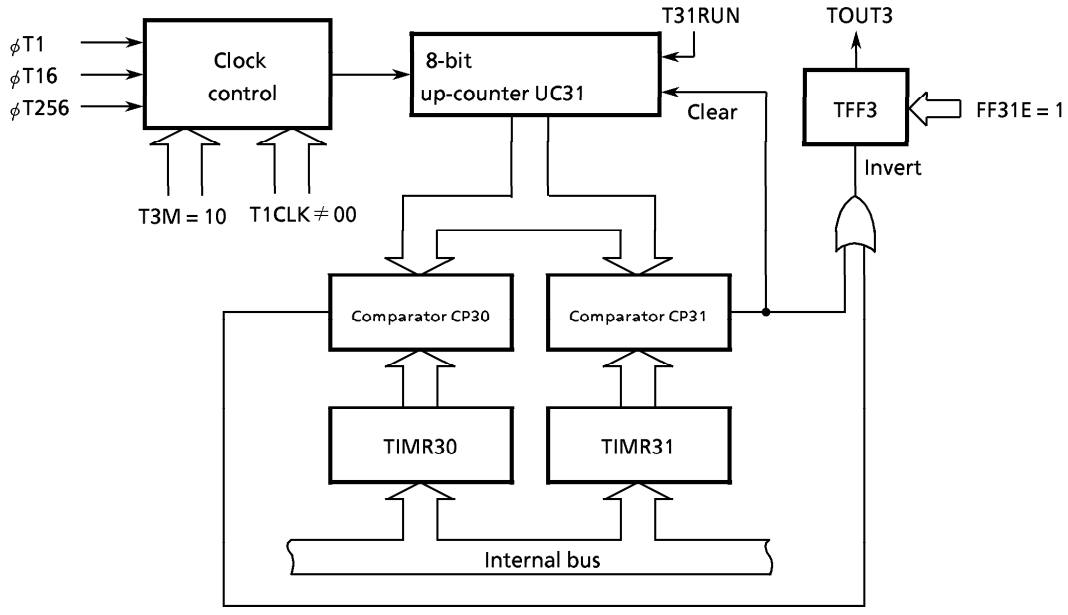
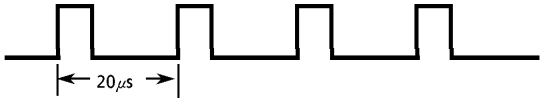


Figure 8.3 8-Bit PPG Output Mode Block Diagram

Example : To output a 50kHz pulse with 1/4 duty (@fc = 16MHz)



- Determine the value to be set in the timer register.
  To set the frequency to 50kHz, generate a waveform with interval t = 1/50kHz = 20μs.
  If φT1 = 0.5μs (@16MHz) is used,
  20μs + 0.5μs = 40
  Therefore, set main timer register 31 (TIMR31) to 40 = $28
  Next, for a 1/4 duty: t × (1 − 1/4) = 20 × 3/4 = 15μs
  15μs ÷ 0.5μs = 30
  Therefore, set sub-timer register 30 (TIMR30) to 30 = $1E.

8.6 8-Bit PWM Mode

In this mode, TMP68303 can generate up to two pulse width modulation (PWM) outputs (PWM3 and PWM4) with 8-bit resolution.

With timer 3, PWM output is via the TOUT3 pin (multiplexed with P00). With timer 4, PWM output is via TOUT4 pin (multiplexed with P10).

The following description uses timer 3 (PWM3) as an example. (Timer 4 operates the same way.)

The timer output is inverted when the up-counter (UC31) value matches the timer register TIMR31 setting and when a 2 n - 1 (where n = 6, 7, or 8, as set in PWM3 of TMDR) counter overflow occurs. The up-counter (UC31) is cleared by the 2 n-1 overflow.

In PWM mode, the following conditions must be satisfied.

(timer register setting) < (2^n - 1 counter overflow setting)
(timer register setting) ≠ 0

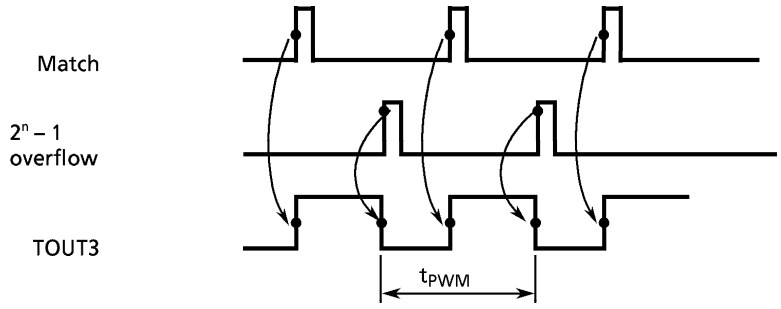


Figure 8.4 illustrates this mode using a block diagram.

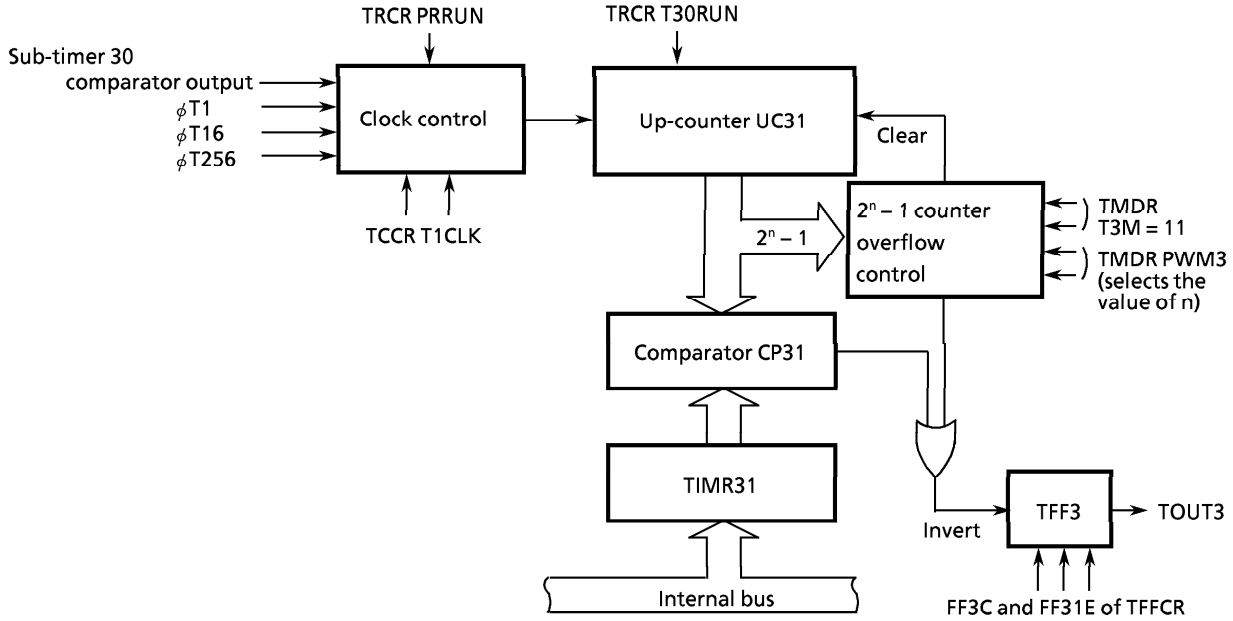
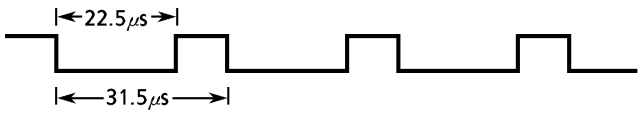


Figure 8.4 8-Bit PWM Output Mode Block Diagram

Setting example : To output the following PWM waveform to pin TOUT3 using timer 3 when fc = 16MHz:



To obtain a PWM interval of 31.5μs at ϕT1 = 0.5μs @fc = 16MHz

$31.5\mu s \div 0.5\mu s = 63 = 2^6 - 1$

Therefore, set n = 6. (TMDR1, 0 = 01)

The L level duration is 22.5μs. Therefore, at ϕT1 = 0.5μs, set

$22.5\mu s \div 0.5\mu s = 45 = \$2D$

in TIMR31.

Table 8.3 PWM Period and $2^n - 1$ Counter Setting

	PWM Period (@fc = 16MHx)		
	ϕT1	ϕT16	ϕT256
2^6-1	31.5μs	504μs	8.064ms
2^7-1	63.5μs	1016μs	16.256ms
2^8-1	127.5μs	2040μs	32.64ms

9. DMA Controller

The DMA controller (DMAC) has three 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 $\overline{\text{DREQ0 - 2}}$ external pins for receiving DMA requests, and $\overline{\text{DACK0 - 2}}$ external pins for responding to requests. In addition, a channel is provided with shared pin $\overline{\text{DTEND}}$, which can suspend DMA operation or notify end of DMA service.

Three independent DMA channels
Maximum 16M bytes of address space
Maximum of 64K - 1 byte for block transfer
Transfer operands can be either bytes or words
Maximum transfer speed of 8M bytes per second
Byte, demand, or continuous bus mode
Single- or dual-address mode
Demand or continuous cycle-steal
Transfer direction: memory ↔ memory or memory ↔ I/O device
An interrupt can be generated at DMA service completion or when an error occurs.

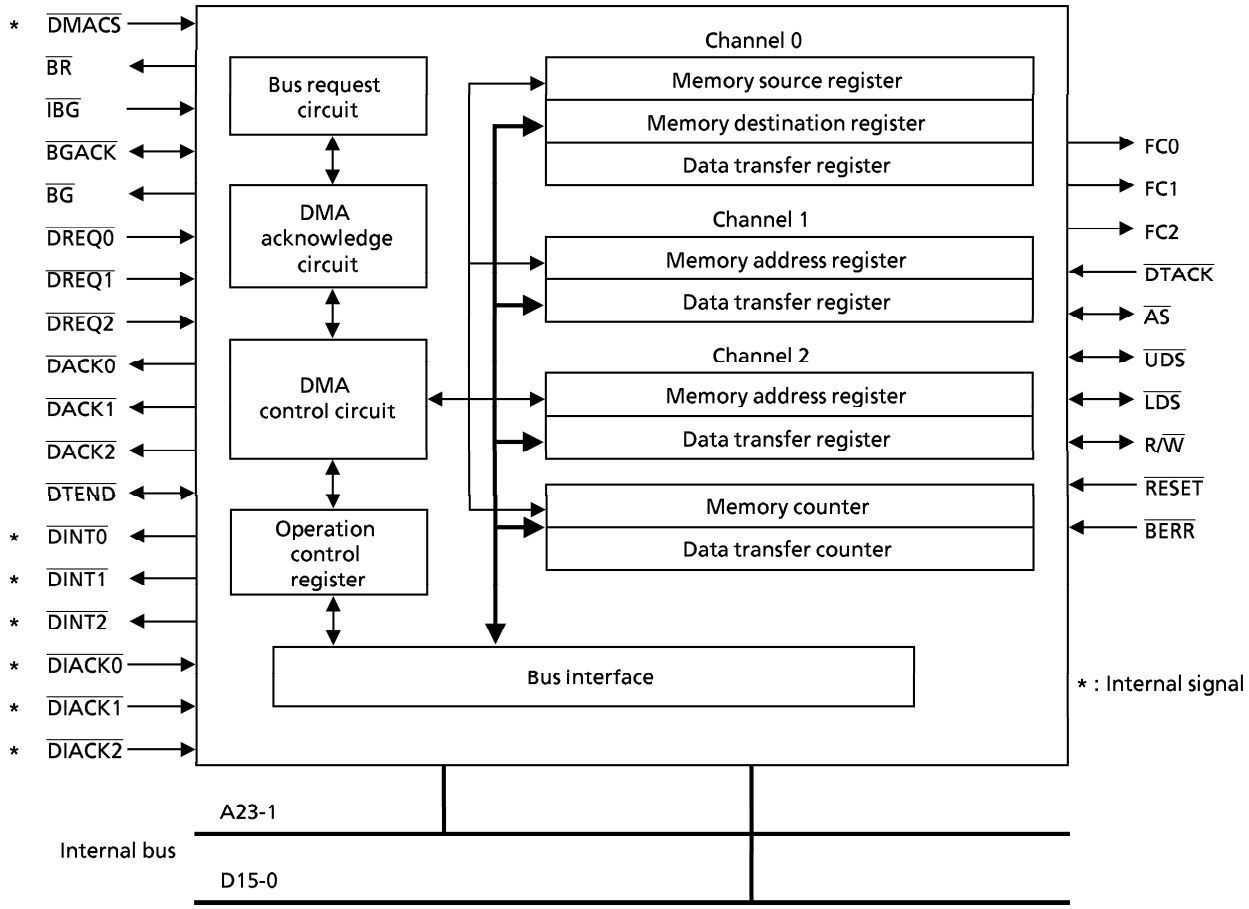


Figure 6.1 DMA Controller Block Diagram

9.1 Bus Arbitration

The DMAC asserts the $\overline{BR}$ (bus request) signal to obtain bus mastership. The core detects the $\overline{BR}$ 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.

9.2 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. In this case, the request is generated by setting the DST bit of the channel control register (CHCR0).

	Ch0	Ch1	Ch2
$\overline{DREQ}$ pin	○	○	○
DST bit	○	×	×

9.2.1 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 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 $\overline{DREQ}$ remains low for at least one clock cycle. When level mode is set, $\overline{DREQ}$ must be held low until the $\overline{BGACK}$ signal is output.

9.2.2 Auto Request

Auto request is supported for memory-to-memory (or memory-mapped I/O device) transfers only. The request is generated by software (supported by ch0 only). Setting the DMAE bit of OPCR and CHCR like for an external request and the DST bit of CHCR generates a DMA request.

9.3 Ending and Suspending Transfer

When the DTCR value reaches $0000, block transfer ends for that channel and the service completes. Asserting the $\overline{DTEND}$ signal during a DMA transfer suspends the transfer.

9.3.1 Service Complete

When the DTCR value reaches $0000, block transfer ends for that channel. The OPC bit of CHSR is set, and a service completion interrupt is output to notify the internal interrupt controller of the DMA transfer end (provided the INTE bit of CHCR and OPCR is set). At this time, the $\overline{DTEND}$ signal is also asserted externally (between S6 and S7).

While the OPC bit is set, DMA transfer is disabled. The OPC bit can only be cleared by writing data in DTCR.

9.3.2 Suspension

Asserting $\overline{\text{DTEND}}$ to $\overline{\text{AS}}$ (low) to the external input pin for a minimum of one system clock suspends the current channel DMA transfer after completion of the transfer bus cycle and causes requests to be ignored.

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.

9.4 Channel Priority

The DMA request channel priority is fixed at ch0 > ch1 > ch2. 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 on a high priority channel while a low priority channel is performing burst transfer, the high priority channel 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 high priority channel transfer is performed. At completion of the high priority channel transfer, transfer is resumed on the interrupted channel.

9.5 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 $\overline{\text{BG}}$ 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.

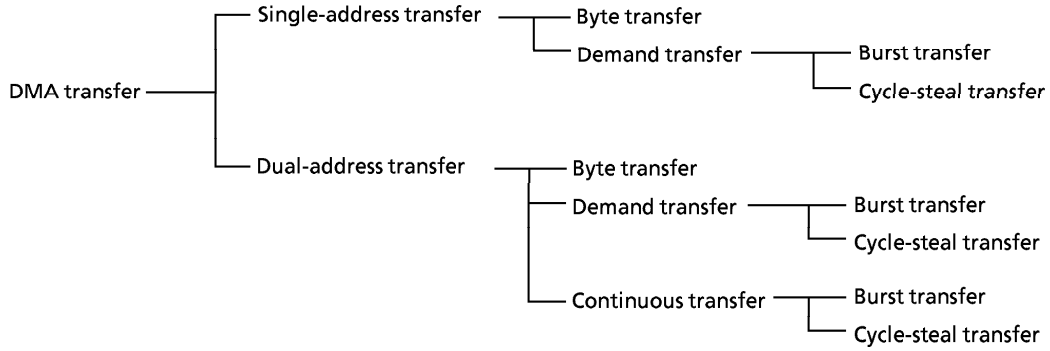
If the $\overline{\text{DTEND}}$ signal is continuously asserted while the DMAC is not operating, all internal DMAC transfer requests are ignored enabling the external bus master to have the highest priority.

9.6 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 acknowledge 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, no limitation applies to external requests. In this case, the data read from the memory specified by the source address are stored in the DMAC temporary register.



9.6.1 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. 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.

Transfer requests use external request mode. When the memory and I/O device $\overline{DTACK}$ signal is generated externally, a slow $\overline{DTACK}$ signal must be input by the external circuit. However, when a memory-side $\overline{DTACK}$ signal is generated by the internal address decoder, the internal address decoder automatically selects a slower $\overline{DTACK}$ signal; thus, no external circuit for selecting slower $\overline{DTACK}$ is necessary. (See Automatic $\overline{DTACK}$ Generation in the Address Decoder section.)

9.6.2 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. DMAC transfers data by dividing the bus cycle into read and write cycles. The read data are temporarily stored in the DMAC temporary register before being output in the write cycle.

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. When a bus error occurs (at a read), execution is stopped.

The transfer requests can be made either external requests or auto requests.

9.7 Transfer Mode

The transfer modes are byte, demand, and continuous transfer. Demand and continuous transfer modes can perform either burst transfer or cycle-steal transfer.

9.7.1 Byte Transfer Mode

In this mode, when a transfer request is generated for the DMAC, after bus arbitration, bus mastership is obtained, and the DMA cycle is generated. However, the bus is always released after one data transfer (8- or 16-bit data). Accordingly, the transfer ends after one bus cycle for single-address transfers and after two bus cycles for dual-address transfers.

Before another transfer can be requested, the previous request must be cleared.

9.7.2 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.

9.7.3 Continuous Transfer Mode

As this mode is supported only by channel 0 in response to a $\overline{DREQ}$ request in demand transfer mode, a transfer request must be generated by software. Setting the DSSH, CONTI and DST bits of the CHCR0 register executes the DMA cycle.

9.7.4 Cycle-Steal Transfer Mode

In this mode, when a transfer request is generated for the DMAC, after bus arbitration, bus mastership is obtained, and the DMA cycle is generated. After transfer of one byte or word, bus mastership is released to the core. After the core completes one bus cycle, it releases bus mastership back to the DMAC, which executes next DMA cycle. In this mode, transfer continues repeatedly.

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

9.7.5 Burst Transfer Mode

In this mode, the DMAC executes the DMA cycle after obtaining bus mastership. However, until transfer ends or until the completion conditions are satisfied, the DMAC has exclusive possession of the bus, during which time it continuously executes DMA cycles.

Conditions can be one of the following.

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

These conditions also apply to cycle-steal transfer mode.

9.8 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{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.

9.9 I/O Device Support

In single-address transfer mode, only external memory and I/O devices with an acknowledge function can transfer data to the DMAC. In dual-address transfer mode, only external memory and external memory-mapped I/O devices 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.

Table 9.1

	Address mode	Device type	Request		Operand size	Device size	
			Channel	Mode		8 bit	16 bit
DMA transfer	Single address	I/O device with acknowledge function	Ch0, 1, 2	External ($\overline{DREQ}$)	Byte	○	–
					Word	–	○
	Dual address	Memory mapped I/O device	Ch0	External ($\overline{DREQ}$) Auto (DST bit)	Byte	○	○
					Word	–	○

9.10 Addressing and Data Maps

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

Table 9.2

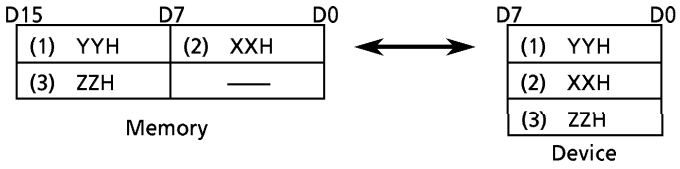
	Operand size	Device size	Memory address increment		Device address increment	
			Single	Dual	Single	Dual
DMA transfer	Byte	8 bit	+ 1	+ 1	–	+ 2
		16 bit	–	+ 1	–	+ 1
	Word	16 bit	+ 2	+ 2	–	+ 2

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 $\overline{\text{DACK}}$ signal and switch the bus in accordance with the $\overline{\text{UDS}}$/$\overline{\text{LDS}}$ signal.

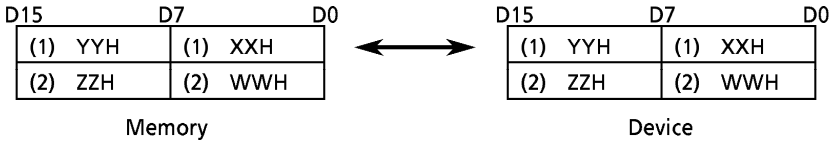
The start memory address can be either even-numbered or odd-numbered.



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.

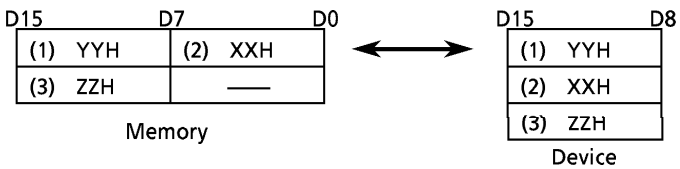
The start memory address must be even-numbered.



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.)

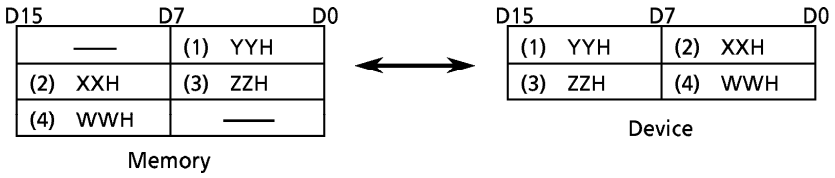
While the memory start address can be either even- or odd-numbered, the start device address must correspond to the side of the data bus to which the device is connected. For example, if the device is connected to the upper data bus, the address must be even-numbered.



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

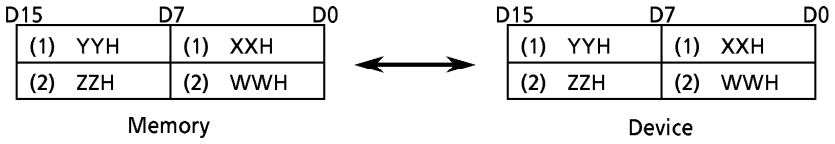
Under these conditions, no particular limitations apply because the I/O device and memory are both connected to a 16-bit data bus.

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.



9.11 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
CHCR0~2 : $0000
CHSR : $0000
SMAR : Undefined
DMAR : Undefined
MAR1~2 : Undefined
DTCR0~2 : 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.

9.13 Register Configuration

9.13.1 Operation Control Register (OPCR)

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

Offset address	7	6	5	4	3	2	1	0	
$28D					RWE	FC2	DMAE	INTE	OPCR
Reset	0	0	0	0					
	R	R	R	R					R : Read only

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 not set DMA operation.
- 1 – Enable DMA operation.

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

9.13.2 Channel Control Registers 0 - 2 (CHCR0 - 2)

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

Offset address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$280			ERC	MAC	DPW	OPW	DST	CONT 1	DSSH	TDIR 1	TDIR 0	TMOD 1	TMOD 0	RSS	DMAE	INTE	CHCR0 For ch0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	R	R															
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$290							ERC	MAC	DPW	OPW	TMOD 1	TMOD 0	RSS	TDIR	DMAE	INTE	CHCR1 For ch1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	R	R	R	R	R	R											
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$2A0							ERC	MAC	DPW	OPW	TMOD 1	TMOD 0	RSS	TDIR	DMAE	INTE	CHCR2 For ch2
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
	R	R	R	R	R	R											

R : Read only

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

MAC : MAR countup function setting
0 – Disable not set MAR countup function.
1 – Enable MAR countup function.

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

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

DST : DMA request initiation control using software (ch0 only)
(Automatically cleared after service completion)
0 – Does not initiate DMA request by software.
1 – Initiates DMA request by software.

CONT1 : Continuous transfer control (ch0 only)
0 – Disables continuous transfer.
1 – Enables continuous transfer.

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 setting (ch0 only)

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

TDIR : Data transfer direction setting (ch1, 2)

TDIR	Data transfer direction
0	Memory → I/O
1	I/O → Memory

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

9.13.3 Channel Status Register (CHSR)

This register indicates the operation status of all 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	ERRCD 21	ERRCD 20	ERRCD 11	ERRCD 10	ERRCD 01	ERRCD 00	ERR2	ERR1	ERR0	REQ2	REQ1	REQ0	OPC2	OPC1	OPC0	CHCR
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

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

ERRCDx1, 0 : Error code

ERRCD21	ERRCD20	Error Code
0	1	Channel 2 bus error
1	0	Channel 2 address error
1	1	Channel 2 count error

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

ERR2, 1, 0 : DMA channel 2, 1, 0 error
0 – No error
1 – Error

REQ2, 1, 0 : DMA channel 2, 1, 0 request
0 – No DMA request
1 – DMA request received

OPC2, 1, 0 : DMA channel 2, 1, 0 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.

9.13.4 Data Transfer Count Registers 0 - 2 (DTCR0 - 2)

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 depends on the byte counts. When the operand width is word, convert the count number in byte counts.

Offset address	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	DTCR0 For ch0
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	
$292	DT15	DT14	DT13	DT12	DT11	DT10	DT9	DT8	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0	DTCR1 For ch1
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	
$2A2	DT15	DT14	DT13	DT12	DT11	DT10	DT9	DT8	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0	DTCR2 For ch2
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

9.13.5 Source Memory Address Register (SMAR)

This 24-bit register stores the DMA channel source memory address.

At transfer between memory and I/O, the register functions the same as ch1/2 memory address register.

After reset, the register is undefined.

Offset address	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	SMAR For ch0 (Upper)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	
$286	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	SMAR For ch0 (Lower)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

9.13.6 Destination Memory Address Register (DMAR)

This 24-bit register stores the DMA channel destination memory address.

This register is not used for transfer between memory and I/O.

Offset address	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$289									DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16	DMAR For ch0 (Upper)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	
$28A	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	DMAR For ch0 (Lower)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

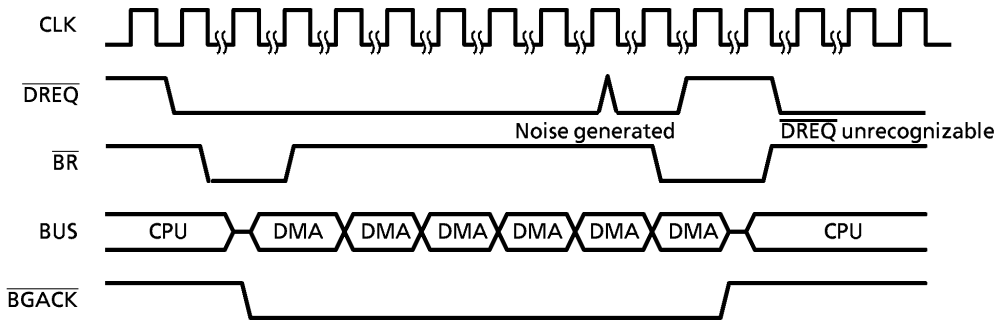
9.13.7 Memory Address Registers 1, 2 (MAR1, 2)

These 24-bit registers store the DMA channel source memory address.
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									A23	A22	A21	A20	A19	A18	A17	A16	MAR1 For ch1 (upper)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$296	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	MAR1 For ch1 (lower)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$2A4									A23	A22	A21	A20	A19	A18	A17	A16	MAR2 For ch2 (upper)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
$2A6	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	MAR2 For ch2 (lower)
Reset	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	

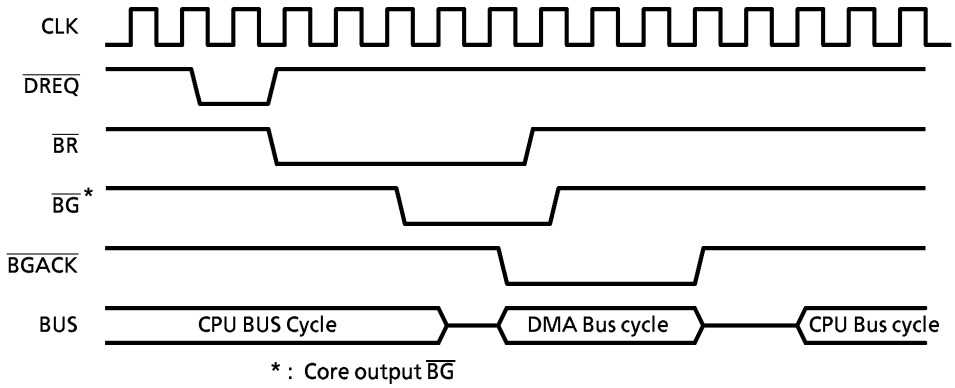
9.13.8 Cautions

1. At word access transfer, write even-numbered values in the source memory address register and the destination memory address register.
2. At single address transfer, use only the source memory address register.
3. When writing to registers, first set the RWE bit of the operation control register.
4. At DMA transfer, do not specify an assigned address in a DMAC register.
5. When an address error and a count error occur simultaneously, the count error in the status register is valid.
6. 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.

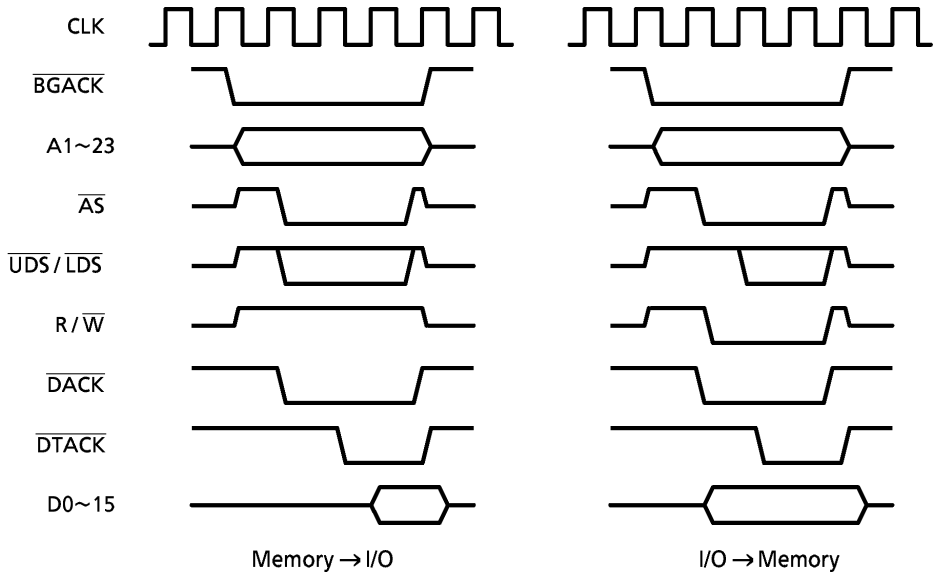


9.14 Bus Cycle Timing Chart

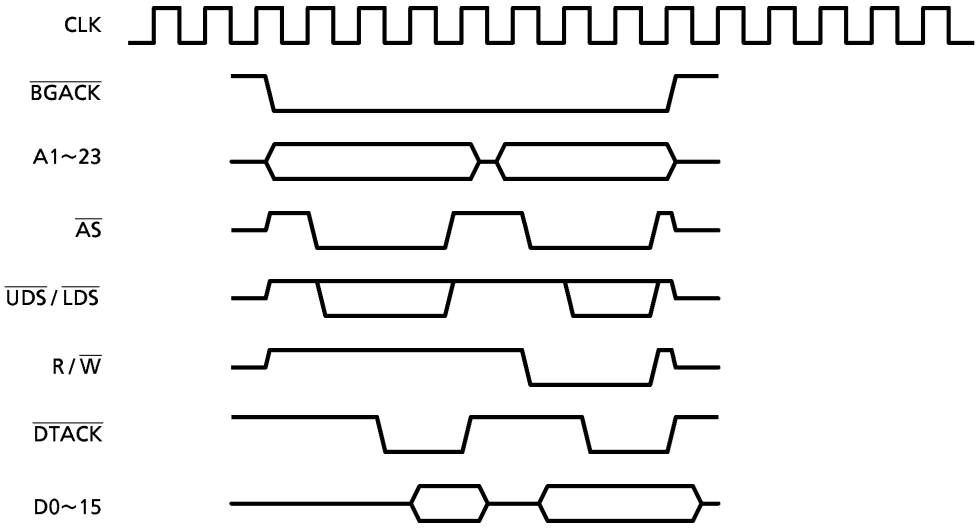
(1) DMAC bus arbitration



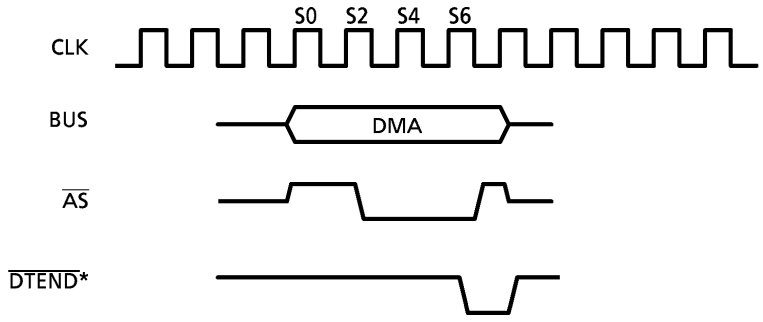
(2) Single address transfer



(3) Dual address transfer (memory to memory)



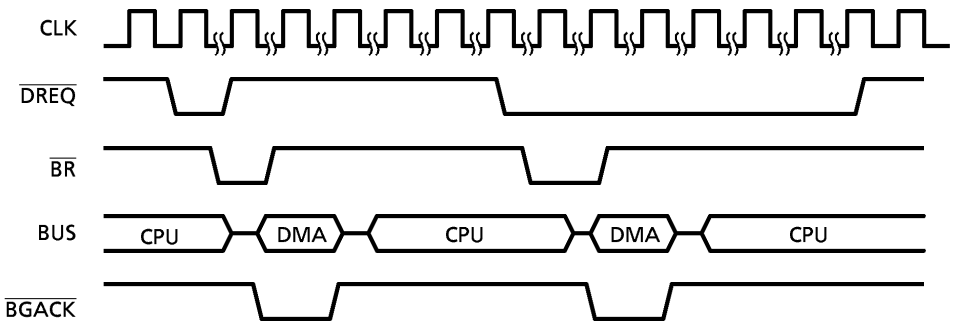
(4) Service complete



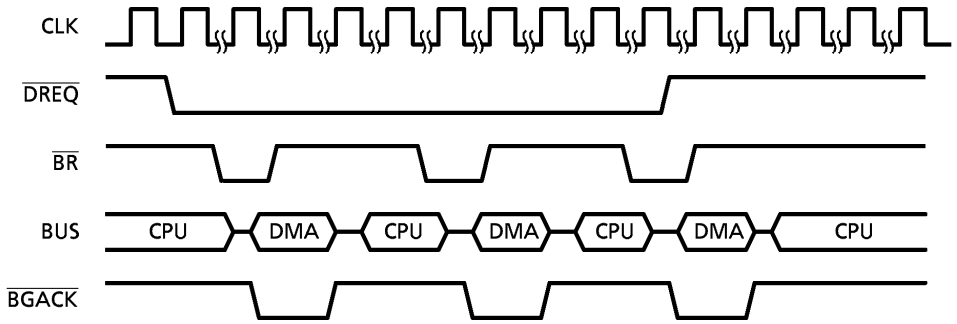
* Open drain output

9.14.1 Transfer Modes

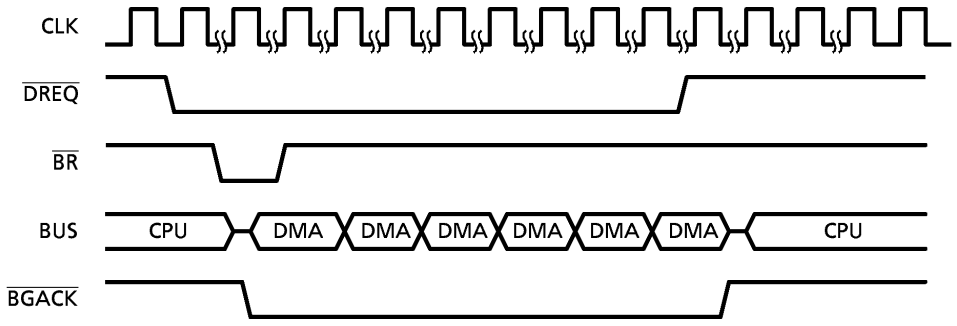
(1) Byte transfer



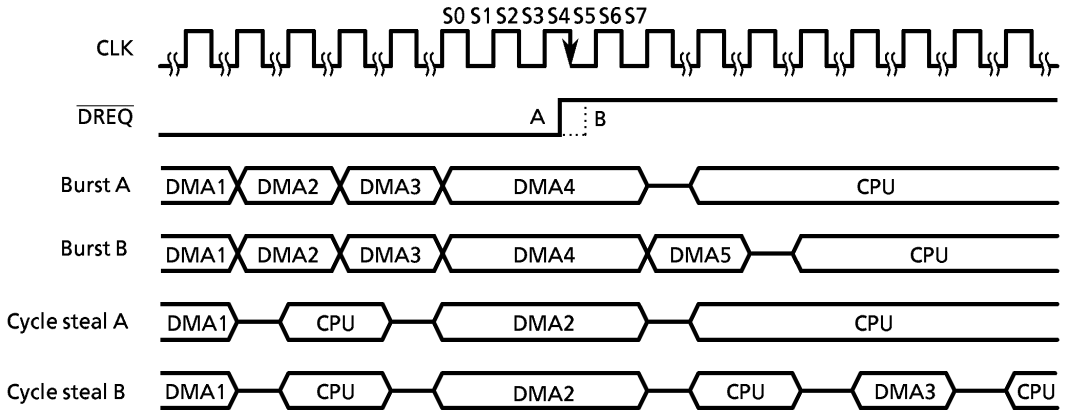
(2) Cycle steal transfer



(3) Burst transfer

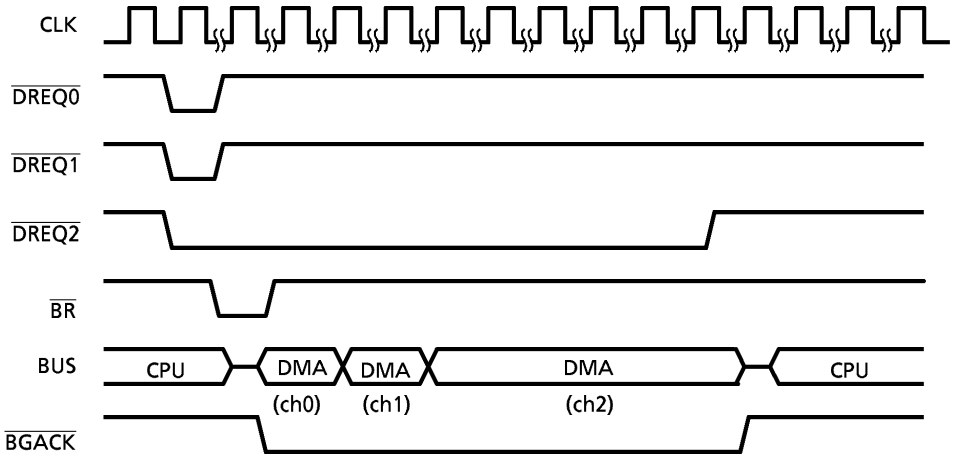


(4) Transfer suspended due to $\overline{\text{DREQ}}$ (at burst or cycle steal transfer)

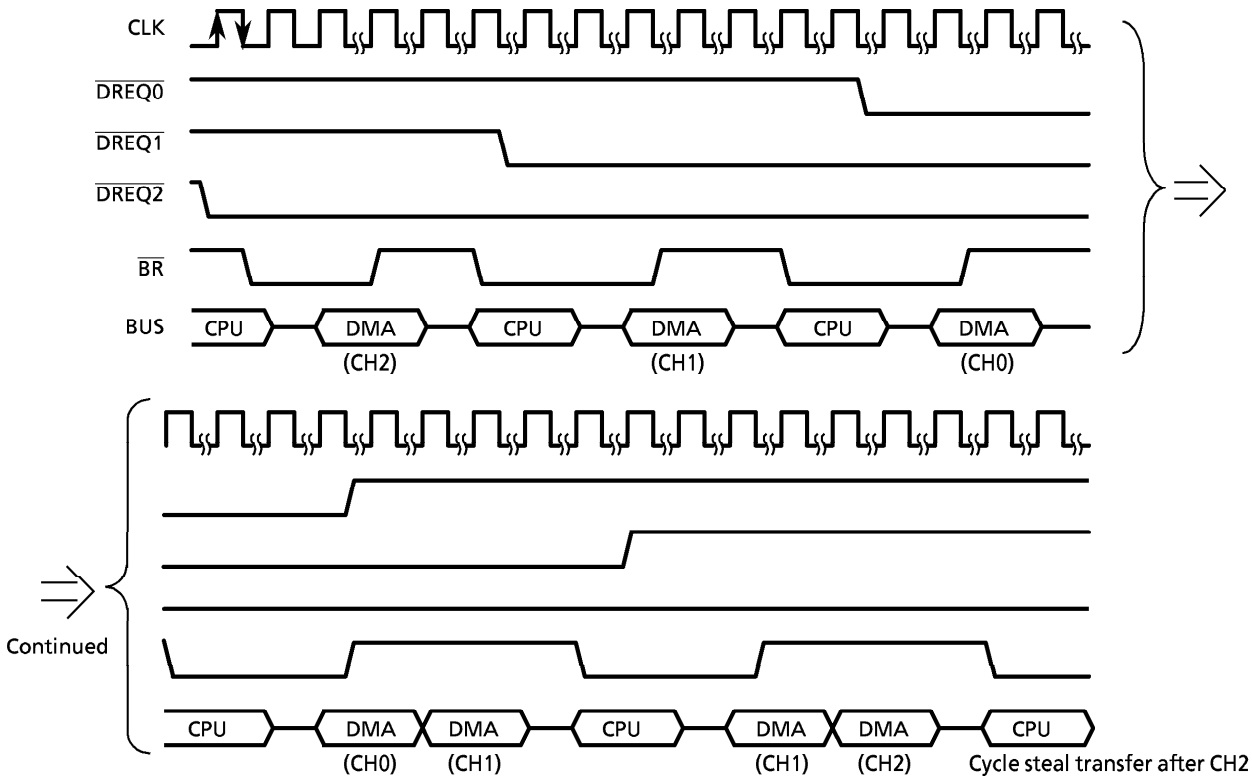


9.14.2 Multiple Requests and Priority

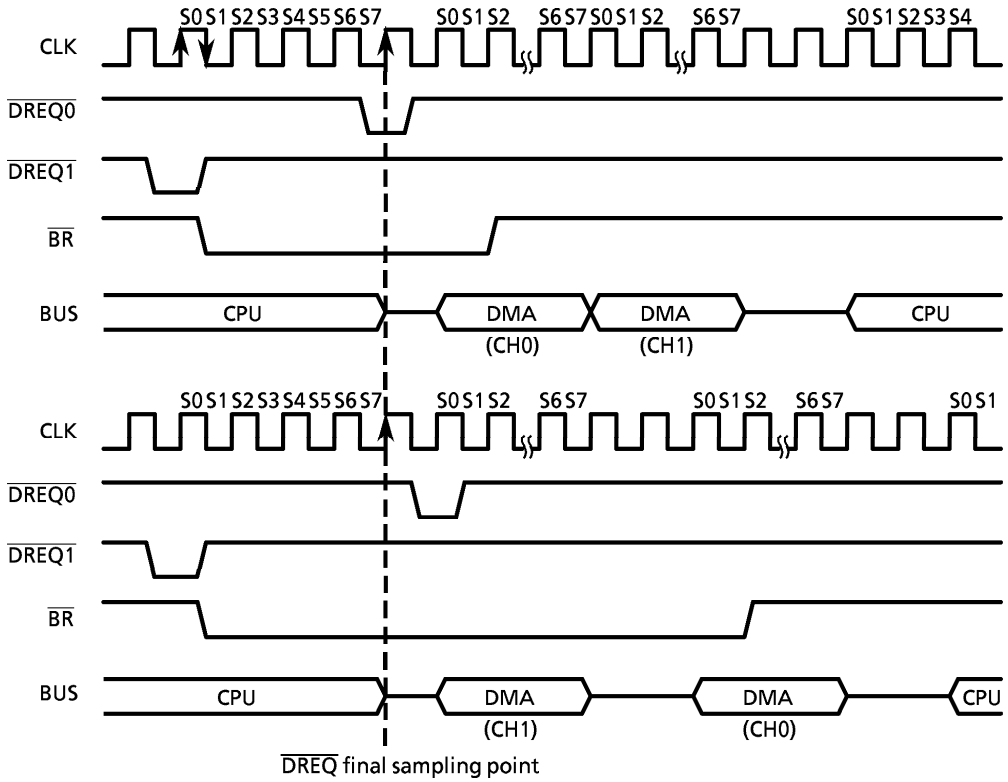
(1) Multiple requests



(2) Priority at cycle steal transfer

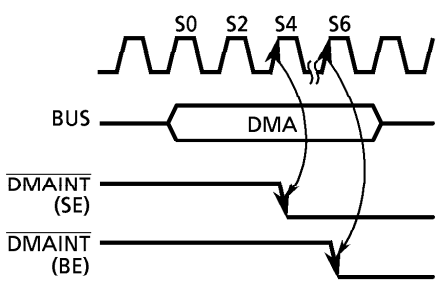


(3) Receive timing at multiple requests

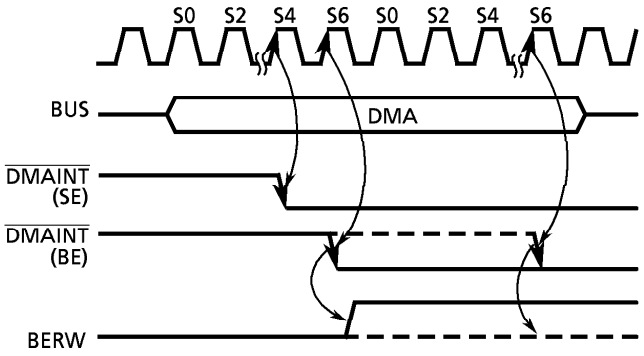


9.14.3 Interrupt Request Timings

(1) Service complete bus error

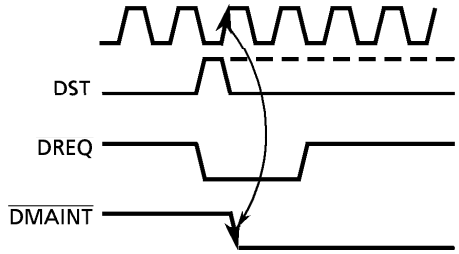


Single



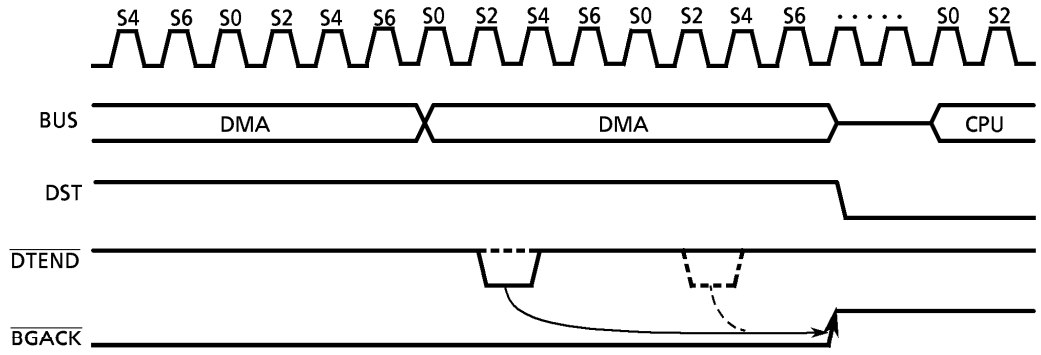
Dual

(2) Address error/count error



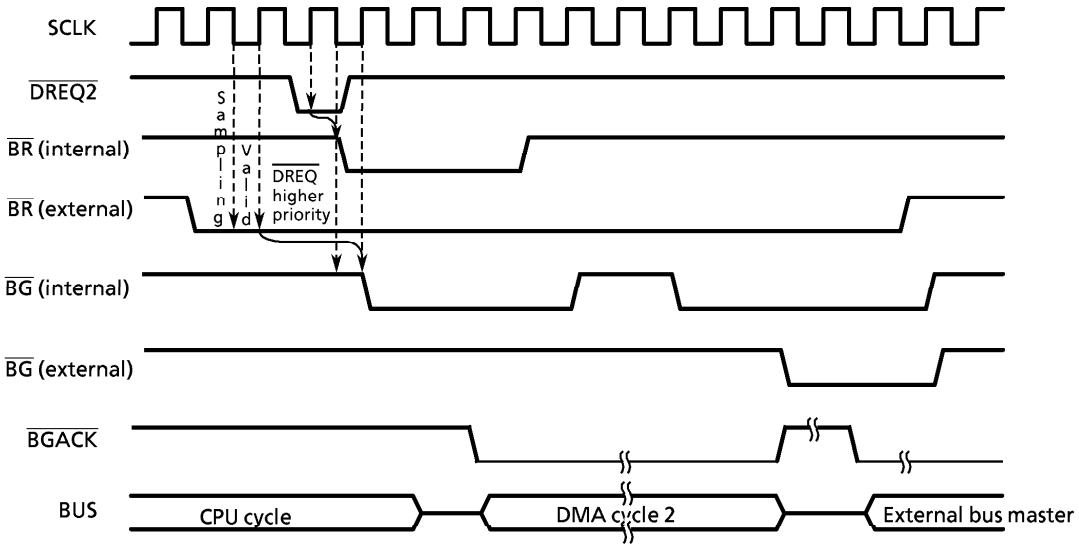
DST is cleared when error occurs.

(3) Suspension



Transfer suspended during dual address transfer

9.14.4 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 $\overline{BR}$.

10. DRAM Controller

The DRAM controller (henceforth, "DRAMC") can generate the control signals ($\overline{RAS}$, $\overline{CAS}$) needed for interfacing with the DRAM. In the DRAM read/write cycle, DRAMC can output, via the internal address multiplexer, row/column addresses to A1 to A11.

In addition, the $\overline{RAS}$ and $\overline{CAS}$ signals are multiplexed with the chip select signals ($\overline{CS0}$, $\overline{CS1}$) from the internal address decoder. When the DRAMC is not in use, the pin functions as a chip select pin. When the refresh and write cycles overlap in DRAM of 4M bits or larger, some may be set to test mode. Therefore, during the refresh cycle, an external circuit is required to set the R/$\overline{W}$ signal to high.

Generate dummy cycle.

Refresh method	:	CAS before RAS refresh
Refresh cycle	:	80 to 288 states selectable
Refresh cycle length	:	2 to 9 states selectable
Memory access cycle length	:	0 to 7 waits (when internal $\overline{DTACK}$ in use) selectable *1
Memory access address length	:	9, 10, 11- bit length selectable

*1 : Specify in address decoder (ARCR0).
*2 : Internal signal
Note : 1 state = 1 system clock (60ns at 16.67MHz)

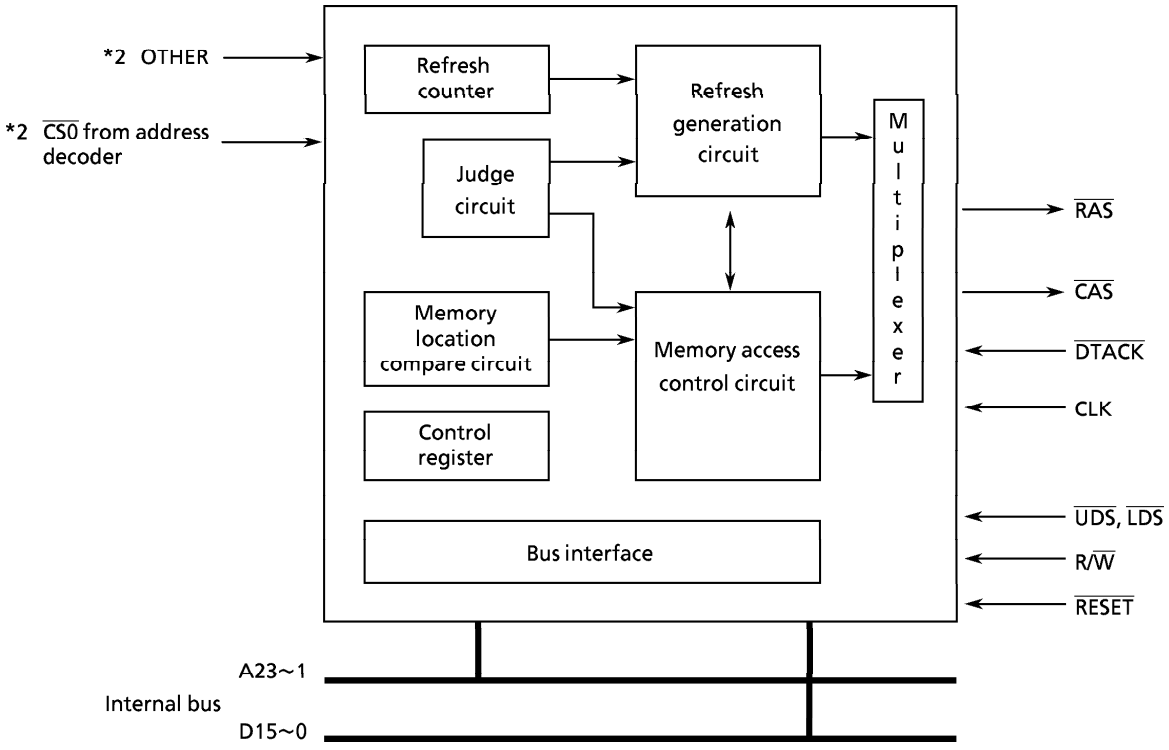


Figure 10.1 Block Diagram

10.1 DRAM Initialization

The DRAMC can generate the number of consecutive CAS before RAS dummy cycles (four clocks) required for initialization at DRAM power on. The dummy cycle is controlled by bit 7 (RINI) of the refresh control register. The dummy cycle is asynchronous with the CPU cycle.

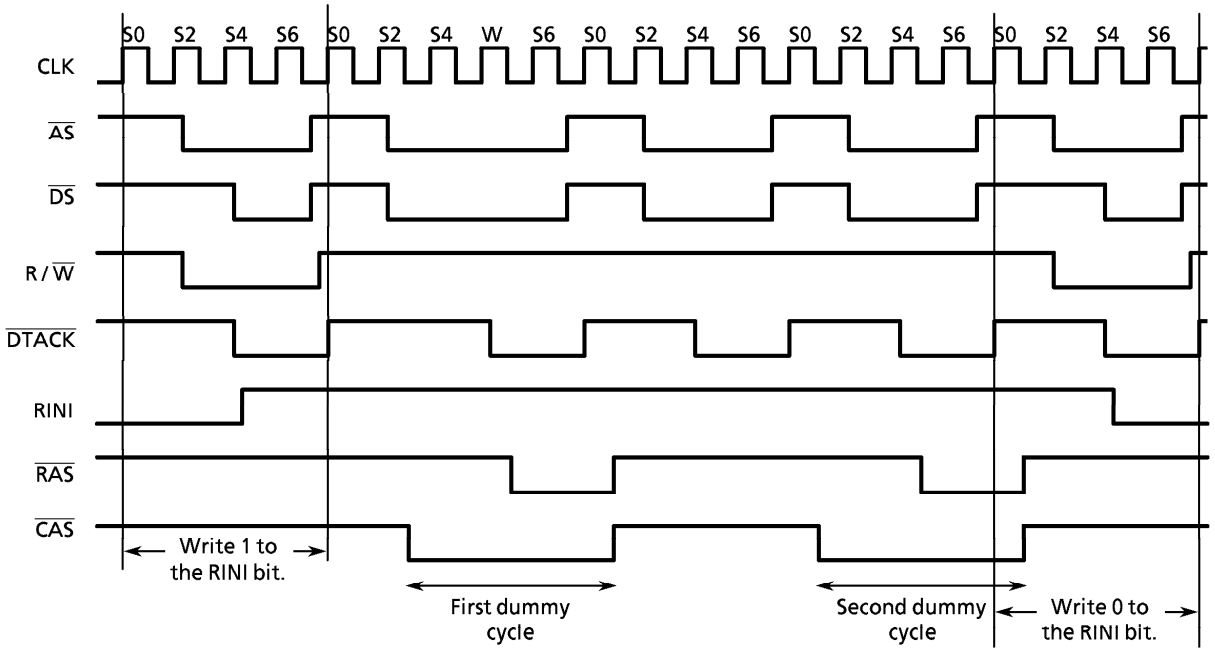


Figure 10.2

10.2 DRAM Refresh

The DRAMC can issue refresh cycles at the intervals set in the refresh control register. Refresh cycles of 80, 112, 144, 176, 208, 224, 256, or 288 clocks can be selected. Refresh cycle lengths of 2 to 9 clocks can also be selected. The refresh cycle is asynchronous with the CPU cycle.

※ Because the refresh follows the $\overline{CAS}$ before $\overline{RAS}$ method, the internal DRAM refresh address counter is used. Consequently, TMP68303F does not output a refresh address.

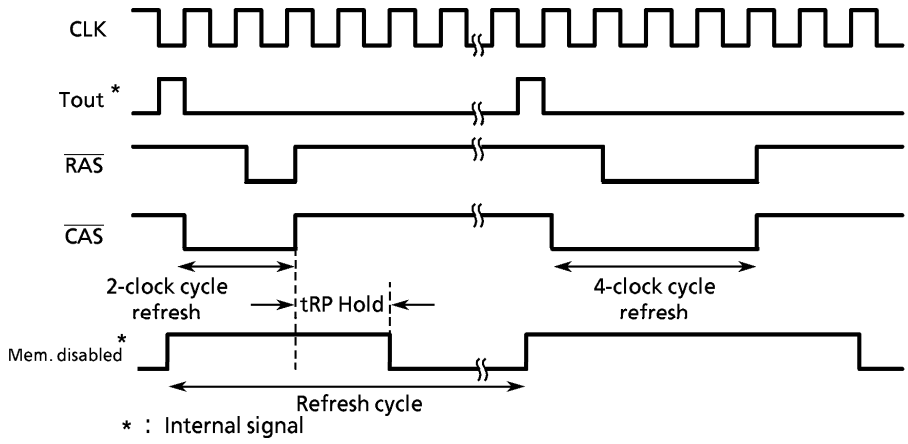


Figure 10.3

10.3 DRAM Read/Write

When memory area 0 specified by the internal address decoder is accessed, the DRAMC outputs a valid signal ($\overline{RAS}$, $\overline{CAS}$) to DRAM. At this time, set the MC bit of DRAM controller register MCR to 1, enable the EN bit of address decoder register ARCR0, and allocate DRAM to the CS0 area. When the DRAM controller is in use, the CS1 area and ARCR1 internal $\overline{DTACK}$ cannot be used.

The $\overline{CAS}$ and $\overline{RAS}$ pulse widths are determined by the number of $\overline{DTACK}$ waits. Specifying bit 3 (RCAWS) of the memory control register synchronizes the $\overline{CAS}$ assert with the $\overline{DTACK}$ assert.

The DRAMC can output multiplexed row and column addresses to A1 to A9 (A10/A11) during the read/write cycle. At this time, the DRAMC outputs normal addresses to upper addresses A10 (A11/A12) to A23.

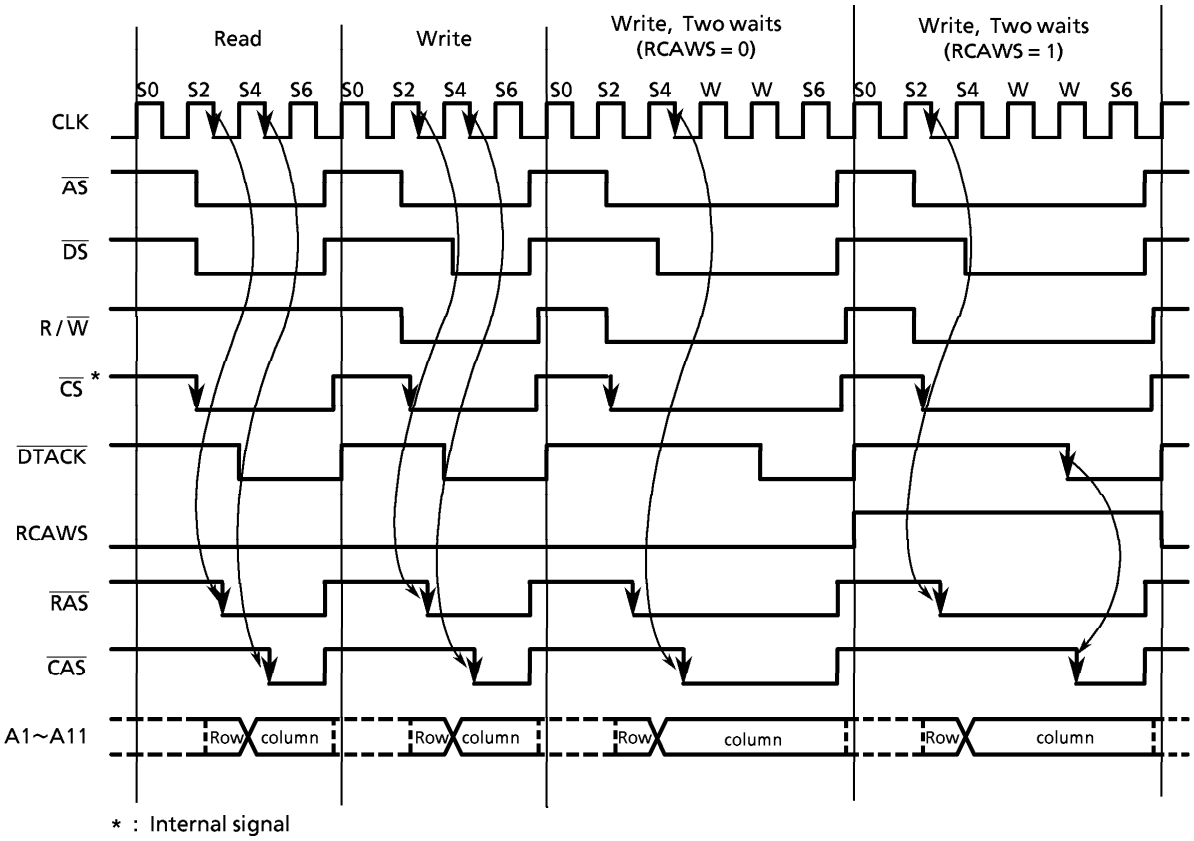


Figure 10.4 DRAM Read/Write Cycle

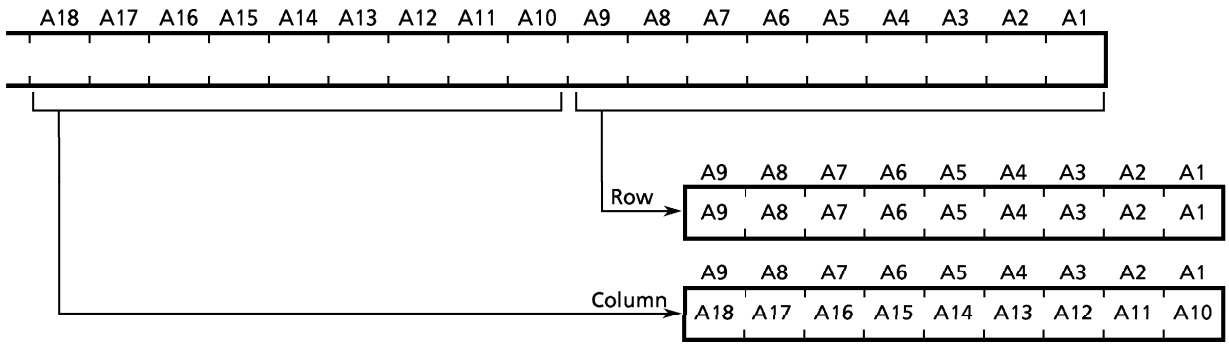


Figure 10.5 Address Multiplex (For 9-Bit Address)

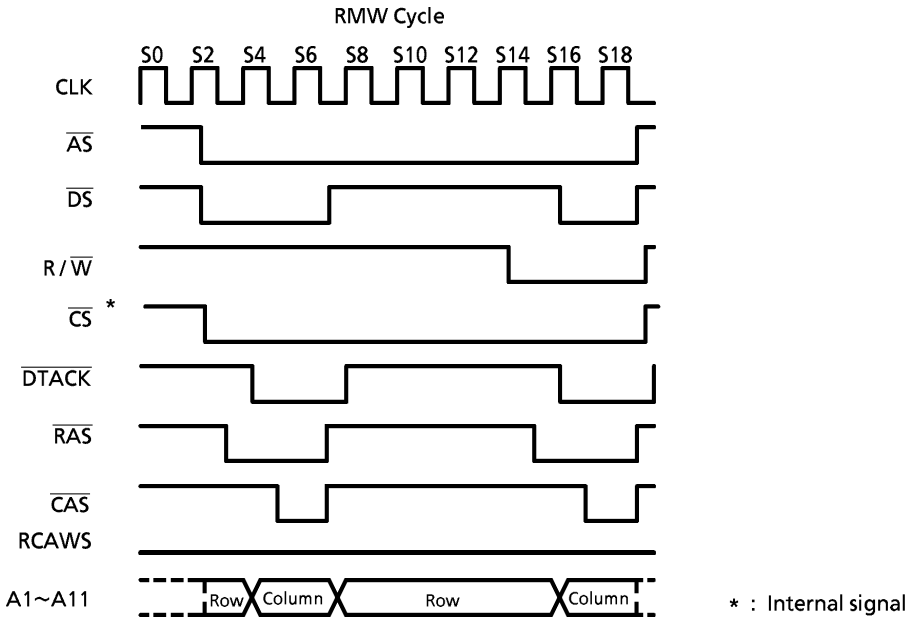


Figure 10.5

10.4 Priority

As refresh requests to DRAM are asynchronous with the processor bus cycle, these may overlap with read/write requests to DRAM. In such a case, the request asserted first receives the higher priority from the DRAMC. When processing a refresh request before a memory access cycle, the DRAMC automatically inserts one or more waits (specified in the address decoder, when internal $\overline{\text{DTACK}}$ is set) in the memory access cycle. Using an external $\overline{\text{DTACK}}$ requires additional external circuit.

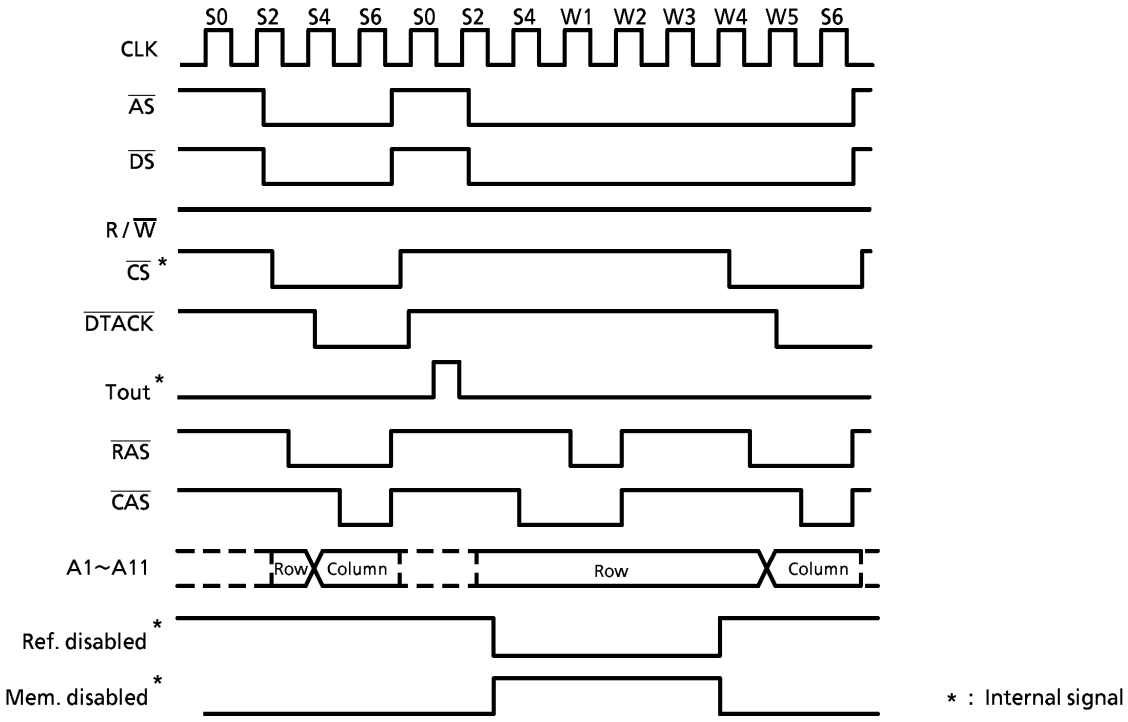


Figure 10.6

10.5 Register Configuration

10.5.1 Refresh Control Register (RCR)

This register controls the refresh operation.

Offset address $301 Reset	7	6	5	4	3	2	1	0	
	RINI	RS2	RS1	RS0	RW2	RW1	RW0	RC	RCR
	0	0	0	0	0	0	0	0	

RINI : Dummy cycle control

0 : Disables dummy cycle.

1 : Enables dummy cycle.

RS2, 1, 0 : Set refresh cycle.

Setting			Refresh cycle	Refresh cycle times vs operating frequencies (μs)				
RS2	RS1	RS0	(Clock)	8MHz	10MHz	12.5MHz	16MHz	16.67MHz
0	0	0	80	10	8	6.4	5	4.8
0	0	1	112	14	11.2	8.96	7	6.72
0	1	0	144	18	14.4	11.5	9	8.64
0	1	1	176	22	17.6	14.1	11	10.6
1	0	0	208	26	20.8	16.6	13	12.5
1	0	1	224	28	22.4	17.9	14	13.4
1	1	0	256	32	25.6	20.5	16	15.4
1	1	1	288	36	28.8	23.0	18	17.3

RW2, 1, 0 : Set refresh cycle.

Setting			Refresh cycle
RW2	RW1	RW0	length
0	0	0	2 clocks
0	0	1	3 clocks
0	1	0	4 clocks
0	1	1	5 clocks
1	0	0	6 clocks
1	0	1	7 clocks
1	1	0	8 clocks
1	1	1	9 clocks

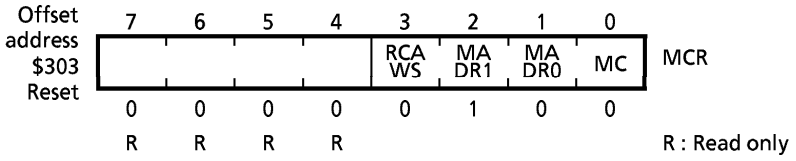
RC : Refresh operation control

0 : Disables refresh.

1 : Enables refresh.

10.5.2 Memory Control Register (MCR)

This register controls DRAM read/write.



RCAWS : $\overline{CAS}$ assert timing setting

0 : Normal

1 : Synchronizes with $\overline{DTACK}$.

MADR1, 0 : Refresh address length setting

MADR1	MADR0	Address length
0	0	Single address
0	1	9 bits
1	0	10 bits
1	1	11 bits

MC : DRAMC memory access control

0 : Disables DRAMC memory access.

1 : Enables DRAMC memory access.

(To permit $\overline{CAS}$ and $\overline{RAS}$ output, the $\overline{CS0}$ and $\overline{CS1}$ functions are disabled. The internal $\overline{DTACK}$ function for the $\overline{CS1}$ area is also disabled.)

10.6 Cautions

1. When only the internal $\overline{DTACK}$ generated by the address decoder is used for DRAM read/write cycles, and the DRAM controller is operating in $\overline{CAS}$ assert $\overline{DTACK}$ synchronous mode with zero wait states, the $\overline{CAS}$ pulse width may be abnormally short.
2. When both $\overline{CS0}$ and CLK are low, the $\overline{RAS}$ assert is internally generated . At the final stage, $\overline{RAS}$ is output externally with $\overline{DS}$. Accordingly, if $\overline{DS}$ input from an external source in emulator mode is delayed (the same applies to the external bus master operating in normal mode), the $\overline{RAS}$ assert is delayed to the same extent. Therefore, be sure to follow the 68000 AC specifications.
3. If the emulator or external bus master use the DRAM controller to generate the read/write DRAM cycle, addresses input externally and the column addresses output by TMP68303F cause bus contention. Take care to avoid this.
   (See Chapter 13, Development Environment.)

10.7 Signal Relations at DRAM access

Read cycle

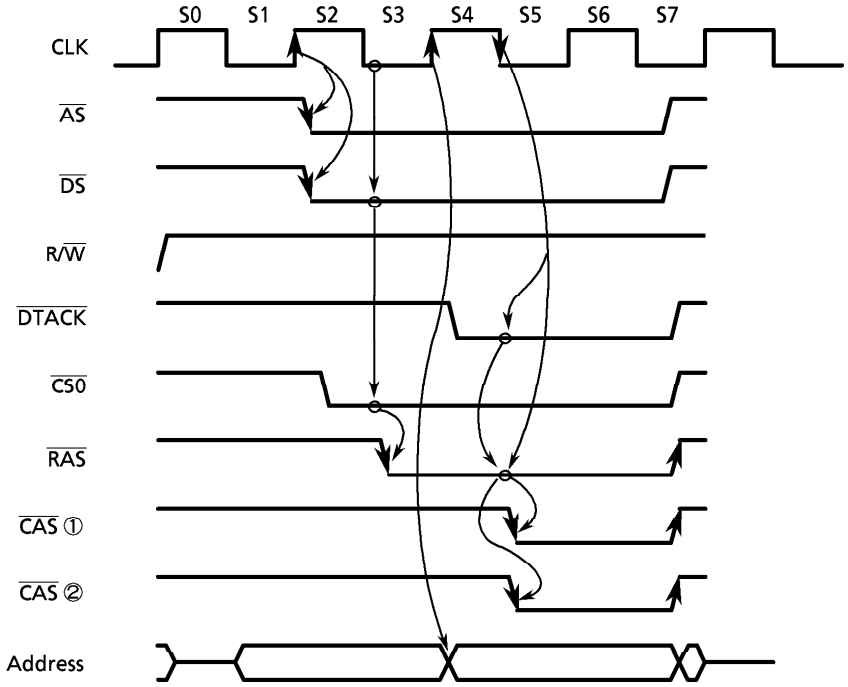


Figure 10.7

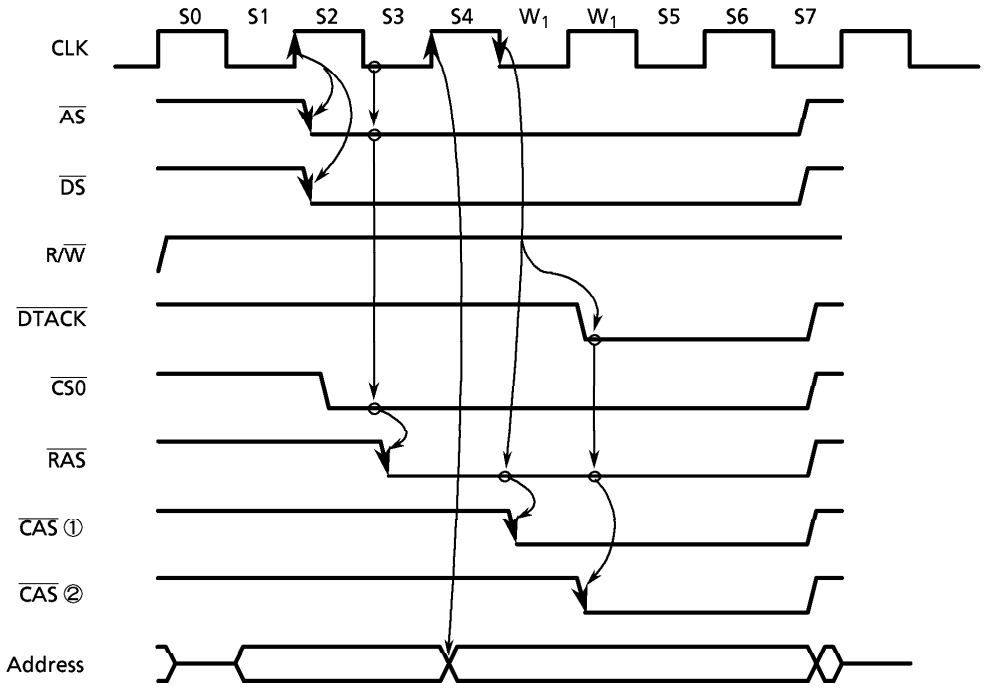


Figure 10.8

When $\overline{DS}$ (high) or $\overline{CS0}$ (high) is negated, both $\overline{RAS}$ and $\overline{CAS}$ are negated in the read cycle.
※ Synchronizes $\overline{CAS}$ ① = normal; $\overline{CAS}$ ② = $\overline{DTACK}$.

Write cycle

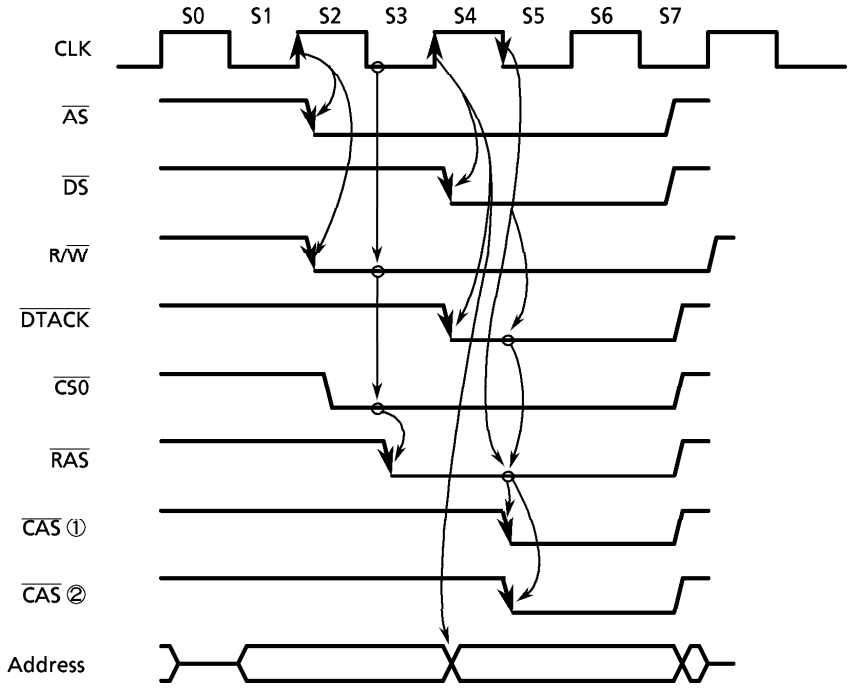


Figure 10.9

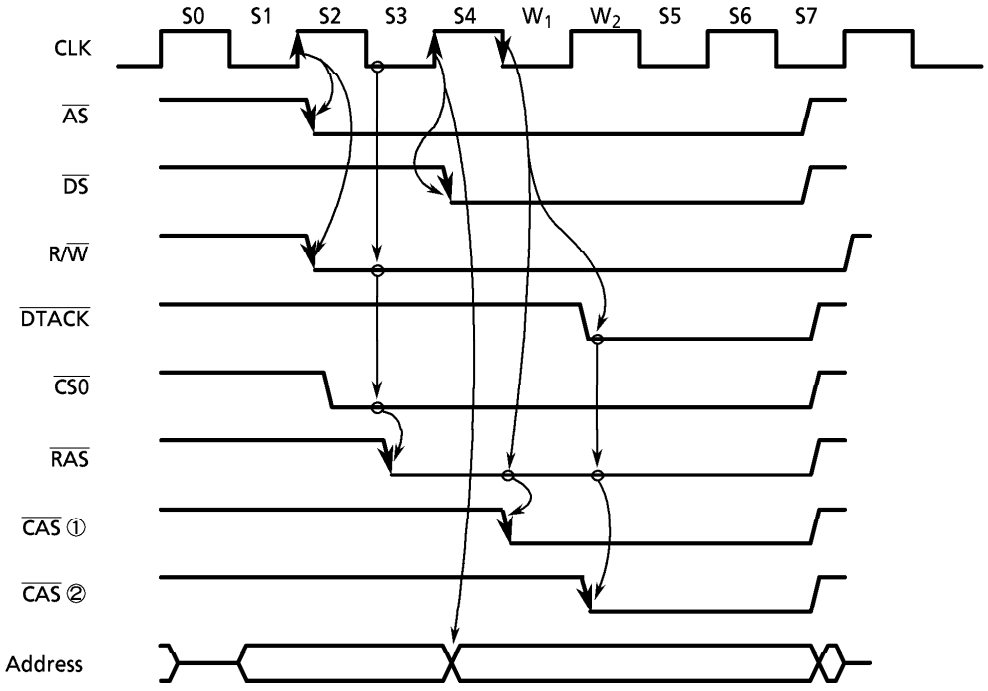


Figure 10.10

$\overline{RAS}$ and $\overline{CAS}$ negation in write cycle is $\overline{CS0}$ (high).

10.8 DRAM Access Timing Examples

Figure number	Bus cycle	Read/write	Contention with refresh	Address decoder internal $\overline{\text{DTACK}}$	I/O $\overline{\text{DTACK}}$	$\overline{\text{CAS}}$ assert in sync with $\overline{\text{DTACK}}$
10.11	CPU	Read	No	0 wait	—	×
10.12	CPU	Read	No	1 wait	—	×
10.13	CPU	Read	No	0 wait	—	○
10.14	CPU	Read	No	1 wait	—	○
10.15	CPU	Write	No	0 wait	—	×
10.16	CPU	Write	No	1 wait	—	×
10.17	CPU	Write	No	0 wait	—	○
10.18	CPU	Write	No	1 wait	—	○
10.19	CPU	Read	yes	0 wait	—	×
10.20	CPU	Read	yes	1 wait	—	×
10.21	CPU	Write	yes	0 wait	—	×
10.22	CPU	Write	yes	1 wait	—	×
10.23	DMA	Read	No	0 - 3 waits *1	3 wait	×
10.24	DMA	Read	No	0 - 3 waits *1	3 wait	○
10.25	DMA	Write	No	0 - 3 waits *1	3 wait	×
10.26	DMA	Write	No	0 - 3 waits *1	3 wait	○
10.27	DMA	Read	yes	0 wait	3 wait	×
10.28	DMA	Read	yes	0 wait	3 wait	○
10.29	DMA	Write	yes	1 wait	3 wait	×
10.30	DMA	Write	yes	1 wait	3 wait	○

*1) At single address transfer, 3 waits is valid for the I/O side; thus, regardless of the wait setting (from 0 to 3), 3 waits is used.

1. CPU cycle

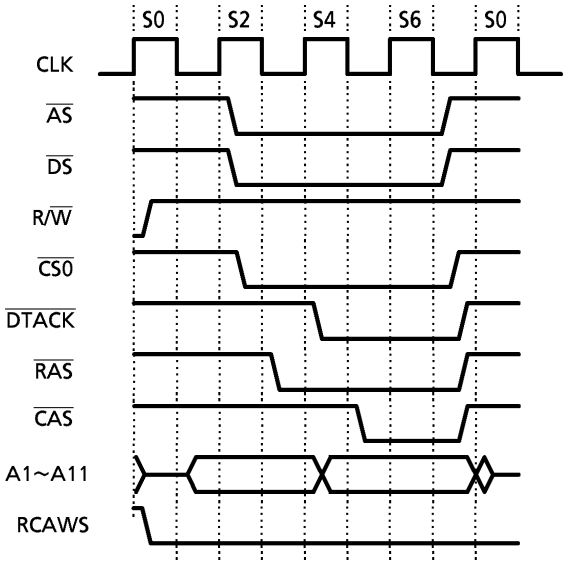


Figure 10.11

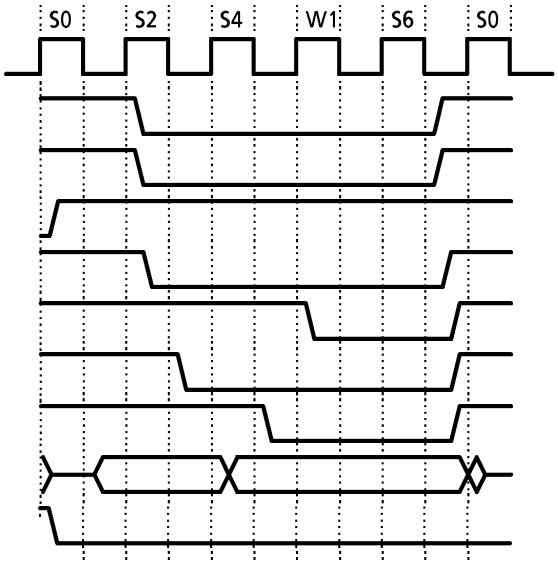


Figure 10.12

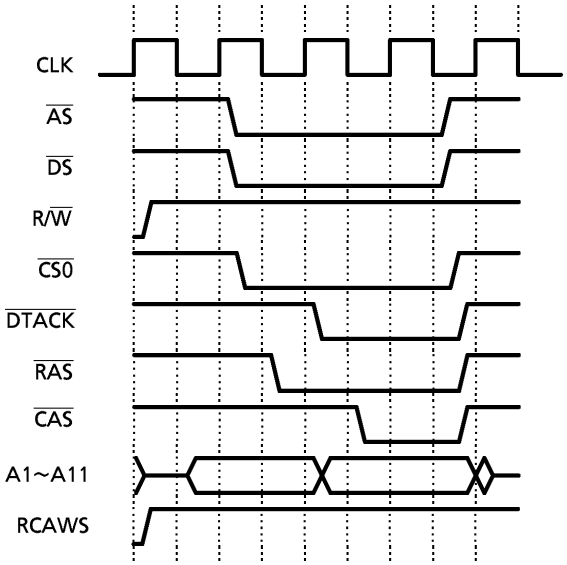


Figure 10.13

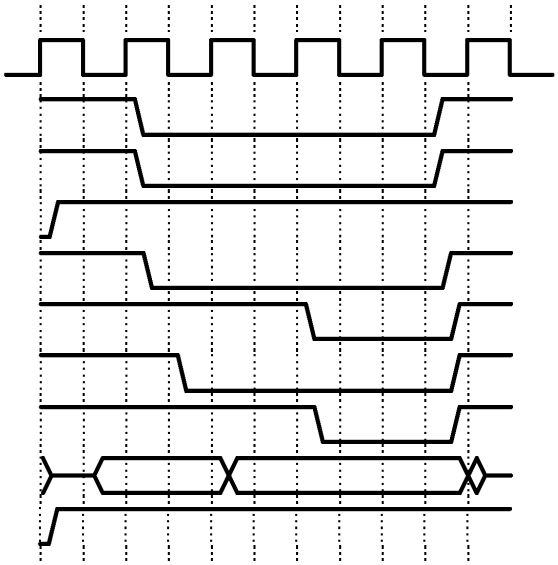


Figure 10.14

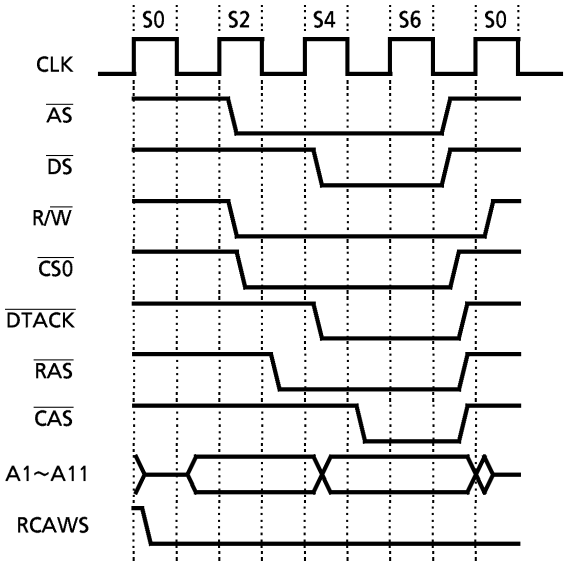


Figure 10.15

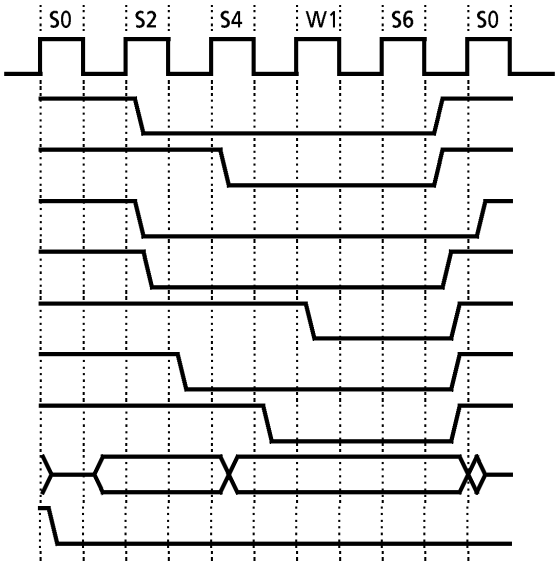


Figure 10.16

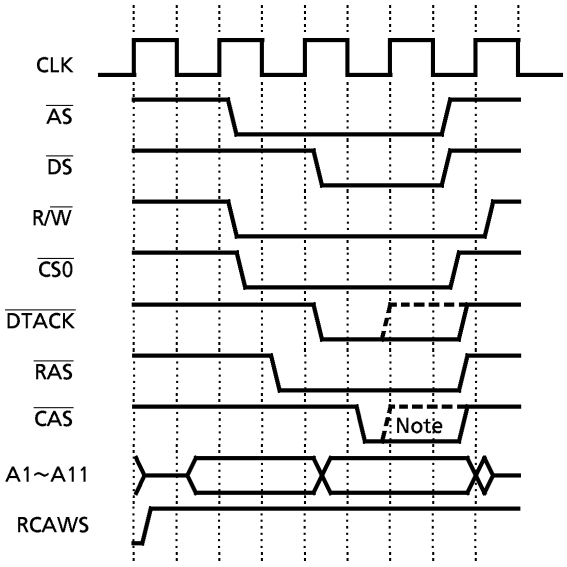


Figure 10.17

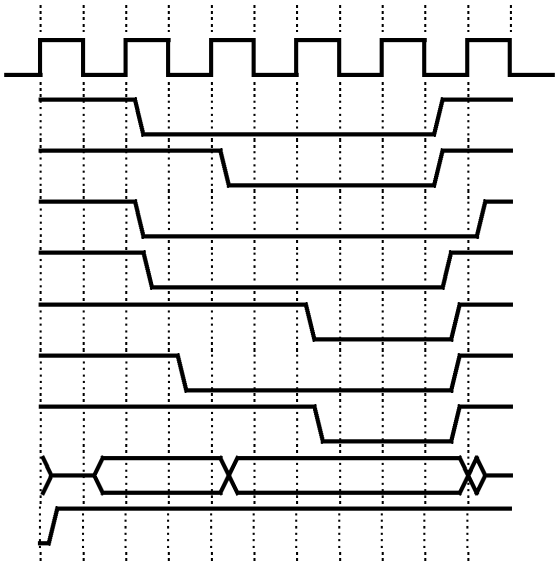


Figure 10.18

Note : Internal 0 wait
Write cycle
$\overline{CAS}$-$\overline{DTACK}$ sync mode
The $\overline{CAS}$ pulse width may become abnormal.

No problem with 1 wait.

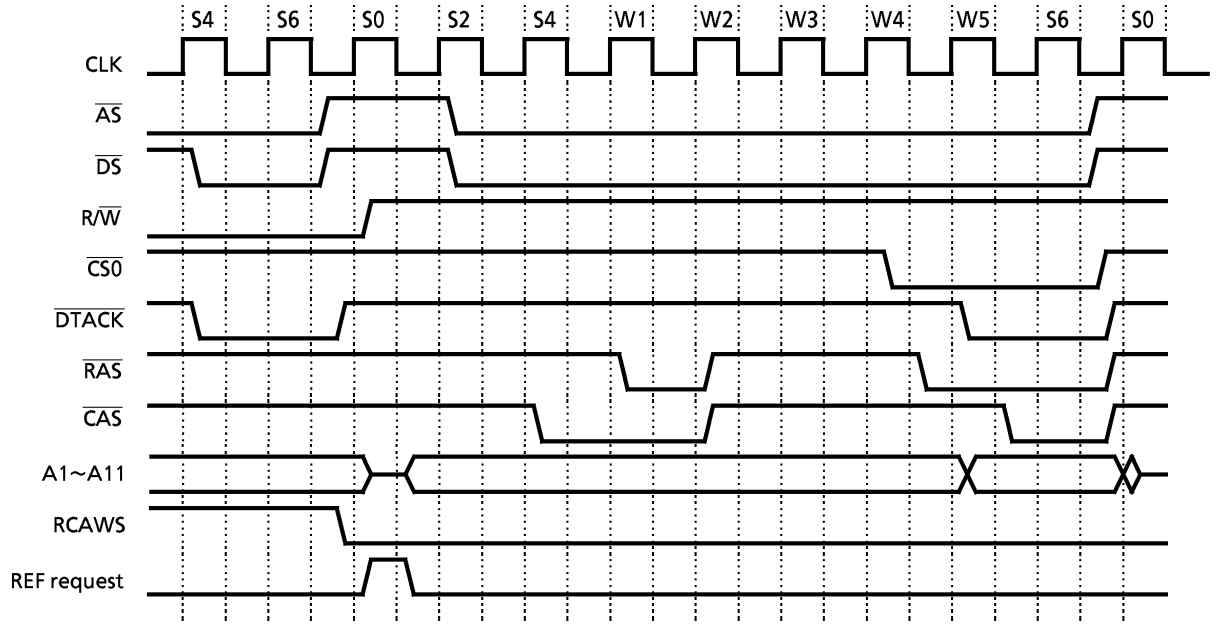


Figure 10.19

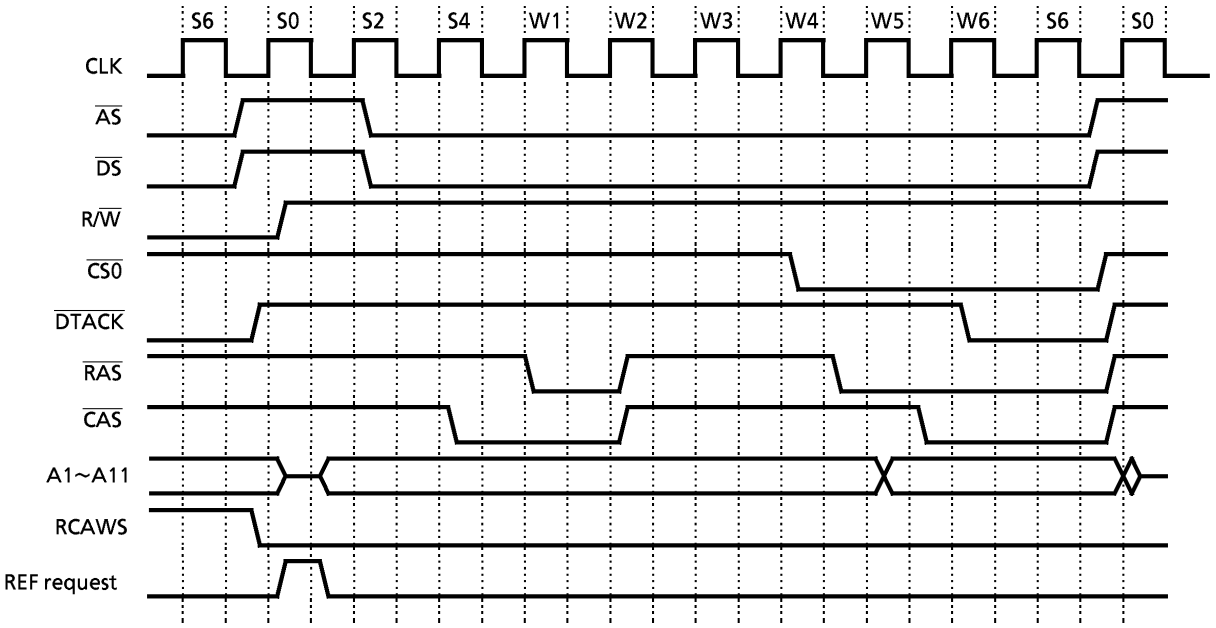


Figure 10.20

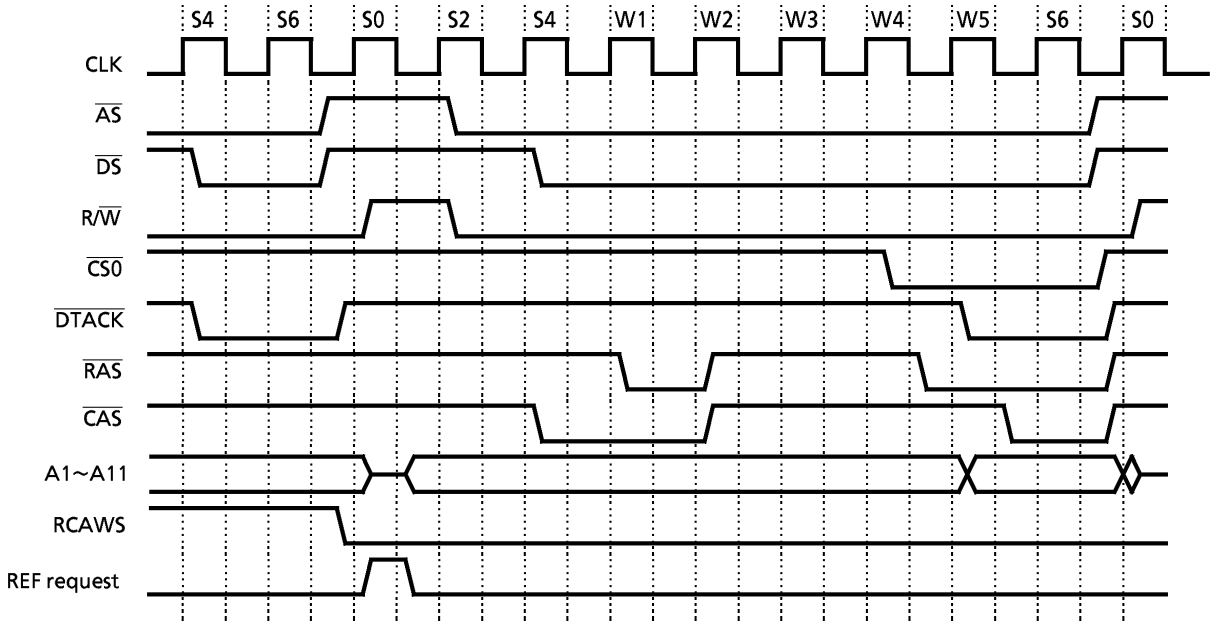


Figure 10.21

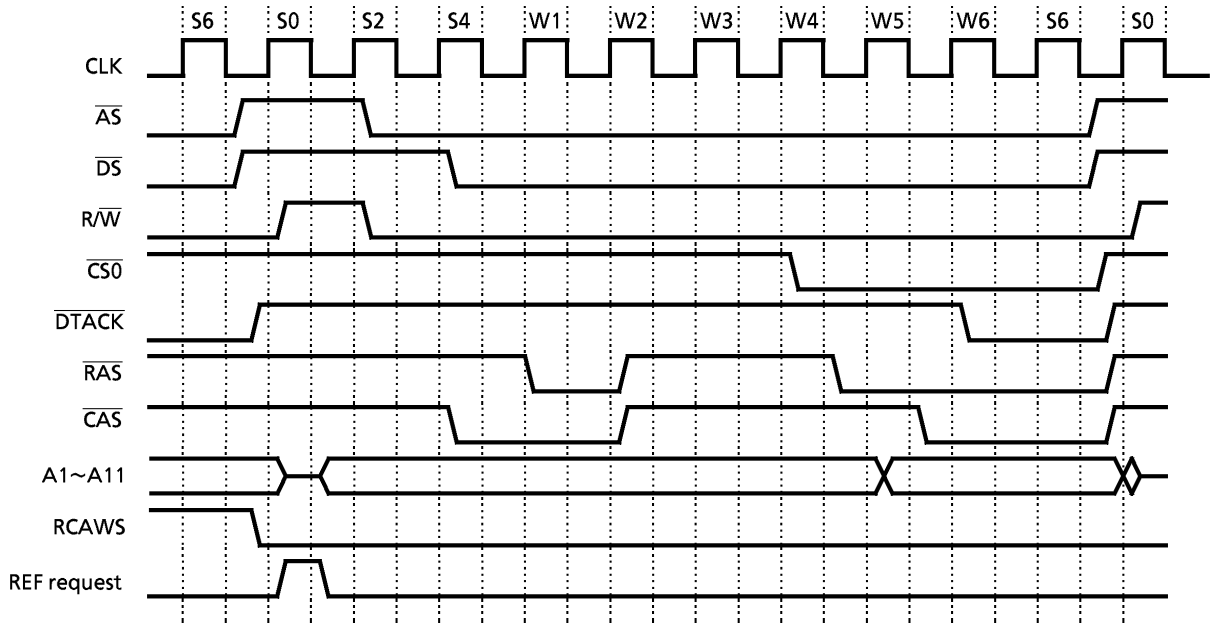


Figure 10.22

2. DMA cycle

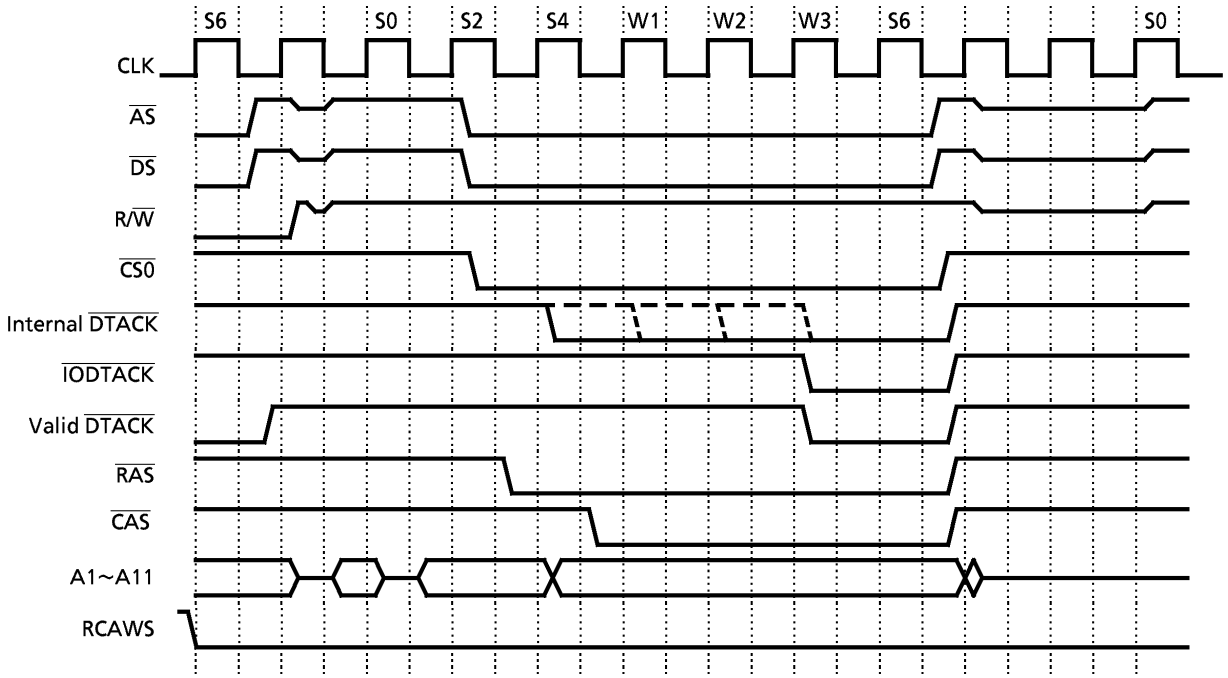


Figure 10.23

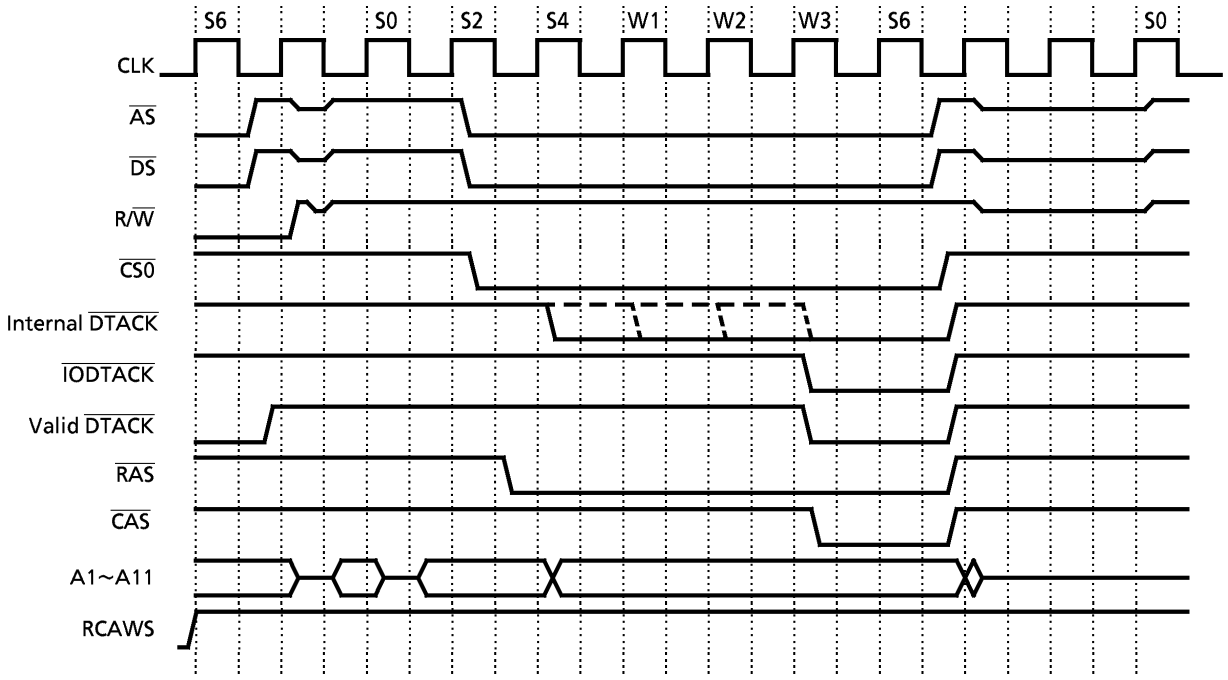


Figure 10.24

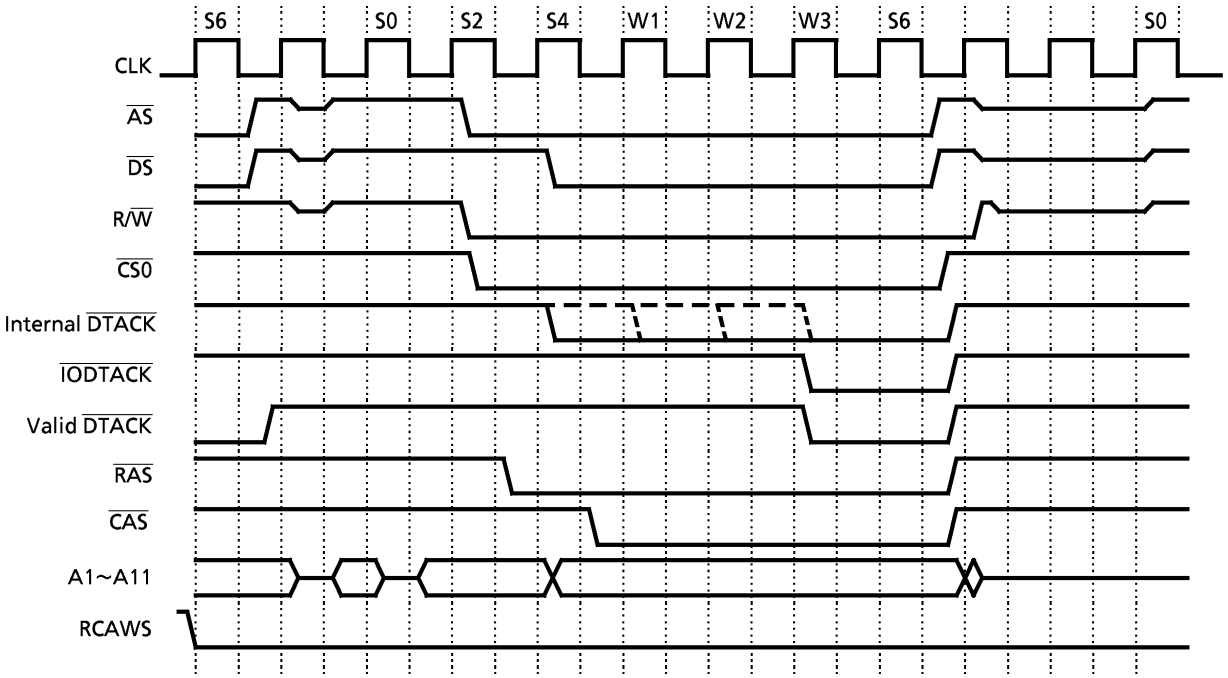


Figure 10.25

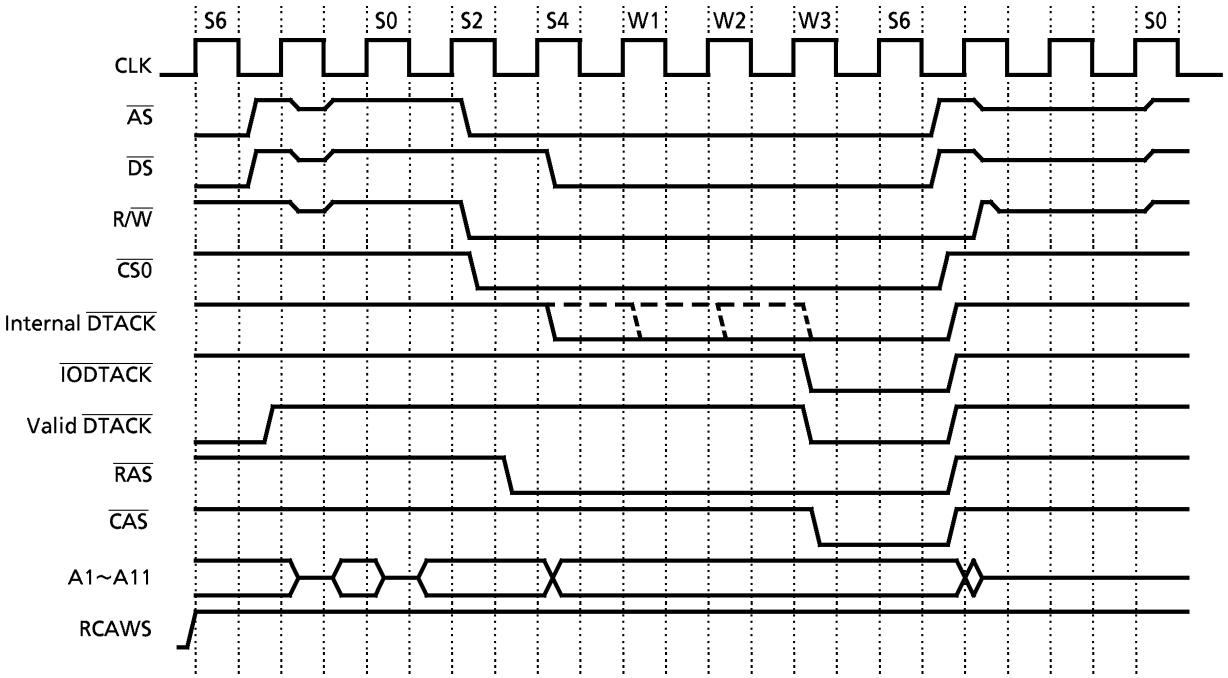


Figure 10.26

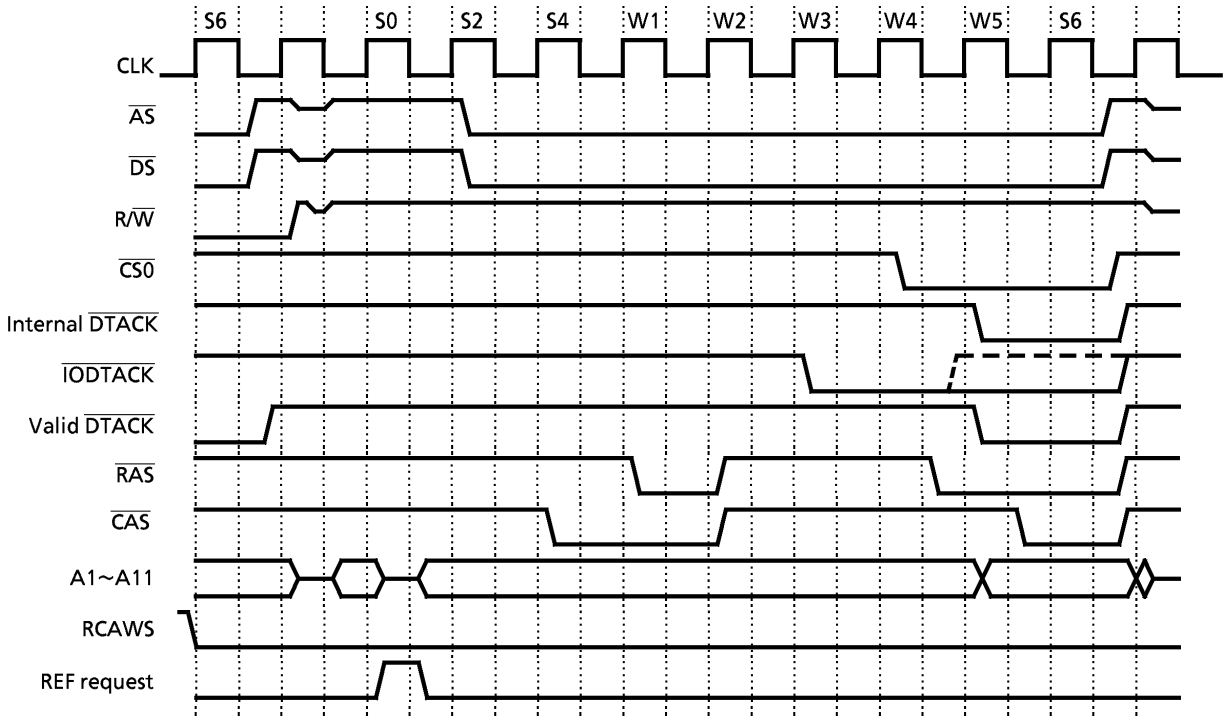


Figure 10.27

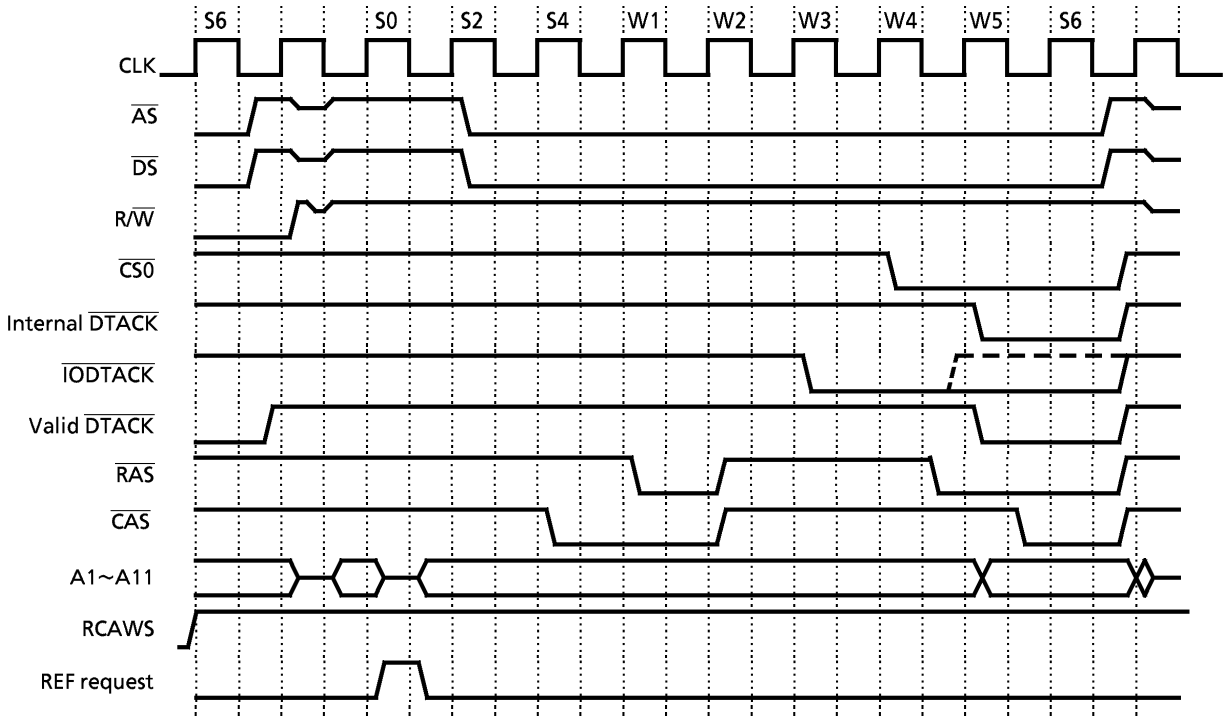


Figure 10.28

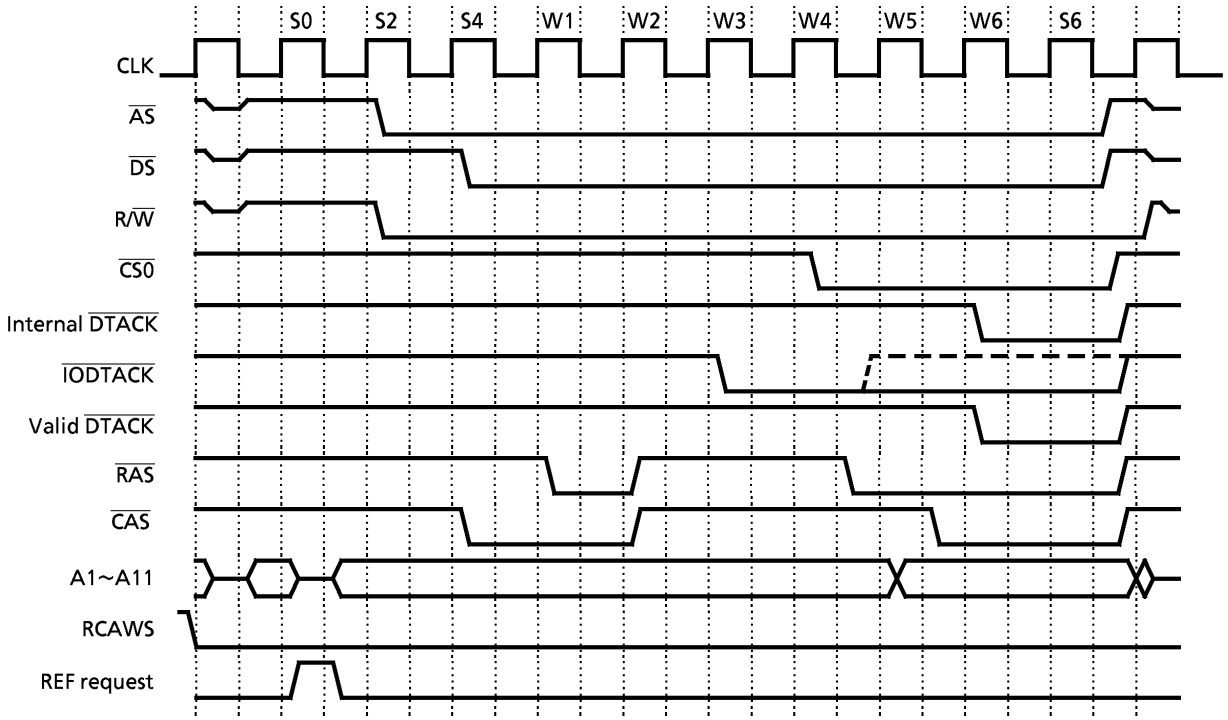


Figure 10.29

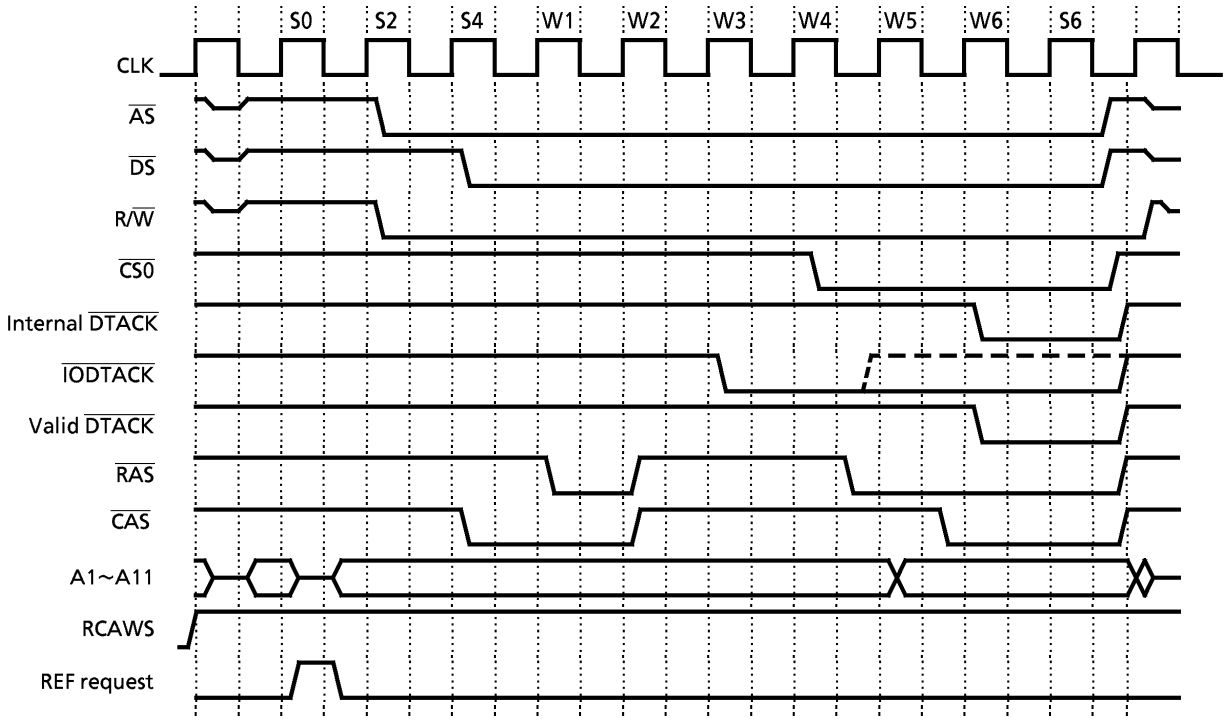


Figure 10.30

11. Stepping Motor Controller

TMP68303 incorporates two channels (M0 and M1) for hardware stepping motor control ports (henceforth, "SMC ports"), which are linked to the 8-bit timers. The SMC ports also serve as 4-bit I/O ports P0 and P1.

Channel 0 (M0) updates output synchronously with the timer flip-flop TFF3 trigger signal. Channel 1 (M1) updates output synchronously with the timer flip-flop TFF4 trigger signal.

The SMC ports are controlled by three control registers (P01CR, SMCMR, and SMCRCR). Drive methods are selectable between 4-phase 1-step/2-step excitation and 4-phase 1-2 step excitation.

Two-channel hardware stepping motor I/F
For 4-phase motor (1/2 or 1-2 step excitation)
Normal rotation/reverse rotation motor control
Variable pulse cycle
Connected to internal 8-bit timers (2 channels)

11.1 Register Configuration

11.1.1 Port P0 and P1 Input/Output Specification Register (P01CR)

This register specifies the input/output for each bit of 4-bit I/O ports P0 and P1. At reset, all bits of the register are cleared to 0; ports P0 and P1 function as input ports.

This register is write-only and cannot be read.

Offset address	7	6	5	4	3	2	1	0	
$337	P13C	P12C	P11C	P10C	P03C	P02C	P01C	P00C	P01CR
Reset	0	0	0	0	0	0	0	0	
	W	W	W	W	W	W	W	W	

W : Write only

P13C-P10C : Port 1 I/O specification
P03C-P00C : Port 0 I/O specification

0 : Input
1 : Output

Note: After this register is read, it is always set to $FF.

11.1.2 SMC Port Mode Register (SMCMR)

Ports P0 and P1 also function as SMC ports (M0, M1) and timer output ports (TOUT3, TOUT4) as selected by P0C and P1C.

To use port 0 as an SMC port (M0), set P0C to 10 or 11.

To use port 1 as an SMC port (M1), set P1C to 10 or 11.

The SM0M and SM1M bits select the excitation method. Setting the bits to "10" selects full-step (1-step/2-step) excitation. Setting the bits to "11" selects half-step (1-2 step) excitation. When full-step is selected, the initial output value determines the 1-step excitation or 2-step excitation option.

Offset address	7	6	5	4	3	2	1	0	
$331	SM1M		P1C		SM0M		P0C		SMCMR
Reset	0	0	0	0	0	0	0	0	
	W				W				

W : Write only

SM1M, SM0M : Stepping Motor Control Ports P1, P0

When parallel ports are used, these bits are not used.

SM1M		SM0M		Function
bit7	bit6	bit3	bit2	
0	0	0	0	—
0	1	0	1	—
1	0	1	0	4-phase 1-step excitation or 4-phase 2-step excitation (full step)
1	1	1	1	4-phase 1-2 step excitation

Note: After bits 7 and 3 are read, they are always set to 1.

P1C : Port P1 function setting

P1C		P13	P12	P11	P10	SMC trigger signal
bit5	bit4					
0	0	IN/OUT	IN/OUT	IN/OUT	IN/OUT	—
0	1	IN/OUT	IN/OUT	IN/OUT	IN/Tout4	—
1	0	IN/M13	IN/M12	IN/M11	IN/M10	Timer 4
1	1					

P0C : Port P0 function setting

P0C		P03	P02	P01	P00	SMC trigger signal
bit1	bit0					
0	0	IN/OUT	IN/OUT	IN/OUT	IN/OUT	—
0	1	IN/OUT	IN/OUT	IN/OUT	IN/Tout3	—
1	0	IN/M03	IN/M02	IN/M01	IN/M00	Timer 3
1	1					

IN : Input port OUT : Output port Tout3/Tout4 : Timer output ports

M03-00 : Stepping motor control port P0

M13-10 : Stepping motor control port P1

11.1.3 SMC Port Rotation Direction Control Register (SMCRCR)

This register controls the direction of the rotation. CCW0 sets the direction for channel 0 (M0). CCW1 sets the direction for channel 1 (M1). For normal rotation, set to 0; for reverse rotation, set to 1.

Offset address $339	7	6	5	4	3	2	1	0	
				CCW 1				CCW 0	SMCRCR
Reset	1	1	1	0	1	1	1	0	
	R	R	R		R	R	R		

R : Read only

CCW1 : SMC port 1 rotation direction
CCW0 : SMC port 0 rotation direction

0 : Normal rotation
1 : Reverse rotation

11.1.4 Ports P0, P1

These are 4-bit I/O ports. The lower four bits are for port 0, 1. The upper four bits function as the shifter alternate register (SA0, SA1) for driving the stepping motor at 1-2 step excitation.

Offset address $333	7	6	5	4	3	2	1	0	
	SA03	SA02	SA01	SA00	P03	P02	P01	P00	P0R
Reset	–	–	–	–	0	0	0	0	
	W	W	W	W					

W : Write only

Offset address $335	7	6	5	4	3	2	1	0	
	SA13	SA12	SA11	SA10	P13	P12	P11	P10	P1R
Reset	–	–	–	–	0	0	0	0	
	W	W	W	W					

W : Write only

SA13-10: Shifter alternate register 1 for stepping motor control
SA03-00: Shifter alternate register 0 for stepping motor control
P13-10 : Port P1
P03-00 : Port P0

11.2 4-Phase 1-Step/2-Step Excitation

Figures 11.1 and 11.2 show 4-phase 1-step/2-step excitation output waveforms for channel 0.

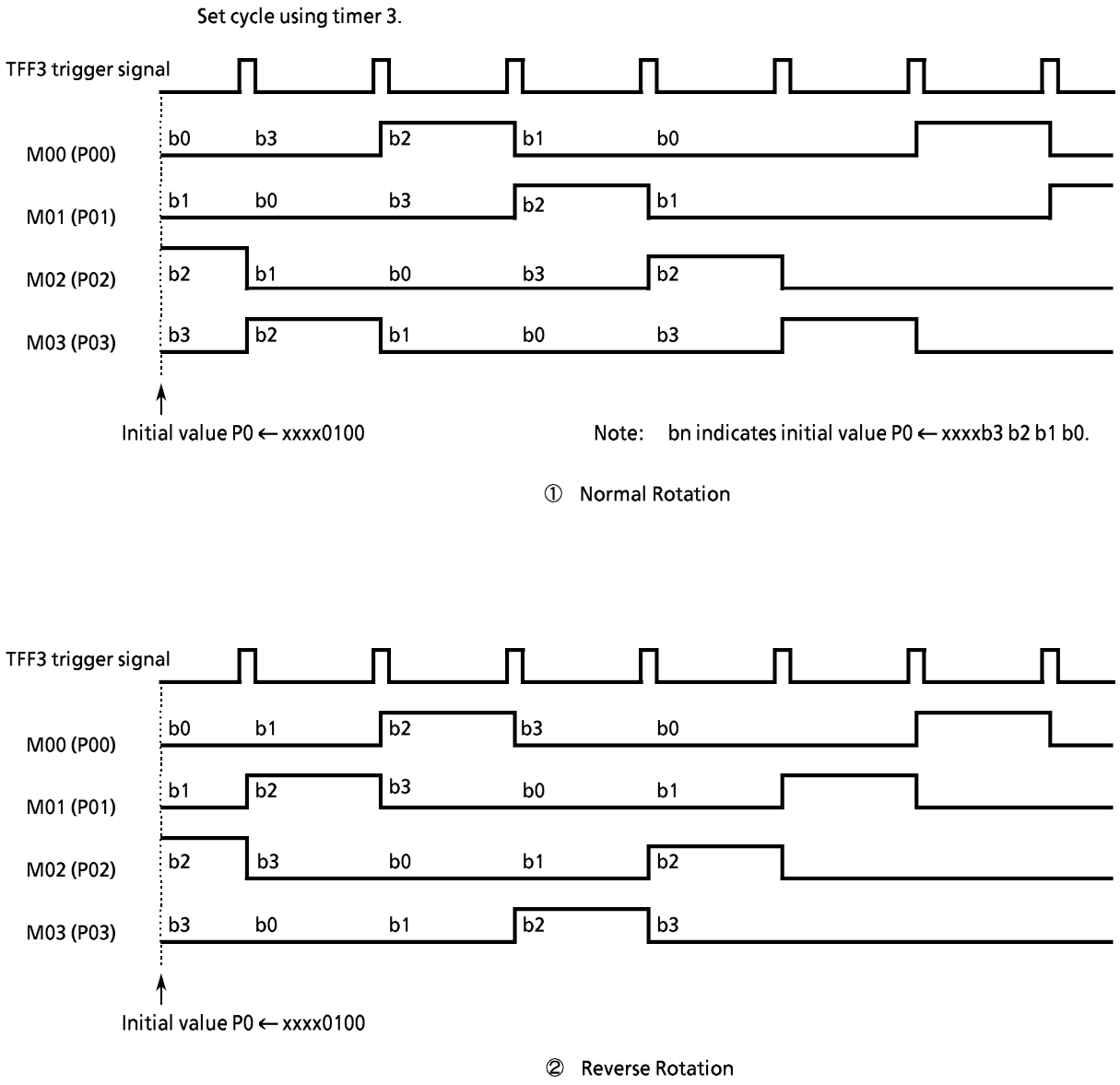


Figure 11.1 4-Phase 1-Step Excitation Output Waveform

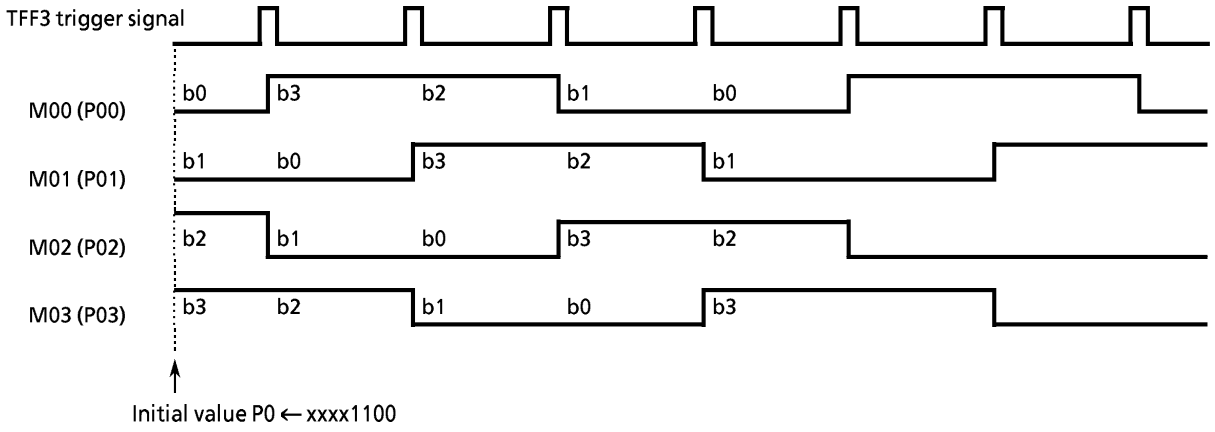


Figure 11.2 4-Phase 2-Step Excitation Output Waveform (Normal Rotation)

The following description uses channel 0 as its example.

The M0 (also functions as P0) output latch rotates and is output to the port on the rising edge of the timer flip-flop TFF3 trigger signal.

The rotation direction is determined by SMCRCR. Setting CCW0 to 0 selects normal rotation (M00 → M01 → M02 → M03). Setting CCW0 to 1 sets reverse rotation (M00 ← M01 ← M02 ← M03). At the port 1 initial settings, setting just one bit to 1 selects 4-phase 1-step excitation. Setting two consecutive bits to 1 selects 4-phase 2-step excitation.

Figure 11.3 shows the block diagram.

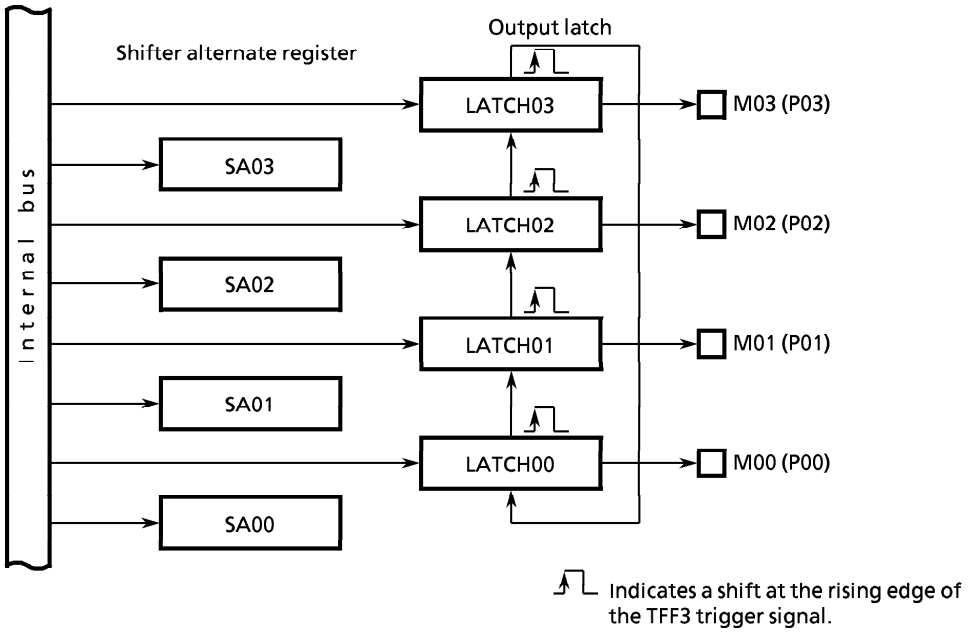


Figure 11.3 4-Phase 1-Step/2-Step Excitation (Normal Rotation) Block Diagram

11.3 4-Phase 1-2 Step Excitation

Figure 11.4 shows the 4-phase 1-2 step excitation output waveform for channel 0.

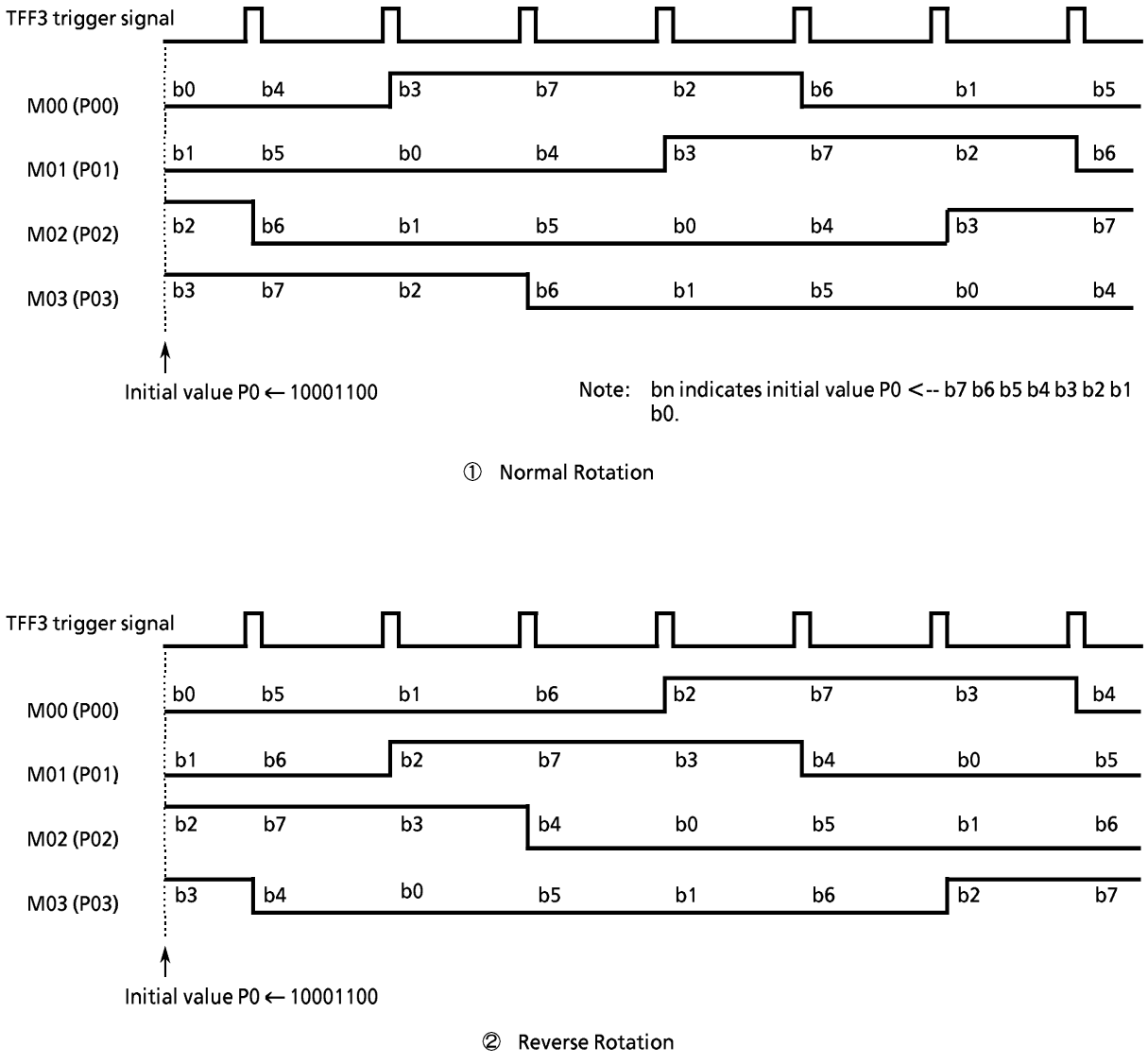


Figure 11.4 4-Phase 1-2 Step Excitation Output Waveform (Normal/Reverse Rotation)

The 4-phase 1-2 step excitation initial setting is as follows.

When initial value	b7 b6 b5 b4 b3 b2 b1 b0	is listed as	b3 b7 b2 b6 b1 b5 b0 b4

set three consecutive bits to 1 and the other bits to 0. (Positive logic)

For example, setting b3, b7, and b2 to 1 obtains an initial value of 10001100 and an output waveform as shown in Figure 11.4.

To obtain a negative logic output waveform, set the reversed initial values. For example, to convert the output waveform shown in Figure 11.4 to a negative logic output waveform, set the initial value to 01110011.

The following description uses channel 0 as its example.

The M0 output latch (also P0) and the shift alternate register (SA0) for the SMC rotate at the rising edge of the timer flip-flop TFF3 trigger signal and are output to the port. The SMCRCR register controls the rotation direction.

Figure 11.5 is a block diagram of 4-phase 1-2 step excitation (normal rotation).

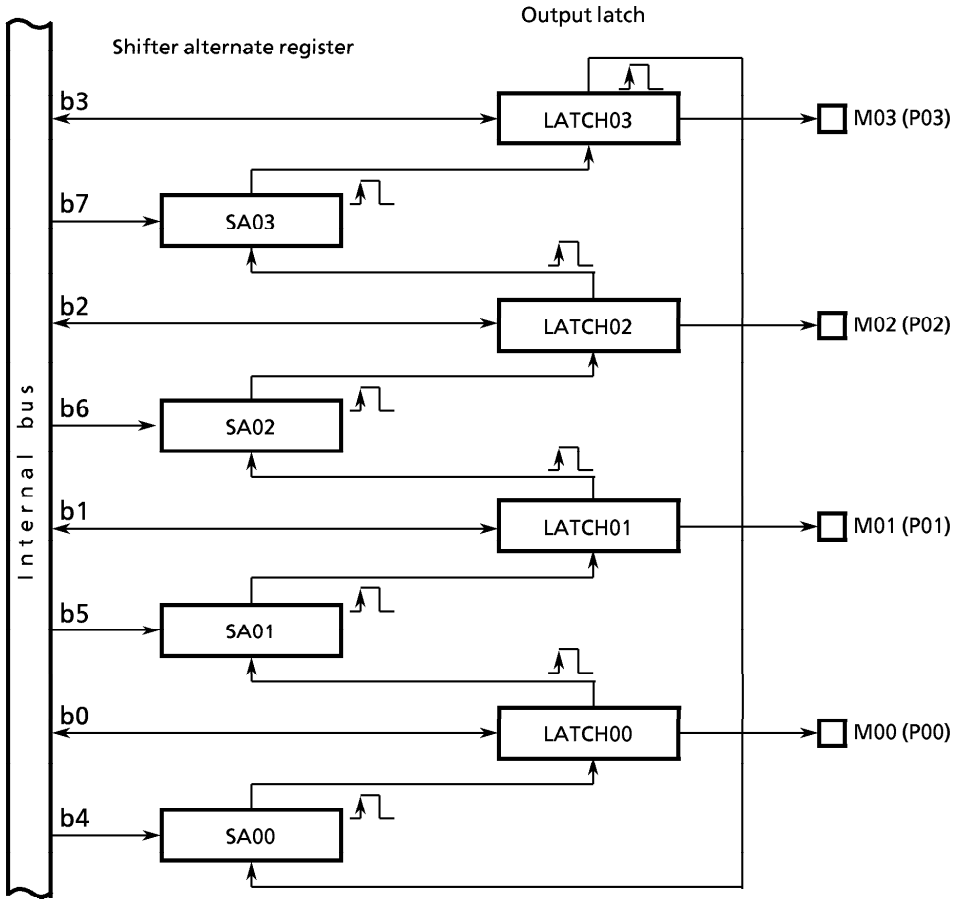


Figure 11.5 4-Phase 1-2 Step Excitation (Normal Rotation) Block Diagram

The 4-phase 1-2 step excitation initial setting is as follows.

When initial value	b7 b6 b5 b4 b3 b2 b1 b0	is listed as	b3 b7 b2 b6 b1 b5 b0 b4

set three consecutive bits to 1 and the other bits to 0. (Positive logic)

For example, setting b3, b7, and b2 to 1 obtains an initial value of 10001100 and an output waveform as shown in Figure 11.4.

To obtain a negative logic output waveform, set the reversed initial values. For example, to convert the output waveform shown in Figure 11.4 to a negative logic output waveform, set the initial value to 01110011.

The following description uses channel 0 as its example.

The M0 output latch (also P0) and the shift alternate register (SA0) for the SMC rotate at the rising edge of the timer flip-flop TFF3 trigger signal and are output to the port. The SMCRCR register controls the rotation direction.

Figure 11.5 is a block diagram of 4-phase 1-2 step excitation (normal rotation).

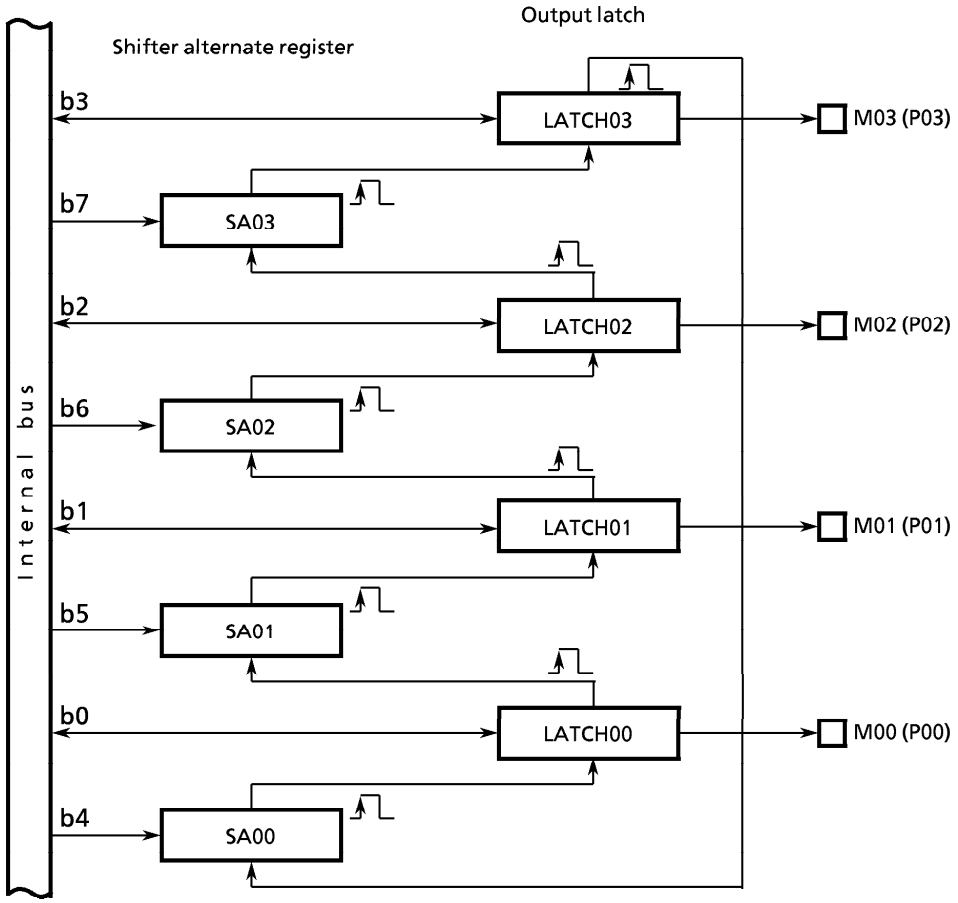


Figure 11.5 4-Phase 1-2 Step Excitation (Normal Rotation) Block Diagram

12. Internal Peripheral Circuit Register Map

12.1 Register map (1)

○ Address decoder

Offset address	15 ---------8	7 --------- 0	Offset address
$000	AMAR0	AAMR0	$001
$002		AACR0	$003
$004	AMAR1	AAMR1	$005
$006		AACR1	$007
$008		AACR2	$009
$00A		ATOR	$00B
$00C	ARELR		$00D
$00E			$00F

○ Interrupt Controller

Offset address	15 ---------8	7 --------- 0	Offset address
$080		ICR0	$081
$082		ICR1	$083
$084		ICR2	$085
$086		ICR3	$087
$088		ICR4	$089
$08A		ICR5	$08B
$08C	ICR6	ICR7	$08D
$08E		ICR8	$08F
$090		ICR9	$091
$092	ICR10	ICR11	$093
$094	IMR		$095
$096	IPR		$097
$098	IISR		$099
$09A		IVNR	$09B
$09C			$09D
$09E			$09F

○ Serial Interface

Offset address	15 ---------8	7 --------- 0	Offset address
$180		SMR0	$181
$182		SCMR0	$183
$184		SBRR0	$185
$186		SSR0	$187
$188		SDR0	$189
$18A			$18B
$18C		SPR	$18D
$18E		SCR	$18F
$190		SMR1	$191
$192		SCMR1	$193
$194		SBRR1	$195
$196		SSR1	$197
$198		SDR1	$199
$19A			$19B
$19C			$19D
$19E			$19F

○ 16-bit timer

Offset address	15 ---------8 7 --------- 0	Offset address
$200	TCR0	$201
$202		$203
$204	TMCR01	$205
$206		$207
$208		$209
$20A		$20B
$20C	TCTR0	$20D
$20E		$20F
$210		$211
$212		$213
$214		$215
$216		$217
$218		$219
$21A		$21B
$21C		$21D
$21E		$21F
$220	TCR1	$221
$222		$223
$224	TMCR11	$225
$226		$227
$228	TMCR12	$229
$22A		$22B
$22C	TCTR1	$22D
$22E		$22F
$230		$231
$232		$233
$234		$235
$236		$237
$238		$239
$23A		$23B
$23C		$23D
$23E		$23F
$240	TCR2	$241
$242		$243
$244	TMCR21	$245
$246		$247
$248	TMCR22	$249
$24A		$24B
$24C	TCTR2	$24D
$24E		$24F

○ 8-bit timer

Offset address	15 ---------8 7 --------- 0	Offset address
$260	TIMR30	$261
$262	TIMR31	$263
$264	TIMR40	$265
$266	TIMR41	$267
$268	TCCR	$269
$26A	TFFCR	$26B
$26C	TMDR	$26D
$26E	TRCR	$26F
$270	TIRCR	$271

○ DRAM controller

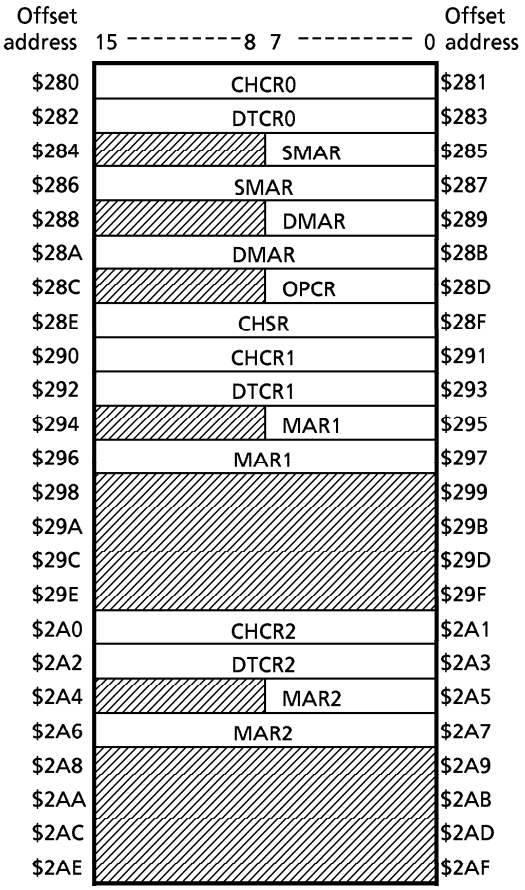
Offset address	15 ---------8 7 --------- 0	Offset address
$300	RCR	$301
$302	MCR	$303

○ Parallel interface/stepping motor controller

Offset address	15 ---------8 7 --------- 0	Offset address
$330	SMCMR	$331
$332	P0R	$333
$334	P1R	$335
$336	P01CR	$337
$338	SMCCR	$339

Offset address	15 ---------8 7 --------- 0	Offset address
$1B0	P2R	$1B1
$1B2	P2CR	$1B3

○ DMA controller



12.2 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
AMAR0	Memory Address Register 0	$000	A23	A22	A21	A20	A19	A18	A17	A16
			0	0	0	0	0	0	0	0
	For CS0		R/W							
AAMR0	Address Mask Register 0	$001	M22	M21	M20	M19	M18	M17	M16	M15
			1	1	1	1	1	1	1	1
	For CS0		R/W							
AACR0	Area Control Register 0	$003			EN	ED	ID	WAIT		
			0	0	1	1	1	1	0	1
	For CS0		R		R/W					
AMAR1	Memory Address Register 1	$004	A23	A22	A21	A20	A19	A18	A17	A16
			–	–	–	–	–	–	–	–
	For CS1		R/W							
AAMR1	Address Mask Register 1	$005	M21	M20	M19	M18	M17	M16	M15-M9	M8
			–	–	–	–	–	–	–	–
	For CS1		R/W							
AACR1	Area Control Register 1	$007			EN	ED	ID	WAIT		
			0	0	0	1	1	0	0	0
	For CS1		R		R/W					
AACR2	Area Control Register 2	$009				ED	ID	WAIT		
			0	0	0	1	1	0	0	0
	For IACK cycle		R			R/W				
ATOR	Time Out Register	$00B					256	128	64	32
			0	0	0	0	1	0	0	0
	For BERR generation		R				R/W			
ARELR	Relocaton Register	$00C	A23	A22	A21	A20	A19	A18	A17	A16
			1	1	1	1	1	1	1	1
			R/W							
		$00D	A15	A14	A13	A12	A11	A10		
			1	1	1	1	1	1	0	0
	For internal registers		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 serial ch0	$087						Level		
			0	0	0	0	0	1	1	1
			R					R/W		
ICR4	Interrupt Control Register 4 For serial ch1	$089						Level		
			0	0	0	0	0	1	1	1
			R					R/W		
ICR5	Interrupt Control Register 5 For DMAC ch0	$08B						Level		
			0	0	0	0	0	1	1	1
			R					R/W		
ICR6	Interrupt Control Register 6 For DMAC ch2	$08C						Level		
			0	0	0	0	0	1	1	1
			R					R/W		
ICR7	Interrupt Control Register 7 For DMAC ch1	$08D						Level		
			0	0	0	0	0	1	1	1
			R					R/W		
ICR8	Interrupt Control Register 8 For timer ch1	$08F						Level		
			0	0	0	0	0	1	1	1
			R					R/W		
ICR9	Interrupt Control Register 9 For timer ch2	$091						Level		
			0	0	0	0	0	1	1	1
			R					R/W		
ICR10	Interrupt Control Register 10 For timer ch4	$092						Level		
			0	0	0	0	0	1	1	1
			R					R/W		
ICR11	Interrupt Control Register 11 For timer ch3	$093						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		D2	D1	D0	T4	T3	T2	T1
			0	1	1	1	1	1	1	1
			R	R/W						
		$095			S1	S0		E2	E1	E0
			0	0	1	1	0	1	1	1
			R		R/W		R	R/W		
IPR	Interrupt Pending Register	$096		D2	D1	D0	T4	T2	T1	T0
			0	0	0	0	0	0	0	0
			R	R/W						
		$097			S1	S0		E2	E1	E0
			0	0	0	0	0	0	0	0
			R		R/W		R	R/W		
IISR	Interrupt In Service Register	$098		D2	D1	D0	T4	T3	T2	T1
			0	0	0	0	0	0	0	0
			R	R/W						
		$099			S1	S0		E2	E1	E0
			0	0	0	0	0	0	0	0
			R		R/W		R	R/W		
IVNR	Interrupt Vector Number Register	$09B	Vector							
			0	0	0	0	0	0	0	0
			R/W			R				

○ Serial interface

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 For ch0	$181	RxINTM	ErINTM	PE0	PEN	CL1	CL0	TxINTM	ST
			1	1	–	–	–	–	1	–
			R/W							
SCMR0	Serial Command Register 0 For ch0	$183			RTS	ERS	SBRK	RxEN		TxEN
			0	0	0	1	0	0	0	0
			R		R/W				R	R/W
SBRR0	Serial Baudrate Register 0 For ch0	$185	B7	B6	B5	B4	B3	B2	B1	B0
			0	0	0	0	0	0	0	0
			R/W							
SSR0	Serial Status Register For ch0	$187		RBRK	FE	OE	PE	TxE	RxRDY	TxRDY
			1	0	0	0	0	1	0	0
			R	R/W						
SDR0	Serial Data Register 0 For ch0	$189	D7	D6	D5	D4	D3	D2	D1	D0
			–	–	–	–	–	–	–	–
			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
SMR1	Serial Mode Register 1 For ch1	$191	RxINTM	ErINTM	PE0	PEN	CL1	CL0	TxINTM	ST
			1	1	–	–	–	–	1	–
			R/W							
SCMR1	Serial Command Register 1 For ch1	$193				ERS	SBRK	RxEN		TxEN
			0	0	0	1	0	0	0	0
			R			R/W			R	R/W
SBRR1	Serial Baudrate Register 1 For ch1	$195	B7	B6	B5	B4	B3	B2	B1	B0
			0	0	0	0	0	0	0	0
			R/W							
SSR1	Serial Status Register 1 For ch1	$197		RBRK	FE	OE	PE	TxE	RxRDY	TxRDY
			0	0	0	0	0	1	0	0
			R	R/W						
SDR1	Serial Data Register 1 For ch1	$199	D7	D6	D5	D4	D3	D2	D1	D0
			–	–	–	–	–	–	–	–
			R/W							

○ 16-bit timer (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
TCR0	Timer Control Register 0	$200		CK1	P4	P3	P2	P1	T2	T1
			0	0	0	0	0	0	0	0
			R	R/W						
		$201							CS	TS
			1	1	0	1	0	0	1	0
	For ch0		R						R/W	
TMCR01	Timer MAX Count Register 01	$204	M15	M14	M13	M12	M11	M10	M9	M8
			–	–	–	–	–	–	–	–
			R/W							
		$205	M7	M6	M5	M4	M3	M2	M1	M0
			–	–	–	–	–	–	–	–
	For ch0		R/W							
TCTR0	Timer Count Register 0	$20C	C15	C14	C13	C12	C11	C10	C9	C8
			0	0	0	0	0	0	0	0
			R							
		$20D	C7	C6	C5	C4	C3	C2	C1	C0
			0	0	0	0	0	0	0	0
	For ch0		R							
TCR1	Timer Control Register 1	$220	CK2	CK1	P4	P3	P2	P1	T2	T1
			0	0	0	0	0	0	0	0
			R/W							
		$221	N/1	R/P	MR2	MR1		INT	CS	TS
			0	0	0	0	0	0	1	0
	For ch1		R/W				R	R/W		
TMCR11	Timer MAX Count Register 11	$224	M15	M14	M13	M12	M11	M10	M9	M8
			–	–	–	–	–	–	–	–
			R/W							
		$225	M7	M6	M5	M4	M3	M2	M1	M0
			–	–	–	–	–	–	–	–
	For ch1		R/W							
TMCR12	Timer MAX Count Register 12	$228	M15	M14	M13	M12	M11	M10	M9	M8
			–	–	–	–	–	–	–	–
			R/W							
		$229	M7	M6	M5	M4	M3	M2	M1	M0
			–	–	–	–	–	–	–	–
	For ch1		R/W							

○ 16-bit timer (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
TCTR1	Timer Count Register 1	$22C	C15	C14	C13	C12	C11	C10	C9	C8
			0	0	0	0	0	0	0	0
			R							
		$22D	C7	C6	C5	C4	C3	C2	C1	C0
			0	0	0	0	0	0	0	0
	For ch1		R							
TCR2	Timer Control Register 2	$240	CK2	CK1	P4	P3	P2	P1	T2	T1
			0	0	0	0	0	0	0	0
			R/W							
		$241	N/1	R/P	MR2	MR1		INT	CS	TS
			0	0	0	0	0	0	0	0
	For ch2		R/W				R	R/W		
TMCR21	Timer MAX Count Register 21	$244	M15	M14	M13	M12	M11	M10	M9	M8
			–	–	–	–	–	–	–	–
			R/W							
		$245	M7	M6	M5	M4	M3	M2	M1	M0
			–	–	–	–	–	–	–	–
	For ch2		R/W							
TMCR22	Timer MAX Count Register 22	$248	M15	M14	M13	M12	M11	M10	M9	M8
			–	–	–	–	–	–	–	–
			R/W							
		$249	M7	M6	M5	M4	M3	M2	M1	M0
			–	–	–	–	–	–	–	–
	For ch2		R/W							
TCTR2	Timer Count Register 2	$24C	C15	C14	C13	C12	C11	C10	C9	C8
			0	0	0	0	0	0	0	0
			R							
		$24D	C7	C6	C5	C4	C3	C2	C1	C0
			0	0	0	0	0	0	0	0
	For ch2		R							

○ 8-bit 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
TIMR30	Timer 30 Register	$261	T307	T306	T305	T304	T303	T302	T301	T300
			–	–	–	–	–	–	–	–
			R/W							
TIMR31	Timer 31 Register	$263	T315	T314	T313	T312	T311	T310	T309	T308
			–	–	–	–	–	–	–	–
			R/W							
TIMR40	Timer 40 Register	$265	T407	T406	T405	T404	T403	T402	T401	T400
			–	–	–	–	–	–	–	–
			R/W							
TIMR41	Timer 41 Register	$267	T415	T414	T413	T412	T411	T410	T409	T408
			–	–	–	–	–	–	–	–
			R/W							
TCCR	Timer Clock Control Register	$269	T3CLK		T2CLK		T1CLK		T0CLK	
			0	0	0	0	0	0	0	0
			R/W							
TFFCR	Timer F/F Control Register	$26B	FF4C		TFF41E	FF4IS	FF3C		FF31E	TFF3IS
			–	–	0	0	–	–	0	0
			W		R/W		W		R/W	
TMDR	Timer Mode Control Register	$26D	T4M		PWM4		T3M		PWM3	
			0	0	0	0	0	0	0	0
			R/W							
TRCR	Timer Control Register	$26F			PRRUN		T41RUN	T40RUN	T31RUN	T30RUN
			0	0	0	0	0	0	0	0
			R		R/W	R	R/W			
TIRCR	Timer Interrupt Request Control Register	$271							IT4E	IT3E
			1	1	1	1	1	1	0	0
			R						R/W	

○ 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
CHCR0	Channel Control Register 0 For ch0	$280			ERC	MAC	DPW	OPW	DST	CONTI
			0	0	0	0	0	0	0	0
			R	R/W						
		$281	DSSH	TDIR1	TDIR	TMOD1	TMOD0	RSS	DMAE	INTE
			0	0	0	0	0	0	0	0
			R/W							
DTCR0	Data Transfer Count Register 0 For ch0	$282	DT15	DT14	DT13	DT12	DT11	DT10	DT9	DT8
			–	–	–	–	–	–	–	–
			R/W							
		$283	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0
			–	–	–	–	–	–	–	–
			R/W							
SMAR	Source Memory Address Register	$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
			–	–	–	–	–	–	–	–
			R/W							
DMAR	Distination Memory Address Register	$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
			–	–	–	–	–	–	–	–
			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	ERRCD21	ERRCD20	ERRCD11	ERRCD10	ERRCD01	ERRCD00	ERR2
			0	0	0	0	0	0	0	0
			R		R/W					
		$28F	ERR1	ERR0	REQ2	REQ1	REQ0	OPC2	OPC1	OPC0
			0	0	0	0	0	0	0	0
			R/W							
CHCR1	Channel Control Register 1	$290							ERC	MAC
			0	0	0	0	0	0	0	0
			R						R/W	
		$291	DPW	OPW	TMOD1	TMOD0	RSS	TDIR	DMAE	INTE
			0	0	0	0	0	0	0	0
	For ch1		R/W							
DTCR1	Data Transfer Count Register 1	$292	DT15	DT14	DT13	DT12	DT11	DT10	DT9	DT8
			–	–	–	–	–	–	–	–
			R/W							
		$293	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0
			–	–	–	–	–	–	–	–
	For ch1		R/W							
MAR1	Memory Address Register 1	$295	A23	A22	A21	A20	A19	A18	A17	A16
			–	–	–	–	–	–	–	–
			R/W							
		$296	A15	A14	A13	A12	A11	A10	A9	A8
			–	–	–	–	–	–	–	–
			R/W							
		$297	A7	A6	A5	A4	A3	A2	A1	A0
			–	–	–	–	–	–	–	–
			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
CHCR2	Channel Control Register 2	$2A0							ERC	MAC
			0	0	0	0	0	0	0	0
			R						R/W	
		$2A1	DPW	OPW	TMOD1	TMOD0	RSS	TDIR	DMAE	INTE
			0	0	0	0	0	0	0	0
	For ch2		R/W							
DTCR2	Data Transfer Count Register 2	$2A2	DT15	DT14	DT13	DT12	DT11	DT10	DT9	DT8
			–	–	–	–	–	–	–	–
			R/W							
		$2A3	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0
			–	–	–	–	–	–	–	–
	For ch2		R/W							
MAR2	Memory Address Register 2	$2A5	A23	A22	A21	A20	A19	A18	A17	A16
			–	–	–	–	–	–	–	–
			R/W							
		$2A6	A15	A14	A13	A12	A11	A10	A9	A8
			–	–	–	–	–	–	–	–
			R/W							
		$2A7	A7	A6	A5	A4	A3	A2	A1	A0
			–	–	–	–	–	–	–	–
			R/W							

○ DRAM controller

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
RCR	Refresh Control Register	$301	RINI	RS2	RS1	RS0	RW2	RW1	RW0	RC
			0	0	0	0	0	0	0	0
			R/W							
MCR	Memory Control Register	$303					RCAWS	MADR1	MADR0	MC
			0	0	0	0	0	1	0	0
			R				R/W			

○ Stepping motor controller/parallel interface

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
SMCMR	Stepping Moter Control port Mode Register	$331	SM1M		P1C		SM0M		P0C	
			0	0	0	0	0	0	0	0
			R	R/W			R	R/W		
P0R	Port 0 Register	$333	SA03	SA02	SA01	SA00	P03	P02	P01	P00
			–	–	–	–	0	1	0	0
			W				R/W			
P1R	Port 1 Register	$335	SA13	SA12	SA11	SA10	P13	P12	P11	P10
			–	–	–	–	0	1	0	0
			W				R/W			
P01CR	Port 0 and 1 I/O Control Register	$337	P13C	P12C	P11C	P10C	P03C	P02C	P01C	P00C
			0	0	0	0	0	0	0	0
			W							
SMCRCR	Stepping Motor Control port Rotaing direction Control Register	$339				CCW1				CCW0
			1	1	1	0	1	1	1	0
			R			R/W	R			R/W
P2R	Port 2 Register	$1B1							P21	P20
			0	0	0	0	0	0	0	0
			R						R/W	
P2CR	Port 2 Control Register	$1B3	EMSI0						P21CR	P20CR
			0	0	0	0	0	0	0	0
			R/W	R					R/W	

13. Electrical Characteristics

This section describes the electrical characteristics and timings of TMP68303.

13.1 Maximum Ratings

Parameter	Symbol	Rating	Unit
		TMP68303	
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

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.

13. Electrical Characteristics

This section describes the electrical characteristics and timings of TMP68303.

13.1 Maximum Ratings

Parameter	Symbol	Rating	Unit
		TMP68303	
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

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.

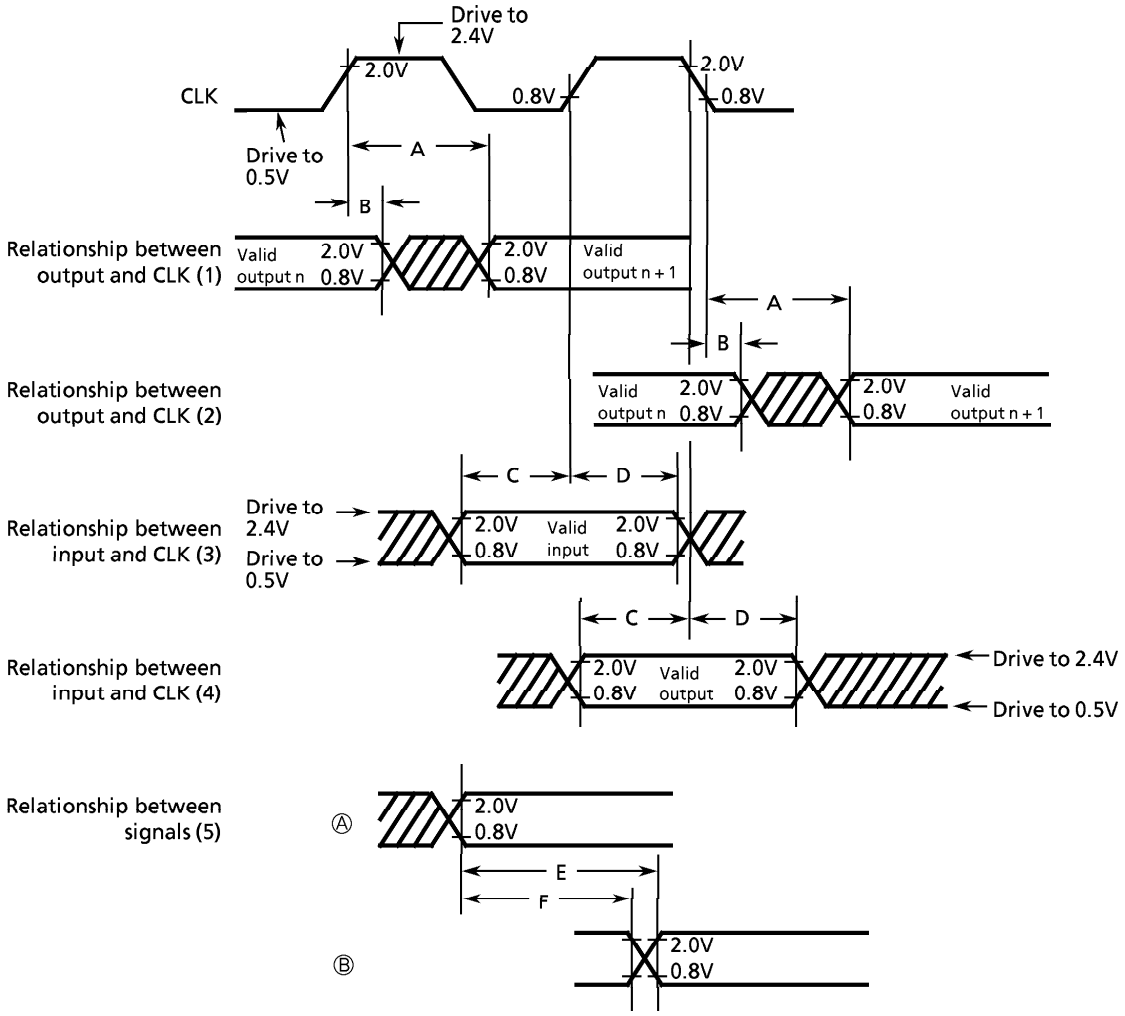
13.2 DC Characteristics

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

Parameter		Symbol	TMP68303 Min	TMP68303 Max	Unit
High-level input voltage	Except CLK, ports P0 and P1.	V_{IH}	2.0	V_{CC}	V
High-level input voltage	(CLK)	V_{IHCK}	2.2	V_{CC}	V
High-level input voltage	(ports P0 and P1)	V_{IHP}	$V_{CC} \times 0.7$	$V_{CC}+0.3$	V
Low-level input voltage	Except ports P0 and P1.	V_{IL}	GND − 0.3	0.8	V
Low-level input voltage	(ports P0 and P1)	V_{ILP}	− 0.3	$V_{CC} \times 0.3$	V
Input leakage current (5.25V)	$\overline{BERR}$, $\overline{BGACK}$, $\overline{BR}$, $\overline{DTACK}$, $\overline{HALT}$, $\overline{RESET}$, CLK, NOR / $\overline{EMU}$, TIN, INT0~INT2, RxD0, RxD1, DREQ0~DREQ2, $\overline{DTEND}$, P00~P03, P10~P13, P20, P21	I_{IN}	– – –	2.5 2.5 20	μA μA
Tri-state (off state) input current	(2.4V/0.4V) $\overline{AS}$, A1~A23, D0~D15, FC0~FC2, $\overline{LDS}$, $\overline{UDS}$, R / $\overline{W}$	I_{TSI}	– –	20 20	μA μA
High-level output voltage (IOH = -400μA)	$\overline{AS}$, A1~A23, D0~D15, FC0~FC2, $\overline{LDS}$, $\overline{UDS}$, R / $\overline{W}$, $\overline{BG}$, P00~P03, P10~P13, P20, P21 Tout1, Tout2, $\overline{RAS}$, $\overline{CAS}$, TxD0, TxD1, $\overline{DACK0}$~$\overline{DACK2}$, $\overline{IACK0}$~$\overline{IACK2}$ IPL2~IPL0	V_{OH}	– V_{CC} − 0.75	–	V
Low-level output voltage		V_{OL}			V
(IOL = 1.6mA)	$\overline{HALT}$, $\overline{RESET}$		–	0.5	
(IOL = 3.2mA)	A1~A23, $\overline{BG}$, FC0~FC2		–	0.5	
(IOL = 5.3mA)	$\overline{AS}$, D0~D15, $\overline{LDS}$, $\overline{UDS}$, R / $\overline{W}$, P00~P03, P10~P13, P20, P21, Tout1, Tout2, $\overline{RAS}$, $\overline{CAS}$, TxD0, TxD1, $\overline{DACK0}$~$\overline{DACK2}$, $\overline{IACK0}$~$\overline{IACK2}$, $\overline{DTACK}$, $\overline{BERR}$, $\overline{DTEND}$, $\overline{BR}$, $\overline{BGACK}$, $\overline{IPL2}$~$\overline{IPL0}$		–	0.5	
Current dissipation	f = 16.67MHz	I_D	–	100	mA
Power dissipation	f = 16.67MHz	P_D	–	0.525	W
Input capacitance (Vin = 0V, Ta = 25℃ : Frequency = 1MHz)*		C_{IN}	–	20.0	pF
Load capacitance	$\overline{HALT}$ Other pins	C_L	– –	70 130	pF

* : Input capacitance is periodically sampled rather than 100% tested.

AC Electrical Characteristics (VCC = 5.0V ± 5%)



A : Maximum output delay

B : Minimum output hold time

C : Minimum input setup time

D : Minimum input hold time

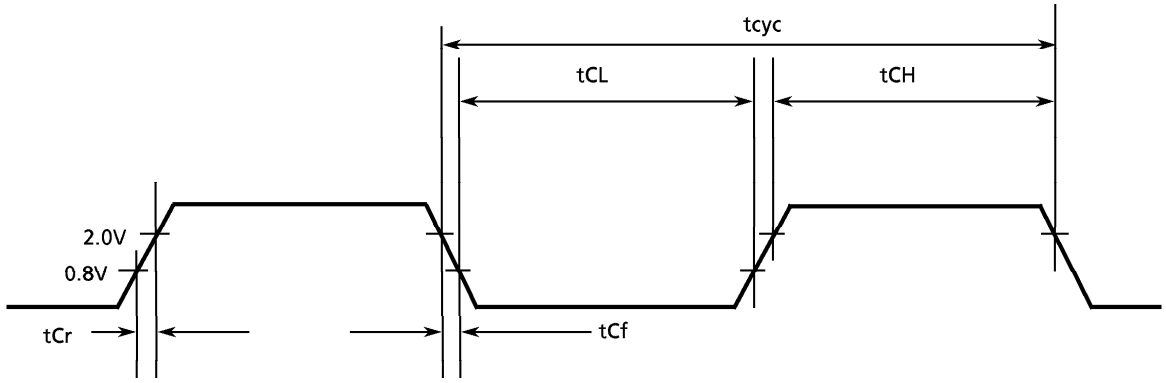
E : Signal Ⓐ valid - signal Ⓑ valid time

F : Signal Ⓐ valid - signal Ⓑ invalid time

13.3 AC Electrical Characteristics - Clock Timings

(V_{CC} = 5.0V ± 5%, GND = 0V, Ta = 0~70°C; See Figure 13.1.)

Parameter	Symbol	16.67MHz		Unit
		Min.	Max.	
Operating frequency	f	8.0	16.67	MHz
Cycle time	tcyc	60	125	ns
Clock pulse width	tCL / tCH	27	62.5	ns
Rise and fall times	tCr / tCf	–	5	ns



Note: 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.

Figure 13.1 Clock Input Timing

13.4 AC Electrical Characteristics - Read and Write Cycles (1/2)

(VCC = 5.0V ± 5%, GND = 0V, Ta = 0~70°C; See figures 13.2 and 13.3.)

Number	Parameter	Symbol	16.67MHz		Unit	Note
			Min.	Max.		
1	Clock cycle	tCYC	60	125	ns	
2	Clock width low	tCL	27	62.5	ns	
3	Clock width high	tCH	27	62.5	ns	
4	Clock fall time	tCf	–	5	ns	
5	Clock rise time	tCr	–	5	ns	
6	Clock low to address valid	tCLAV	–	50	ns	7
6A	Clock high to FC valid	tCHFCV	–	45	ns	
7	Clock high to address, data bus high impedance (maximum)	tCHADZ	–	50	ns	
8	Clock high to address, FC invalid (minimum)	tCHAFI	0	–	ns	
9	Clock high to $\overline{AS}$, $\overline{DS}$ low	tCHSL	3	40	ns	1, 7
11	Address valid to $\overline{AS}$, $\overline{DS}$ low (read) /Address valid to $\overline{AS}$ low (write)	tAVSL	10	–	ns	2
11A	FC valid to $\overline{AS}$, $\overline{DS}$ low (read)/FC valid to $\overline{AS}$ low (write)	tFCVSL	30	–	ns	2
12	Clock low to $\overline{AS}$, $\overline{DS}$ high	tCLSH	3	40	ns	1, 7
13	$\overline{AS}$, $\overline{DS}$ high to address/FC invalid	tSHAFI	10	–	ns	2
14	$\overline{AS}$, $\overline{DS}$ width low (read) /$\overline{AS}$ width low (write)	tSL	120	–	ns	2
14A	$\overline{DS}$ width low (write)	tDSL	60	–	ns	2
15	$\overline{AS}$, $\overline{DS}$ width high	tSH	60	–	ns	2
16	Clock high to control bus high impedance	tCHCZ	–	50	ns	
17	$\overline{AS}$, $\overline{DS}$ high to R/W high (read)	tSHRH	10	–	ns	2
18	Clock high to R/$\overline{W}$ high	tCHRH	0	40	ns	1, 7
20	Clock to R/$\overline{W}$ low (write)	tCHRL	0	40	ns	1, 7
20A	AS low to R/W valid (write)	tASRV	–	10	ns	2,6
21	Address valid to R/$\overline{W}$ low (write)	tAVRL	0	–	ns	2
21A	FC valid to R/$\overline{W}$ low (write)	tFCVRL	20	–	ns	2
22	R/$\overline{W}$ low to $\overline{DS}$ low (write)	tRLSL	20	–	ns	2
23	Clock low to data out valid (write)	tCLDO	–	50	ns	7
25	$\overline{AS}$, $\overline{DS}$ high to data out invalid (write)	tSHDOI	15	–	ns	2
26	Data out valid to $\overline{DS}$ low (write)	tDOSL	10	–	ns	2

13.4 AC Electrical Characteristics - Read and Write Cycles (2/2)

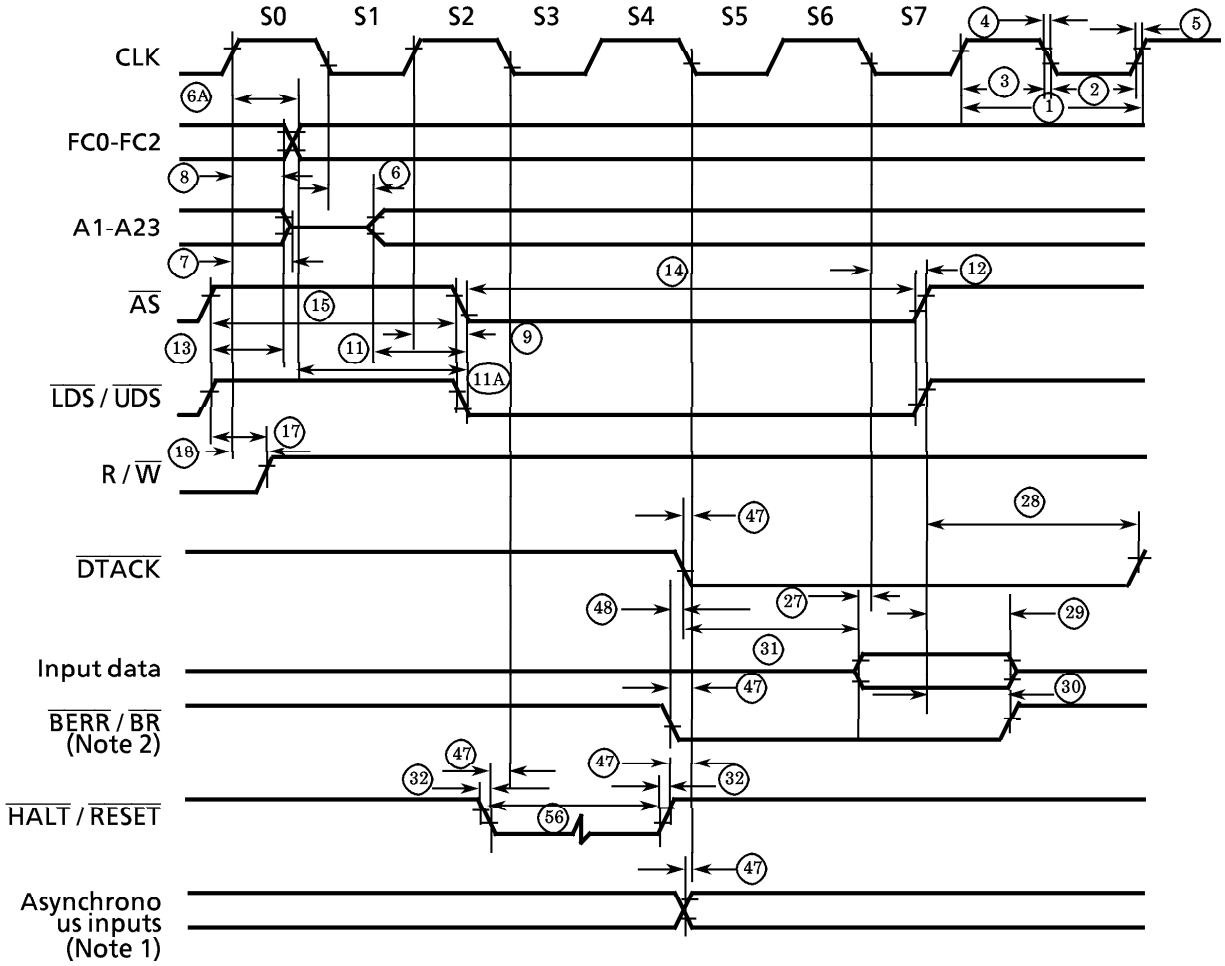
(VCC = 5.0V ± 5%, GND = 0V, Ta = 0~70°C; See figures 13.2 and 13.3.)

Number	Parameter	Symbol	16.67MHz		Unit	Note
			Min.	Max.		
27	Data in to clock low (setup time on read)	tDICL	7	–	ns	5
28	$\overline{AS}$, $\overline{DS}$ high to $\overline{DTACK}$ high	tSHDAH	0	110	ns	2
29	$\overline{AS}$, $\overline{DS}$ negate to data invalid (hold time for read)	tSHDII	0	–	ns	
30	$\overline{AS}$, $\overline{DS}$ high to $\overline{BERR}$ high	tSHBEH	0	–	ns	
31	$\overline{DTACK}$ low setup time (input data)	tDALDI	–	40	ns	2,5
32	$\overline{HALT}$ and $\overline{RESET}$ input transition	tRHr, f	0	150	ns	
47	Asynchronous input setup time	tASI	10	–	ns	5
48	$\overline{BERR}$ low to $\overline{DTACK}$ low	tBELDAL	10	–	ns	2,3
53	Clock high to data out invalid	tCHDOI	0	–	ns	
55	R/$\overline{W}$ low to data bus drive	tRLDBD	0	–	ns	
56	$\overline{HALT}$/$\overline{RESET}$ pulse width	tHRPW	12	–	Clk. Per.	4

Notes:

1. For a loading capacitance of less than or equal to 50 picofarads, subtract five nanoseconds from the value given in the maximum columns.
2. Actual value depends on clock cycle.
3. If #47 is satisfied for both $\overline{DTACK}$ and $\overline{BERR}$, #48 may be 0ns.
4. At power on, hold $\overline{RESET}$ and $\overline{HALT}$ into low status for 1 to 100μs microseconds 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 clock-low setup time (#27) requirement for the following cycle.
6. When $\overline{AS}$ and R/$\overline{W}$ are equally loaded (±20%), subtract 10ns (8 - 12.5MHz) or 5ns (16.67MHz) from the tASRV maximum value.
7. The maximum values of signals input externally in emulation mode or during external bus master operation is 30ns at 16.67MHz.

The waveforms below should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Functional descriptions and their related device operation diagrams are given elsewhere.

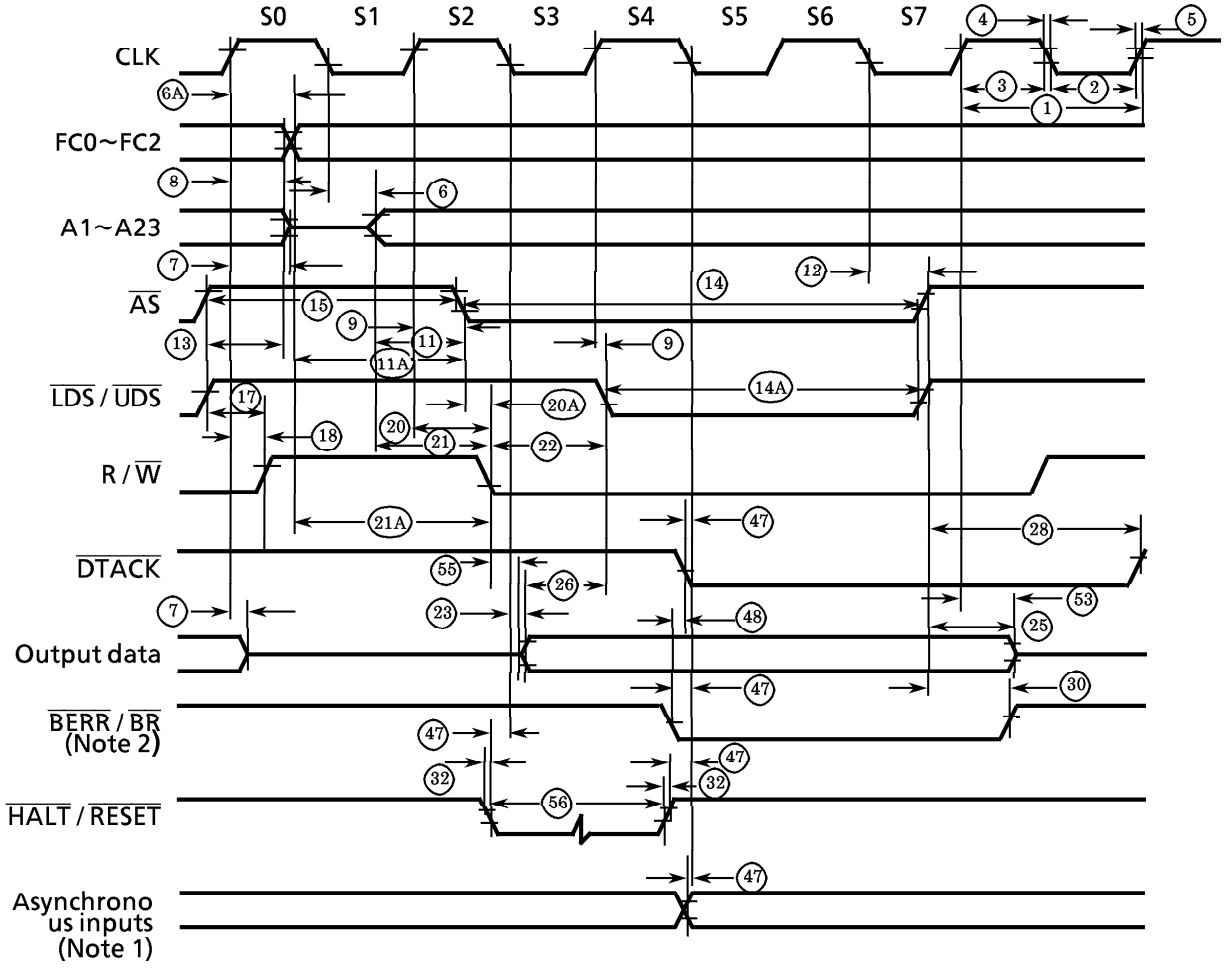


Notes:

1. Asynchronous input $\overline{\text{BGACK}}$ is detected at the clock'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 0.8V and 2.0V.

Figure 13.2 Read Cycle Timing

The waveforms below should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Functional descriptions and their related device operation diagrams are given elsewhere.



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 13.3 Write Cycle Timing

13.5 AC Electrical Characteristics - Bus Arbitration

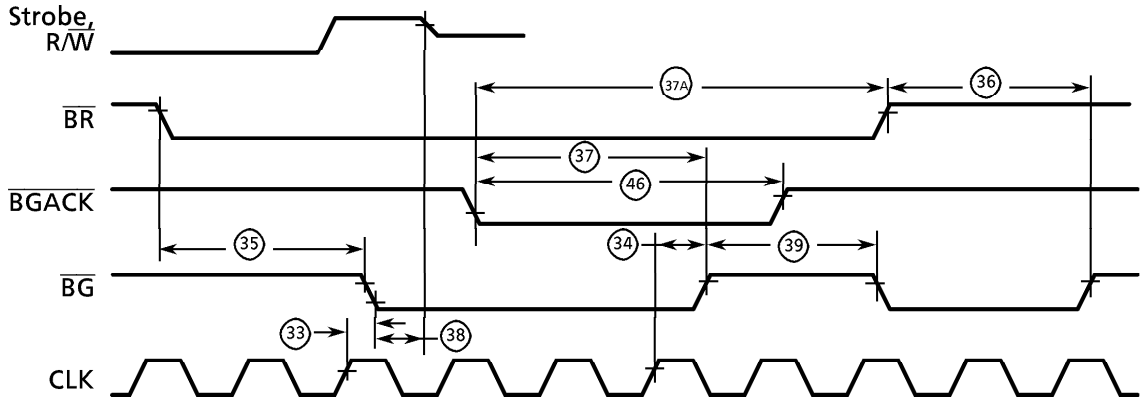
(VCC = 5.0V ± 5%, GND = 0V, Ta = 0~70°C; See figure 13.4)

Number	Parameter	Symbol	16.67MHz		Unit	Note
			Min.	Max.		
7	Clock high to address, data bus high impedance	tCHADZ	–	50	ns	
16	Clock high to control bus high impedance	tCHCZ	–	50	ns	
33	Clock high to $\overline{BG}$ low	tCHGL	–	40	ns	
34	Clock high to $\overline{BG}$ high	tCHGH	–	40	ns	
35	$\overline{BR}$ high to $\overline{BG}$ high	tBRLGL	1.5	3.5	Clk. Per.	
36	$\overline{BGACK}$ low to $\overline{BG}$ high	tBKHGH	1.5	3.5	Clk. Per.	1
37	$\overline{BGACK}$ low to $\overline{BR}$ high	tGALGH	1.5	3.5	Clk. Per.	
37A	$\overline{BGACK}$ low to $\overline{BR}$ high	tGALBRH	10	1.5 Clocks	ns	2
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.	
57	$\overline{BGACK}$ high to control bus drive	tGABD	1.5	–	Clk. Per.	
58	$\overline{BG}$ high to control bus drive	tGHBD	1.5	–	Clk. Per.	1

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.

The waveforms below should only be referenced in regard to the edge-to-edge measurement of the timing specifications. They are not intended as a functional description of the input and output signals. Functional descriptions and their related device operation diagrams are given elsewhere.



Note: Asynchronous inputs $\overline{\text{BERR}}$, $\overline{\text{BGACK}}$, $\overline{\text{BR}}$, and $\overline{\text{DTACK}}$ are detected at the clock's falling edge.

Figure 13.4 Bus Arbitration Timing

When the external bus master accesses the registers of internal devices, addresses, data, and control signals must be input in accordance with the read/write cycle timing.

13.6 AC Electrical Characteristics - Peripherals (1/2)

(VCC = 5.0V ± 5%, GND = 0V, Ta = 0~70°C; See figures 13.5 - 13.13)

Number	Parameter	Symbol	16.67MHz		Unit
			Min.	Max.	
47	Asynchronous input setup time	tASI	10	—	ns
101	CLK to $\overline{CS}$, $\overline{IACK}$, $\overline{DACK}$ delay time	tDAS	—	50	ns
102	CLK low to TOUT	tCLTO	—	70	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
108	$\overline{LDS}$ high to $\overline{RTS0}$	tDSMC	—	140	ns
109	$\overline{CST0}$ to $\overline{LDS}$ low	tMCDS	50	—	ns
113	CLK to $\overline{RAS}$, $\overline{CAS}$ assert	tCLRC	—	30	ns
114	$\overline{AS}$, $\overline{DS}$ high to $\overline{RAS}$, $\overline{CAS}$ negate	tADSRC	—	30	ns
115	$\overline{RAS}$ precharge time	tRP	1.5	—	Clk. Per.
117	CLK high to column address valid	tCLCO	—	30	ns
118	$\overline{RAS}$-$\overline{CAS}$ delay time and $\overline{CAS}$ setup time	tRCD	1.0	—	Clk. Per.
119	$\overline{RAS}$ pulse width (at read/write)	tRAS	2.0	—	Clk. Per.
120	$\overline{DTACK}$ low to $\overline{CAS}$ assert	tDTCCA	—	30	ns
121	CLK high to low address hold time	tCLRCA	15	—	ns
122	CLK high to $\overline{RAS}$, $\overline{CAS}$ negate	tCLRCN	—	50	ns
123	$\overline{RAS}$, $\overline{CAS}$ negate to column address hold time	tCSR	0	—	ns
124	CLK (Low) to P0, P1 OUTPUT	tDPW1		40	ns
125	CLK low to P0, P1 output	tPRD1	1.5	—	Clk. Per.
126	P0, P1 input setup time	tPHD1	0	—	ns
127	$\overline{LDS}$ low to P2 output	tDPW2	—	40	ns
128	P2 input setup time	tPRD2	10	—	ns
129	P2 input hold time	tPHD2	10	—	ns
130	$\overline{DREQ}$ input setup time	tDASI	30	—	ns
131	$\overline{DREQ}$ input hold time	tDCH	30	—	ns
132	$\overline{BGACK}$ low to $\overline{DREQ}$ high	tGALRQH	0	—	ns
133	CLK high to $\overline{BR}$ low	tCHBRL	—	10	ns

13.6 AC Electrical Characteristics - Peripherals (2/2)

(VCC = 5.0V ± 5%, GND = 0V, Ta = 0~70°C; See figures 13.4 - 13.18)

Number	Parameter	Symbol	16.67MHz		Unit
			Min.	Max.	
134	$\overline{BGACK}$ low to bus drive	tGALDBD	—	30	ns
135	$\overline{BGACK}$ high to bus high impedance	tGAHZ	—	30	ns
136	CPU bus high impedance to $\overline{BGACK}$ low	tZGAL	—	1.0	Clk. Per.
137	CLK high to $\overline{BGACK}$ low	tCHGAL	—	30	ns
138	CLK high to $\overline{BGACK}$ high	tCHGAH	—	30	ns
139	$\overline{AS}$ low to $\overline{DTEND}$ input setup time	tASLEDL	0	—	ns
140	$\overline{DTEND}$ low width	tEDW	1.0	—	Clk. Per.
141	CLK high to $\overline{DTEND}$ low	tCHEDL	—	20	ns
142	CLK high to $\overline{DTEND}$ high	tCHEDH	—	20	ns
143	$\overline{BGACK}$ low to FC0-2 valid	tGALFC	—	30	ns
144	$\overline{BGACK}$ high to FC0-2 high impedance	tDFCZ	—	20	ns
145	$\overline{BGACK}$ low to control bus valid	tGALCSV	—	30	ns
146	$\overline{BGACK}$ high to control bus high impedance	tGAHCSZ	—	20	ns

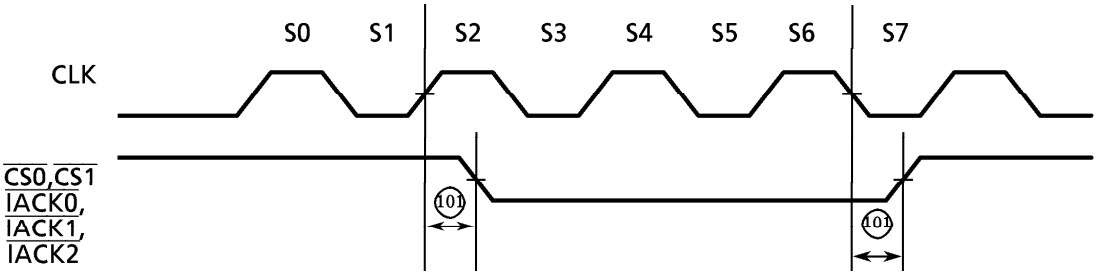


Figure 13.5 $\overline{CS}$, $\overline{IACK}$, $\overline{DACK}$ Timing

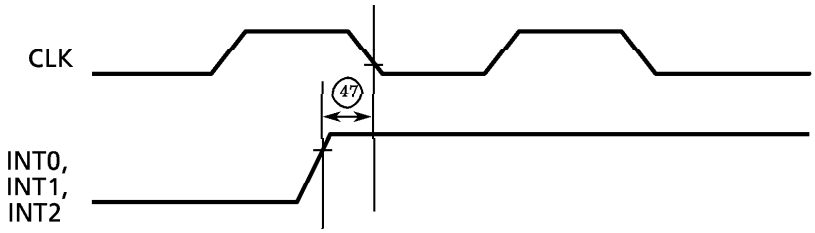


Figure 13.6 Interrupt Request Timing

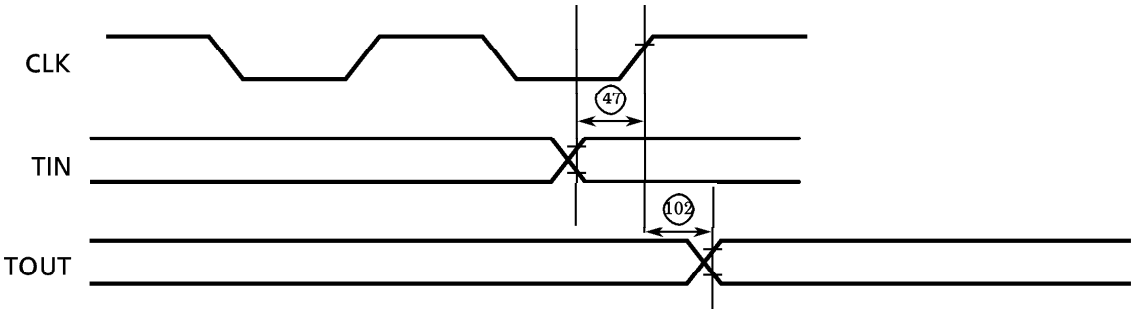


Figure 13.7 Timer I/O Timing

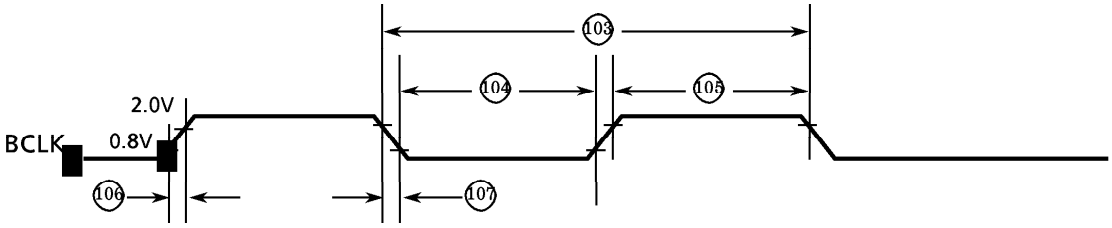


Figure 13.8 Baud Rate Clock Timing

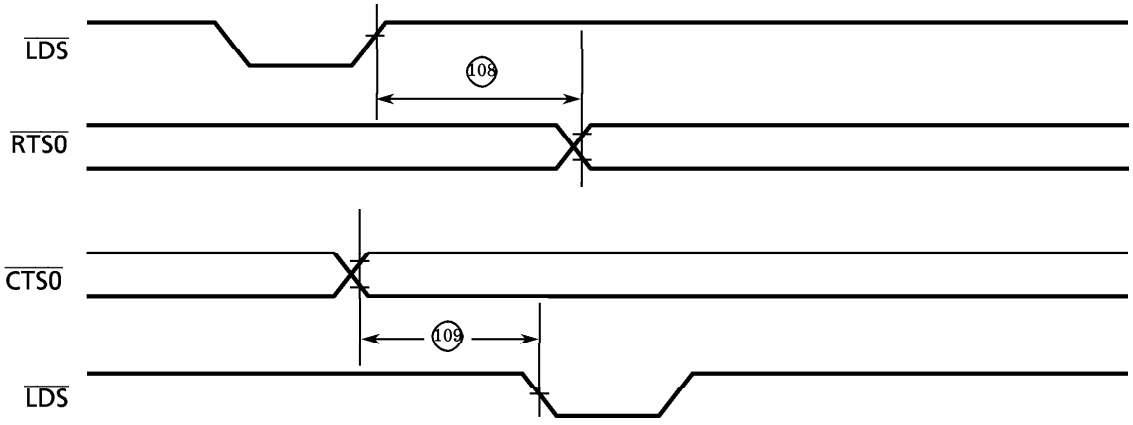


Figure 13.9 Serial Port Timing

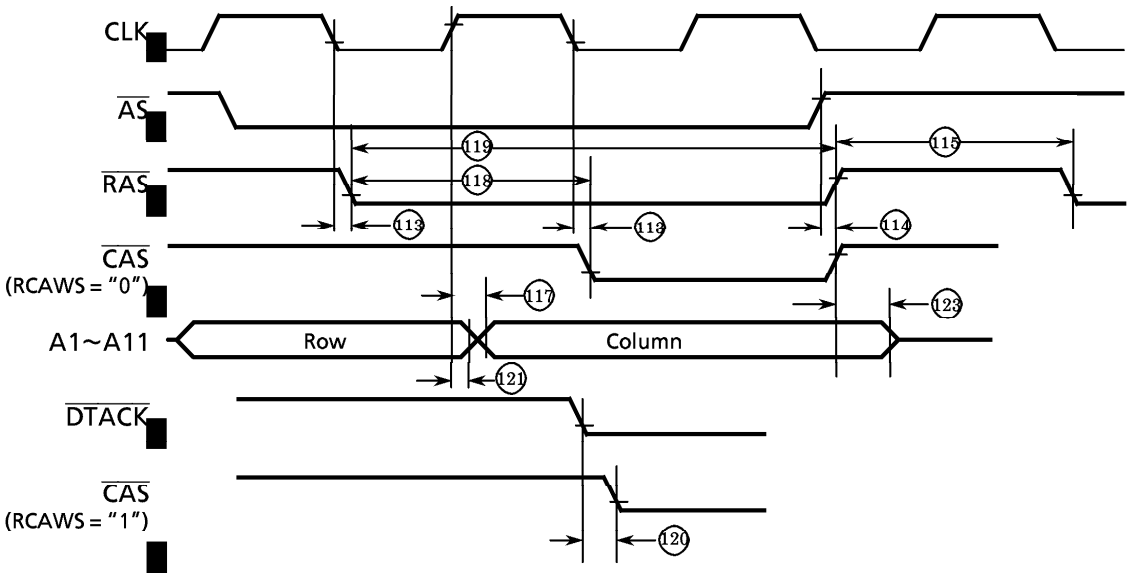


Figure 13.10 DRAM Read/Write Timing

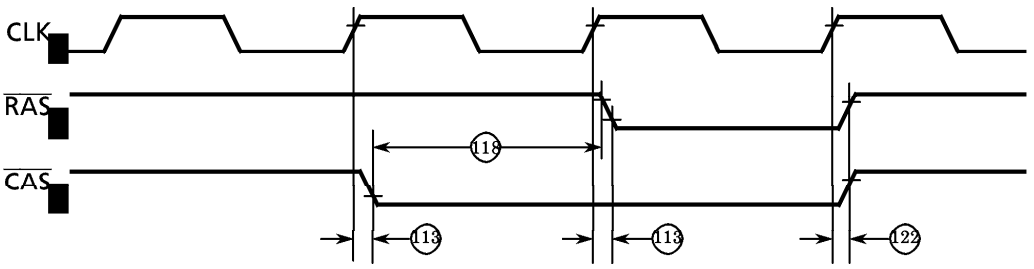


Figure 13.11 DRAM Refresh Timing

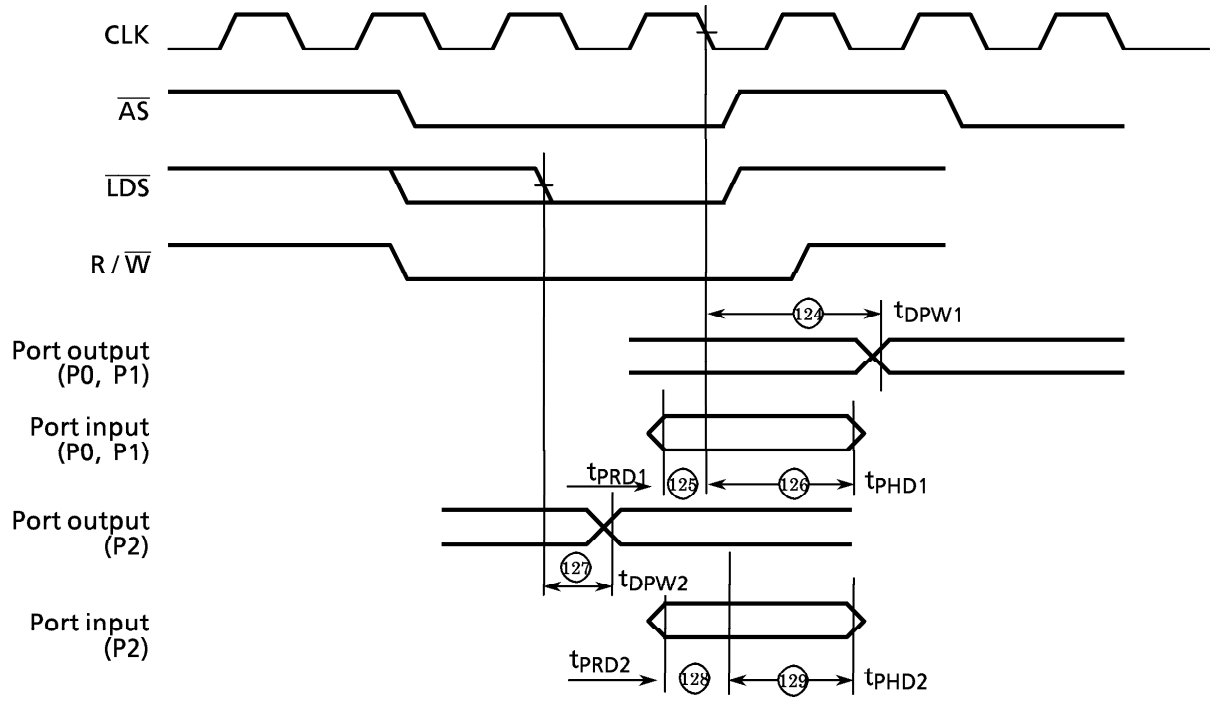


Figure 13.12 I/O Port Timing

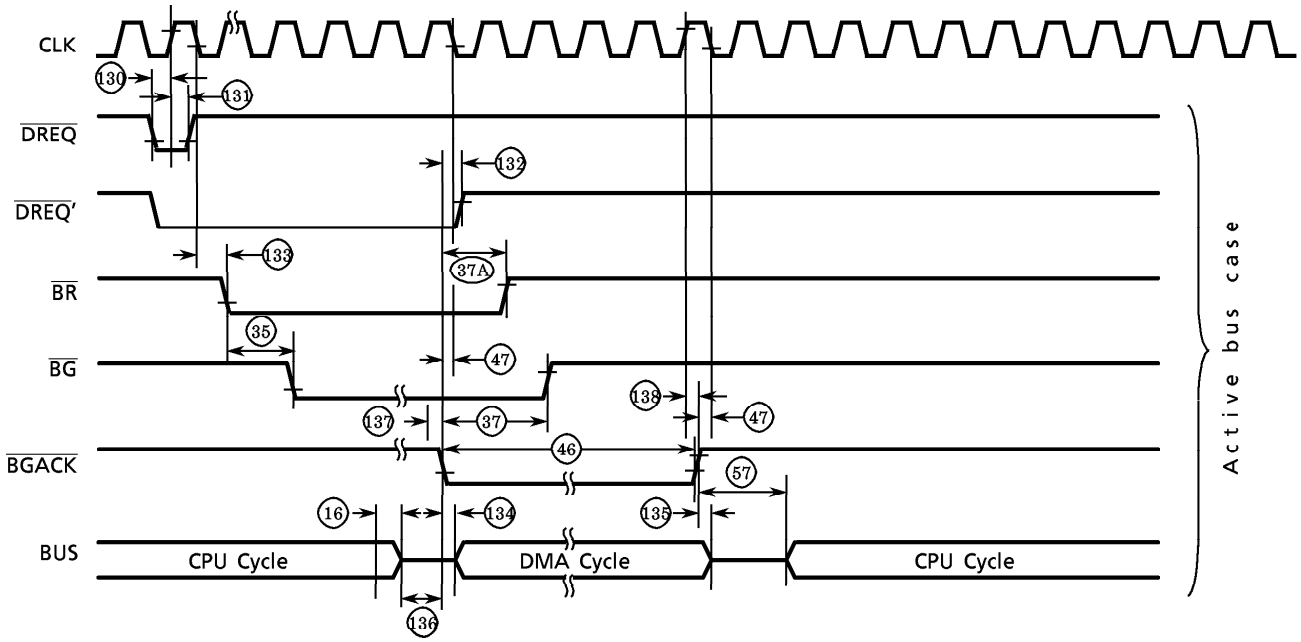


Figure 13.13 DMA Bus Arbitration Timing

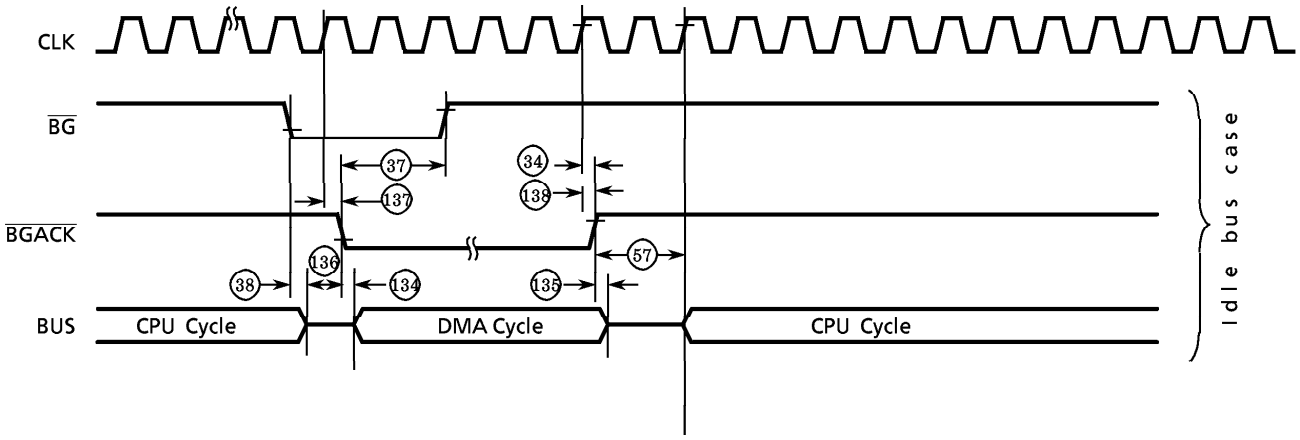


Figure 13.14 DMA/Bus Arbitration Timing

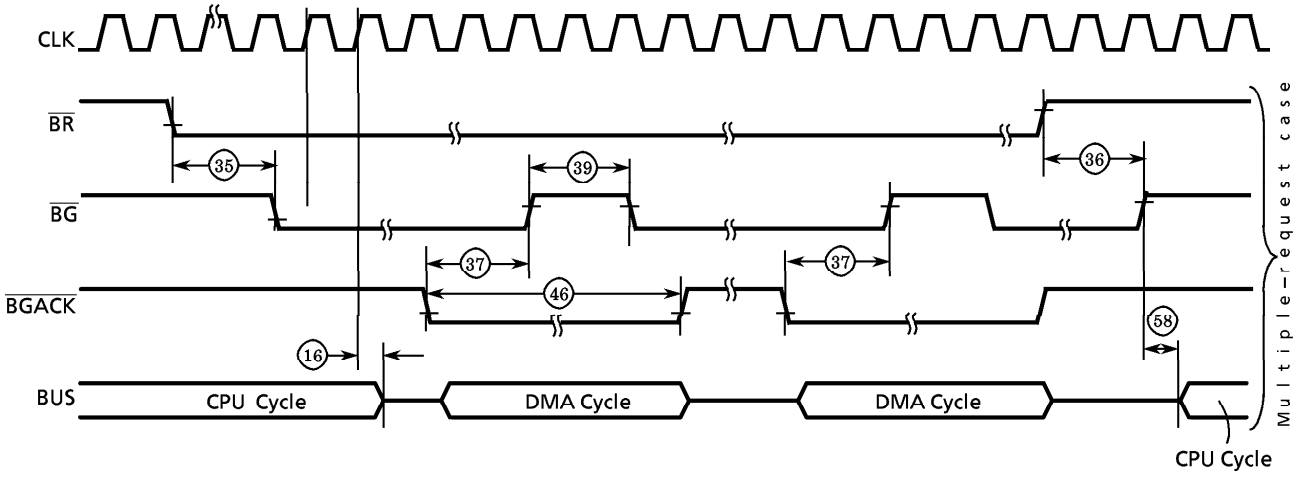


Figure 13.15 DMA/Bus Arbitration Timing

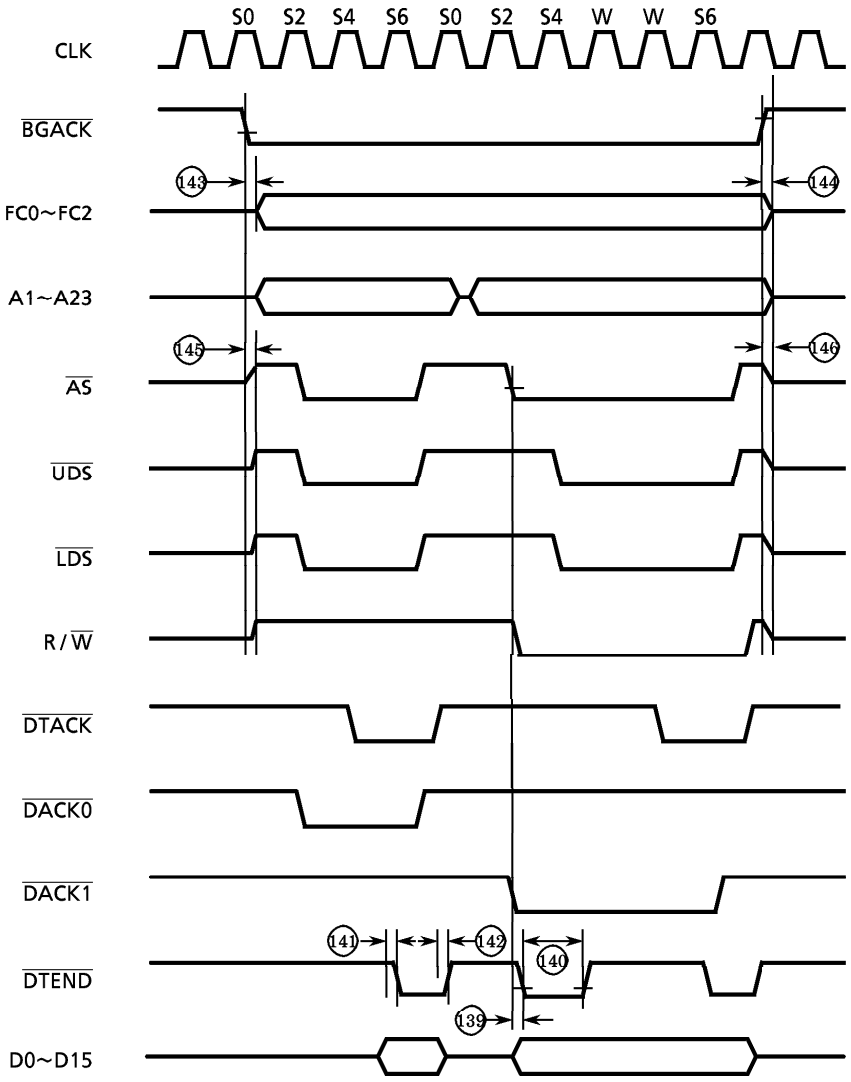


Figure 13.16 DMA Timing

14. Development Environment

14.1 Emulation mode

To put TMP68303 in emulation mode (henceforth, EMU mode), set the NOR/$\overline{\text{EMU}}$ pin to low. In EMU mode, the TMP68303 CPU is completely disconnected and only peripheral devices are active. In EMU mode, the emulator controls the peripheral devices.

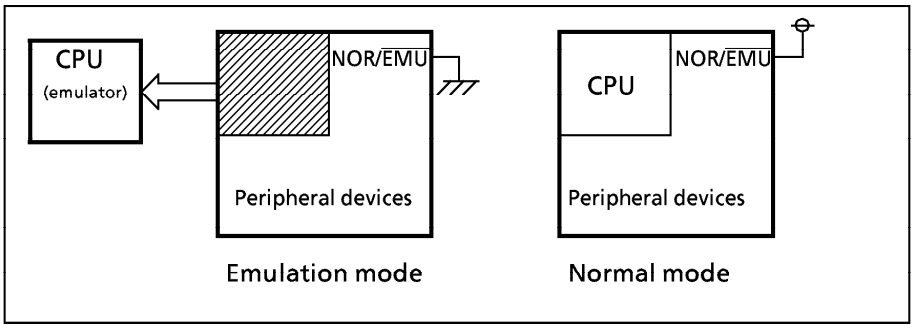


Figure 14.1 Outline of Normal and Emulation Modes

14.1.1 Pin Functions

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

(2) $\overline{\text{IPL0}-2}$: In EMU mode, IPL0 to IPL2 are interrupt request (IPL) signals output from the TMP68303F internal peripheral circuits to the emulator. In normal mode, these signals are input signals. As they have no significance, fix to low level. (See *1 in the figures)

(3) $\overline{\text{BGIN}}$: In $\overline{\text{EMU}}$ mode, this pin receives $\overline{\text{BG}}$ signals from the emulator. If there is an external bus master, connect the signal output from the TMP68303F $\overline{\text{BG}}$ pin to the external bus master.
As these signals have no significance in normal mode, fix to high level. (See *1 in the figures)

14.2 Connecting to General-Purpose 68000 Emulator

In principle, TMP68303F emulation can be performed using any commercially available general-purpose 68000 emulator. Note, however, the following points:

(1) When starting the DRAM controller and performing read/write operations to DRAM, the row and column addresses output by the DRAM controller to pins A1 to A9 (A10/A11) contend with addresses output from the emulator. To avoid this contention, an external circuit* is required. (When an external bus master runs the DRAM controller, this circuit is necessary regardless of the mode.)
※ : Supported by adaptor board BM1155.

(2) As TMP68303F 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.

(3) The 68000 interrupt vector automatic generation function cannot be used because TMP68303F does not have a $\overline{VPA}$ signal. However, TMP68303F has a similar function in its internal interrupt controller. For interrupt requests from external devices, use the interrupt input pins INT0 to INT2 connected to the interrupt controller instead of using the IPL. In EMU mode, TMP68303F pins IPL0 to IPL2 are used for interrupt requests from internal peripheral devices or from external devices (interrupt requests to INT0 - INT2).

(4) $\overline{RESET}/\overline{HALT}$ input to TMP68303F must be asserted at least 12 clocks to match the built-in watchdog timer.

(5) The $\overline{DTACK}$ and $\overline{BERR}$ signals are open drain outputs in -EMU mode. Therefore, the $\overline{DTACK}$ 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 1KΩ.

(6) The following problems can occur due to a slow rise time for the $\overline{AS}$ 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 14.4) 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 14.4)

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 clocks to the emulator and to 68303, as shown in Figure 14.5.

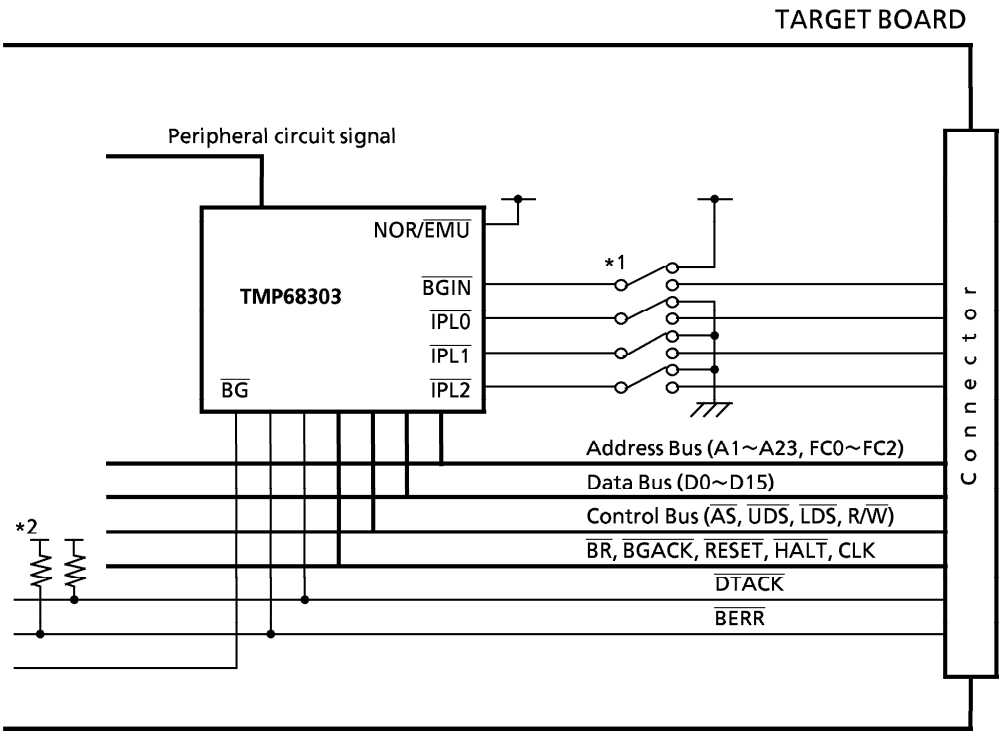


Figure 14.2 Wiring in Normal Mode

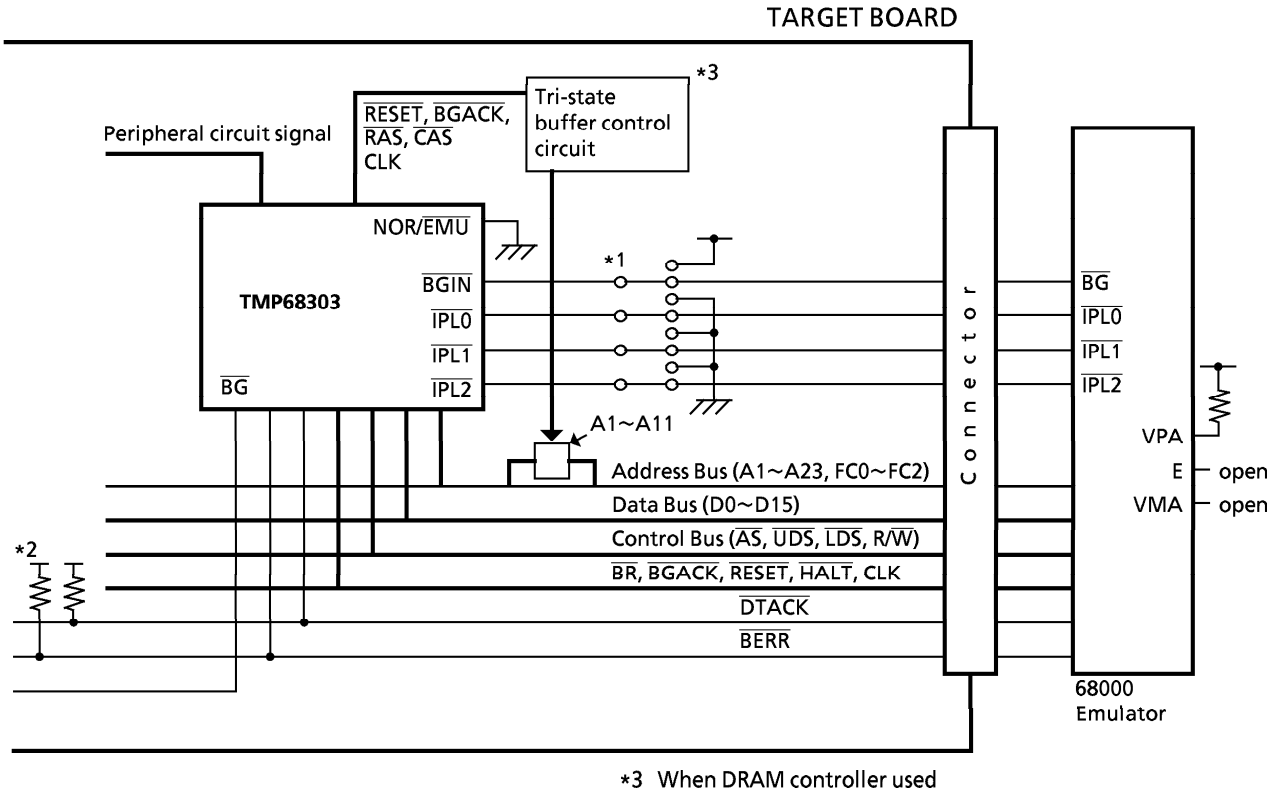


Figure 14.3 Connection to Emulator in EMU Mode

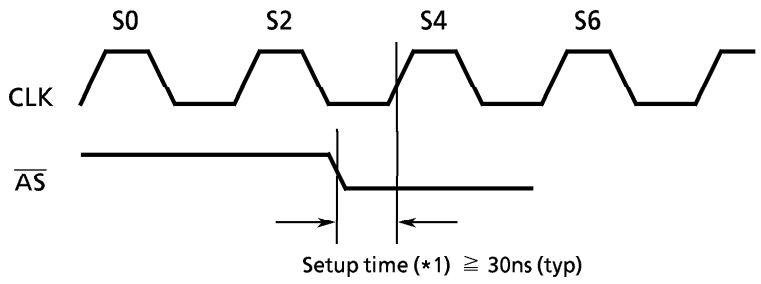


Figure 14.4 $\overline{AS}$ Signal Setup Time during Emulation

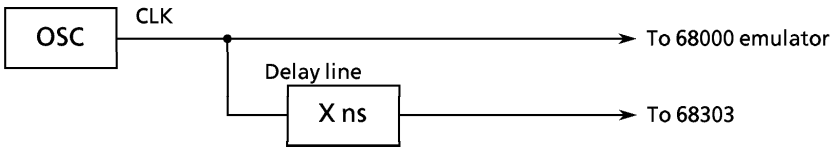


Figure 14.5 Clock Adjustments for Emulation

14.3 Adapter Board

The adapter board (BM1155) is provided for debugging user systems which utilize the TLCS-68000 ASSP family, TMP68303F. It consists of interface circuits connected to the emulator. Figure 14.6 shows connection to the emulator. Figure 14.7 shows the circuits.

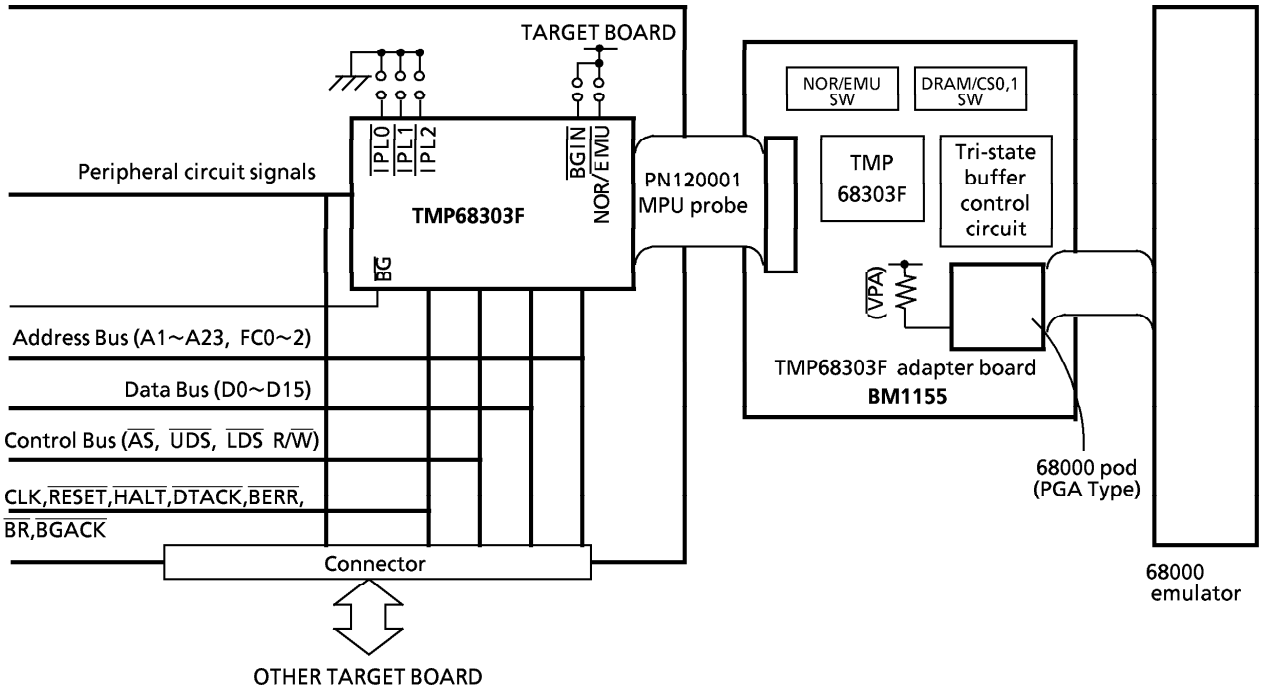


Figure 14.6 Adapter Board Connection to Emulator

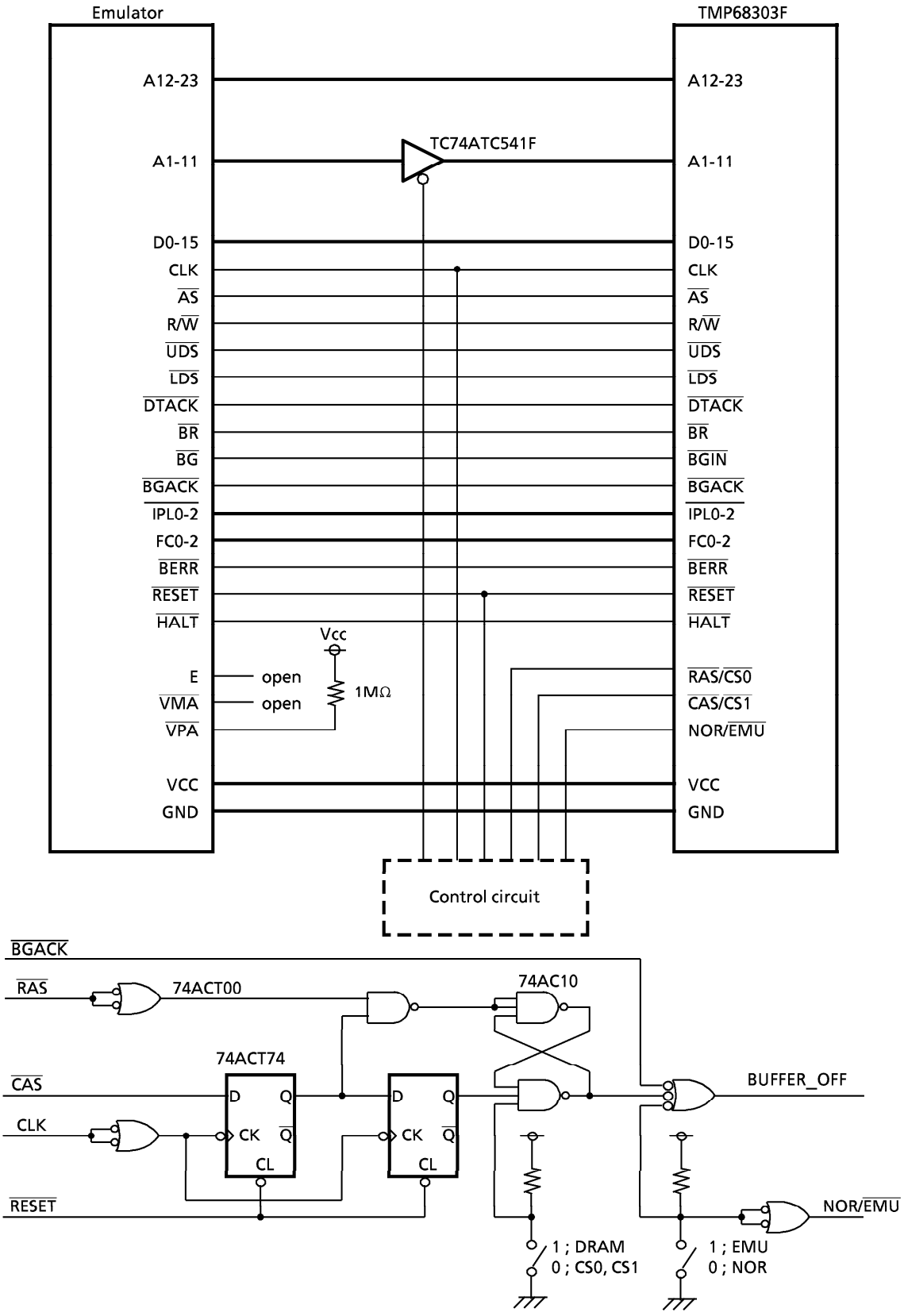
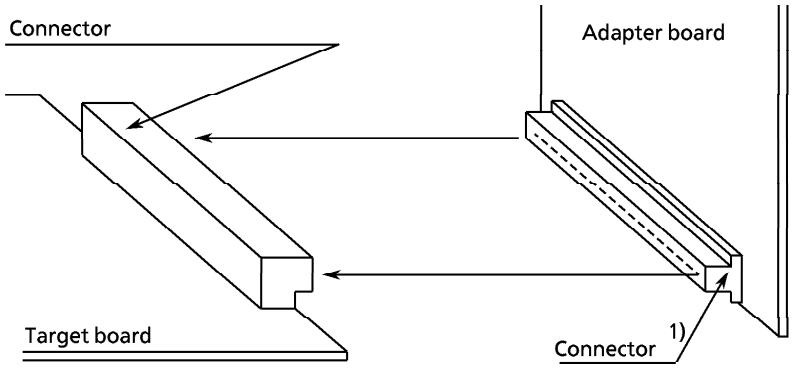


Figure 14.7 Control Circuits

Note: When using the above circuit, design according to the AC timing specifications.

(1) Target board connection example

Connect the adapter board (BM1155) connector to the target board connector directly (Figure 14.8). TMP68303F can be connected either to BM1155 or the target board. Using the optional MPU probe (PN120001), BM1155 may be directly soldered on the mounting pattern on the target board (Figure 14.10).



1) The example uses a straight-type connector. If a right-angle-type connector is used, the target board and the adapter board are connected vertically instead of horizontally.

Figure 14.8

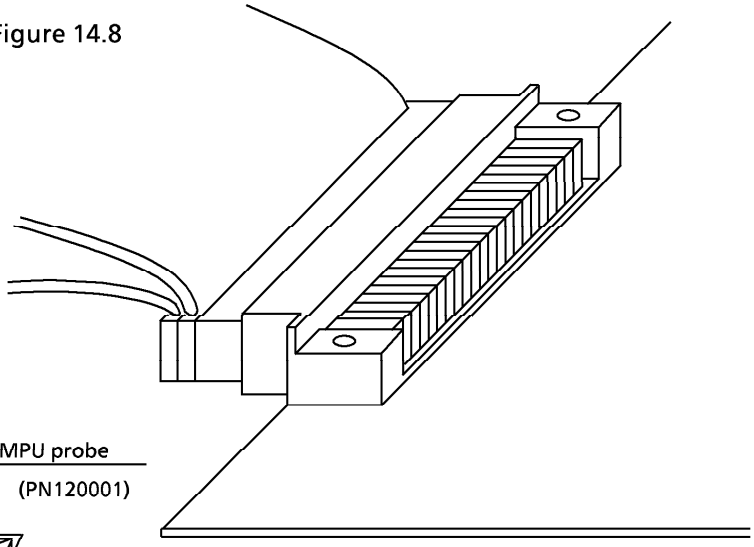


Figure 14.9

※ When cable is used as shown in Figure 14. 9, the BM1155 connector cannot be used. For details, refer to the BM1155 Instruction Manual.

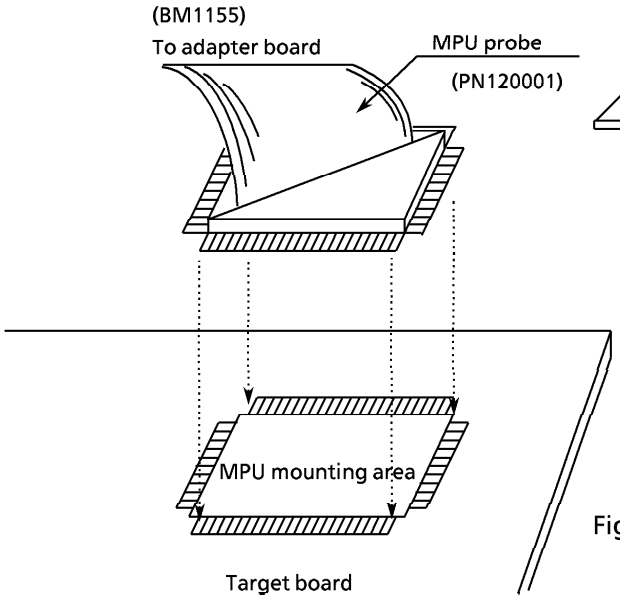


Figure 14.10