FUJITSU SEMICONDUCTOR
DATA SHEET

DS07-13705-1E

16-bit Proprietary Microcontroller

CMOS

F^{2}MC-16LX MB90595 Series

MB90598/F598/V595

■ DESCRIPTION

The MB90595-series with FULL-CAN and FLASH ROM is specially designed for automotive and industrial applications. Its main feature is the on board CAN Interface, that is conform to V2.0 Part A and Part B, supporting very flexible message buffering. Thus, offering more functions than a normal full CAN approach. In the new 0.5µm Technology Fujitsu now also offer FLASH-ROM. An internal voltage booster substitutes the necessity of a second programming voltage.
An on board voltage regulator provides 3V to the internal MCU core. This constitutes a major advantage in terms of EMI and power consumption.
The internal PLL clock frequency multiplier, provides an internal 62.5 nsec instruction cycle time with an external 4 MHz clock.
The unit features 4 Stepping Motor Controllers with high current outputs.
Further more it features 4 channels Output Capture Units and 4 channels Input Capture Units with a 16-bit free running timer. Two UARTs constitute additional functionality for communication purposes.

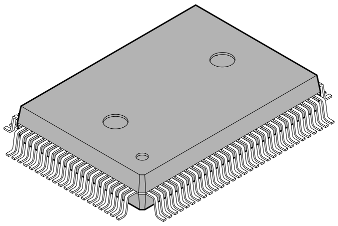
■ FEATURES

- 16-bit core CPU; 4MHz external clock (16 MHz internal, 62.5 nsec instr. cycle time)
- New 0.5 µm CMOS Process Technology
- Internal voltage regulator supports 3V MCU core, offering low EMI and low power consumption figures
- FULL-CAN interface; conform to Version 2.0 Part A and Part B, flexible message buffering (mailbox and FIFO buffering can be mixed)
- EI^{2}OS - Automatic transfer function indep.of CPU; 10 ch. of intelligent I/O Services
- 18-bit Time-base counter

(Continued)

■ PACKAGE

100-pin Plastic QFP



(FPT-100P-M06)

- Powerful interrupt functions (8 progr. priority levels; 8 external interrupts)
- Watchdog Timer
- 2 full duplex UARTs; UART0 supports 10.4 KBaud (USA standard), UART 1 also for serial transfer with clock (SCI) programmable
- Serial I/O: 1ch for synchronous data transfer
- A/D Converter: 8 ch. analog inputs (Resolution 10 bits or 8 bits)
- 16-bit reload timer * 2ch
- ICU (Input capture) 16bit * 4 ch
- OCU (Output compare) 16bit * 4ch
- 16-bit Programmable Pulse Generator 6ch
- Stepping Motor Controller 4ch
- Optimized instruction set for controller applications (bit, byte, word and long-word data types; 23 different addressing modes; barrel shift; variety of pointers)
- 4-byte instruction execution queue
- signed multiply (16bit*16bit) and divide (32bit/16bit) instructions available
- Program Patch Function
- Fast Interrupt processing
- Low Power Consumption - 7 different power saving modes: (Sleep, Stop, CPU intermittent mode, Hardware standby pin,...)
- Package: 100-pin plastic QFP

Controller Area Network (CAN) - License of Robert Bosch GmbH

■ PRODUCT LINEUP

The following table provides a quick outlook of the MB90595 Series

Features	MB90V595	MB90F598	MB90598
CPU	F^{2}MC-16LX CPU		
System clock	On-chip PLL clock multiplier (x1, x2, x3, x4, 1/2 when PLL stop) Minimum instruction execution time: 62.5 ns (4 MHz osc. PLL x4)		
ROM	External	Boot-block Flash memory 128 Kbytes Hard-wired reset vector	Mask ROM 128 Kbytes
RAM	6 Kbytes	4 Kbytes	4 Kbytes
Technology	0.5 µm CMOS with on-chip voltage regulator for internal power supply	0.5 µm CMOS with on-chip voltage regulator for internal power supply + Flash memory On-chip charge pump for programming voltage	0.5 µm CMOS with on-chip voltage regulator for internal power supply
Operating voltage range	5 V ± 10%		
Temperature range	– 40 to 85 °C		
Package	PGA-256	QFP100	
UART0	Full duplex double buffer Supports asynchronous/synchronous (with start/stop bit) transfer Baud rate: 4808/5208/9615/10417/19230/38460/62500/500000bps (asynchronous) 500K/1M/2Mbps (synchronous) at System clock = 16MHz		
UART1 (SCI)	Full duplex double buffer Asynchronous (start-stop synchronized) and CLK-synchronous communication Baud rate: 1202/2404/4808/9615/31250bps (asynchronous) 62.5K/125K/250K/500K/1Mbps (synchronous) at 6,8,10,12,16 MHz		
Serial IO	Transfer can be started from MSB or LSB Supports internal clock synchronized transfer and external clock synchronized transfer Supports positive-edge and negative-edge clock synchronization Baud rate : 31.25K/62.5K/125K/500K/1Mbps at System clock = 16MHz		
A/D Converter	10-bit or 8-bit resolution 8 input channels Conversion time: 26.3µs (per one channel)		
16-bit Reload Timer (2 channels)	Operation clock frequency: fsys/2^1, fsys/2^3, fsys/2^5 (fsys = System clock frequency) Supports External Event Count function		
Stepper Motor Controller (4 channels)	Four high current outputs for each channel Synchronized two 8-bit PWM’s for each channel Succeeds to MB89940 design resource		
16-bit IO Timer	Signals an interrupt when overflow Supports Timer Clear when a match with Output Compare(Channel 0) Operation clock freq.: fsys/2^2, fsys/2^4, fsys/2^6, fsys/2^8(fsys = System clock freq.)		

(Continued)

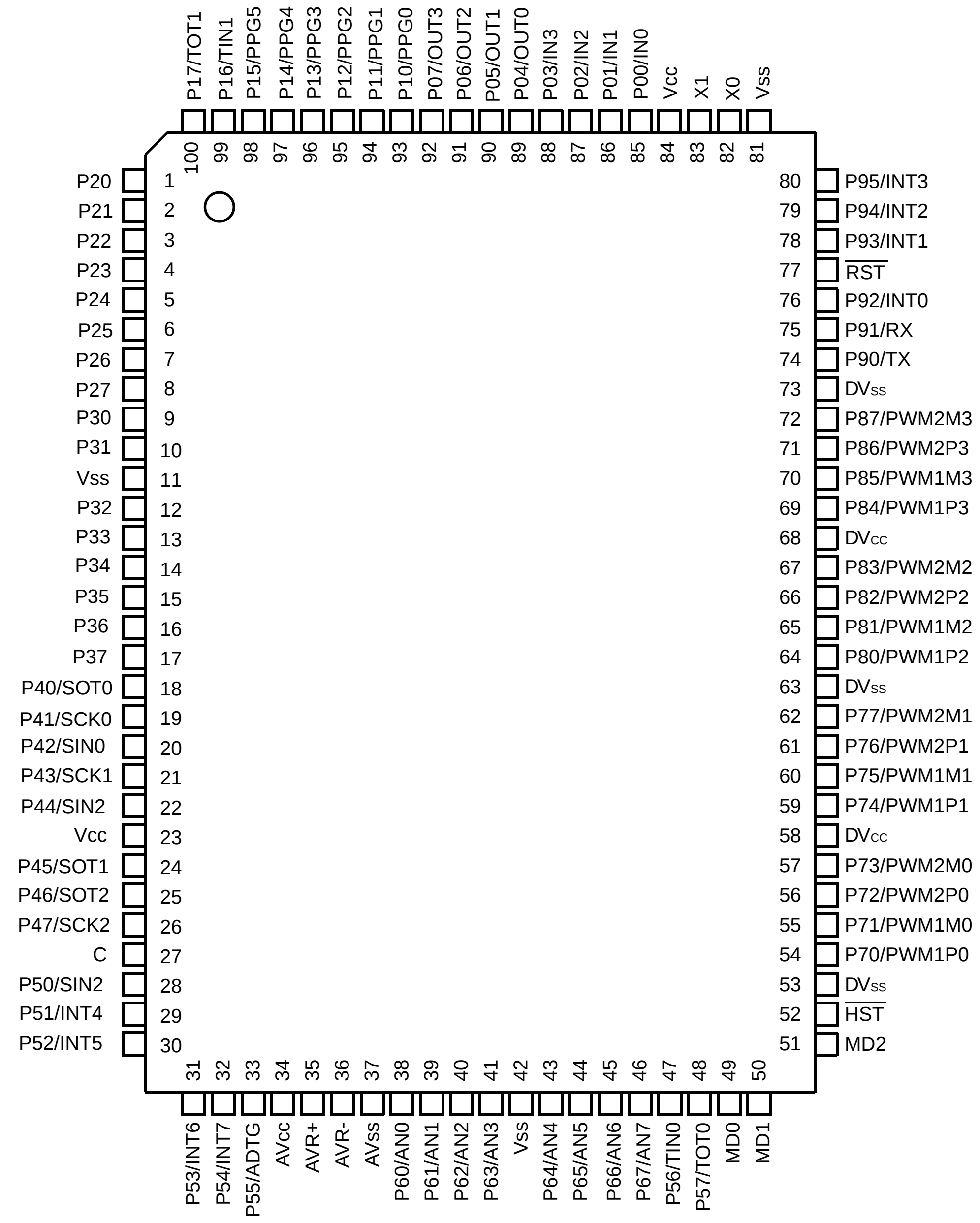
(Continued)

Features	MB90V595	MB90F598	MB90598
16-bit Output Compare (4 channels)	Signals an interrupt when a match with 16-bit IO Timer Four 16-bit compare registers A pair of compare registers can be used to generate an output signal		
16-bit Input Capture (4 channels)	Rising edge, falling edge or rising & falling edge sensitive Four 16-bit Capture registers Signals an interrupt upon external event		
8/16-bit Programmable Pulse Generator (6channels)	Supports 8-bit and 16-bit operation modes Twelve 8-bit reload counters Twelve 8-bit reload registers for L pulse width Twelve 8-bit reload registers for H pulse width A pair of 8-bit reload counters can be configured as one 16-bit reload counter or as 8-bit prescaler plus 8-bit reload counter 6 output pins Operation clock freq.: fsys, fsys/2^1, fsys/2^2, fsys/2^3, fsys/2^4 or 128µs@fosc=4MHz (fsys = System clock frequency, fosc = Oscillation clock frequency)		
CAN Interface	Conforms to CAN Specification Version 2.0 Part A and B Automatic re-transmission in case of error Automatic transmission responding to Remote Frame Prioritized 16 message buffers for data and ID’s Supports multiple messages Flexible configuration of acceptance filtering: Full bit compare / Full bit mask / Two partial bit masks Supports up to 1Mbps		
External Interrupt (8 channels)	Can be programmed edge sensitive or level sensitive		
IO Ports	Virtually all external pins can be used as general purpose IO All push-pull outputs and schmitt trigger inputs Bit-wise programmable as input/output or peripheral signal		
Flash Memory	—	Supports automatic programming, Embedded Algorithm$^{\text{TM}}$*1 Write/Erase/Erase-Suspend/Resume commands A flag indicating completion of the algorithm Number of erase cycles: 10,000 times Data retention time: 10 years Hard-wired reset vector available in order to point to a fixed boot sector in Flash Memory Flash Writer from Minato Electronics Inc. Boot block configuration Erase can be performed on each block Block protection with external programming voltage	—

*1: Embedded Algorithm is a trade mark of Advanced Micro Devices Inc

■ PIN ASSIGNMENT

(Top view)



(FPT-100P-M06)

■ PIN DESCRIPTION

No.	Pin name	Circuit type	Function
82	X0	A	Oscillation input
83	X1		Oscillation output
77	$\overline{\text{RST}}$	B	Reset input
52	$\overline{\text{HST}}$	C	Hardware standby input
85 to 88	P00 to P03	G	General purpose IO
	IN0 to IN3		Inputs for the Input Captures
89 to 92	P04 to P07	G	General purpose IO
	OUT0 to OUT3		Outputs for the Output Compares.
93 to 98	P10 to P15	D	General purpose IO
	PPG0 to PPG5		Outputs for the Programmable Pulse Generators
99	P16	D	General purpose IO
	TIN1		TIN input for the 16-bit Reload Timer 1
100	P17	D	General purpose IO
	TOT1		TOT output for the 16-bit Reload Timer 1
1 to 8	P20 to P27	G	General purpose IO
9 to 10	P30 to P31	G	General purpose IO
12 to 16	P32 to P36	G	General purpose IO
17	P37	D	General purpose IO
18	P40	G	General purpose IO
	SOT0		SOT output for UART 0
19	P41	G	General purpose IO
	SCK0		SCK input/output for UART 0
20	P42	G	General purpose IO
	SIN0		SIN input for UART 0
21	P43	G	General purpose IO
	SIN1		SIN input for UART 1
22	P44	G	General purpose IO
	SCK1		SCK input/output for UART 1
24	P45	G	General purpose IO
	SOT1		SOT output for UART 1
25	P46	G	General purpose IO
	SOT2		SOT output for the Serial IO
26	P47	G	General purpose IO
	SCK2		SCK input/output for the Serial IO

(Continued)

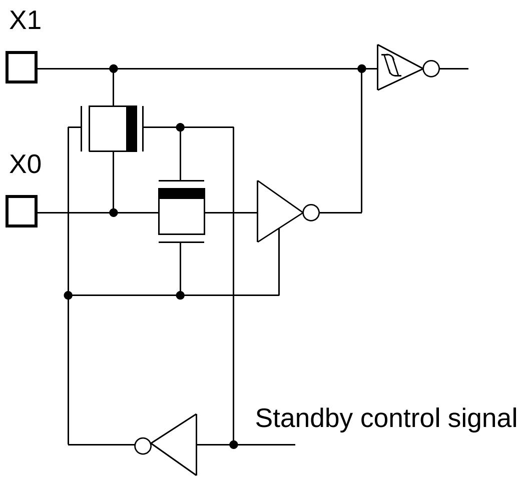
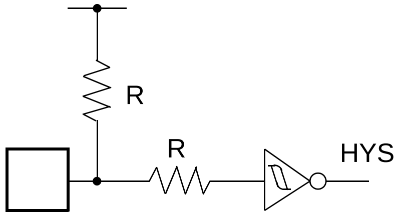
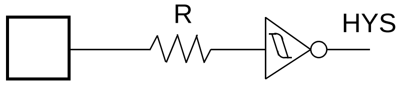
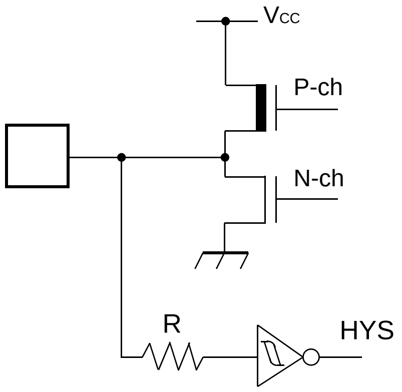
No.	Pin name	Circuit type	Function
28	P50	D	General purpose IO
	SIN2		SIN Input for the Serial IO
29 to 32	P51 to P54	D	General purpose IO
	INT4 to INT7		External interrupt input for INT4 to INT7
33	P55	D	General purpose IO
	ADTG		Input for the external trigger of the A/D Converter
38 to 41	P60 to P63	E	General purpose IO
	AN0 to AN3		Inputs for the A/D Converter
43 to 46	P64 to P67	E	General purpose IO
	AN4 to AN7		Inputs for the A/D Converter
47	P56	D	General purpose IO
	TIN0		TIN input for the 16-bit Reload Timer 0
48	P57	D	General purpose IO
	TOT0		TOT output for the 16-bit Reload Timer 0
54 to 57	P70 to P73	F	General purpose IO
	PWM1P0 PWM1M0 PWM2P0 PWM2M0		Output for Stepper Motor Controller channel 0
59 to 62	P74 to P77	F	General purpose IO
	PWM1P1 PWM1M1 PWM2P1 PWM2M1		Output for Stepper Motor Controller channel 1
64 to 67	P80 to P83	F	General purpose IO
	PWM1P2 PWM1M2 PWM2P2 PWM2M2		Output for Stepper Motor Controller channel 2
69 to 72	P84 to P87	F	General purpose IO
	PWM1P3 PWM1M3 PWM2P3 PWM2M3		Output for Stepper Motor Controller channel 3
74	P90	D	General purpose IO
	TX		TX output for CAN Interface
75	P91	D	General purpose IO
	RX		RX input for CAN Interface

(Continued)

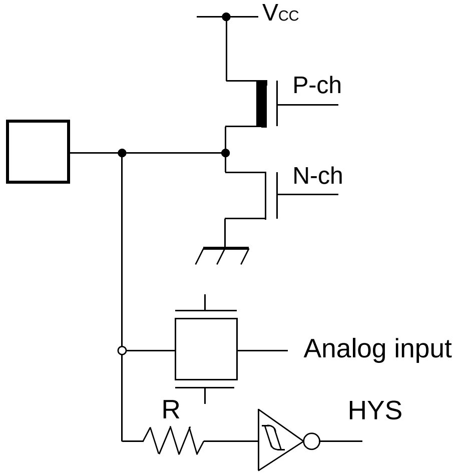
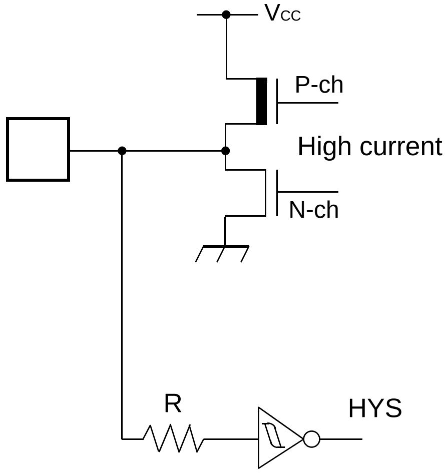
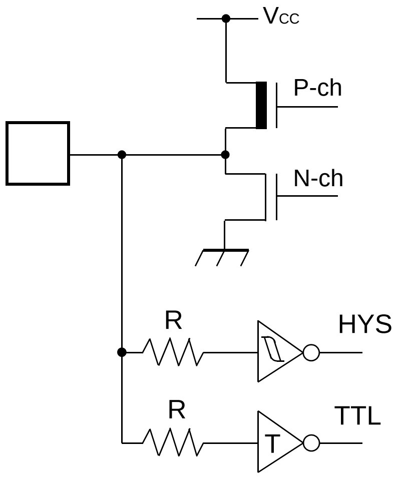
(Continued)

No.	Pin name	Circuit type	Function
76	P92	D	General purpose IO
	INT0		External interrupt input for INT0
78 to 80	P93 to P95	D	General purpose IO
	INT1 to INT3		External interrupt input for INT1 to INT3
58, 68	DV$_{CC}$		Dedicated power supply pins for the high current outputbuffers (Pin No. 54 to 72)
53, 63, 73	DV$_{SS}$		Dedicated ground pins for the high current output buffers (Pin No. 54 to 72)
34	AV$_{CC}$		Dedicated power supply pin for the A/D Converter
37	AV$_{SS}$		Dedicated ground pin for the A/D Converter
35	AVR+		Upper reference voltage input for the A/D Converter
36	AVR-		Lower reference voltage input for the A/D Converter
49, 50	MD0 MD1	C	Test mode inputs. These pins should be connected to VCC
51	MD2	H	Test mode input. This pin should be connected to VSS
27	C		External capacitor pin. A capacitor of 0.1µF should be connected to this pin and VSS.
23, 84	V$_{CC}$		Power supply pins
11, 42, 81	V$_{SS}$		Ground pins

■ I/O CIRCUIT TYPE

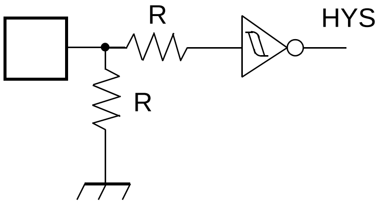
Circuit Type	Circuit	Remarks
A		• Oscillation feedback resistor: 1 MΩ approx.
B		• Hysteresis input with pull-up Resistor: 50 kΩ approx.
C		• Hysteresis input
D		• CMOS output • Hysteresis input

(Continued)

Circuit Type	Circuit	Remarks
E		• CMOS output • Hysteresis input • Analog input
F		• CMOS high current output • Hysteresis input
G		• CMOS output • Hysteresis input • TTL input (MB90F598, only in Flash mode)

(Continued)

(Continued)

Circuit Type	Circuit	Remarks
H		• Hysteresis input Pull-down Resistor: 50 Ω approx. (except MB90F598)

■ HANDLING DEVICES

(1)Preventing latch-up

CMOS IC chips may suffer latch-up under the following conditions:

- A voltage higher than Vcc or lower than Vss is applied to an input or output pin.
- A voltage higher than the rated voltage is applied between Vcc and Vss.
- The AVcc power supply is applied before the Vcc voltage.

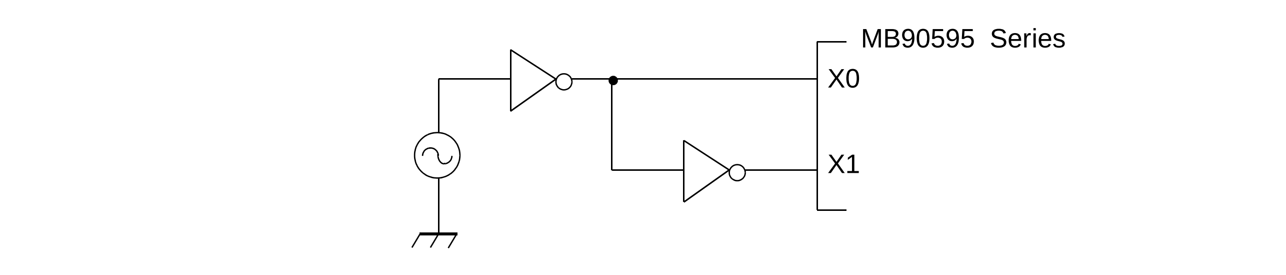
Latch-up may increase the power supply current drastically, causing thermal damage to the device.

(2)Handling unused input pins

Do not leave unused input pins open, as doing so may cause misoperation of the device. Use a pull-up or pull-down resistor.

(3)Using external clock

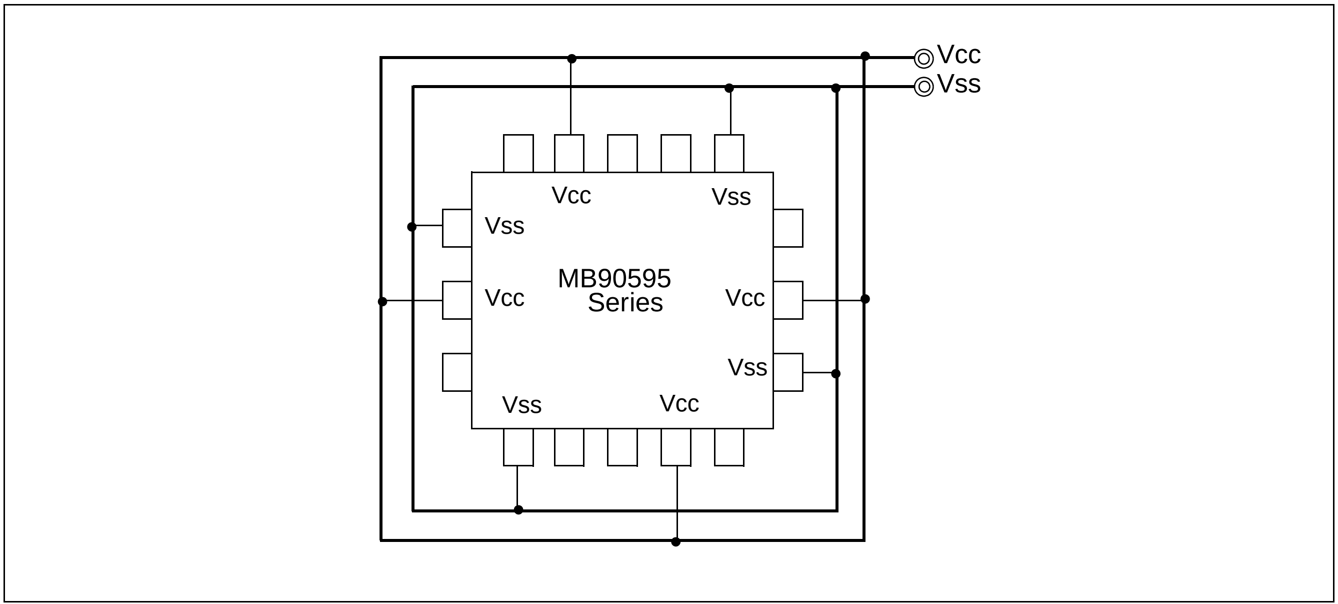
To use external clock, drive the X0 and X1 pins in reverse phase. See diagram below.



Using external clock

(4)Power supply pins (Vcc/Vss)

Ensure that all Vcc-level power supply pins are at the same potential. In addition, ensure the same for all Vss-level power supply pins. (See the figure below.) If there are more than one Vcc or Vss system, the device may operate incorrectly even within the guaranteed operating range.



(5) Pull-up/down resistors

The MB90595 Series does not support internal pull-up/down resistors. Use external components where needed.

(6) Crystal Oscillator Circuit

Noises around X0 or X1 pins may be possible causes of abnormal operations. Make sure to provide bypass capacitors via shortest distance from X0, X1 pins, crystal oscillator (or ceramic resonator) and ground lines, and make sure, to the utmost effort, that lines of oscillation circuit not cross the lines of other circuits.

It is highly recommended to provide a printed circuit board art work surrounding X0 and X1 pins with an grand area for stabilizing the operation.

(7) Turning-on Sequence of Power Supply to A/D Converter and Analog Inputs

Make sure to turn on the A/D converter power supply(AV$_{CC}$, AVR + , AVR –) and analog inputs (AN0 to AN7) after turning-on the digital power supply (V$_{CC}$).

Turn-off the digital power after turning off the A/D converter supply and analog inputs. In this case, make sure that the voltage not exceed AVR + or AV$_{CC}$ (turning on/off the analog and digital power supplies simultaneously is acceptable).

(8) Connection of Unused Pins of A/D Converter

Connect unused pins of A/D converter to AV$_{CC}$ = V$_{CC}$, AV$_{SS}$ = AVR + = V$_{SS}$.

(9) N.C. Pin

The N.C. (internally connected) pin must be opened for use.

(10) Notes on Energization

To prevent the internal regulator circuit from malfunctioning, set the voltage rise time during energization at 50 or more ms (0.2 V to 2.7 V).

(11) Initialization

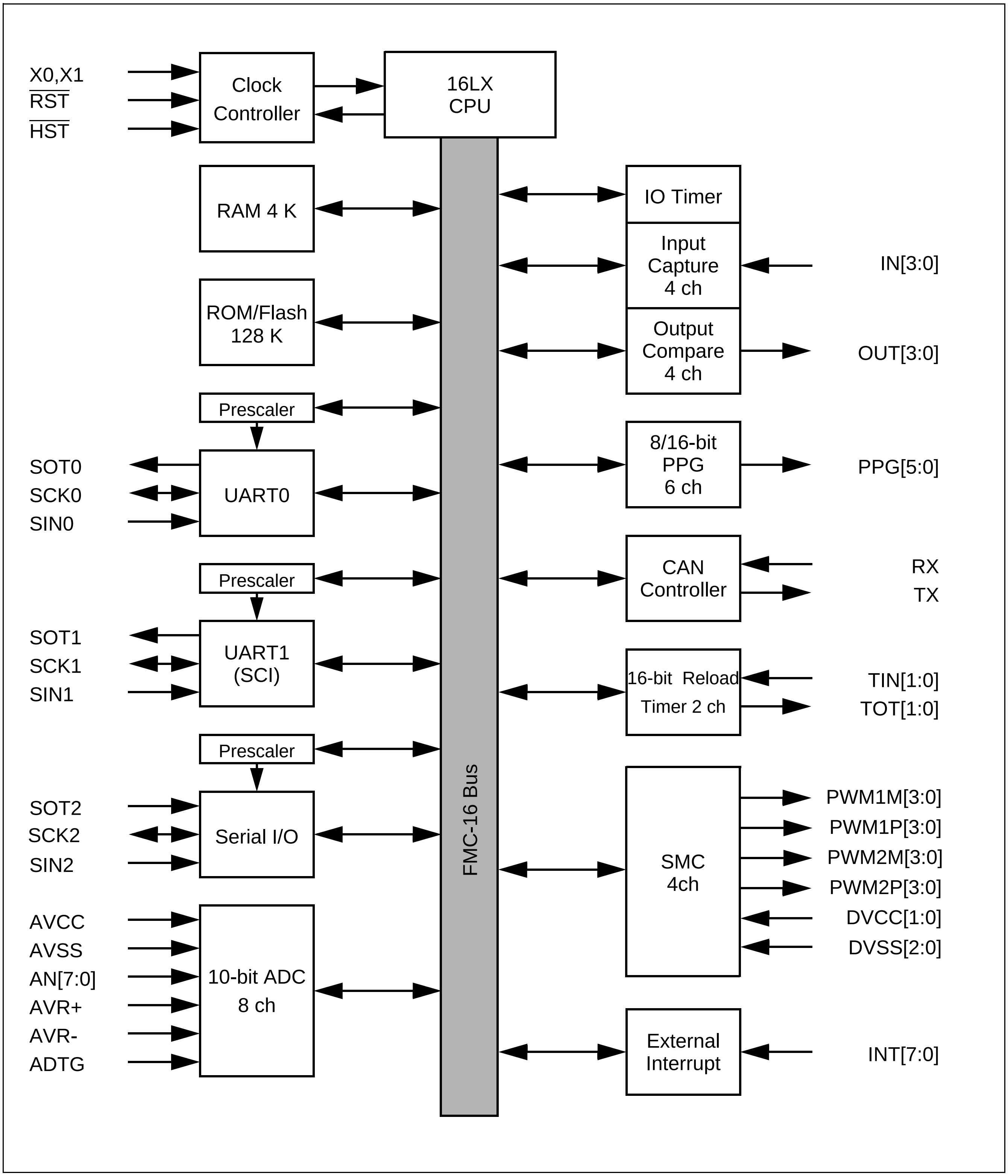
In the device, there are internal registers which is initialized only by a power-on reset. To initialize these registers turning on the power again.

(12) Directions of “DIV A, Ri” and “DIVW A, RWi” instructions

In the Signed multiplication and division instructions (“DIV A, Ri” and “DIVW A, RWi”), the value of the corresponding bank register (DTB, ADB, USB, SSB) is set in “00h”.

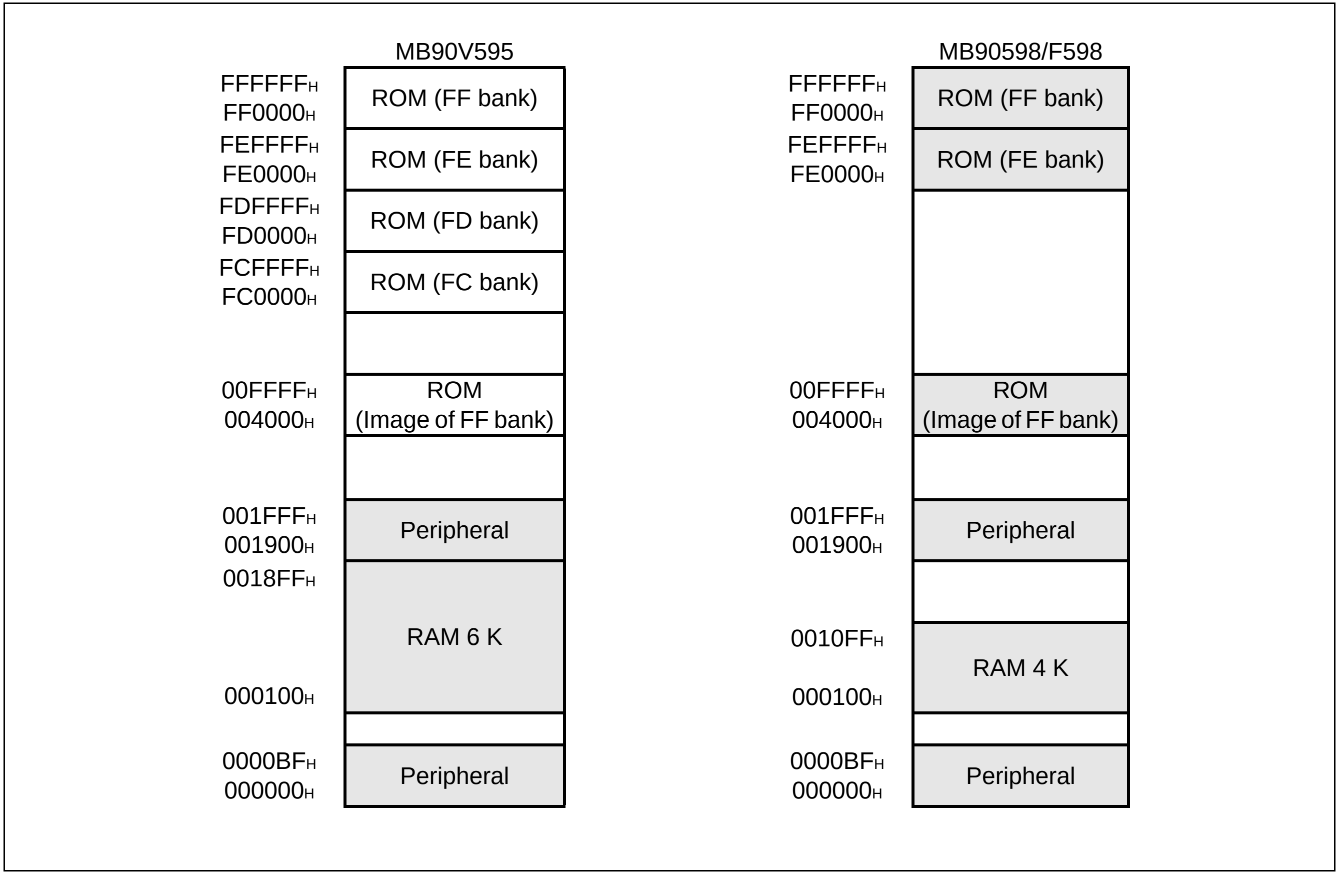
If the values of the corresponding bank register (DTB,ADB,USB,SSB) are setting other than “00h”, the remainder by the execution result of the instruction is not stored in the register of the instruction operand.

■ BLOCK DIAGRAM



■ MEMORY SPACE

The memory space of the MB90595 Series is shown below



Memory space map

The high-order portion of bank 00 gives the image of the FF bank ROM to make the small model of the C compiler effective. Since the low-order 16 bits are the same, the table in ROM can be referenced without using the far specification in the pointer declaration.
For example, an attempt to access 00C000H accesses the value at FFC000H in ROM.
The ROM area in bank FF exceeds 48 Kbytes, and its entire image cannot be shown in bank 00.
The image between FF4000H and FFFFFFH is visible in bank 00, while the image between FF0000H and FF3FFFH is visible only in bank FF.

■ I/O MAP

Address	Register	Abbreviation	Access	Pripheral	Initial value
00 H	Port 0 data register	PDR0	R/W	Port 0	XXXXXXXX B
01 H	Port 1 data register	PDR1	R/W	Port 1	XXXXXXXX B
02 H	Port 2 data register	PDR2	R/W	Port 2	XXXXXXXX B
03 H	Port 3 data register	PDR3	R/W	Port 3	XXXXXXXX B
04 H	Port 4 data register	PDR4	R/W	Port 4	XXXXXXXX B
05 H	Port 5 data register	PDR5	R/W	Port 5	XXXXXXXX B
06 H	Port 6 data register	PDR6	R/W	Port 6	XXXXXXXX B
07 H	Port 7 data register	PDR7	R/W	Port 7	XXXXXXXX B
08 H	Port 8 data register	PDR8	R/W	Port 8	XXXXXXXX B
09 H	Port 9 data register	PDR9	R/W	Port 9	_ _ XXXXXX B
0A to 0F H	Reserved				
10 H	Port 0 direction register	DDR0	R/W	Port 0	0 0 0 0 0 0 0 0 B
11 H	Port 1 direction register	DDR1	R/W	Port 1	0 0 0 0 0 0 0 0 B
12 H	Port 2 direction register	DDR2	R/W	Port 2	0 0 0 0 0 0 0 0 B
13 H	Port 3 direction register	DDR3	R/W	Port 3	0 0 0 0 0 0 0 0 B
14 H	Port 4 direction register	DDR4	R/W	Port 4	0 0 0 0 0 0 0 0 B
15 H	Port 5 direction register	DDR5	R/W	Port 5	0 0 0 0 0 0 0 0 B
16 H	Port 6 direction register	DDR6	R/W	Port 6	0 0 0 0 0 0 0 0 B
17 H	Port 7 direction register	DDR7	R/W	Port 7	0 0 0 0 0 0 0 0 B
18 H	Port 8 direction register	DDR8	R/W	Port 8	0 0 0 0 0 0 0 0 B
19 H	Port 9 direction register	DDR9	R/W	Port 9	_ _ 0 0 0 0 0 0 B
1A H	Reserved				
1B H	Analog Input Enable	ADER	R/W	Port 6, A/D	1 1 1 1 1 1 1 1 B
1C to 1F H	Reserved				
20 H	Serial Mode Control Register 0	UMC0	R/W	UART0	0 0 0 0 0 1 0 0 B
21 H	Status Register 0	USR0	R/W		0 0 0 1 0 0 0 0 B
22 H	Input/Output Data Register 0	UIDR0/ UODR0	R/W		XXXXXXXX B
23 H	Rate and Data Register 0	URD0	R/W		0 0 0 0 0 0 0 X B
24 H	Serial Mode Register 1	SMR1	R/W	UART1	0 0 0 0 0 0 0 0 B
25 H	Serial Control Register 1	SCR1	R/W		0 0 0 0 0 1 0 0 B
26 H	Input/Output Data Register 1	SIDR1/ SODR1	R/W		XXXXXXXX B
27 H	Serial Status Register 1	SSR1	R/W		0 0 0 0 1 _ 0 0 B
28 H	UART1 Prescaler Control Register	U1CDCR	R/W		0 _ _ _ 1 1 1 1 B

(Continued)

Address	Register	Abbreviation	Access	Pripheral	Initial value
29 to 2A H	Reserved				
2B H	Serial IO Prescaler	SCDCR	R/W	Serial IO	0 _ _ _ 1 1 1 1 B
2C H	Serial Mode Control	SMCS	R/W		_ _ _ _ 0 0 0 0 B
2D H	Serial Mode Control	SMCS	R/W		0 0 0 0 0 0 1 0 B
2E H	Serial Data	SDR	R/W		XXXXXXXX B
2F H	Edge Selector	SES	R/W		_ _ _ _ _ _ _ 0 B
30 H	External Interrupt Enable	ENIR	R/W	External Interrupt	0 0 0 0 0 0 0 0 B
31 H	External Interrupt Request	EIRR	R/W		XXXXXXXX B
32 H	External Interrupt Level	ELVR	R/W		0 0 0 0 0 0 0 0 B
33 H	External Interrupt Level	ELVR	R/W		0 0 0 0 0 0 0 0 B
34 H	A/D Control Status 0	ADCS0	R/W	A/D Converter	0 0 0 0 0 0 0 0 B
35 H	A/D Control Status 1	ADCS1	R/W		0 0 0 0 0 0 0 0 B
36 H	A/D Data 0	ADCR0	R		XXXXXXXX B
37 H	A/D Data 1	ADCR1	R/W		0 0 0 0 1 _ XX B
38 H	PPG0 operation mode control register	PPGC0	R/W	16-bit Programable Pulse Generator 0/1	0 _ 0 0 0 _ _ 1 B
39 H	PPG1 operation mode control register	PPGC1	R/W		0 _ 0 0 0 0 0 1 B
3A H	PPG0 and PPG1 clock select register	PPG01	R/W		0 0 0 0 0 0 _ _ B
3B H	Reserved				
3C H	PPG2 operation mode control register	PPGC2	R/W	16-bit Programable Pulse Generator 2/3	0 _ 0 0 0 _ _ 1 B
3D H	PPG3 operation mode control register	PPGC3	R/W		0 _ 0 0 0 0 0 1 B
3E H	PPG2 and PPG3 clock select register	PPG23	R/W		0 0 0 0 0 0 _ _ B
3F H	Reserved				
40 H	PPG4 operation mode control register	PPGC4	R/W	16-bit Programable Pulse Generator 4/5	0 _ 0 0 0 _ _ 1 B
41 H	PPG5 operation mode control register	PPGC5	R/W		0 _ 0 0 0 0 0 1 B
42 H	PPG4 and PPG5 clock select register	PPG45	R/W		0 0 0 0 0 0 _ _ B
43 H	Reserved				
44 H	PPG6 operation mode control register	PPGC6	R/W	16-bit Programable Pulse Generator 6/7	0 _ 0 0 0 _ _ 1 B
45 H	PPG7 operation mode control register	PPGC7	R/W		0 _ 0 0 0 0 0 1 B
46 H	PPG6 and PPG7 clock select register	PPG67	R/W		0 0 0 0 0 0 _ _ B
47 H	Reserved				
48 H	PPG8 operation mode control register	PPGC8	R/W	16-bit Programable Pulse Generator 8/9	0 _ 0 0 0 _ _ 1 B
49 H	PPG9 operation mode control register	PPGC9	R/W		0 _ 0 0 0 0 0 1 B
4A H	PPG8 and PPG9 clock select register	PPG89	R/W		0 0 0 0 0 0 _ _ B
4B H	Reserved				

(Continued)

Address	Register	Abbreviation	Access	Pripheral	Initial value
4C $_H$	PPGA operation mode control register	PPGCA	R/W	16-bit Programable Pulse Generator A/B	0 _ 0 0 0 _ _ 1 $_B$
4D $_H$	PPGB operation mode control register	PPGCB	R/W		0 _ 0 0 0 0 0 1 $_B$
4E $_H$	PPGA and PPGB clock select register	PPGAB	R/W		0 0 0 0 0 0 _ _ $_B$
4F $_H$	Reserved				
50 $_H$	Timer Control Status 0	TMCSR0	R/W	16-bit Reload Timer 0	0 0 0 0 0 0 0 0 $_B$
51 $_H$	Timer Control Status 0	TMCSR0	R/W		_ _ _ _ 0 0 0 0 $_B$
52 $_H$	Timer 0/Reload 0	TMR0/ TMRLR0	R/W		XXXXXXXX $_B$
53 $_H$	Timer 0/Reload 0	TMR0/ TMRLR0	R/W		XXXXXXXX $_B$
54 $_H$	Timer Control Status 1	TMCSR1	R/W	16-bit Reload Timer 1	0 0 0 0 0 0 0 0 $_B$
55 $_H$	Timer Control Status 1	TMCSR1	R/W		_ _ _ _ 0 0 0 0 $_B$
56 $_H$	Timer 1/Reload 1	TMR1/ TMRLR1	R/W		XXXXXXXX $_B$
57 $_H$	Timer 1/Reload 1	TMR1/ TMRLR1	R/W		XXXXXXXX $_B$
58 $_H$	Output Compare Control Status 0	OCS0	R/W	Output Compare 0/1	0 0 0 0 _ _ 0 0 $_B$
59 $_H$	Output Compare Control Status 1	OCS1	R/W		_ _ _ 0 0 0 0 0 $_B$
5A $_H$	Output Compare Control Status 2	OCS2	R/W	Output Compare 2/3	0 0 0 0 _ _ 0 0 $_B$
5B $_H$	Output Compare Control Status 3	OCS3	R/W		_ _ _ 0 0 0 0 0 $_B$
5C $_H$	Input Capture Control Status 0/1	ICS01	R/W	Input Capture 0/1	0 0 0 0 0 0 0 0 $_B$
5D $_H$	Input Capture Control Status 2/3	ICS23	R/W	Input Capture 2/3	0 0 0 0 0 0 0 0 $_B$
5E $_H$	PWM Control 0	PWC0	R/W	Stepping Motor Controller 0	0 0 0 0 0 _ _ 0 $_B$
5F $_H$	Reserved				
60 $_H$	PWM Control 1	PWC1	R/W	Stepping Motor Controller 1	0 0 0 0 0 _ _ 0 $_B$
61 $_H$	Reserved				
62 $_H$	PWM Control 2	PWC2	R/W	Stepping Motor Controller 2	0 0 0 0 0 _ _ 0 $_B$
63 $_H$	Reserved				
64 $_H$	PWM Control 3	PWC3	R/W	Stepping Motor Controller 3	0 0 0 0 0 _ _ 0 $_B$
65 $_H$	Reserved				
66 $_H$	Timer Data	TCDT	R/W	IO Timer	0 0 0 0 0 0 0 0 $_B$
67 $_H$	Timer Data	TCDT	R/W		0 0 0 0 0 0 0 0 $_B$
68 $_H$	Timer Control	TCCS	R/W		0 0 0 0 0 0 0 0 $_B$
69 to 6E $_H$	Reserved				

(Continued)

Address	Register	Abbreviation	Access	Pripheral	Initial value
6F H	ROM Mirror	ROMM	R/W	ROM Mirror	_ _ _ _ _ _ _ 1 B
70 H	PWM1 Compare 0	PWC10	R/W	Stepping Motor Controller 0	XXXXXXXX B
71 H	PWM2 Compare 0	PWC20	R/W		XXXXXXXX B
72 H	PWM1 Select 0	PWS10	R/W		_ _ 0 0 0 0 0 0 B
73 H	PWM2 Select 0	PWS20	R/W		_ 0 0 0 0 0 0 0 B
74 H	PWM1 Compare 1	PWC11	R/W	Stepping Motor Controller 1	XXXXXXXX B
75 H	PWM2 Compare 1	PWC21	R/W		XXXXXXXX B
76 H	PWM1 Select 1	PWS11	R/W		_ _ 0 0 0 0 0 0 B
77 H	PWM2 Select 1	PWS21	R/W		_ 0 0 0 0 0 0 0 B
78 H	PWM1 Compare 2	PWC12	R/W	Stepping Motor Controller 2	XXXXXXXX B
79 H	PWM2 Compare 2	PWC22	R/W		XXXXXXXX B
7A H	PWM1 Select 2	PWS12	R/W		_ _ 0 0 0 0 0 0 B
7B H	PWM2 Select 2	PWS22	R/W		_ 0 0 0 0 0 0 0 B
7C H	PWM1 Compare 3	PWC13	R/W	Stepping Motor Controller 3	XXXXXXXX B
7D H	PWM2 Compare 3	PWC23	R/W		XXXXXXXX B
7E H	PWM1 Select 3	PWS13	R/W		_ _ 0 0 0 0 0 0 B
7F H	PWM2 Select 3	PWS23	R/W		_ 0 0 0 0 0 0 0 B
80 to 8F H	CAN Controller. Refer to section about CAN Controller				
90 to 9D H	Reserved				
9E H	ROM Correction Control Status	PACSR	R/W	ROM Correction	0 0 0 0 0 0 0 0 B
9F H	Delayed Interrupt/release	DIRR	R/W	Delayed Interrupt	_ _ _ _ _ _ _ 0 B
A0 H	Low-power Mode	LPMCR	R/W	Low Power Controller	0 0 0 1 1 0 0 0 B
A1 H	Clock Selector	CKSCR	R/W	Low Power Controller	1 1 1 1 1 1 0 0 B
A2 to A7 H	Reserved				
A8 H	Watchdog Control	WDTC	R/W	Watchdog Timer	XXXXX 1 1 1 B
A9 H	Time Base Timer Control	TBTC	R/W	Time Base Timer	1 _ _ 0 0 1 0 0 B
AA to AD H	Reserved				
AE H	Flash Control Status (MB90F598 only. Otherwise reserved)	FMCS	R/W	Flash Memory	0 0 0 X 0 0 0 0 B
AF H	Reserved				
B0 H	Interrupt control register 00	ICR00	R/W	Interrupt controller	0 0 0 0 0 1 1 1 B
B1 H	Interrupt control register 01	ICR01	R/W		0 0 0 0 0 1 1 1 B
B2 H	Interrupt control register 02	ICR02	R/W		0 0 0 0 0 1 1 1 B
B3 H	Interrupt control register 03	ICR03	R/W		0 0 0 0 0 1 1 1 B

(Continued)

Address	Register	Abbreviation	Access	Pripheral	Initial value
B4$_H$	Interrupt control register 04	ICR04	R/W	Interrupt controller	00000111$_B$
B5$_H$	Interrupt control register 05	ICR05	R/W		00000111$_B$
B6$_H$	Interrupt control register 06	ICR06	R/W		00000111$_B$
B7$_H$	Interrupt control register 07	ICR07	R/W		00000111$_B$
B8$_H$	Interrupt control register 08	ICR08	R/W		00000111$_B$
B9$_H$	Interrupt control register 09	ICR09	R/W		00000111$_B$
BA$_H$	Interrupt control register 10	ICR10	R/W		00000111$_B$
BB$_H$	Interrupt control register 11	ICR11	R/W		00000111$_B$
BC$_H$	Interrupt control register 12	ICR12	R/W		00000111$_B$
BD$_H$	Interrupt control register 13	ICR13	R/W		00000111$_B$
BE$_H$	Interrupt control register 14	ICR14	R/W		00000111$_B$
BF$_H$	Interrupt control register 15	ICR15	R/W		00000111$_B$
CO to FF$_H$	Reserved				
1900$_H$	Reload L	PRLL0	R/W	16-bit Programable Pulse Generator 0/1	XXXXXXXX$_B$
1901$_H$	Reload H	PRLH0	R/W		XXXXXXXX$_B$
1902$_H$	Reload L	PRLL1	R/W		XXXXXXXX$_B$
1903$_H$	Reload H	PRLH1	R/W		XXXXXXXX$_B$
1904$_H$	Reload L	PRLL2	R/W	16-bit Programable Pulse Generator 2/3	XXXXXXXX$_B$
1905$_H$	Reload H	PRLH2	R/W		XXXXXXXX$_B$
1906$_H$	Reload L	PRLL3	R/W		XXXXXXXX$_B$
1907$_H$	Reload H	PRLH3	R/W		XXXXXXXX$_B$
1908$_H$	Reload L	PRLL4	R/W	16-bit Programable Pulse Generator 4/5	XXXXXXXX$_B$
1909$_H$	Reload H	PRLH4	R/W		XXXXXXXX$_B$
190A$_H$	Reload L	PRLL5	R/W		XXXXXXXX$_B$
190B$_H$	Reload H	PRLH5	R/W		XXXXXXXX$_B$
190C$_H$	Reload L	PRLL6	R/W	16-bit Programable Pulse Generator 6/7	XXXXXXXX$_B$
190D$_H$	Reload H	PRLH6	R/W		XXXXXXXX$_B$
190E$_H$	Reload L	PRLL7	R/W		XXXXXXXX$_B$
190F$_H$	Reload H	PRLH7	R/W		XXXXXXXX$_B$
1910$_H$	Reload L	PRLL8	R/W	16-bit Programable Pulse Generator 8/9	XXXXXXXX$_B$
1911$_H$	Reload H	PRLH8	R/W		XXXXXXXX$_B$
1912$_H$	Reload L	PRLL9	R/W		XXXXXXXX$_B$
1913$_H$	Reload H	PRLH9	R/W		XXXXXXXX$_B$
1914$_H$	Reload L	PRLLA	R/W	16-bit Programable Pulse Generator A/B	XXXXXXXX$_B$
1915$_H$	Reload H	PRLHA	R/W		XXXXXXXX$_B$

(Continued)

Address	Register	Abbreviation	Access	Pripheral	Initial value
1916$_H$	Reload L	PRLLB	R/W	16-bit Programable Pulse Generator A/B	XXXXXXXX$_B$
1917$_H$	Reload H	PRLHB	R/W		XXXXXXXX$_B$
1918 to 191F$_H$	Reserved				
1920$_H$	Input Capture 0	IPCP0	R	Input Captue 0/1	XXXXXXXX$_B$
1921$_H$	Input Capture 0	IPCP0	R		XXXXXXXX$_B$
1922$_H$	Input Capture 1	IPCP1	R		XXXXXXXX$_B$
1923$_H$	Input Capture 1	IPCP1	R		XXXXXXXX$_B$
1924$_H$	Input Capture 2	IPCP2	R	Input Captue 2/3	XXXXXXXX$_B$
1925$_H$	Input Capture 2	IPCP2	R		XXXXXXXX$_B$
1926$_H$	Input Capture 3	IPCP3	R		XXXXXXXX$_B$
1927$_H$	Input Capture 3	IPCP3	R		XXXXXXXX$_B$
1928$_H$	Output Compare 0	OCCP0	R/W	Output Compare 0/1	XXXXXXXX$_B$
1929$_H$	Output Compare 0	OCCP0	R/W		XXXXXXXX$_B$
192A$_H$	Output Compare 1	OCCP1	R/W		XXXXXXXX$_B$
192B$_H$	Output Compare 1	OCCP1	R/W		XXXXXXXX$_B$
192C$_H$	Output Compare 2	OCCP2	R/W	Output Compare 2/3	XXXXXXXX$_B$
192D$_H$	Output Compare 2	OCCP2	R/W		XXXXXXXX$_B$
192E$_H$	Output Compare 3	OCCP3	R/W		XXXXXXXX$_B$
192F$_H$	Output Compare 3	OCCP3	R/W		XXXXXXXX$_B$
1930 to 19FF$_H$	Reserved				
1A00 to 1AFF$_H$	CAN Controller. Refer to section about CAN Controller				
1B00 to 1BFF$_H$	CAN Controller. Refer to section about CAN Controller				
1C00 to 1EFF$_H$	Reserved				
1FF0$_H$	ROM Correction Address 0	PADR0	R/W	ROM Correction	XXXXXXXX$_B$
1FF1$_H$	ROM Correction Address 1	PADR0	R/W		XXXXXXXX$_B$
1FF2$_H$	ROM Correction Address 2	PADR0	R/W		XXXXXXXX$_B$
1FF3$_H$	ROM Correction Address 3	PADR1	R/W		XXXXXXXX$_B$
1FF4$_H$	ROM Correction Address 4	PADR1	R/W		XXXXXXXX$_B$
1FF5$_H$	ROM Correction Address 5	PADR1	R/W		XXXXXXXX$_B$
1FF6 to 1FFF$_H$	Reserved				

Note Initial value of “_” represents unused bit, “X” represents unknown value.
Addresses in the range 0000$_H$ to 00FF$_H$, which are not listed in the table, are reserved for the primary functions of the MCU. A read access to these reserved addresses results reading “X” and any write access should not be performed.

■ CAN CONTROLLER

The CAN controller has the following features:

- Conforms to CAN Specification Version 2.0 Part A and B
  - Supports transmission/reception in standard frame and extended frame formats
- Supports transmitting of data frames by receiving remote frames
- 16 transmitting/receiving message buffers
  - 29-bit ID and 8-byte data
  - Multi-level message buffer configuration
- Provides full-bit comparison, full-bit mask, acceptance register 0/acceptance register 1 for each message buffer as 1D acceptance mask
  - Two acceptance mask registers in either standard frame format or extended frame formats
- Bit rate programmable from 10 Kbits/s to 2 Mbits/s (when input clock is at 16 MHz)

List of Control Registers

Address	Register	Abbreviation	Access	Initial Value
000080$_H$	Message buffer valid register	BVALR	R/W	00000000 00000000$_B$
000081$_H$				
000082$_H$	Transmit request register	TREQR	R/W	00000000 00000000$_B$
000083$_H$				
000084$_H$	Transmit cancel register	TCANR	W	00000000 00000000$_B$
000085$_H$				
000086$_H$	Transmit complete register	TCR	R/W	00000000 00000000$_B$
000087$_H$				
000088$_H$	Receive complete register	RCR	R/W	00000000 00000000$_B$
000089$_H$				
00008A$_H$	Remote request receiving register	RRTRR	R/W	00000000 00000000$_B$
00008B$_H$				
00008C$_H$	Receive overrun register	ROVRR	R/W	00000000 00000000$_B$
00008D$_H$				
00008E$_H$	Receive interrupt enable register	RIER	R/W	00000000 00000000$_B$
00008F$_H$				
001B00$_H$	Control status register	CSR	R/W, R	00---000 0----0-1$_B$
001B01$_H$				
001B02$_H$	Last event indicator register	LEIR	R/W	-------- 000-0000$_B$
001B03$_H$				
001B04$_H$	Receive/transmit error counter	RTEC	R	00000000 00000000$_B$
001B05$_H$				
001B06$_H$	Bit timing register	BTR	R/W	-1111111 11111111$_B$
001B07$_H$				

(Continued)

(Continued)

Address	Register	Abbreviation	Access	Initial Value
001B08$_H$	IDE register	IDER	R/W	XXXXXXXX XXXXXXXX$_B$
001B09$_H$				
001B0A$_H$	Transmit RTR register	TRTRR	R/W	00000000 00000000$_B$
001B0B$_H$				
001B0C$_H$	Remote frame receive waiting register	RFWTR	R/W	XXXXXXXX XXXXXXXX$_B$
001B0D$_H$				
001B0E$_H$	Transmit interrupt enable register	TIER	R/W	00000000 00000000$_B$
001B0F$_H$				
001B10$_H$	Acceptance mask select register	AMSR	R/W	XXXXXXXX XXXXXXXX$_B$
001B11$_H$				
001B12$_H$				XXXXXXXX XXXXXXXX$_B$
001B13$_H$				
001B14$_H$	Acceptance mask register 0	AMR0	R/W	XXXXXXXX XXXXXXXX$_B$
001B15$_H$				
001B16$_H$				XXXXX--- XXXXXXXX$_B$
001B17$_H$				
001B18$_H$	Acceptance mask register 1	AMR1	R/W	XXXXXXXX XXXXXXXX$_B$
001B19$_H$				
001B1A$_H$				XXXXX--- XXXXXXXX$_B$
001B1B$_H$				

List of Message Buffers (ID Registers)

Address	Register	Abbreviation	Access	Initial Value
001A00$_H$ to 001A1F$_H$	General-purpose RAM	--	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001A20$_H$	ID register 0	IDR0	R/W	XXXXXXXX XXXXXXXX$_B$
001A21$_H$				
001A22$_H$				XXXXX--- XXXXXXXX$_B$
001A23$_H$				
001A24$_H$	ID register 1	IDR1	R/W	XXXXXXXX XXXXXXXX$_B$
001A25$_H$				
001A26$_H$				XXXXX--- XXXXXXXX$_B$
001A27$_H$				
001A28$_H$	ID register 2	IDR2	R/W	XXXXXXXX XXXXXXXX$_B$
001A29$_H$				
001A2A$_H$				XXXXX--- XXXXXXXX$_B$
001A2B$_H$				
001A2C$_H$	ID register 3	IDR3	R/W	XXXXXXXX XXXXXXXX$_B$
001A2D$_H$				
001A2E$_H$				XXXXX--- XXXXXXXX$_B$
001A2F$_H$				
001A30$_H$	ID register 4	IDR4	R/W	XXXXXXXX XXXXXXXX$_B$
001A31$_H$				
001A32$_H$				XXXXX--- XXXXXXXX$_B$
001A33$_H$				
001A34$_H$	ID register 5	IDR5	R/W	XXXXXXXX XXXXXXXX$_B$
001A35$_H$				
001A36$_H$				XXXXX--- XXXXXXXX$_B$
001A37$_H$				
001A38$_H$	ID register 6	IDR6	R/W	XXXXXXXX XXXXXXXX$_B$
001A39$_H$				
001A3A$_H$				XXXXX--- XXXXXXXX$_B$
001A3B$_H$				
001A3C$_H$	ID register 7	IDR7	R/W	XXXXXXXX XXXXXXXX$_B$
001A3D$_H$				
001A3E$_H$				XXXXX--- XXXXXXXX$_B$
001A3F$_H$				

(Continued)

(Continued)

Address	Register	Abbreviation	Access	Initial Value
001A40$_H$	ID register 8	IDR8	R/W	XXXXXXXX XXXXXXXX$_B$
001A41$_H$				
001A42$_H$				XXXXX--- XXXXXXXX$_B$
001A43$_H$				
001A44$_H$	ID register 9	IDR9	R/W	XXXXXXXX XXXXXXXX$_B$
001A45$_H$				
001A46$_H$				XXXXX--- XXXXXXXX$_B$
001A47$_H$				
001A48$_H$	ID register 10	IDR10	R/W	XXXXXXXX XXXXXXXX$_B$
001A49$_H$				
001A4A$_H$				XXXXX--- XXXXXXXX$_B$
001A4B$_H$				
001A4C$_H$	ID register 11	IDR11	R/W	XXXXXXXX XXXXXXXX$_B$
001A4D$_H$				
001A4E$_H$				XXXXX--- XXXXXXXX$_B$
001A4F$_H$				
001A50$_H$	ID register 12	IDR12	R/W	XXXXXXXX XXXXXXXX$_B$
001A51$_H$				
001A52$_H$				XXXXX--- XXXXXXXX$_B$
001A53$_H$				
001A54$_H$	ID register 13	IDR13	R/W	XXXXXXXX XXXXXXXX$_B$
001A55$_H$				
001A56$_H$				XXXXX--- XXXXXXXX$_B$
001A57$_H$				
001A58$_H$	ID register 14	IDR14	R/W	XXXXXXXX XXXXXXXX$_B$
001A59$_H$				
001A5A$_H$				XXXXX--- XXXXXXXX$_B$
001A5B$_H$				
001A5C$_H$	ID register 15	IDR15	R/W	XXXXXXXX XXXXXXXX$_B$
001A5D$_H$				
001A5E$_H$				XXXXX--- XXXXXXXX$_B$
001A5F$_H$				

List of Message Buffers (DLC Registers and Data Registers)

Address	Register	Abbreviation	Access	Initial Value
001A60$_{H}$	DLC register 0	DLCR0	R/W	----XXXX$_{B}$
001A61$_{H}$				
001A62$_{H}$	DLC register 1	DLCR1	R/W	----XXXX$_{B}$
001A63$_{H}$				
001A64$_{H}$	DLC register 2	DLCR2	R/W	----XXXX$_{B}$
001A65$_{H}$				
001A66$_{H}$	DLC register 3	DLCR3	R/W	----XXXX$_{B}$
001A67$_{H}$				
001A68$_{H}$	DLC register 4	DLCR4	R/W	----XXXX$_{B}$
001A69$_{H}$				
001A6A$_{H}$	DLC register 5	DLCR5	R/W	----XXXX$_{B}$
001A6B$_{H}$				
001A6C$_{H}$	DLC register 6	DLCR6	R/W	----XXXX$_{B}$
001A6D$_{H}$				
001A6E$_{H}$	DLC register 7	DLCR7	R/W	----XXXX$_{B}$
001A6F$_{H}$				
001A70$_{H}$	DLC register 8	DLCR8	R/W	----XXXX
001A71$_{H}$				
001A72$_{H}$	DLC register 9	DLCR9	R/W	----XXXX$_{B}$
001A73$_{H}$				
001A74$_{H}$	DLC register 10	DLCR10	R/W	----XXXX$_{B}$
001A75$_{H}$				
001A76$_{H}$	DLC register 11	DLCR11	R/W	----XXXX$_{B}$
001A77$_{H}$				
001A78$_{H}$	DLC register 12	DLCR12	R/W	----XXXX$_{B}$
001A79$_{H}$				
001A7A$_{H}$	DLC register 13	DLCR13	R/W	----XXXX$_{B}$
001A7B$_{H}$				
001A7C$_{H}$	DLC register 14	DLCR14	R/W	----XXXX$_{B}$
001A7D$_{H}$				
001A7E$_{H}$	DLC register 15	DLCR15	R/W	----XXXX$_{B}$
001A7F$_{H}$				
001A80$_{H}$ to 001A87$_{H}$	Data register 0 (8 bytes)	DTR0	R/W	XXXXXXXX$_{B}$ to XXXXXXXX$_{B}$

(Continued)

(Continued)

Address	Register	Abbreviation	Access	Initial Value
001A88$_H$ to 001A8F$_H$	Data register 1 (8 bytes)	DTR1	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001A90$_H$ to 001A97$_H$	Data register 2 (8 bytes)	DTR2	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001A98$_H$ to 001A9F$_H$	Data register 3 (8 bytes)	DTR3	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001AA0$_H$ to 001AA7$_H$	Data register 4 (8 bytes)	DTR4	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001AA8$_H$ to 001AAF$_H$	Data register 5 (8 bytes)	DTR5	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001AB0$_H$ to 001AB7$_H$	Data register 6 (8 bytes)	DTR6	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001AB8$_H$ to 001ABF$_H$	Data register 7 (8 bytes)	DTR7	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001AC0$_H$ to 001AC7$_H$	Data register 8 (8 bytes)	DTR8	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001AC8$_H$ to 001ACF$_H$	Data register 9 (8 bytes)	DTR9	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001AD0$_H$ to 001AD7$_H$	Data register 10 (8 bytes)	DTR10	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001AD8$_H$ to 001ADF$_H$	Data register 11 (8 bytes)	DTR11	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001AE0$_H$ to 001AE7$_H$	Data register 12 (8 bytes)	DTR12	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001AE8$_H$ to 001AEF$_H$	Data register 13 (8 bytes)	DTR13	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001AF0$_H$ to 001AF7$_H$	Data register 14 (8 bytes)	DTR14	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$
001AF8$_H$ to 001AFF$_H$	Data register 15 (8 bytes)	DTR15	R/W	XXXXXXXX$_B$ to XXXXXXXX$_B$

■ INTERRUPT MAP

Interrupt cause	I^{2}OS clear	Interrupt vector		Interrupt control register	
		Number	Address	Number	Address
Reset	N/A	# 08	FFFFDC$_H$	——	——
INT9 instruction	N/A	# 09	FFFFD8$_H$	——	——
Exception	N/A	# 10	FFFFD4$_H$	——	——
CAN RX	N/A	# 11	FFFFD0$_H$	ICR00	0000B0$_H$
CAN TX/NS	N/A	# 12	FFFFCC$_H$		
External Interrupt (INT0/INT1)	*1	# 13	FFFFC8$_H$	ICR01	0000B1$_H$
Time Base Timer	N/A	# 14	FFFFC4$_H$		
16-bit Reload Timer 0	*1	# 15	FFFFC0$_H$	ICR02	0000B2$_H$
A/D Converter	*1	# 16	FFFFBC$_H$		
I/O Timer	N/A	# 17	FFFFB8$_H$	ICR03	0000B3$_H$
External Interrupt (INT2/INT3)	*1	# 18	FFFFB4$_H$		
Serial I/O	*1	# 19	FFFFB0$_H$	ICR04	0000B4$_H$
External Interrupt (INT4/INT5)	*1	# 20	FFFFAC$_H$		
Input Capture 0	*1	# 21	FFFFA8$_H$	ICR05	0000B5$_H$
PPG 0/1	N/A	# 22	FFFFA4$_H$		
Output Compare 0	*1	# 23	FFFFA0$_H$	ICR06	0000B6$_H$
PPG 2/3	N/A	# 24	FFFF9C$_H$		
External Interrupt (INT6/INT7)	*1	# 25	FFFF98$_H$	ICR07	0000B7$_H$
Input Capture 1	*1	# 26	FFFF94$_H$		
PPG 4/5	N/A	# 27	FFFF90$_H$	ICR08	0000B8$_H$
Output Compare 1	*1	# 28	FFFF8C$_H$		
PPG 6/7	N/A	# 29	FFFF88$_H$	ICR09	0000B9$_H$
Input Capture 2	*1	# 30	FFFF84$_H$		
PPG 8/9	N/A	# 31	FFFF80$_H$	ICR10	0000BA$_H$
Output Compare 2	*1	# 32	FFFF7C$_H$		
Input Capture 3	*1	# 33	FFFF78$_H$	ICR11	0000BB$_H$
PPG A/B	N/A	# 34	FFFF74$_H$		
Output Compare 3	*1	# 35	FFFF70$_H$	ICR12	0000BC$_H$
16-bit Reload Timer 1	*1	# 36	FFFF6C$_H$		
UART 0 RX	*2	# 37	FFFF68$_H$	ICR13	0000BD$_H$
UART 0 TX	*1	# 38	FFFF64$_H$		
UART 1 RX	*2	# 39	FFFF60$_H$	ICR14	0000BE$_H$
UART 1 TX	*1	# 40	FFFF5C$_H$		
Flash Memory	N/A	# 41	FFFF58$_H$	ICR15	0000BF$_H$
Delayed interrupt	N/A	# 42	FFFF54$_H$		

*1: The interrupt request flag is cleared by the I^{2}OS interrupt clear signal.

*2: The interrupt request flag is cleared by the I^{2}OS interrupt clear signal. A stop request is available.

N/A:The interrupt request flag is not cleared by the I^{2}OS interrupt clear signal.

Note: For a peripheral module with two interrupt causes for a single interrupt number, both interrupt request flags are cleared by the I^{2}OS interrupt clear signal.

Note: At the end of I^{2}OS, the I^{2}OS clear signal will be asserted for all the interrupt flags assigned to the same interrupt number. If one interrupt flag starts the I^{2}OS and in the meantime another interrupt flag is set by hardware event, the later event is lost because the flag is cleared by the I^{2}OS clear signal caused by the first event. So it is recommended not to use the I^{2}OS for this interrupt number.

Note: If I^{2}OS is enabled, I^{2}OS is initiated when one of the two interrupt signals in the same interrupt control register (ICR) is asserted. This means that different interrupt sources share the same I^{2}OS Descriptor which should be unique for each interrupt source.. For this reason, when one interrupt source uses the I^{2}OS, the other interrupt should be disabled.

■ ELECTRICAL CHARACTERISTICS

1. Absolute Maximum Ratings

(V_{SS} = AV_{SS} = 0 V)

Parameter	Symbol	Rated Value		Units	Remarks
		Min.	Max.		
Power supply voltage	V_{CC}	V_{SS} – 0.3	V_{SS} + 6.0	V	
	AV_{CC}	V_{SS} – 0.3	V_{SS} + 6.0	V	V_{CC} = AV_{CC} *1
	AVR±	V_{SS} – 0.3	V_{SS} + 6.0	V	AV_{CC} ≥ AVR±, AVR+ ≥ AVR–
	DV_{CC}	V_{SS} – 0.3	V_{SS} + 6.0	V	V_{CC} ≥ DV_{CC}
Input voltage	V_I	V_{SS} – 0.3	V_{SS} + 6.0	V	*2
Output voltage	V_O	V_{SS} – 0.3	V_{SS} + 6.0	V	*2
Clamp Current	I_{CLAMP}	–2.0	2.0	mA	
"L" level max. output current	I_{OL1}	—	15	mA	Normal outputs
"L" level avg. output current	I_{OLAV1}	—	4	mA	Normal outputs, average value
"L" level max. output current	I_{OL2}	—	40	mA	High current outputs
"L" level avg. output current	I_{OLAV2}	—	30	mA	High current outputs, average value
"L" level max. overall output current	ΣI_{OL1}	—	100	mA	Sum of all normal outputs
"L" level max. overall output current	ΣI_{OL2}		330	mA	Sum of all high current outputs
"L" level avg. overall output current	ΣI_{OLAV1}	—	50	mA	Sum of all normal outputs, average value
"L" level avg. overall output current	ΣI_{OLAV2}		250	mA	Sum of all high current outputs, average value
"H" level max. output current	I_{OH1}	—	–15	mA	Normal outputs
"H" level avg. output current	I_{OHAV1}	—	–4	mA	Normal outputs, average value
"H" level max. output current	I_{OH2}	—	–40	mA	High current outputs
"H" level avg. output current	I_{OHAV2}	—	–30	mA	High current outputs, average value
"H" level max. overall output current	ΣI_{OH1}	—	–100	mA	Sum of all normal outputs
"H" level max. overall output current	ΣI_{OH2}	—	–330	mA	Sum of all high current outputs
"H" level avg. overall output current	ΣI_{OHAV1}	—	–50	mA	Sum of all normal outputs, average value
"H" level avg. overall output current	ΣI_{OHAV2}	—	–250	mA	Sum of all high current outputs, average value
Power consumption	P_D	—	500	mW	MB90F598
			400	mW	MB90598
Operating temperature	T_A	–40	+85	°C	
Storage temperature	T_{STG}	–55	+150	°C	

*1: Set AV_{CC} and V_{CC} to the same voltage. Make sure that AV_{CC} does not exceed V_{CC} and that the voltage at the analog inputs does not exceed AV_{CC} when the power is switched on.

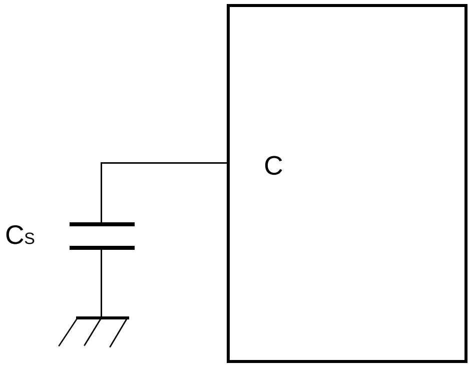
*2: V_I and V_O should not exceed V_{CC} + 0.3V. V_I should not exceed the specified ratings. However if the maximun current to/from a input is limited by some means with external components, the I_I rating supercedes the V_I rating.

2. Recommended Conditions

(V_{SS} = AV_{SS} = 0 V)

Parameter	Symbol	Rated Value			Units	Remarks
		Min.	Typ.	Max.		
Power supply voltage	V_{CC} AV_{CC}	4.5	5.0	5.5	V	Under normal operation
		3V		5.5	V	Maintains RAM data in stop mode
Input H voltage	V_{IHS}	0.8 V_{CC}		V_{CC} +0.3	V	CMOS hysteresis input pin
	V_{IHM}	V_{CC} – 0.3		V_{CC} +0.3	V	MD input pin
Input L voltage	V_{ILS}	V_{SS} – 0.3		0.2 V_{CC}	V	CMOS hysteresis input pin
	V_{ILM}	V_{SS} – 0.3		V_{SS} +0.3	V	MD input pin
Smooth capacitor	C_S	0.022	0.1	1.0	µF	Use a ceramic capacitor or capacitor of better AC characteristics. Capacitor at the VCC should be greater than this capacitor.
Operating temperature	T_A	–40		+85	°C	

- **C Pin Connection Diagram**



3. DC Characteristics

(V_{CC} = 5.0 V±10%, V_{SS} = AV_{SS} = 0V, T_A = –40 °C to +85 °C)

Parameter	Symbol	Pin	Condition	Value			Units	Remarks
				Min.	Typ.	Max.		
Output H voltage	V_{OH1}	Normal outputs	V_{CC} = 4.5 V, I_{OH1} = –4.0 mA	V_{CC} – 0.5	—	—	V	
Output H voltage	V_{OH2}	High current outputs	V_{CC} = 4.5 V, I_{OH2} = –30.0 mA	V_{CC} – 0.5	—	—	V	
Output L voltage	V_{OL1}	Normal outputs	V_{CC} = 4.5 V, I_{OL1} = 4.0 mA	—	—	0.4	V	
Output L voltage	V_{OL2}	High current outputs	V_{CC} = 4.5 V, I_{OL2} = 30.0 mA	—	—	0.5	V	
Input leak current	I_{IL}		V_{CC} = 5.5 V, $V_{SS} < V_I < V_{CC}$	–5	—	5	µA	
Power supply current *	I_{CC}	V_{CC}	V_{CC} = 5.0 V±10%, Internal frequency: 16 MHz, At normal operating	—	TBD	TBD	mA	MB90598
				—	60	90	mA	MB90F598
	I_{CCS}		V_{CC} = 5.0 V±10%, Internal frequency: 16 MHz, At sleep	—	TBD	TBD	mA	MB90598
				—	15	23	mA	MB90F598
	I_{CTS}		V_{CC} = 5.0 V±1%, Internal frequency: 2 MHz, At timer mode	—	TBD	TBD	mA	MB90598
				—	—	0.6	mA	MB90F598
	I_{CCH}		V_{CC} = 5.0 V±10%, At stop, T_A = 25°C	—	TBD	TBD	µA	MB90598
				—	—	20	µA	MB90F598
Input capacity	C_{IN}	Other than C, AV_{CC}, AV_{SS}, AVR+, AVR-, V_{CC}, V_{SS}, DV_{CC}, DV_{SS}	—	—	10	80	pF	

*: Current values are tentative. They are subject to change without notice according to improvements in the characteristics. The power supply current testing conditions are when using the external clock.

4. AC Characteristics

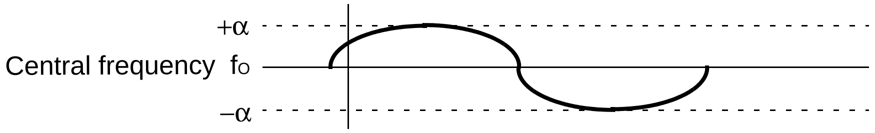
(1) Clock Timing

(V_{CC} = 5.0 V±10%, V_{SS} = AV_{SS} = 0V, T_A = –40 °C to +85 °C)

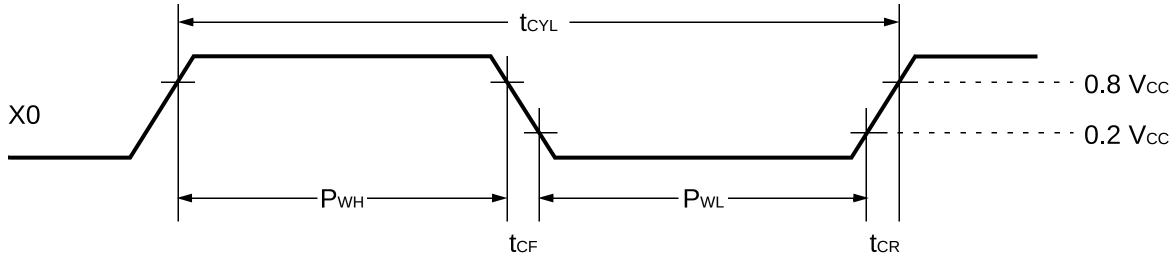
Parameter	Symbol	Pin	Value			Units	Remarks
			Min.	Typ.	Max.		
Oscillation frequency	f_C	X0, X1	3	—	16	MHz	
Oscillation cycle time	t_{CYL}	X0, X1	62.5	—	333	ns	
Frequency deviation with PLL *	Δf	—	—	—	5	%	
Input clock pulse width	P_{WH}, P_{WL}	X0	10	—	—	ns	Duty ratio is about 30 to 70%.
Input clock rise and fall time	t_{CR}, t_{CF}	X0	—	—	5	ns	When using external clock
Machine clock frequency	f_{CP}	—	1.5	—	16	MHz	
Machine clock cycle time	t_{CP}	—	62.5	—	666	ns	

* : Frequency deviation indicates the maximum frequency difference from the target frequency when using a multiplied clock.

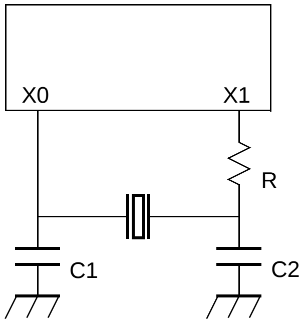
$$\Delta f = \frac{|\alpha|}{fo} \times 100\%$$



- **Clock Timing**

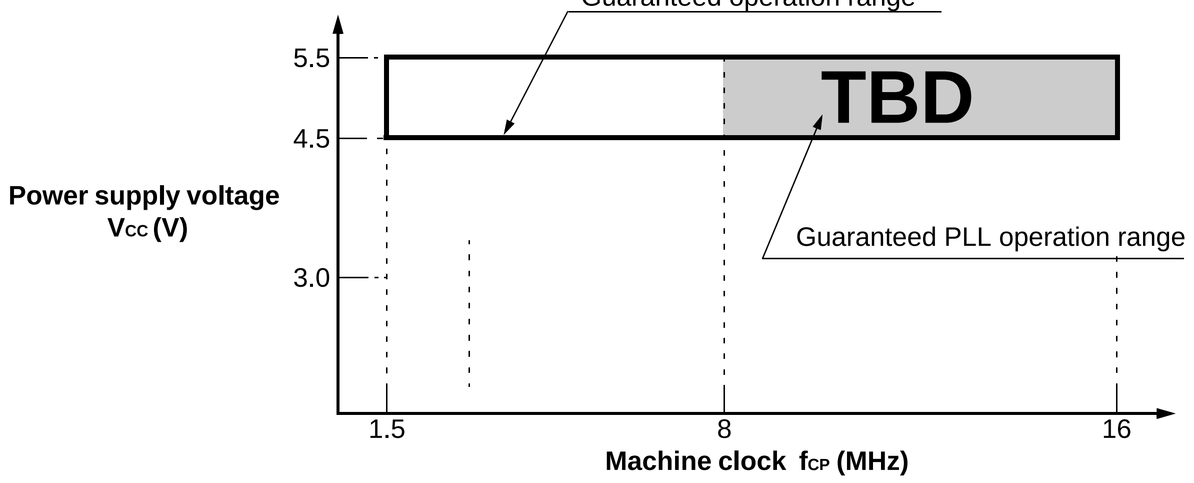


Example of Oscillation circuit

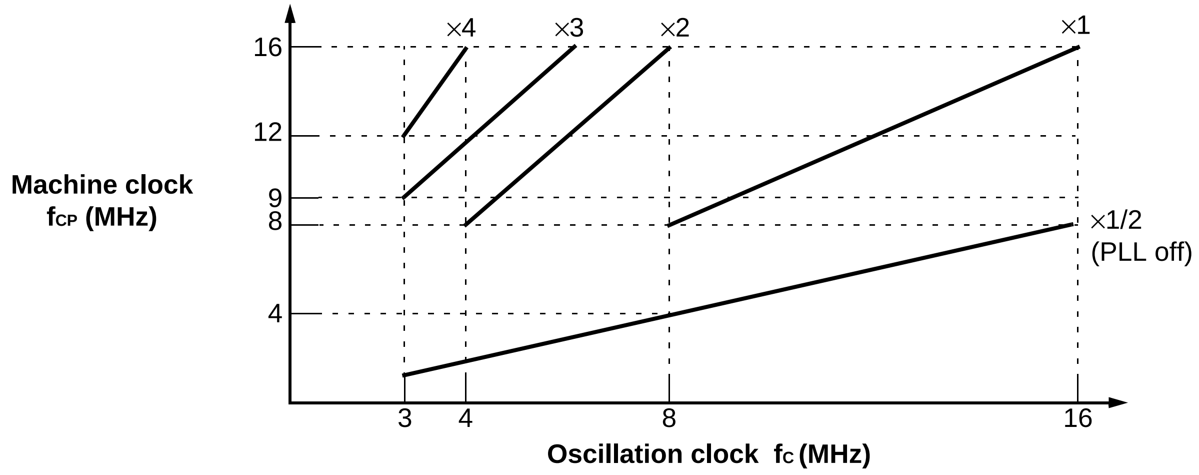


Make	Oscillator	Frequency (MHz)	C1 (pF)	C2 (pF)	R (Ω)
TBD	TBD	4MHz	TBD	TBD	TBD

- **Guaranteed operation range**

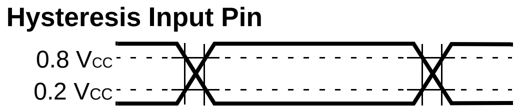


- **Ocsillation clock frequency and Machine clock frequency**

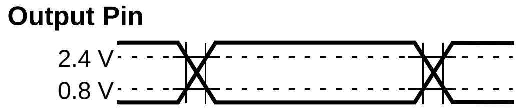


AC characteristics are set to the measured reference voltage values below.

- **Input signal waveform**



- **Output signal waveform**

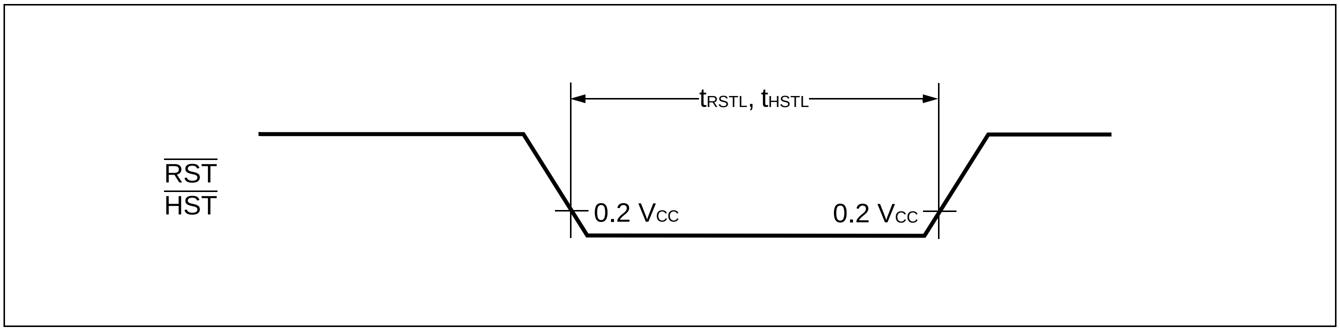


(2) Reset and Hardware Standby Input

(V_{CC} = 5.0 V±10%, V_{SS} = AV_{SS} = 0V, T_A = –40 °C to +85 °C)

Parameter	Symbol	Pin	Value		Units	Remarks
			Min.	Max.		
Reset input time	t_{RSTL}	$\overline{RST}$	16 t_{CP}	—	ns	
Hardware standby input time	t_{HSTL}	$\overline{HST}$	16 t_{CP}	—	ns	

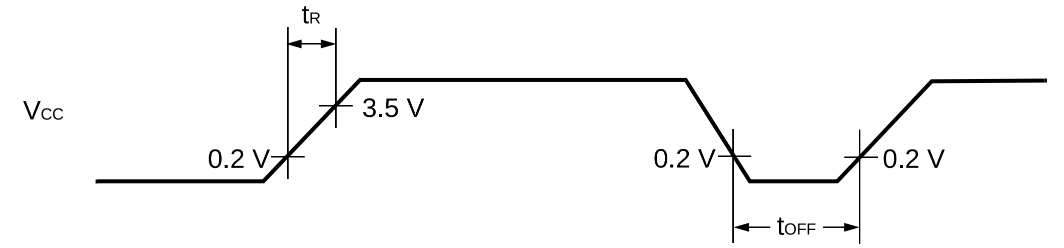
"t_{cp}" represents one cycle time of the machine clock.
Any reset can not fully initialize the Flash Memory if it is performing the automatic algorithm.



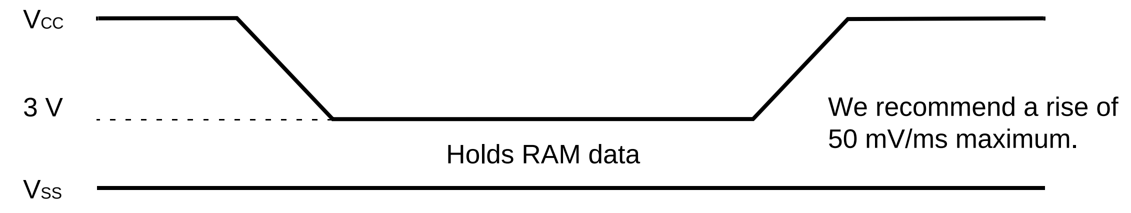
(3)Power On Reset

(V_{CC} = 5.0 V±10%, V_{SS} = AV_{SS} = 0V, T_A = –40 °C to +85 °C)

Parameter	Symbol	Pin	Test Condition	Value		Units	Remarks
				Min.	Max.		
Power on rise time	t_R	V_{CC}	—	0.05	30	ms	
Power off time	t_{OFF}	V_{CC}		50	—	ms	Due to repetitive operation



If you change the power supply voltage too rapidly, a power on reset may occur. We recommend that you startup smoothly by restraining voltages when changing the power supply voltage during operation, as shown in the figure below. Perform while not using the PLL clock. However, if voltage drops are within 1 mV/sec, you can operate while using the PLL clock.



(4) UART0/1, Serial I/O Timing

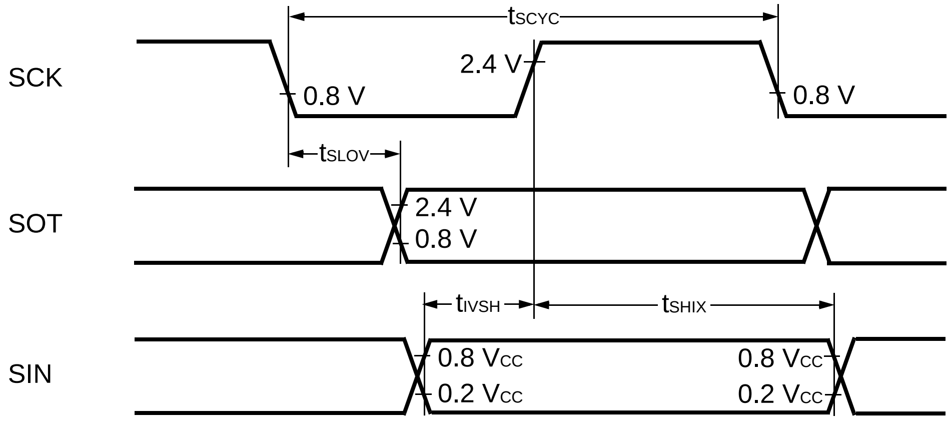
(V_{CC} = 5.0 V±10%, V_{SS} = AV_{SS} = 0V, T_A = –40 °C to +85 °C)

Parameter	Symbol	Pin Symbol	Test Condition	Value		Units	Remarks
				Min.	Max.		
Serial clock cycle time	t_{SCYC}	SCK0 to SCK2	Internal clock operation output pins are C_L = 80 pF + 1 TTL.	8 t_{CP}	—	ns	
SCK ↓ ⇒ SOT delay time	t_{SLOV}	SCK0 to SCK2, SOT0 to SOT2		–80	80	ns	
Valid SIN ⇒ SCK ↑	t_{IVSH}	SCK0 to SCK2, SIN0 to SIN2		100	—	ns	
SCK ↑ ⇒ Valid SIN hold time	t_{SHIX}	SCK0 to SCK2, SIN0 to SIN2		60	—	ns	
Serial clock "H" pulse width	t_{SHSL}	SCK0 to SCK2	External clock operation output pins are C_L = 80 pF + 1 TTL.	4 t_{CP}	—	ns	
Serial clock "L" pulse width	t_{SLSH}	SCK0 to SCK2		4 t_{CP}	—	ns	
SCK ↓ ⇒ SOT delay time	t_{SLOV}	SCK0 to SCK2, SOT0 to SOT2		—	150	ns	
Valid SIN ⇒ SCK ↑	t_{IVSH}	SCK0 to SCK2, SIN0 to SIN2		60	—	ns	
SCK ↑ ⇒ Valid SIN hold time	t_{SHIX}	SCK0 to SCK2, SIN0 to SIN2		60	—	ns	

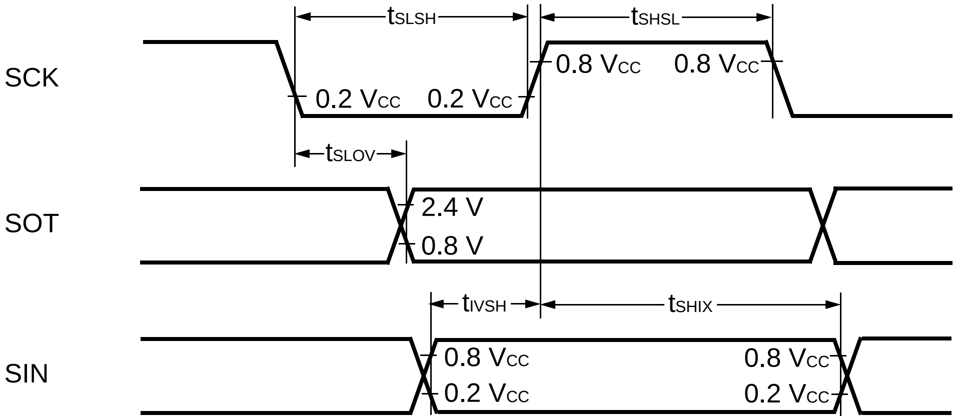
Note:

1. AC characteristic in CLK synchronized mode.
2. C_L is load capacity value of pins when testing.
3. t_{CP} is the machine cycle (Unit: ns).

- **Internal Shift Clock Mode**



- **External Shift Clock Mode**

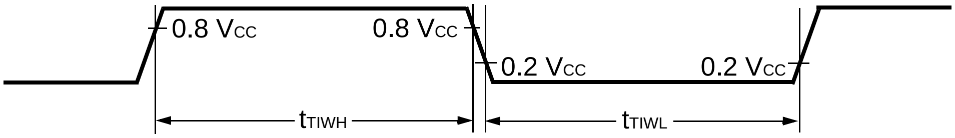


(5) Timer Related Resource Input Timing

(V_{CC} = 5.0 V±10%, V_{SS} = AV_{SS} = 0V, T_A = –40 °C to +85 °C)

Parameter	Symbol	Pin	Condition	Value		Units	Remarks
				Min.	Max.		
Input pulse width	t_{TIWH} t_{TIWL}	TIN0, TIN1 IN0 to IN3	—	4 t_{CP}	—	ns	

- **Timer Input Timing**

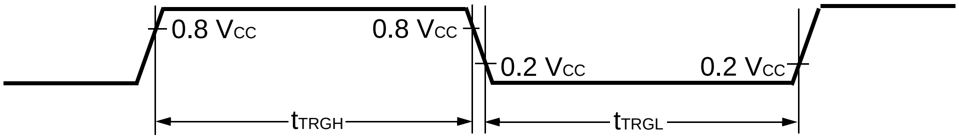


(6) Trigger Input Timing

(V_{CC} = 5.0 V±10%, V_{SS} = AV_{SS} = 0V, T_A = –40 °C to +85 °C)

Parameter	Symbol	Pin	Condition	Value		Units	Remarks
				Min.	Max.		
Input pulse width	t_{TRGH} t_{TRGL}	INT0 to INT7, ADTG	—	5 t_{CP}	—	ns	

- **Trigger Input Timing**



5. A/D Converter

(V_{CC} = AV_{CC} = 5.0 V±10%, V_{SS} = AV_{SS} = 0 V,3.0 V ≤ AVR_+ – AVR_-, T_A = –40 °C to +85 °C)

Parameter	Symbol	Pin	Value			Units	Remarks
			Min.	Typ.	Max.		
Resolution	—	—	—		10	bit	
Conversion error	—	—	—	—	±5.0	LSB	
Nonlinearity error	—	—	—	—	±2.5	LSB	
Differential nonlinearity error	—	—	—	—	±1.9	LSB	
Zero reading voltage	V_{OT}	AN0 to AN7	AVR_- – 3.5	AVR_- +0.5	AVR_- + 4.5	mV	
Full scale reading voltage	V_{FST}	AN0 to AN7	AVR_+ – 6.5	AVR_+ – 1.5	AVR_+ + 1.5	mV	
Conversion time	—	—	—	352t_{CP}	—	ns	
Sampling time	—	—	—	64t_{CP}	—	ns	
Analog port input current	I_{AIN}	AN0 to AN7	–10	—	10	µA	
Analog input voltage range	V_{AIN}	AN0 to AN7	AVR-	—	AVR_+	V	
Reference voltage range	—	AVR_+	AVR_- + 2.7	—	AV_{CC}	V	
	—	AVR_-	0	—	AVR_+ – 2.7	V	
Power supply current	I_A	AV_{CC}	—	5	—	mA	
	I_{AH}	AV_{CC}	—	—	5	µA	*
Reference voltage current	I_R	AVR_+	200	400	600	µA	
	I_{RH}	AVR_+	—	—	5	µA	*
Offset between input channels	—	AN0 to AN7	—	—	4	LSB	

* : When not operating A/D converter, this is the current (V_{CC} = AV_{CC} = AVR_+ = 5.0 V) when the CPU is stopped.

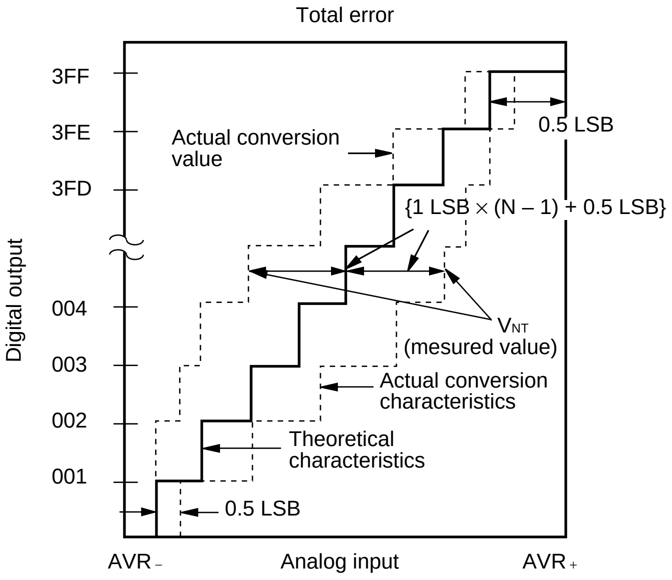
6. A/D Converter Glossary

Resolution: Analog changes that are identifiable with the A/D converter

Linearity error: The deviation of the straight line connecting the zero transition point (“00 0000 0000” ↔ “00 0000 0001”) with the full-scale transition point (“11 1111 1110” ↔ “11 1111 1111”) from actual conversion characteristics

Differential linearity error: The deviation of input voltage needed to change the output code by 1 LSB from the theoretical value

Total error: The total error is defined as a difference between the actual value and the theoretical value, which includes zero-transition error/full-scale transition error and linearity error.



1 LSB = (Theoretical value) $\frac{AVR_+ - AVR_-}{1024}$ [V]

Total error for digital output N = $\frac{V_{NT} - \{1\,LSB \times (N - 1) + 0.5\,LSB\}}{1\,LSB}$ [LSB]

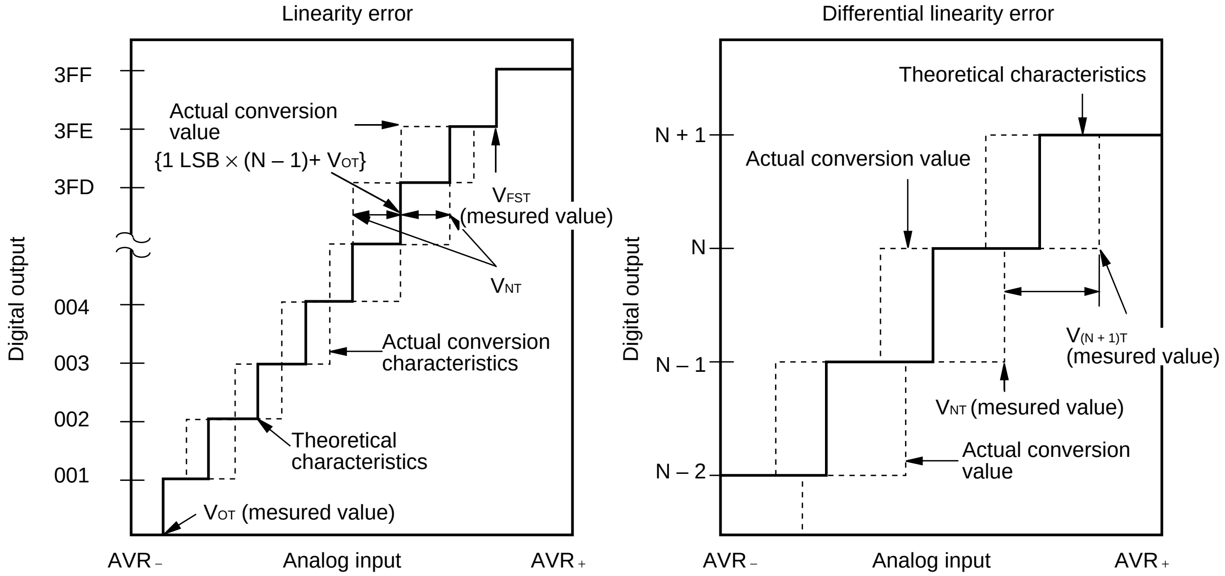
V_{OT} (Theoretical value) = AVR_- + 0.5 LSB[V]

V_{NT}: Voltage at a transition of digital output from (N – 1) to N

V_{FST} (Theoretical value) = AVR_+ – 1.5 LSB[V]

(Continued)

(Continued)



Linearity error of digital output N $= \dfrac{V_{NT} - \{1\ \text{LSB} \times (N-1) + V_{OT}\}}{1\ \text{LSB}}$ [LSB]

Differential linearity error of digital N $= \dfrac{V_{(N+1)T} - V_{NT}}{1\ \text{LSB}} - 1\ \text{LSB}$ [LSB]

$1\ \text{LSB} = \dfrac{V_{FST} - V_{OT}}{1022}$ [V]

V_{OT}: Voltage at transition of digital output from "000_H" to "001_H"
V_{FST}: Voltage at transition of digital output from "$3FE_H$" to "$3FF_H$"

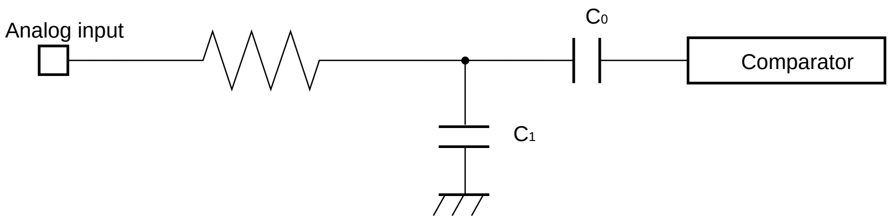
7. Notes on Using A/D Converter

Select the output impedance value for the external circuit of analog input according to the following conditions. Output impedance values of the external circuit of 15 kΩ or lower are recommended.

When capacitors are connected to external pins, the capacitance of several thousand times the internal capacitor value is recommended to minimized the effect of voltage distribution between the external capacitor and internal capacitor.

When the output impedance of the external circuit is too high, the sampling period for analog voltages may not be sufficient (sampling period = 4.00 µs @machine clock of 16 MHz).

- **Equipment of analog input circuit model**



Note: Listed values must be considered as standards.

- **Error**

The smaller the | AVR+ – AVR– |, the greater the error would become relatively.

■ INSTRUCTIONS (340 INSTRUCTIONS)

Table 1 Explanation of Items in Tables of Instructions

Item	Meaning
Mnemonic	Upper-case letters and symbols: Represented as they appear in assembler. Lower-case letters: Replaced when described in assembler. Numbers after lower-case letters: Indicate the bit width within the instruction code.
#	Indicates the number of bytes.
~	Indicates the number of cycles. m : When branching n : When not branching See Table 4 for details about meanings of other letters in items.
RG	Indicates the number of accesses to the register during execution of the instruction. It is used calculate a correction value for intermittent operation of CPU.
B	Indicates the correction value for calculating the number of actual cycles during execution of the instruction. (Table 5) The number of actual cycles during execution of the instruction is the correction value summed with the value in the “~” column.
Operation	Indicates the operation of instruction.
LH	Indicates special operations involving the upper 8 bits of the lower 16 bits of the accumulator. Z : Transfers “0”. X : Extends with a sign before transferring. – : Transfers nothing.
AH	Indicates special operations involving the upper 16 bits in the accumulator. * : Transfers from AL to AH. – : No transfer. Z : Transfers 00$_H$ to AH. X : Transfers 00$_H$ or FF$_H$ to AH by signing and extending AL.
I	Indicates the status of each of the following flags: I (interrupt enable), S (stack), T (sticky bit), N (negative), Z (zero), V (overflow), and C (carry). * : Changes due to execution of instruction. – : No change. S : Set by execution of instruction. R : Reset by execution of instruction.
S	
T	
N	
Z	
V	
C	
RMW	Indicates whether the instruction is a read-modify-write instruction. (a single instruction that reads data from memory, etc., processes the data, and then writes the result to memory.) * : Instruction is a read-modify-write instruction. – : Instruction is not a read-modify-write instruction. Note: A read-modify-write instruction cannot be used on addresses that have different meanings depending on whether they are read or written.

- **Number of execution cycles**

  The number of cycles required for instruction execution is acquired by adding the number of cycles for each instruction, a corrective value depending on the condition, and the number of cycles required for program fetch. Whenever the instruction being executed exceeds the two-byte (word) boundary, a program on an internal ROM connected to a 16-bit bus is fetched. If data access is interfered with, therefore, the number of execution cycles is increased.

  For each byte of the instruction being executed, a program on a memory connected to an 8-bit external data bus is fetched. If data access in interfered with, therefore, the number of execution cycles is increased. When a general-purpose register, an internal ROM, an internal RAM, an internal I/O device, or an external bus is accessed during intermittent CPU operation, the CPU clock is suspended by the number of cycles specified by the CG1/0 bit of the low-power consumption mode control register. When determining the number of cycles required for instruction execution during intermittent CPU operation, therefore, add the value of the number of times access is done × the number of cycles suspended as the corrective value to the number of ordinary execution cycles.

Table 2 Explanation of Symbols in Tables of Instructions

Symbol	Meaning
A	32-bit accumulator The bit length varies according to the instruction. Byte : Lower 8 bits of AL Word : 16 bits of AL Long : 32 bits of AL and AH
AH AL	Upper 16 bits of A Lower 16 bits of A
SP	Stack pointer (USP or SSP)
PC	Program counter
PCB	Program bank register
DTB	Data bank register
ADB	Additional data bank register
SSB	System stack bank register
USB	User stack bank register
SPB	Current stack bank register (SSB or USB)
DPR	Direct page register
brg1	DTB, ADB, SSB, USB, DPR, PCB, SPB
brg2	DTB, ADB, SSB, USB, DPR, SPB
Ri	R0, R1, R2, R3, R4, R5, R6, R7
RWi	RW0, RW1, RW2, RW3, RW4, RW5, RW6, RW7
RWj	RW0, RW1, RW2, RW3
RLi	RL0, RL1, RL2, RL3
dir	Compact direct addressing
addr16 addr24 ad24 0 to 15 ad24 16 to 23	Direct addressing Physical direct addressing Bit 0 to bit 15 of addr24 Bit 16 to bit 23 of addr24
io	I/O area (000000$_H$ to 0000FF$_H$)
imm4 imm8 imm16 imm32 ext (imm8)	4-bit immediate data 8-bit immediate data 16-bit immediate data 32-bit immediate data 16-bit data signed and extended from 8-bit immediate data
disp8 disp16	8-bit displacement 16-bit displacement
bp	Bit offset
vct4 vct8	Vector number (0 to 15) Vector number (0 to 255)
( )b	Bit address
rel	PC relative addressing
ear eam	Effective addressing (codes 00 to 07) Effective addressing (codes 08 to 1F)
rlst	Register list

Table 3 Effective Address Fields

Code	Notation			Address format	Number of bytes in address extension *
00 01 02 03 04 05 06 07	R0 R1 R2 R3 R4 R5 R6 R7	RW0 RW1 RW2 RW3 RW4 RW5 RW6 RW7	RL0 (RL0) RL1 (RL1) RL2 (RL2) RL3 (RL3)	Register direct “ea” corresponds to byte, word, and long-word types, starting from the left	—
08 09 0A 0B	@RW0 @RW1 @RW2 @RW3			Register indirect	0
0C 0D 0E 0F	@RW0 + @RW1 + @RW2 + @RW3 +			Register indirect with post-increment	0
10 11 12 13 14 15 16 17	@RW0 + disp8 @RW1 + disp8 @RW2 + disp8 @RW3 + disp8 @RW4 + disp8 @RW5 + disp8 @RW6 + disp8 @RW7 + disp8			Register indirect with 8-bit displacement	1
18 19 1A 1B	@RW0 + disp16 @RW1 + disp16 @RW2 + disp16 @RW3 + disp16			Register indirect with 16-bit displacement	2
1C 1D 1E 1F	@RW0 + RW7 @RW1 + RW7 @PC + disp16 addr16			Register indirect with index Register indirect with index PC indirect with 16-bit displacement Direct address	0 0 2 2

Note : The number of bytes in the address extension is indicated by the “+” symbol in the “#” (number of bytes) column in the tables of instructions.

Table 4 Number of Execution Cycles for Each Type of Addressing

Code	Operand	(a)	Number of register accesses for each type of addressing
		Number of execution cycles for each type of addressing	
00 to 07	Ri RWi RLi	Listed in tables of instructions	Listed in tables of instructions
08 to 0B	@RWj	2	1
0C to 0F	@RWj +	4	2
10 to 17	@RWi + disp8	2	1
18 to 1B	@RWj + disp16	2	1
1C 1D 1E 1F	@RW0 + RW7 @RW1 + RW7 @PC + disp16 addr16	4 4 2 1	2 2 0 0

Note : “(a)” is used in the “~” (number of states) column and column B (correction value) in the tables of instructions.

Table 5 Compensation Values for Number of Cycles Used to Calculate Number of Actual Cycles

Operand	(b) byte		(c) word		(d) long	
	Cycles	Access	Cycles	Access	Cycles	Access
Internal register	+0	1	+0	1	+0	2
Internal memory even address Internal memory odd address	+0 +0	1 1	+0 +2	1 2	+0 +4	2 4
Even address on external data bus (16 bits) Odd address on external data bus (16 bits)	+1 +1	1 1	+1 +4	1 2	+2 +8	2 4
External data bus (8 bits)	+1	1	+4	2	+8	4

Notes:
- “(b)”, “(c)”, and “(d)” are used in the “~” (number of states) column and column B (correction value) in the tables of instructions.
- When the external data bus is used, it is necessary to add in the number of wait cycles used for ready input and automatic ready.

Table 6 Correction Values for Number of Cycles Used to Calculate Number of Program Fetch Cycles

Instruction	Byte boundary	Word boundary
Internal memory	—	+2
External data bus (16 bits)	—	+3
External data bus (8 bits)	+3	—

Notes:
- When the external data bus is used, it is necessary to add in the number of wait cycles used for ready input and automatic ready.
- Because instruction execution is not slowed down by all program fetches in actuality, these correction values should be used for “worst case” calculations.

Table 7 Transfer Instructions (Byte) [41 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOV A, dir	2	3	0	(b)	byte (A) ← (dir)	Z	*	–	–	–	*	*	–	–	–
MOV A, addr16	3	4	0	(b)	byte (A) ← (addr16)	Z	*	–	–	–	*	*	–	–	–
MOV A, Ri	1	2	1	0	byte (A) ← (Ri)	Z	*	–	–	–	*	*	–	–	–
MOV A, ear	2	2	1	0	byte (A) ← (ear)	Z	*	–	–	–	*	*	–	–	–
MOV A, eam	2+	3+ (a)	0	(b)	byte (A) ← (eam)	Z	*	–	–	–	*	*	–	–	–
MOV A, io	2	3	0	(b)	byte (A) ← (io)	Z	*	–	–	–	*	*	–	–	–
MOV A, #imm8	2	2	0	0	byte (A) ← imm8	Z	*	–	–	–	*	*	–	–	–
MOV A, @A	2	3	0	(b)	byte (A) ← ((A))	Z	–	–	–	–	*	*	–	–	–
MOV A, @RLi+disp8	3	10	2	(b)	byte (A) ← ((RLi)+disp8)	Z	*	–	–	–	*	*	–	–	–
MOVN A, #imm4	1	1	0	0	byte (A) ← imm4	Z	*	–	–	–	R	*	–	–	–
MOVX A, dir	2	3	0	(b)	byte (A) ← (dir)	X	*	–	–	–	*	*	–	–	–
MOVX A, addr16	3	4	0	(b)	byte (A) ← (addr16)	X	*	–	–	–	*	*	–	–	–
MOVX A, Ri	2	2	1	0	byte (A) ← (Ri)	X	*	–	–	–	*	*	–	–	–
MOVX A, ear	2	2	1	0	byte (A) ← (ear)	X	*	–	–	–	*	*	–	–	–
MOVX A, eam	2+	3+ (a)	0	(b)	byte (A) ← (eam)	X	*	–	–	–	*	*	–	–	–
MOVX A, io	2	3	0	(b)	byte (A) ← (io)	X	*	–	–	–	*	*	–	–	–
MOVX A, #imm8	2	2	0	0	byte (A) ← imm8	X	*	–	–	–	*	*	–	–	–
MOVX A, @A	2	3	0	(b)	byte (A) ← ((A))	X	–	–	–	–	*	*	–	–	–
MOVX A,@RWi+disp8	2	5	1	(b)	byte (A) ← ((RWi)+disp8)	X	*	–	–	–	*	*	–	–	–
MOVX A, @RLi+disp8	3	10	2	(b)	byte (A) ← ((RLi)+disp8)	X	*	–	–	–	*	*	–	–	–
MOV dir, A	2	3	0	(b)	byte (dir) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV addr16, A	3	4	0	(b)	byte (addr16) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV Ri, A	1	2	1	0	byte (Ri) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV ear, A	2	2	1	0	byte (ear) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV eam, A	2+	3+ (a)	0	(b)	byte (eam) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV io, A	2	3	0	(b)	byte (io) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV @RLi+disp8, A	3	10	2	(b)	byte ((RLi) +disp8) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV Ri, ear	2	3	2	0	byte (Ri) ← (ear)	–	–	–	–	–	*	*	–	–	–
MOV Ri, eam	2+	4+ (a)	1	(b)	byte (Ri) ← (eam)	–	–	–	–	–	*	*	–	–	–
MOV ear, Ri	2	4	2	0	byte (ear) ← (Ri)	–	–	–	–	–	*	*	–	–	–
MOV eam, Ri	2+	5+ (a)	1	(b)	byte (eam) ← (Ri)	–	–	–	–	–	*	*	–	–	–
MOV Ri, #imm8	2	2	1	0	byte (Ri) ← imm8	–	–	–	–	–	*	*	–	–	–
MOV io, #imm8	3	5	0	(b)	byte (io) ← imm8	–	–	–	–	–	–	–	–	–	–
MOV dir, #imm8	3	5	0	(b)	byte (dir) ← imm8	–	–	–	–	–	–	–	–	–	–
MOV ear, #imm8	3	2	1	0	byte (ear) ← imm8	–	–	–	–	–	*	*	–	–	–
MOV eam, #imm8	3+	4+ (a)	0	(b)	byte (eam) ← imm8	–	–	–	–	–	–	–	–	–	–
MOV @AL, AH /MOV @A, T	2	3	0	(b)	byte ((A)) ← (AH)	–	–	–	–	–	*	*	–	–	–
XCH A, ear	2	4	2	0	byte (A) ↔ (ear)	Z	–	–	–	–	–	–	–	–	–
XCH A, eam	2+	5+ (a)	0	2× (b)	byte (A) ↔ (eam)	Z	–	–	–	–	–	–	–	–	–
XCH Ri, ear	2	7	4	0	byte (Ri) ↔ (ear)	–	–	–	–	–	–	–	–	–	–
XCH Ri, eam	2+	9+ (a)	2	2× (b)	byte (Ri) ↔ (eam)	–	–	–	–	–	–	–	–	–	–

Note : For an explanation of “(a)” to “(d)”, refer to Table 4, “Number of Execution Cycles for Each Type of Addressing,” and Table 5, “Correction Values for Number of Cycles Used to Calculate Number of Actual Cycles.”

Table 8 Transfer Instructions (Word/Long Word) [38 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVW A, dir	2	3	0	(c)	word (A) ← (dir)	–	*	–	–	–	*	*	–	–	–
MOVW A, addr16	3	4	0	(c)	word (A) ← (addr16)	–	*	–	–	–	*	*	–	–	–
MOVW A, SP	1	1	0	0	word (A) ← (SP)	–	*	–	–	–	*	*	–	–	–
MOVW A, RWi	1	2	1	0	word (A) ← (RWi)	–	*	–	–	–	*	*	–	–	–
MOVW A, ear	2	2	1	0	word (A) ← (ear)	–	*	–	–	–	*	*	–	–	–
MOVW A, eam	2+	3+ (a)	0	(c)	word (A) ← (eam)	–	*	–	–	–	*	*	–	–	–
MOVW A, io	2	3	0	(c)	word (A) ← (io)	–	*	–	–	–	*	*	–	–	–
MOVW A, @A	2	3	0	(c)	word (A) ← ((A))	–	–	–	–	–	*	*	–	–	–
MOVW A, #imm16	3	2	0	0	word (A) ← imm16	–	*	–	–	–	*	*	–	–	–
MOVW A, @RWi+disp8	2	5	1	(c)	word (A) ← ((RWi) +disp8)	–	*	–	–	–	*	*	–	–	–
MOVW A, @RLi+disp8	3	10	2	(c)	word (A) ← ((RLi) +disp8)	–	*	–	–	–	*	*	–	–	–
MOVW dir, A	2	3	0	(c)	word (dir) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW addr16, A	3	4	0	(c)	word (addr16) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW SP, A	1	1	0	0	word (SP) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW RWi, A	1	2	1	0	word (RWi) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW ear, A	2	2	1	0	word (ear) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW eam, A	2+	3+ (a)	0	(c)	word (eam) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW io, A	2	3	0	(c)	word (io) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW @RWi+disp8, A	2	5	1	(c)	word ((RWi) +disp8) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW @RLi+disp8, A	3	10	2	(c)	word ((RLi) +disp8) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW RWi, ear	2	3	2	(0)	word (RWi) ← (ear)	–	–	–	–	–	*	*	–	–	–
MOVW RWi, eam	2+	4+ (a)	1	(c)	word (RWi) ← (eam)	–	–	–	–	–	*	*	–	–	–
MOVW ear, RWi	2	4	2	0	word (ear) ← (RWi)	–	–	–	–	–	*	*	–	–	–
MOVW eam, RWi	2+	5+ (a)	1	(c)	word (eam) ← (RWi)	–	–	–	–	–	*	*	–	–	–
MOVW RWi, #imm16	3	2	1	0	word (RWi) ← imm16	–	–	–	–	–	*	*	–	–	–
MOVW io, #imm16	4	5	0	(c)	word (io) ← imm16	–	–	–	–	–	–	–	–	–	–
MOVW ear, #imm16	4	2	1	0	word (ear) ← imm16	–	–	–	–	–	*	*	–	–	–
MOVW eam, #imm16	4+	4+ (a)	0	(c)	word (eam) ← imm16	–	–	–	–	–	–	–	–	–	–
MOVW @AL, AH /MOVW@A, T	2	3	0	(c)	word ((A)) ← (AH)	–	–	–	–	–	*	*	–	–	–
XCHW A, ear	2	4	2	0	word (A) ↔ (ear)	–	–	–	–	–	–	–	–	–	–
XCHW A, eam	2+	5+ (a)	0	2× (c)	word (A) ↔ (eam)	–	–	–	–	–	–	–	–	–	–
XCHW RWi, ear	2	7	4	0	word (RWi) ↔ (ear)	–	–	–	–	–	–	–	–	–	–
XCHW RWi, eam	2+	9+ (a)	2	2× (c)	word (RWi) ↔ (eam)	–	–	–	–	–	–	–	–	–	–
MOVL A, ear	2	4	2	0	long (A) ← (ear)	–	–	–	–	–	*	*	–	–	–
MOVL A, eam	2+	5+ (a)	0	(d)	long (A) ← (eam)	–	–	–	–	–	*	*	–	–	–
MOVL A, #imm32	5	3	0	0	long (A) ← imm32	–	–	–	–	–	*	*	–	–	–
MOVL ear, A	2	4	2	0	long (ear) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVL eam, A	2+	5+ (a)	0	(d)	long (eam) ← (A)	–	–	–	–	–	*	*	–	–	–

Note : For an explanation of “(a)” to “(d)”, refer to Table 4, “Number of Execution Cycles for Each Type of Addressing,” and Table 5, “Correction Values for Number of Cycles Used to Calculate Number of Actual Cycles.”

Table 9 Addition and Subtraction Instructions (Byte/Word/Long Word) [42 Instructions]

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ADD	A,#imm8	2	2	0	0	byte (A) ← (A) +imm8	Z	–	–	–	–	*	*	*	*	–
ADD	A, dir	2	5	0	(b)	byte (A) ← (A) +(dir)	Z	–	–	–	–	*	*	*	*	–
ADD	A, ear	2	3	1	0	byte (A) ← (A) +(ear)	Z	–	–	–	–	*	*	*	*	–
ADD	A, eam	2+	4+ (a)	0	(b)	byte (A) ← (A) +(eam)	Z	–	–	–	–	*	*	*	*	–
ADD	ear, A	2	3	2	0	byte (ear) ← (ear) + (A)	–	–	–	–	–	*	*	*	*	–
ADD	eam, A	2+	5+ (a)	0	2× (b)	byte (eam) ← (eam) + (A)	Z	–	–	–	–	*	*	*	*	*
ADDC	A	1	2	0	0	byte (A) ← (AH) + (AL) + (C)	Z	–	–	–	–	*	*	*	*	–
ADDC	A, ear	2	3	1	0	byte (A) ← (A) + (ear) + (C)	Z	–	–	–	–	*	*	*	*	–
ADDC	A, eam	2+	4+ (a)	0	(b)	byte (A) ← (A) + (eam) + (C)	Z	–	–	–	–	*	*	*	*	–
ADDDC	A	1	3	0	0	byte (A) ← (AH) + (AL) + (C) (decimal)	Z	–	–	–	–	*	*	*	*	–
SUB	A, #imm8	2	2	0	0	byte (A) ← (A) –imm8	Z	–	–	–	–	*	*	*	*	–
SUB	A, dir	2	5	0	(b)	byte (A) ← (A) – (dir)	Z	–	–	–	–	*	*	*	*	–
SUB	A, ear	2	3	1	0	byte (A) ← (A) – (ear)	Z	–	–	–	–	*	*	*	*	–
SUB	A, eam	2+	4+ (a)	0	(b)	byte (A) ← (A) – (eam)	Z	–	–	–	–	*	*	*	*	–
SUB	ear, A	2	3	2	0	byte (ear) ← (ear) – (A)	–	–	–	–	–	*	*	*	*	–
SUB	eam, A	2+	5+ (a)	0	2× (b)	byte (eam) ← (eam) – (A)	–	–	–	–	–	*	*	*	*	*
SUBC	A	1	2	0	0	byte (A) ← (AH) – (AL) – (C)	Z	–	–	–	–	*	*	*	*	–
SUBC	A, ear	2	3	1	0	byte (A) ← (A) – (ear) – (C)	Z	–	–	–	–	*	*	*	*	–
SUBC	A, eam	2+	4+ (a)	0	(b)	byte (A) ← (A) – (eam) – (C)	Z	–	–	–	–	*	*	*	*	–
SUBDC	A	1	3	0	0	byte (A) ← (AH) – (AL) – (C) (decimal)	Z	–	–	–	–	*	*	*	*	–
ADDW	A	1	2	0	0	word (A) ← (AH) + (AL)	–	–	–	–	–	*	*	*	*	–
ADDW	A, ear	2	3	1	0	word (A) ← (A) +(ear)	–	–	–	–	–	*	*	*	*	–
ADDW	A, eam	2+	4+ (a)	0	(c)	word (A) ← (A) +(eam)	–	–	–	–	–	*	*	*	*	–
ADDW	A, #imm16	3	2	0	0	word (A) ← (A) +imm16	–	–	–	–	–	*	*	*	*	–
ADDW	ear, A	2	3	2	0	word (ear) ← (ear) + (A)	–	–	–	–	–	*	*	*	*	–
ADDW	eam, A	2+	5+ (a)	0	2× (c)	word (eam) ← (eam) + (A)	–	–	–	–	–	*	*	*	*	*
ADDCW	A, ear	2	3	1	0	word (A) ← (A) + (ear) + (C)	–	–	–	–	–	*	*	*	*	–
ADDCW	A, eam	2+	4+ (a)	0	(c)	word (A) ← (A) + (eam) + (C)	–	–	–	–	–	*	*	*	*	–
SUBW	A	1	2	0	0	word (A) ← (AH) – (AL)	–	–	–	–	–	*	*	*	*	–
SUBW	A, ear	2	3	1	0	word (A) ← (A) – (ear)	–	–	–	–	–	*	*	*	*	–
SUBW	A, eam	2+	4+ (a)	0	(c)	word (A) ← (A) – (eam)	–	–	–	–	–	*	*	*	*	–
SUBW	A, #imm16	3	2	0	0	word (A) ← (A) –imm16	–	–	–	–	–	*	*	*	*	–
SUBW	ear, A	2	3	2	0	word (ear) ← (ear) – (A)	–	–	–	–	–	*	*	*	*	–
SUBW	eam, A	2+	5+ (a)	0	2× (c)	word (eam) ← (eam) – (A)	–	–	–	–	–	*	*	*	*	*
SUBCW	A, ear	2	3	1	0	word (A) ← (A) – (ear) – (C)	–	–	–	–	–	*	*	*	*	–
SUBCW	A, eam	2+	4+ (a)	0	(c)	word (A) ← (A) – (eam) – (C)	–	–	–	–	–	*	*	*	*	–
ADDL	A, ear	2	6	2	0	long (A) ← (A) + (ear)	–	–	–	–	–	*	*	*	*	–
ADDL	A, eam	2+	7+ (a)	0	(d)	long (A) ← (A) + (eam)	–	–	–	–	–	*	*	*	*	–
ADDL	A, #imm32	5	4	0	0	long (A) ← (A) +imm32	–	–	–	–	–	*	*	*	*	–
SUBL	A, ear	2	6	2	0	long (A) ← (A) – (ear)	–	–	–	–	–	*	*	*	*	–
SUBL	A, eam	2+	7+ (a)	0	(d)	long (A) ← (A) – (eam)	–	–	–	–	–	*	*	*	*	–
SUBL	A, #imm32	5	4	0	0	long (A) ← (A) –imm32	–	–	–	–	–	*	*	*	*	–

Note : For an explanation of “(a)” to “(d)”, refer to Table 4, “Number of Execution Cycles for Each Type of Addressing,” and Table 5, “Correction Values for Number of Cycles Used to Calculate Number of Actual Cycles.”

Table 10 Increment and Decrement Instructions (Byte/Word/Long Word) [12 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
INC ear	2	2	2	0	byte (ear) ← (ear) +1	–	–	–	–	–	*	*	*	–	–
INC eam	2+	5+ (a)	0	2× (b)	byte (eam) ← (eam) +1	–	–	–	–	–	*	*	*	–	*
DEC ear	2	3	2	0	byte (ear) ← (ear) –1	–	–	–	–	–	*	*	*	–	–
DEC eam	2+	5+ (a)	0	2× (b)	byte (eam) ← (eam) –1	–	–	–	–	–	*	*	*	–	*
INCW ear	2	3	2	0	word (ear) ← (ear) +1	–	–	–	–	–	*	*	*	–	–
INCW eam	2+	5+ (a)	0	2× (c)	word (eam) ← (eam) +1	–	–	–	–	–	*	*	*	–	*
DECW ear	2	3	2	0	word (ear) ← (ear) –1	–	–	–	–	–	*	*	*	–	–
DECW eam	2+	5+ (a)	0	2× (c)	word (eam) ← (eam) –1	–	–	–	–	–	*	*	*	–	*
INCL ear	2	7	4	0	long (ear) ← (ear) +1	–	–	–	–	–	*	*	*	–	–
INCL eam	2+	9+ (a)	0	2× (d)	long (eam) ← (eam) +1	–	–	–	–	–	*	*	*	–	*
DECL ear	2	7	4	0	long (ear) ← (ear) –1	–	–	–	–	–	*	*	*	–	–
DECL eam	2+	9+ (a)	0	2× (d)	long (eam) ← (eam) –1	–	–	–	–	–	*	*	*	–	*

Note : For an explanation of "(a)" to "(d)", refer to Table 4, "Number of Execution Cycles for Each Type of Addressing," and Table 5, "Correction Values for Number of Cycles Used to Calculate Number of Actual Cycles."

Table 11 Compare Instructions (Byte/Word/Long Word) [11 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CMP A	1	1	0	0	byte (AH) – (AL)	–	–	–	–	–	*	*	*	*	–
CMP A, ear	2	2	1	0	byte (A) ← (ear)	–	–	–	–	–	*	*	*	*	–
CMP A, eam	2+	3+ (a)	0	(b)	byte (A) ← (eam)	–	–	–	–	–	*	*	*	*	–
CMP A, #imm8	2	2	0	0	byte (A) ← imm8	–	–	–	–	–	*	*	*	*	–
CMPW A	1	1	0	0	word (AH) – (AL)	–	–	–	–	–	*	*	*	*	–
CMPW A, ear	2	2	1	0	word (A) ← (ear)	–	–	–	–	–	*	*	*	*	–
CMPW A, eam	2+	3+ (a)	0	(c)	word (A) ← (eam)	–	–	–	–	–	*	*	*	*	–
CMPW A, #imm16	3	2	0	0	word (A) ← imm16	–	–	–	–	–	*	*	*	*	–
CMPL A, ear	2	6	2	0	word (A) ← (ear)	–	–	–	–	–	*	*	*	*	–
CMPL A, eam	2+	7+ (a)	0	(d)	word (A) ← (eam)	–	–	–	–	–	*	*	*	*	–
CMPL A, #imm32	5	3	0	0	word (A) ← imm32	–	–	–	–	–	*	*	*	*	–

Note : For an explanation of "(a)" to "(d)", refer to Table 4, "Number of Execution Cycles for Each Type of Addressing," and Table 5, "Correction Values for Number of Cycles Used to Calculate Number of Actual Cycles."

Table 12 Multiplication and Division Instructions (Byte/Word/Long Word) [11 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIVU A	1	*1	0	0	word (AH) /byte (AL) Quotient → byte (AL) Remainder → byte (AH)	–	–	–	–	–	–	–	*	*	–
DIVU A, ear	2	*2	1	0	word (A)/byte (ear) Quotient → byte (A) Remainder → byte (ear)	–	–	–	–	–	–	–	*	*	–
DIVU A, eam	2+	*3	0	*6	word (A)/byte (eam) Quotient → byte (A) Remainder → byte (eam)	–	–	–	–	–	–	–	*	*	–
DIVUW A, ear	2	*4	1	0	long (A)/word (ear) Quotient → word (A) Remainder → word (ear)	–	–	–	–	–	–	–	*	*	–
DIVUW A, eam	2+	*5	0	*7	long (A)/word (eam) Quotient → word (A) Remainder → word (ear)	–	–	–	–	–	–	–	*	*	–
MULU A	1	*8	0	0	byte (AH) *byte (AL) → word (A)	–	–	–	–	–	–	–	–	–	–
MULU A, ear	2	*9	1	0	byte (A) *byte (ear) → word (A)	–	–	–	–	–	–	–	–	–	–
MULU A, eam	2+	*10	0	(b)	byte (A) *byte (eam) → word (A)	–	–	–	–	–	–	–	–	–	–
MULUW A	1	*11	0	0	word (AH) *word (AL) → long (A)	–	–	–	–	–	–	–	–	–	–
MULUW A, ear	2	*12	1	0	word (A) *word (ear) → long (A)	–	–	–	–	–	–	–	–	–	–
MULUW A, eam	2+	*13	0	(c)	word (A) *word (eam) → long (A)	–	–	–	–	–	–	–	–	–	–

*1: 3 when the result is zero, 7 when an overflow occurs, and 15 normally.
*2: 4 when the result is zero, 8 when an overflow occurs, and 16 normally.
*3: 6 + (a) when the result is zero, 9 + (a) when an overflow occurs, and 19 + (a) normally.
*4: 4 when the result is zero, 7 when an overflow occurs, and 22 normally.
*5: 6 + (a) when the result is zero, 8 + (a) when an overflow occurs, and 26 + (a) normally.
*6: (b) when the result is zero or when an overflow occurs, and 2 × (b) normally.
*7: (c) when the result is zero or when an overflow occurs, and 2 × (c) normally.
*8: 3 when byte (AH) is zero, and 7 when byte (AH) is not zero.
*9: 4 when byte (ear) is zero, and 8 when byte (ear) is not zero.
*10: 5 + (a) when byte (eam) is zero, and 9 + (a) when byte (eam) is not 0.
*11: 3 when word (AH) is zero, and 11 when word (AH) is not zero.
*12: 4 when word (ear) is zero, and 12 when word (ear) is not zero.
*13: 5 + (a) when word (eam) is zero, and 13 + (a) when word (eam) is not zero.

Note : For an explanation of "(a)" to "(d)", refer to Table 4, "Number of Execution Cycles for Each Type of Addressing," and Table 5, "Correction Values for Number of Cycles Used to Calculate Number of Actual Cycles."

Table 13 Signed Multiplication and Division Instructions (Byte/Word/Long Word) [11 Instructions]

Mnemonic		#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
DIV	A	2	*1	0	0	word (AH) /byte (AL) Quotient → byte (AL) Remainder → byte (AH)	Z	–	–	–	–	–	–	*	*	–
DIV	A, ear	2	*2	1	0	word (A)/byte (ear) Quotient → byte (A) Remainder → byte (ear)	Z	–	–	–	–	–	–	*	*	–
DIV	A, eam	2 +	*3	0	*6	word (A)/byte (eam) Quotient → byte (A) Remainder → byte (eam)	Z	–	–	–	–	–	–	*	*	–
DIVW	A, ear	2	*4	1	0	long (A)/word (ear) Quotient → word (A) Remainder → word (ear)	–	–	–	–	–	–	–	*	*	–
DIVW	A, eam	2+	*5	0	*7	long (A)/word (eam) Quotient → word (A) Remainder → word (eam)	–	–	–	–	–	–	–	*	*	–
MULU	A	2	*8	0	0	byte (AH) *byte (AL) → word (A)	–	–	–	–	–	–	–	–	–	–
MULU	A, ear	2	*9	1	0	byte (A) *byte (ear) → word (A)	–	–	–	–	–	–	–	–	–	–
MULU	A, eam	2 +	*10	0	(b)	byte (A) *byte (eam) → word (A)	–	–	–	–	–	–	–	–	–	–
MULUW	A	2	*11	0	0	word (AH) *word (AL) → long (A)	–	–	–	–	–	–	–	–	–	–
MULUW	A, ear	2	*12	1	0	word (A) *word (ear) → long (A)	–	–	–	–	–	–	–	–	–	–
MULUW	A, eam	2 +	*13	0	(c)	word (A) *word (eam) → long (A)	–	–	–	–	–	–	–	–	–	–

*1: Set to 3 when the division-by-0, 8 or 18 for an overflow, and 18 for normal operation.

*2: Set to 3 when the division-by-0, 10 or 21 for an overflow, and 22 for normal operation.

*3: Set to 4 + (a) when the division-by-0, 11 + (a) or 22 + (a) for an overflow, and 23 + (a) for normal operation.

*4: Positive dividend: Set to 4 when the division-by-0, 10 or 29 for an overflow, and 30 for normal operation.
Negative dividend: Set to 4 when the division-by-0, 11 or 30 for an overflow and 31 for normal operation.

*5: Positive dividend: Set to 4 + (a) when the division-by-0, 11 + (a) or 30 + (a) for an overflow, and 31 + (a) for normal operation.
Negative dividend: Set to 4 + (a) when the division-by-0, 12 + (a) or 31 + (a) for an overflow, and 32 + (a) for normal operation.

*6: When the division-by-0, (b) for an overflow, and 2 × (b) for normal operation.

*7: When the division-by-0, (c) for an overflow, and 2 × (c) for normal operation.

*8: Set to 3 when byte (AH) is zero, 12 when the result is positive, and 13 when the result is negative.

*9: Set to 3 when byte (ear) is zero, 12 when the result is positive, and 13 when the result is negative.

*10: Set to 4 + (a) when byte (eam) is zero, 13 + (a) when the result is positive, and 14 + (a) when the result is negative.

*11: Set to 3 when word (AH) is zero, 12 when the result is positive, and 13 when the result is negative.

*12: Set to 3 when word (ear) is zero, 16 when the result is positive, and 19 when the result is negative.

*13: Set to 4 + (a) when word (eam) is zero, 17 + (a) when the result is positive, and 20 + (a) when the result is negative.

Notes:
- When overflow occurs during DIV or DIVW instruction execution, the number of execution cycles takes two values because of detection before and after an operation.
- When overflow occurs during DIV or DIVW instruction execution, the contents of AL are destroyed.
- For (a) to (d), refer to “Table 4 Number of Execution Cycles for Effective Address in Addressing Modes” and “Table 5 Correction Values for Number of Cycles for Calculating Actual Number of Cycles.”

Table 14 Logical 1 Instructions (Byte/Word) [39 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
AND A, #imm8	2	2	0	0	byte (A) ← (A) and imm8	–	–	–	–	–	*	*	R	–	–
AND A, ear	2	3	1	0	byte (A) ← (A) and (ear)	–	–	–	–	–	*	*	R	–	–
AND A, eam	2+	4+ (a)	0	(b)	byte (A) ← (A) and (eam)	–	–	–	–	–	*	*	R	–	–
AND ear, A	2	3	2	0	byte (ear) ← (ear) and (A)	–	–	–	–	–	*	*	R	–	–
AND eam, A	2+	5+ (a)	0	2× (b)	byte (eam) ← (eam) and (A)	–	–	–	–	–	*	*	R	–	*
OR A, #imm8	2	2	0	0	byte (A) ← (A) or imm8	–	–	–	–	–	*	*	R	–	–
OR A, ear	2	3	1	0	byte (A) ← (A) or (ear)	–	–	–	–	–	*	*	R	–	–
OR A, eam	2+	4+ (a)	0	(b)	byte (A) ← (A) or (eam)	–	–	–	–	–	*	*	R	–	–
OR ear, A	2	3	2	0	byte (ear) ← (ear) or (A)	–	–	–	–	–	*	*	R	–	–
OR eam, A	2+	5+ (a)	0	2× (b)	byte (eam) ← (eam) or (A)	–	–	–	–	–	*	*	R	–	*
XOR A, #imm8	2	2	0	0	byte (A) ← (A) xor imm8	–	–	–	–	–	*	*	R	–	–
XOR A, ear	2	3	1	0	byte (A) ← (A) xor (ear)	–	–	–	–	–	*	*	R	–	–
XOR A, eam	2+	4+ (a)	0	(b)	byte (A) ← (A) xor (eam)	–	–	–	–	–	*	*	R	–	–
XOR ear, A	2	3	2	0	byte (ear) ← (ear) xor (A)	–	–	–	–	–	*	*	R	–	–
XOR eam, A	2+	5+ (a)	0	2× (b)	byte (eam) ← (eam) xor (A)	–	–	–	–	–	*	*	R	–	*
NOT A	1	2	0	0	byte (A) ← not (A)	–	–	–	–	–	*	*	R	–	–
NOT ear	2	3	2	0	byte (ear) ← not (ear)	–	–	–	–	–	*	*	R	–	–
NOT eam	2+	5+ (a)	0	2× (b)	byte (eam) ← not (eam)	–	–	–	–	–	*	*	R	–	*
ANDW A	1	2	0	0	word (A) ← (AH) and (A)	–	–	–	–	–	*	*	R	–	–
ANDW A, #imm16	3	2	0	0	word (A) ← (A) and imm16	–	–	–	–	–	*	*	R	–	–
ANDW A, ear	2	3	1	0	word (A) ← (A) and (ear)	–	–	–	–	–	*	*	R	–	–
ANDW A, eam	2+	4+ (a)	0	(c)	word (A) ← (A) and (eam)	–	–	–	–	–	*	*	R	–	–
ANDW ear, A	2	3	2	0	word (ear) ← (ear) and (A)	–	–	–	–	–	*	*	R	–	–
ANDW eam, A	2+	5+ (a)	0	2× (c)	word (eam) ← (eam) and (A)	–	–	–	–	–	*	*	R	–	*
ORW A	1	2	0	0	word (A) ← (AH) or (A)	–	–	–	–	–	*	*	R	–	–
ORW A, #imm16	3	2	0	0	word (A) ← (A) or imm16	–	–	–	–	–	*	*	R	–	–
ORW A, ear	2	3	1	0	word (A) ← (A) or (ear)	–	–	–	–	–	*	*	R	–	–
ORW A, eam	2+	4+ (a)	0	(c)	word (A) ← (A) or (eam)	–	–	–	–	–	*	*	R	–	–
ORW ear, A	2	3	2	0	word (ear) ← (ear) or (A)	–	–	–	–	–	*	*	R	–	–
ORW eam, A	2+	5+ (a)	0	2× (c)	word (eam) ← (eam) or (A)	–	–	–	–	–	*	*	R	–	*
XORW A	1	2	0	0	word (A) ← (AH) xor (A)	–	–	–	–	–	*	*	R	–	–
XORW A, #imm16	3	2	0	0	word (A) ← (A) xor imm16	–	–	–	–	–	*	*	R	–	–
XORW A, ear	2	3	1	0	word (A) ← (A) xor (ear)	–	–	–	–	–	*	*	R	–	–
XORW A, eam	2+	4+ (a)	0	(c)	word (A) ← (A) xor (eam)	–	–	–	–	–	*	*	R	–	–
XORW ear, A	2	3	2	0	word (ear) ← (ear) xor (A)	–	–	–	–	–	*	*	R	–	–
XORW eam, A	2+	5+ (a)	0	2× (c)	word (eam) ← (eam) xor (A)	–	–	–	–	–	*	*	R	–	*
NOTW A	1	2	0	0	word (A) ← not (A)	–	–	–	–	–	*	*	R	–	–
NOTW ear	2	3	2	0	word (ear) ← not (ear)	–	–	–	–	–	*	*	R	–	–
NOTW eam	2+	5+ (a)	0	2× (c)	word (eam) ← not (eam)	–	–	–	–	–	*	*	R	–	*

Note : For an explanation of “(a)” to “(d)”, refer to Table 4, “Number of Execution Cycles for Each Type of Addressing,” and Table 5, “Correction Values for Number of Cycles Used to Calculate Number of Actual Cycles.”

Table 15 Logical 2 Instructions (Long Word) [6 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ANDL A, ear	2	6	2	0	long (A) ← (A) and (ear)	–	–	–	–	–	*	*	R	–	–
ANDL A, eam	2+	7+ (a)	0	(d)	long (A) ← (A) and (eam)	–	–	–	–	–	*	*	R	–	–
ORL A, ear	2	6	2	0	long (A) ← (A) or (ear)	–	–	–	–	–	*	*	R	–	–
ORL A, eam	2+	7+ (a)	0	(d)	long (A) ← (A) or (eam)	–	–	–	–	–	*	*	R	–	–
XORL A, ea	2	6	2	0	long (A) ← (A) xor (ear)	–	–	–	–	–	*	*	R	–	–
XORL A, eam	2+	7+ (a)	0	(d)	long (A) ← (A) xor (eam)	–	–	–	–	–	*	*	R	–	–

Table 16 Sign Inversion Instructions (Byte/Word) [6 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NEG A	1	2	0	0	byte (A) ← 0 – (A)	X	–	–	–	–	*	*	*	*	–
NEG ear	2	3	2	0	byte (ear) ← 0 – (ear)	–	–	–	–	–	*	*	*	*	–
NEG eam	2+	5+ (a)	0	2× (b)	byte (eam) ← 0 – (eam)	–	–	–	–	–	*	*	*	*	*
NEGW A	1	2	0	0	word (A) ← 0 – (A)	–	–	–	–	–	*	*	*	*	–
NEGW ear	2	3	2	0	word (ear) ← 0 – (ear)	–	–	–	–	–	*	*	*	*	–
NEGW eam	2+	5+ (a)	0	2× (c)	word (eam) ← 0 – (eam)	–	–	–	–	–	*	*	*	*	*

Table 17 Normalize Instruction (Long Word) [1 Instruction]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NRML A, R0	2	*1	1	0	long (A) ← Shift until first digit is “1” byte (R0) ← Current shift count	–	–	–	–	–	–	*	–	–	–

*1: 4 when the contents of the accumulator are all zeroes, 6 + (R0) in all other cases (shift count).

Note : For an explanation of “(a)” to “(d)”, refer to Table 4, “Number of Execution Cycles for Each Type of Addressing,” and Table 5, “Correction Values for Number of Cycles Used to Calculate Number of Actual Cycles.”

Table 18 Shift Instructions (Byte/Word/Long Word) [18 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
RORC A	2	2	0	0	byte (A) ← Right rotation with carry	–	–	–	–	–	*	*	–	*	–
ROLC A	2	2	0	0	byte (A) ← Left rotation with carry	–	–	–	–	–	*	*	–	*	–
RORC ear	2	3	2	0	byte (ear) ← Right rotation with carry	–	–	–	–	–	*	*	–	*	–
RORC eam	2+	5+ (a)	0	2× (b)	byte (eam) ← Right rotation with carry	–	–	–	–	–	*	*	–	*	*
ROLC ear	2	3	2	0	byte (ear) ← Left rotation with carry	–	–	–	–	–	*	*	–	*	–
ROLC eam	2+	5+ (a)	0	2× (b)	byte (eam) ← Left rotation with carry	–	–	–	–	–	*	*	–	*	*
ASR A, R0	2	*1	1	0	byte (A) ← Arithmetic right barrel shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSR A, R0	2	*1	1	0	byte (A) ← Logical right barrel shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSL A, R0	2	*1	1	0	byte (A) ← Logical left barrel shift (A, R0)	–	–	–	–	–	*	*	–	*	–
ASRW A	1	2	0	0	word (A) ← Arithmetic right shift (A, 1 bit)	–	–	–	–	*	*	*	–	*	–
LSRW A/SHRW A	1	2	0	0	word (A) ← Logical right shift (A, 1 bit)	–	–	–	–	*	R	*	–	*	–
LSLW A/SHLW A	1	2	0	0	word (A) ← Logical left shift (A, 1 bit)	–	–	–	–	–	*	*	–	*	–
ASRW A, R0	2	*1	1	0	word (A) ← Arithmetic right barrel shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSRW A, R0	2	*1	1	0	word (A) ← Logical right barrel shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSLW A, R0	2	*1	1	0	word (A) ← Logical left barrel shift (A, R0)	–	–	–	–	–	*	*	–	*	–
ASRL A, R0	2	*2	1	0	long (A) ← Arithmetic right shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSRL A, R0	2	*2	1	0	long (A) ← Logical right barrel shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSLL A, R0	2	*2	1	0	long (A) ← Logical left barrel shift (A, R0)	–	–	–	–	–	*	*	–	*	–

*1: 6 when R0 is 0, 5 + (R0) in all other cases.
*2: 6 when R0 is 0, 6 + (R0) in all other cases.

Note : For an explanation of “(a)” to “(d)”, refer to Table 4, “Number of Execution Cycles for Each Type of Addressing,” and Table 5, “Correction Values for Number of Cycles Used to Calculate Number of Actual Cycles.”

Table 19 Branch 1 Instructions [31 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
BZ/BEQ rel	2	*1	0	0	Branch when (Z) = 1	–	–	–	–	–	–	–	–	–	–
BNZ/BNE rel	2	*1	0	0	Branch when (Z) = 0	–	–	–	–	–	–	–	–	–	–
BC/BLO rel	2	*1	0	0	Branch when (C) = 1	–	–	–	–	–	–	–	–	–	–
BNC/BHS rel	2	*1	0	0	Branch when (C) = 0	–	–	–	–	–	–	–	–	–	–
BN rel	2	*1	0	0	Branch when (N) = 1	–	–	–	–	–	–	–	–	–	–
BP rel	2	*1	0	0	Branch when (N) = 0	–	–	–	–	–	–	–	–	–	–
BV rel	2	*1	0	0	Branch when (V) = 1	–	–	–	–	–	–	–	–	–	–
BNV rel	2	*1	0	0	Branch when (V) = 0	–	–	–	–	–	–	–	–	–	–
BT rel	2	*1	0	0	Branch when (T) = 1	–	–	–	–	–	–	–	–	–	–
BNT rel	2	*1	0	0	Branch when (T) = 0	–	–	–	–	–	–	–	–	–	–
BLT rel	2	*1	0	0	Branch when (V) xor (N) = 1	–	–	–	–	–	–	–	–	–	–
BGE rel	2	*1	0	0	Branch when (V) xor (N) = 0	–	–	–	–	–	–	–	–	–	–
BLE rel	2	*1	0	0	Branch when ((V) xor (N)) or (Z) = 1	–	–	–	–	–	–	–	–	–	–
BGT rel	2	*1	0	0	Branch when ((V) xor (N)) or (Z) = 0	–	–	–	–	–	–	–	–	–	–
BLS rel	2	*1	0	0	Branch when (C) or (Z) = 1	–	–	–	–	–	–	–	–	–	–
BHI rel	2	*1	0	0	Branch when (C) or (Z) = 0	–	–	–	–	–	–	–	–	–	–
BRA rel	2	*1	0	0	Branch unconditionally	–	–	–	–	–	–	–	–	–	–
JMP @A	1	2	0	0	word (PC) ← (A)	–	–	–	–	–	–	–	–	–	–
JMP addr16	3	3	0	0	word (PC) ← addr16	–	–	–	–	–	–	–	–	–	–
JMP @ear	2	3	1	0	word (PC) ← (ear)	–	–	–	–	–	–	–	–	–	–
JMP @eam	2+	4+ (a)	0	(c)	word (PC) ← (eam)	–	–	–	–	–	–	–	–	–	–
JMPP @ear *3	2	5	2	0	word (PC) ← (ear), (PCB) ← (ear +2)	–	–	–	–	–	–	–	–	–	–
JMPP @eam *3	2+	6+ (a)	0	(d)	word (PC) ← (eam), (PCB) ← (eam +2)	–	–	–	–	–	–	–	–	–	–
JMPP addr24	4	4	0	0	word (PC) ← ad24 0 to 15, (PCB) ← ad24 16 to 23	–	–	–	–	–	–	–	–	–	–
CALL @ear *4	2	6	1	(c)	word (PC) ← (ear)	–	–	–	–	–	–	–	–	–	–
CALL @eam *4	2+	7+ (a)	0	2× (c)	word (PC) ← (eam)	–	–	–	–	–	–	–	–	–	–
CALL addr16 *5	3	6	0	(c)	word (PC) ← addr16	–	–	–	–	–	–	–	–	–	–
CALLV #vct4 *5	1	7	0	2× (c)	Vector call instruction	–	–	–	–	–	–	–	–	–	–
CALLP @ear *6	2	10	2	2× (c)	word (PC) ← (ear) 0 to 15, (PCB) ← (ear) 16 to 23	–	–	–	–	–	–	–	–	–	–
CALLP @eam *6	2+	11+ (a)	0	*2	word (PC) ← (eam) 0 to 15, (PCB) ← (eam) 16 to 23	–	–	–	–	–	–	–	–	–	–
CALLP addr24 *7	4	10	0	2× (c)	word (PC) ← addr0 to 15, (PCB) ← addr16 to 23	–	–	–	–	–	–	–	–	–	–

*1: 4 when branching, 3 when not branching.

*2: (b) + 3 × (c)

*3: Read (word) branch address.

*4: W: Save (word) to stack; R: read (word) branch address.

*5: Save (word) to stack.

*6: W: Save (long word) to W stack; R: read (long word) R branch address.

*7: Save (long word) to stack.

Note : For an explanation of “(a)” to “(d)”, refer to Table 4, “Number of Execution Cycles for Each Type of Addressing,” and Table 5, “Correction Values for Number of Cycles Used to Calculate Number of Actual Cycles.”

Table 20 Branch 2 Instructions [19 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
CBNE A, #imm8, rel	3	*1	0	0	Branch when byte (A) ≠ imm8	–	–	–	–	–	*	*	*	*	–
CWBNE A, #imm16, rel	4	*1	0	0	Branch when word (A) ≠ imm16	–	–	–	–	–	*	*	*	*	–
CBNE ear, #imm8, rel	4	*2	1	0	Branch when byte (ear) ≠ imm8	–	–	–	–	–	*	*	*	*	–
CBNE eam, #imm8, rel*10	4+	*3	0	(b)	Branch when byte (eam) ≠ imm8	–	–	–	–	–	*	*	*	*	–
CWBNE ear, #imm16, rel	5	*4	1	0	Branch when word (ear) ≠ imm16	–	–	–	–	–	*	*	*	*	–
CWBNE eam, #imm16, rel*10	5+	*3	0	(c)	Branch when word (eam) ≠ imm16	–	–	–	–	–	*	*	*	*	–
DBNZ ear, rel	3	*5	2	0	Branch when byte (ear) = (ear) – 1, and (ear) ≠ 0	–	–	–	–	–	*	*	*	–	–
DBNZ eam, rel	3+	*6	2	2× (b)	Branch when byte (eam) = (eam) – 1, and (eam) ≠ 0	–	–	–	–	–	*	*	*	–	*
DWBNZ ear, rel	3	*5	2	0	Branch when word (ear) = (ear) – 1, and (ear) ≠ 0	–	–	–	–	–	*	*	*	–	–
DWBNZ eam, rel	3+	*6	2	2× (c)	Branch when word (eam) = (eam) – 1, and (eam) ≠ 0	–	–	–	–	–	*	*	*	–	*
INT #vct8	2	20	0	8× (c)	Software interrupt	–	–	R	S	–	–	–	–	–	–
INT addr16	3	16	0	6× (c)	Software interrupt	–	–	R	S	–	–	–	–	–	–
INTP addr24	4	17	0	6× (c)	Software interrupt	–	–	R	S	–	–	–	–	–	–
INT9	1	20	0	8× (c)	Software interrupt	–	–	R	S	–	–	–	–	–	–
RETI	1	15	0	*7	Return from interrupt	–	–	*	*	*	*	*	*	*	–
LINK #local8	2	6	0	(c)	At constant entry, save old frame pointer to stack, set new frame pointer, and allocate local pointer area	–	–	–	–	–	–	–	–	–	–
UNLINK	1	5	0	(c)	At constant entry, retrieve old frame pointer from stack.	–	–	–	–	–	–	–	–	–	–
RET *8	1	4	0	(c)	Return from subroutine	–	–	–	–	–	–	–	–	–	–
RETP *9	1	6	0	(d)	Return from subroutine	–	–	–	–	–	–	–	–	–	–

*1: 5 when branching, 4 when not branching

*2: 13 when branching, 12 when not branching

*3: 7 + (a) when branching, 6 + (a) when not branching

*4: 8 when branching, 7 when not branching

*5: 7 when branching, 6 when not branching

*6: 8 + (a) when branching, 7 + (a) when not branching

*7: Set to 3 × (b) + 2 × (c) when an interrupt request occurs, and 6 × (c) for return.

*8: Retrieve (word) from stack

*9: Retrieve (long word) from stack

*10: In the CBNE/CWBNE instruction, do not use the RWj+ addressing mode.

Note : For an explanation of “(a)” to “(d)”, refer to Table 4, “Number of Execution Cycles for Each Type of Addressing,” and Table 5, “Correction Values for Number of Cycles Used to Calculate Number of Actual Cycles.”

Table 21 Other Control Instructions (Byte/Word/Long Word) [36 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
PUSHW A	1	4	0	(c)	word (SP) ← (SP) –2, ((SP)) ← (A)	–	–	–	–	–	–	–	–	–	–
PUSHW AH	1	4	0	(c)	word (SP) ← (SP) –2, ((SP)) ← (AH)	–	–	–	–	–	–	–	–	–	–
PUSHW PS	1	4	0	(c)	word (SP) ← (SP) –2, ((SP)) ← (PS)	–	–	–	–	–	–	–	–	–	–
PUSHW rlst	2	*3	*5	*4	(SP) ← (SP) –2n, ((SP)) ← (rlst)	–	–	–	–	–	–	–	–	–	–
POPW A	1	3	0	(c)	word (A) ← ((SP)), (SP) ← (SP) +2	–	*	–	–	–	–	–	–	–	–
POPW AH	1	3	0	(c)	word (AH) ← ((SP)), (SP) ← (SP) +2	–	–	–	–	–	–	–	–	–	–
POPW PS	1	4	0	(c)	word (PS) ← ((SP)), (SP) ← (SP) +2	–	–	*	*	*	*	*	*	*	–
POPW rlst	2	*2	*5	*4	(rlst) ← ((SP)), (SP) ← (SP) +2n	–	–	–	–	–	–	–	–	–	–
JCTX @A	1	14	0	6× (c)	Context switch instruction	–	–	*	*	*	*	*	*	*	–
AND CCR, #imm8	2	3	0	0	byte (CCR) ← (CCR) and imm8	–	–	*	*	*	*	*	*	*	–
OR CCR, #imm8	2	3	0	0	byte (CCR) ← (CCR) or imm8	–	–	*	*	*	*	*	*	*	–
MOV RP, #imm8	2	2	0	0	byte (RP) ←imm8	–	–	–	–	–	–	–	–	–	–
MOV ILM, #imm8	2	2	0	0	byte (ILM) ←imm8	–	–	–	–	–	–	–	–	–	–
MOVEA RWi, ear	2	3	1	0	word (RWi) ←ear	–	–	–	–	–	–	–	–	–	–
MOVEA RWi, eam	2+	2+ (a)	1	0	word (RWi) ←eam	–	–	–	–	–	–	–	–	–	–
MOVEA A, ear	2	1	0	0	word(A) ←ear	–	*	–	–	–	–	–	–	–	–
MOVEA A, eam	2+	1+ (a)	0	0	word (A) ←eam	–	*	–	–	–	–	–	–	–	–
ADDSP #imm8	2	3	0	0	word (SP) ← (SP) +ext (imm8)	–	–	–	–	–	–	–	–	–	–
ADDSP #imm16	3	3	0	0	word (SP) ← (SP) +imm16	–	–	–	–	–	–	–	–	–	–
MOV A, brgl	2	*1	0	0	byte (A) ← (brgl)	Z	*	–	–	–	*	*	–	–	–
MOV brg2, A	2	1	0	0	byte (brg2) ← (A)	–	–	–	–	–	*	*	–	–	–
NOP	1	1	0	0	No operation	–	–	–	–	–	–	–	–	–	–
ADB	1	1	0	0	Prefix code for accessing AD space	–	–	–	–	–	–	–	–	–	–
DTB	1	1	0	0	Prefix code for accessing DT space	–	–	–	–	–	–	–	–	–	–
PCB	1	1	0	0	Prefix code for accessing PC space	–	–	–	–	–	–	–	–	–	–
SPB	1	1	0	0	Prefix code for accessing SP space	–	–	–	–	–	–	–	–	–	–
NCC	1	1	0	0	Prefix code for no flag change	–	–	–	–	–	–	–	–	–	–
CMR	1	1	0	0	Prefix code for common register bank	–	–	–	–	–	–	–	–	–	–

*1: PCB, ADB, SSB, USB, and SPB : 1 state
DTB, DPR : 2 states

*2: 7 + 3 × (pop count) + 2 × (last register number to be popped), 7 when rlst = 0 (no transfer register)

*3: 29 + (push count) – 3 × (last register number to be pushed), 8 when rlst = 0 (no transfer register)

*4: Pop count × (c), or push count × (c)

*5: Pop count or push count.

Note : For an explanation of "(a)" to "(d)", refer to Table 4, "Number of Execution Cycles for Each Type of Addressing," and Table 5, "Correction Values for Number of Cycles Used to Calculate Number of Actual Cycles."

Table 22 Bit Manipulation Instructions [21 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVB A, dir:bp	3	5	0	(b)	byte (A) ← (dir:bp) b	Z	*	–	–	–	*	*	–	–	–
MOVB A, addr16:bp	4	5	0	(b)	byte (A) ← (addr16:bp) b	Z	*	–	–	–	*	*	–	–	–
MOVB A, io:bp	3	4	0	(b)	byte (A) ← (io:bp) b	Z	*	–	–	–	*	*	–	–	–
MOVB dir:bp, A	3	7	0	2× (b)	bit (dir:bp) b ← (A)	–	–	–	–	–	*	*	–	–	*
MOVB addr16:bp, A	4	7	0	2× (b)	bit (addr16:bp) b ← (A)	–	–	–	–	–	*	*	–	–	*
MOVB io:bp, A	3	6	0	2× (b)	bit (io:bp) b ← (A)	–	–	–	–	–	*	*	–	–	*
SETB dir:bp	3	7	0	2× (b)	bit (dir:bp) b ← 1	–	–	–	–	–	–	–	–	–	*
SETB addr16:bp	4	7	0	2× (b)	bit (addr16:bp) b ← 1	–	–	–	–	–	–	–	–	–	*
SETB io:bp	3	7	0	2× (b)	bit (io:bp) b ← 1	–	–	–	–	–	–	–	–	–	*
CLRB dir:bp	3	7	0	2× (b)	bit (dir:bp) b ← 0	–	–	–	–	–	–	–	–	–	*
CLRB addr16:bp	4	7	0	2× (b)	bit (addr16:bp) b ← 0	–	–	–	–	–	–	–	–	–	*
CLRB io:bp	3	7	0	2× (b)	bit (io:bp) b ← 0	–	–	–	–	–	–	–	–	–	*
BBC dir:bp, rel	4	*1	0	(b)	Branch when (dir:bp) b = 0	–	–	–	–	–	–	*	–	–	–
BBC addr16:bp, rel	5	*1	0	(b)	Branch when (addr16:bp) b = 0	–	–	–	–	–	–	*	–	–	–
BBC io:bp, rel	4	*2	0	(b)	Branch when (io:bp) b = 0	–	–	–	–	–	–	*	–	–	–
BBS dir:bp, rel	4	*1	0	(b)	Branch when (dir:bp) b = 1	–	–	–	–	–	–	*	–	–	–
BBS addr16:bp, rel	5	*1	0	(b)	Branch when (addr16:bp) b = 1	–	–	–	–	–	–	*	–	–	–
BBS io:bp, rel	4	*2	0	(b)	Branch when (io:bp) b = 1	–	–	–	–	–	–	*	–	–	–
SBBS addr16:bp, rel	5	*3	0	2× (b)	Branch when (addr16:bp) b = 1, bit = 1	–	–	–	–	–	–	*	–	–	*
WBTS io:bp	3	*4	0	*5	Wait until (io:bp) b = 1	–	–	–	–	–	–	–	–	–	–
WBTC io:bp	3	*4	0	*5	Wait until (io:bp) b = 0	–	–	–	–	–	–	–	–	–	–

*1: 8 when branching, 7 when not branching

*2: 7 when branching, 6 when not branching

*3: 10 when condition is satisfied, 9 when not satisfied

*4: Undefined count

*5: Until condition is satisfied

Note : For an explanation of “(a)” to “(d)”, refer to Table 4, “Number of Execution Cycles for Each Type of Addressing,” and Table 5, “Correction Values for Number of Cycles Used to Calculate Number of Actual Cycles.”

Table 23 Accumulator Manipulation Instructions (Byte/Word) [6 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
SWAP	1	3	0	0	byte (A) 0 to 7 ↔ (A) 8 to 15	–	–	–	–	–	–	–	–	–	–
SWAPW	1	2	0	0	word (AH) ↔ (AL)	–	*	–	–	–	–	–	–	–	–
EXT	1	1	0	0	byte sign extension	X	–	–	–	–	*	*	–	–	–
EXTW	1	2	0	0	word sign extension	–	X	–	–	–	*	*	–	–	–
ZEXT	1	1	0	0	byte zero extension	Z	–	–	–	–	R	*	–	–	–
ZEXTW	1	1	0	0	word zero extension	–	Z	–	–	–	R	*	–	–	–

Table 24 String Instructions [10 Instructions]

Mnemonic	#	~	RG	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVS/MOVSI	2	*2	*5	*3	Byte transfer @AH+ ← @AL+, counter = RW0	–	–	–	–	–	–	–	–	–	–
MOVSD	2	*2	*5	*3	Byte transfer @AH– ← @AL–, counter = RW0	–	–	–	–	–	–	–	–	–	–
SCEQ/SCEQI	2	*1	*5	*4	Byte retrieval (@AH+) – AL, counter = RW0	–	–	–	–	–	*	*	*	*	–
SCEQD	2	*1	*5	*4	Byte retrieval (@AH–) – AL, counter = RW0	–	–	–	–	–	*	*	*	*	–
FISL/FILSI	2	6m +6	*5	*3	Byte filling @AH+ ← AL, counter = RW0	–	–	–	–	–	*	*	–	–	–
MOVSW/MOVSWI	2	*2	*8	*6	Word transfer @AH+ ← @AL+, counter = RW0	–	–	–	–	–	–	–	–	–	–
MOVSWD	2	*2	*8	*6	Word transfer @AH– ← @AL–, counter = RW0	–	–	–	–	–	–	–	–	–	–
SCWEQ/SCWEQI	2	*1	*8	*7	Word retrieval (@AH+) – AL, counter = RW0	–	–	–	–	–	*	*	*	*	–
SCWEQD	2	*1	*8	*7	Word retrieval (@AH–) – AL, counter = RW0	–	–	–	–	–	*	*	*	*	–
FILSW/FILSWI	2	6m +6	*8	*6	Word filling @AH+ ← AL, counter = RW0	–	–	–	–	–	*	*	–	–	–

m: RW0 value (counter value)
n: Loop count

*1: 5 when RW0 is 0, 4 + 7 × (RW0) for count out, and 7 × n + 5 when match occurs
*2: 5 when RW0 is 0, 4 + 8 × (RW0) in any other case
*3: (b) × (RW0) + (b) × (RW0) when accessing different areas for the source and destination, calculate (b) separately for each.
*4: (b) × n
*5: 2 × (RW0)
*6: (c) × (RW0) + (c) × (RW0) when accessing different areas for the source and destination, calculate (c) separately for each.
*7: (c) × n
*8: 2 × (RW0)

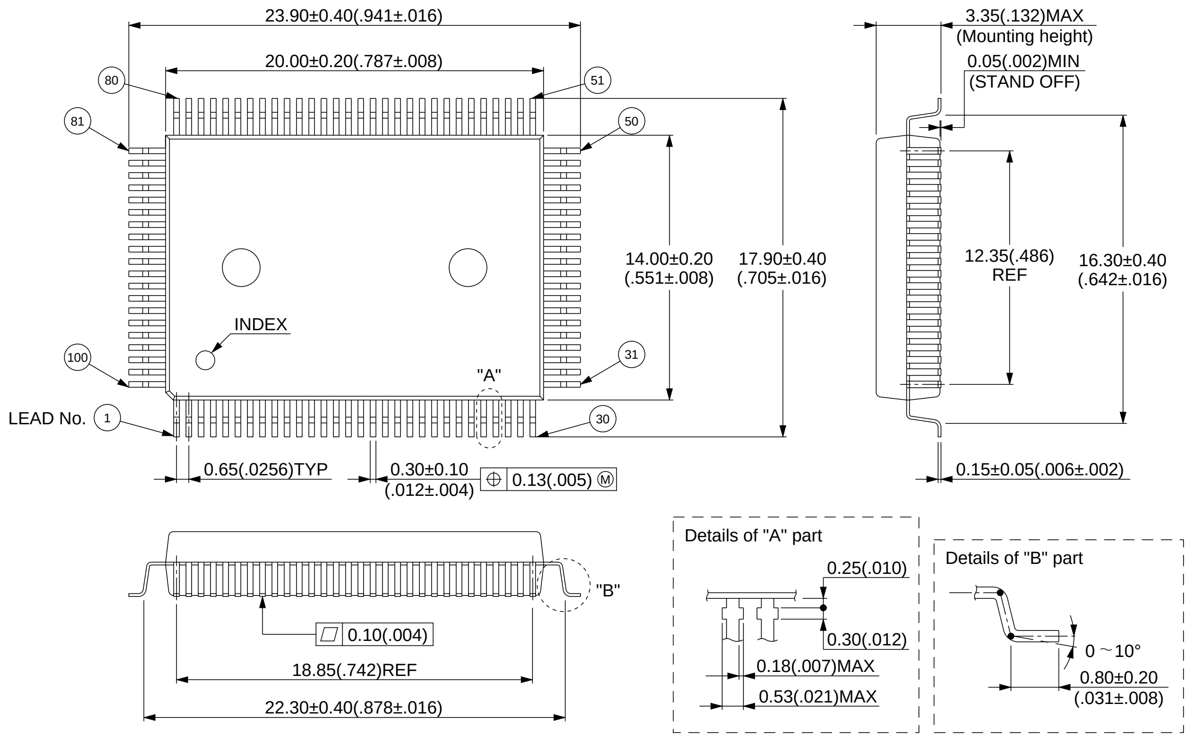
Note : For an explanation of “(a)” to “(d)”, refer to Table 4, “Number of Execution Cycles for Each Type of Addressing,” and Table 5, “Correction Values for Number of Cycles Used to Calculate Number of Actual Cycles.”

■ ORDERING INFORMATION

Part number	Package	Remarks
MB90598PF MB90F598PF	100-pin Plastic QFP (FPT-100P-M06)	
MB90V595CR	256-pin Ceramic PGA (PGA-256C-A01)	For evaluation

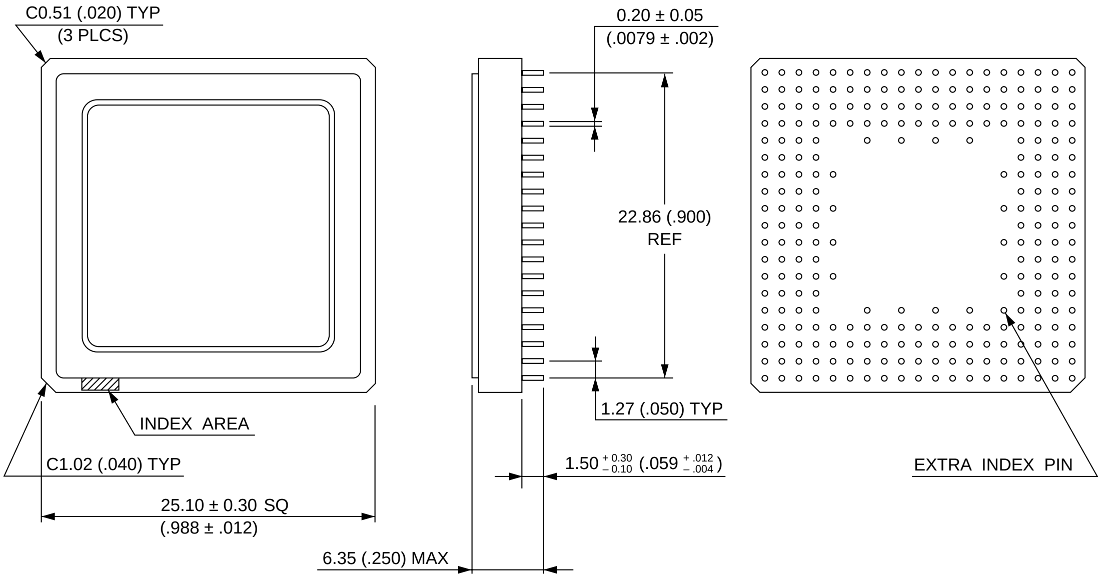
■ PACKAGE DIMENSION

100-pin plastic QFP
(FPT-100P-M06)



Dimensions in mm (inches)

250-pin ceramic PGA
(PGA-256-A01)



Dimensions in mm (inches)

MEMO

FUJITSU LIMITED

For further information please contact:

Japan
FUJITSU LIMITED
Corporate Global Business Support Division
Electronic Devices
KAWASAKI PLANT, 4-1-1, Kamikodanaka
Nakahara-ku, Kawasaki-shi
Kanagawa 211-8588, Japan
Tel: 81(44) 754-3763
Fax: 81(44) 754-3329

http://www.fujitsu.co.jp/

North and South America
FUJITSU MICROELECTRONICS, INC.
Semiconductor Division
3545 North First Street
San Jose, CA 95134-1804, USA
Tel: (408) 922-9000
Fax: (408) 922-9179

Customer Response Center
Mon. - Fri.: 7 am - 5 pm (PST)
Tel: (800) 866-8608
Fax: (408) 922-9179

http://www.fujitsumicro.com/

Europe
FUJITSU MICROELECTRONICS EUROPE GmbH
Am Siebenstein 6-10
D-63303 Dreieich-Buchschlag
Germany
Tel: (06103) 690-0
Fax: (06103) 690-122

http://www.fujitsu-ede.com/

Asia Pacific
FUJITSU MICROELECTRONICS ASIA PTE LTD
#05-08, 151 Lorong Chuan
New Tech Park
Singapore 556741
Tel: (65) 281-0770
Fax: (65) 281-0220

http://www.fmap.com.sg/

F9909
© FUJITSU LIMITED Printed in Japan

The contents of this document are subject to change without notice. Customers are advised to consult with FUJITSU sales representatives before ordering.

The information and circuit diagrams in this document are presented as examples of semiconductor device applications, and are not intended to be incorporated in devices for actual use. Also, FUJITSU is unable to assume responsibility for infringement of any patent rights or other rights of third parties arising from the use of this information or circuit diagrams.

FUJITSU semiconductor devices are intended for use in standard applications (computers, office automation and other office equipment, industrial, communications, and measurement equipment, personal or household devices, etc.).

CAUTION:
Customers considering the use of our products in special applications where failure or abnormal operation may directly affect human lives or cause physical injury or property damage, or where extremely high levels of reliability are demanded (such as aerospace systems, atomic energy controls, sea floor repeaters, vehicle operating controls, medical devices for life support, etc.) are requested to consult with FUJITSU sales representatives before such use. The company will not be responsible for damages arising from such use without prior approval.

Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions.

If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Law of Japan, the prior authorization by Japanese government will be required for export of those products from Japan.