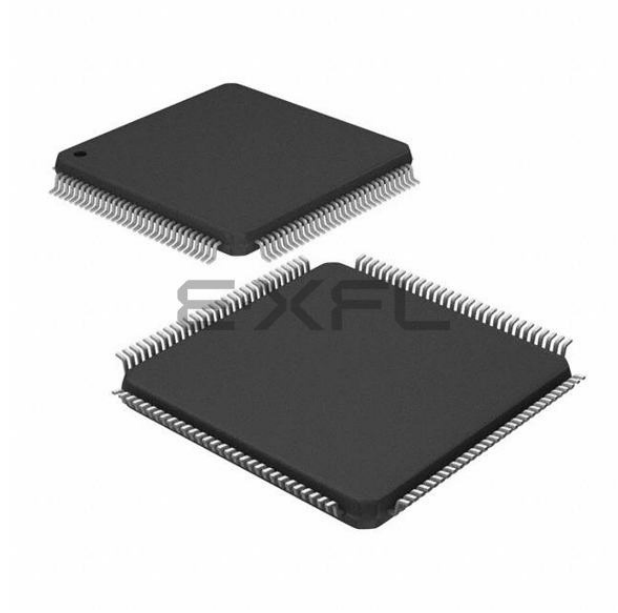


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

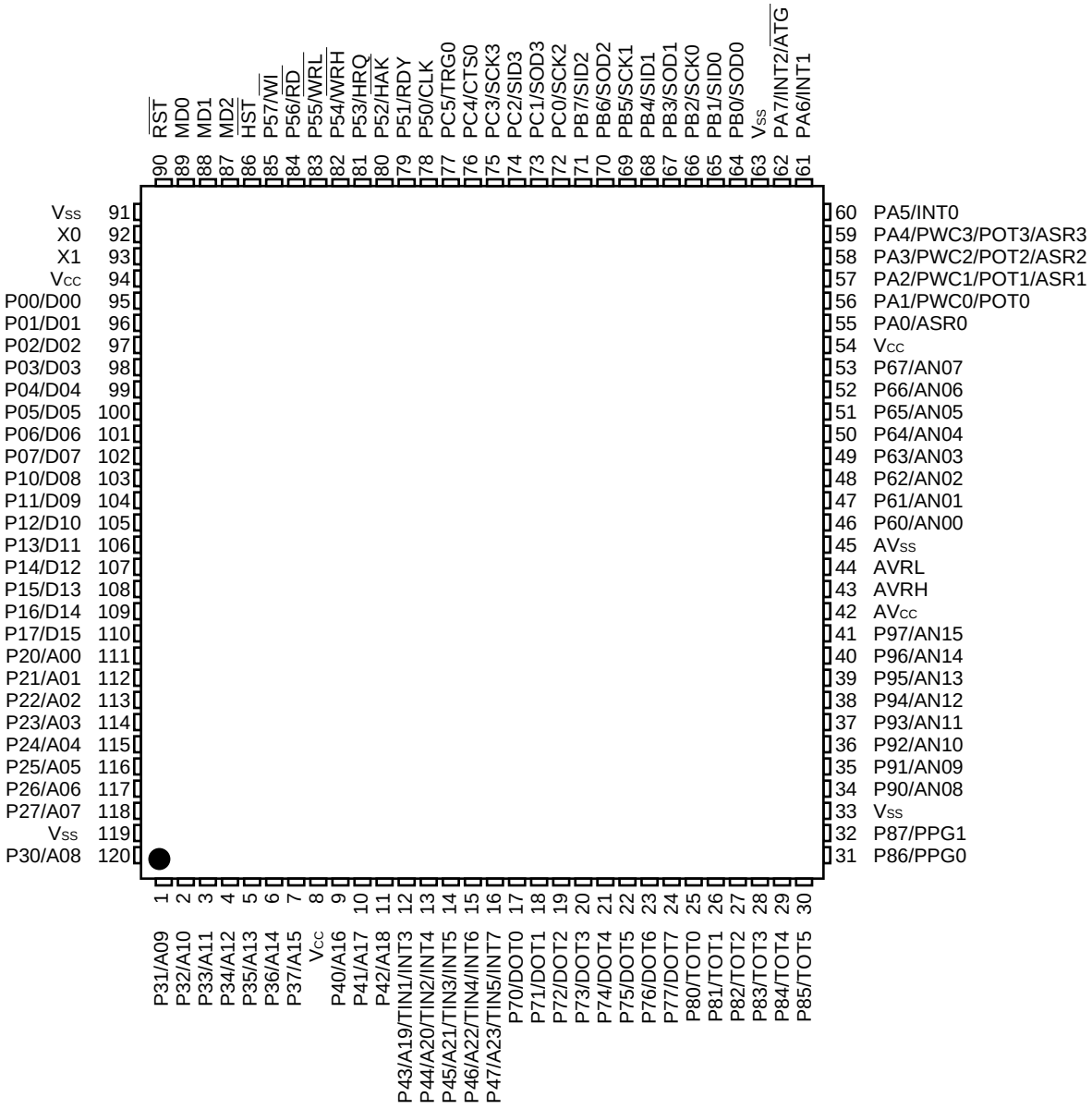
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

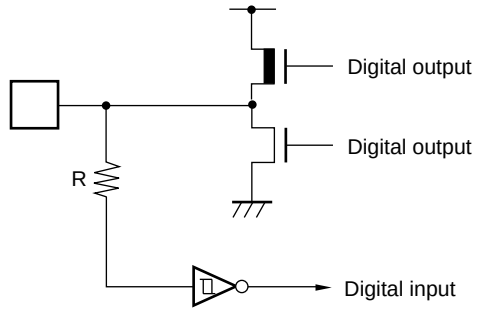
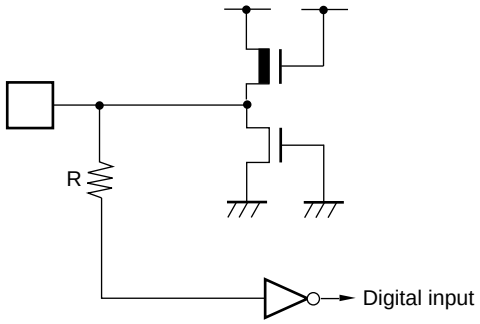
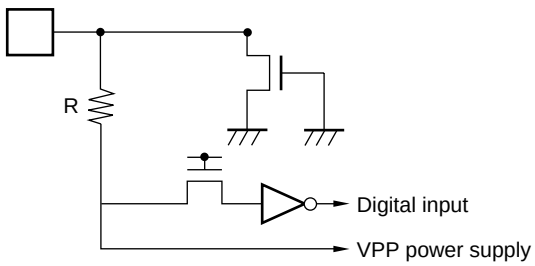
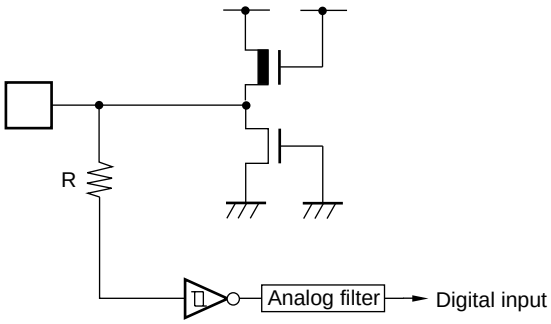
Details	
Product Status	Active
Core Processor	F²MC-16F
Core Size	16-Bit
Speed	16MHz
Connectivity	EBI/EMI, UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	102
Program Memory Size	64KB (64K x 8)
Program Memory Type	Mask ROM
EEPROM Size	-
RAM Size	3K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	120-BQFP
Supplier Device Package	120-QFP (28x28)
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/mb90223pf-gt-375

■ PIN ASSIGNMENT

(Top view)



(FPT-120P-M03)
(FPT-120C-C02)

Type	Circuit	Remarks
C		• CMOS-level output • CMOS-level hysteresis input with standby control
D		• CMOS-level input with no standby control Mask ROM products only: MD2: with pull-down resistor MD1: with pull-up resistor MD0: with pull-down resistor
		• CMOS-level input with no standby control MD2 of OTPROM products/EPROM products only
E		• CMOS-level hysteresis input with no standby control • With input analog filter (40 ns Typ.)

Note: The pull-up and pull-down resistors are always connected, regardless of the state.

(Continued)

8. Power-on Sequence for A/D Converter Power Supplies and Analog Inputs

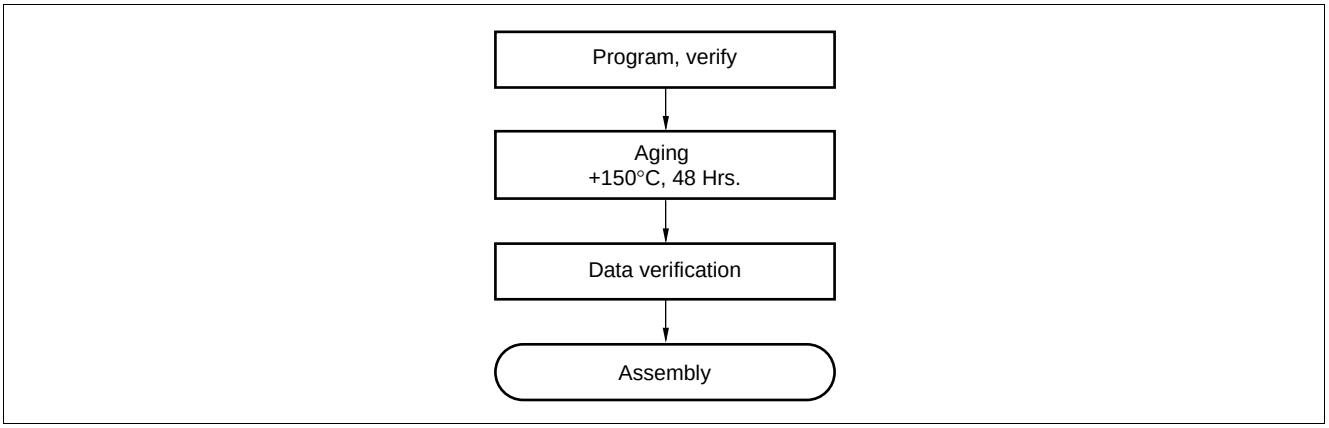
Be sure to turn on the digital power supply (V_{CC}) before applying voltage to the A/D converter power supplies (AV_{CC}, AVRH, and AVRL) and analog inputs (AN00 to AN15).

When turning power supplies off, turn off the A/D converter power supplies (AV_{CC}, AVRH, and AVRL) and analog inputs (AN00 to AN15) first, then the digital power supply (V_{CC}).

When turning AVRH on or off, be careful not to let it exceed AV_{CC}.

5. Recommended Screening Conditions

High temperature aging is recommended as the pre-assembly screening procedure.



6. Programming Yeild

MB90P224A/P224B cannot be write-tested for all bits due to their nature. Therefore the write yield cannot always be guaranteed to be 100%.

7. Pin Assignments in EPROM Mode

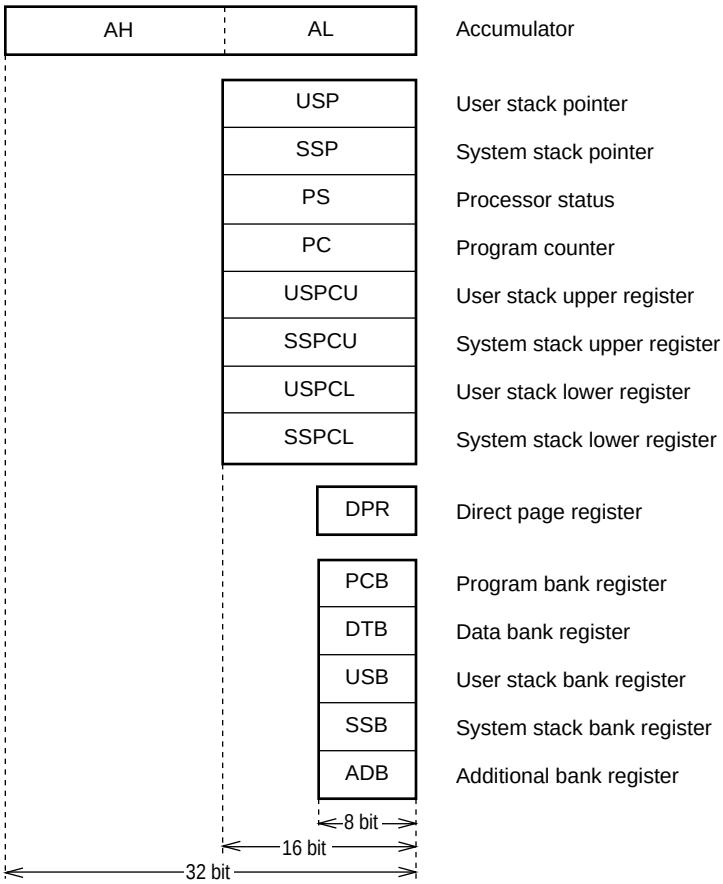
(1) Pins Compatible with MBM27C1000

MBM27C1000		MB90P224A/P224B/ MB90W224A/W224B	
Pin no.	Pin name	Pin no.	Pin name
1	V_{PP}	87	MD2 (V_{PP})
2	OE	83	P55
3	A15	7	P37
4	A12	4	P34
5	A07	118	P27
6	A06	117	P26
7	A05	116	P25
8	A04	115	P24
9	A03	114	P23
10	A02	113	P22
11	A01	112	P21
12	A00	111	P20
13	D00	95	P00
14	D01	96	P01
15	D02	97	P02
16	GND	33, 63, 91,119	V_{SS}

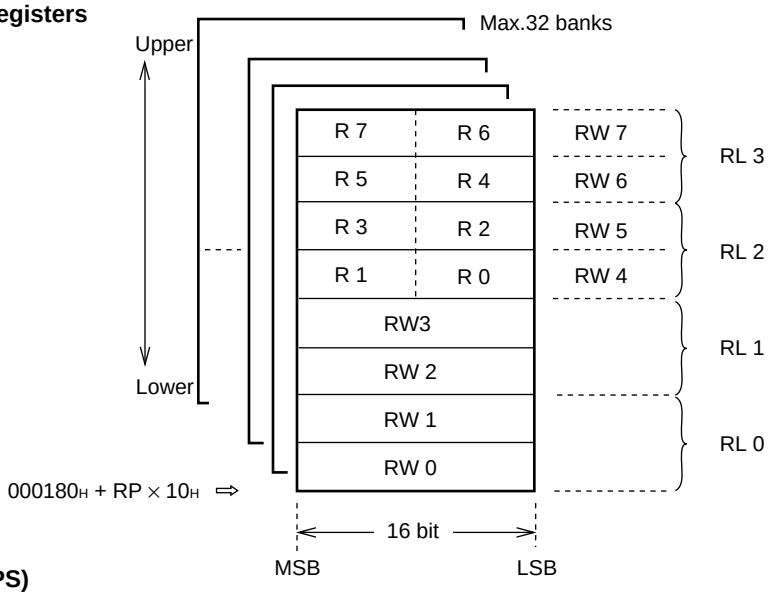
MBM27C1000		MB90P224A/P224B/ MB90W224A/W224B	
Pin no.	Pin name	Pin no.	Pin name
32	V_{CC}	8, 54, 94	V_{CC}
31	PGM	84	P56
30	N.C.	—	—
29	A14	6	P36
28	A13	5	P35
27	A08	120	P30
26	A09	1	P31
25	A11	3	P33
24	A16	9	P40
23	A10	2	P32
22	CE	82	P54
21	D07	102	P07
20	D06	101	P06
19	D05	100	P05
18	D04	99	P04
17	D03	98	P03

■ PROGRAMMING MODEL

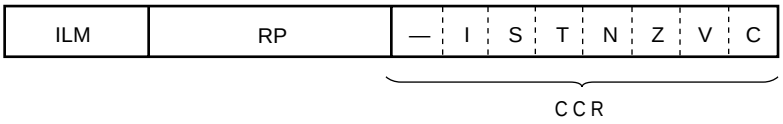
Dedicated Registers



General-purpose Registers



Processor Status (PS)



(Continued)

Address	Register	Register name	Access	Resouce name	Initial value
001F48H	PPG cycle setting register 0	PCSR0	W	16-bit PPG timer 0	XXXXXXXX
001F49H					XXXXXXXX
001F4AH	PPG duty setting register 0	PDUT0	W		XXXXXXXX
001F4BH					XXXXXXXX
001F4CH	PPG cycle setting register 1	PCSR1	W	16-bit PPG timer 1	XXXXXXXX
001F4DH					XXXXXXXX
001F4EH	PPG duty setting register 1	PDUT1	W		XXXXXXXX
001F4FH					XXXXXXXX
001F50H	ICU lower-order data register 0	ICRL0	R	Input capture 0	XXXXXXXX
001F51H					XXXXXXXX
001F52H	ICU higher-order data register 0	ICRH0	R		XXXXXXXX
001F53H					00000000
001F54H	ICU lower-order data register 1	ICRL1	R	Input capture 1	XXXXXXXX
001F55H					XXXXXXXX
001F56H	ICU higher-order data register 1	ICRH1	R		XXXXXXXX
001F57H					00000000
001F58H	ICU lower-order data register 2	ICRL2	R	Input capture 2	XXXXXXXX
001F59H					XXXXXXXX
001F5AH	ICU higher-order data register 2	ICRH2	R		XXXXXXXX
001F5BH					00000000
001F5CH	ICU lower-order data register 3	ICRL3	R	Input capture 3	XXXXXXXX
001F5DH					XXXXXXXX
001F5EH	ICU higher-order data register 3	ICRH3	R		XXXXXXXX
001F5FH					00000000
001F60H to 1FFFH	(Reserved area)*1				

Initial value
0: The initial value of this bit is “0”.
1: The initial value of this bit is “1”.
X: The initial value of this bit is undefined.
–: This bit is not used. The initial value is undefined.
*: The initial value of this bit varies with the reset source.
#: The initial value of this bit varies with the operation mode.
*1: Access prohibited
*2: Only this area is open to external access in the area below address 0000FFH (inclusive). All addresses which are not described in the table are reserved areas, and accesses to these areas are handled in the same manner as for internal areas. The access signal for the external bus is not generated.
*3: When an external bus is enable mode, never access to resisters which are not used as general ports in areas address 000000H to 000005H or 000010H to 000015H.

(Continued)

Interrupt source	EI²OS support	Interrupt vector			Interrupt control register	
		No.		Address	ICR	Address
UART0 (ch.0) reception completed	◎	#39	27H	FFFF60H	ICR14	0000BEH
Delay interrupt generation module	×	#42	2AH	FFFF54H	ICR15	0000BFH
Stack fault	×	#255	FFH	FFFC00H	—	—

◎: EI²OS is supported (with stop request).
□: EI²OS is supported (without stop request).
○: EI²OS is supported; however, since two interrupt sources are allocated to a single ICR, in case EI²OS is used for one of the two, EI²OS and ordinary interrupt are not both available for the other (with stop request).
△: EI²OS is supported; however, since two interrupt sources are allocated to a single ICR, in case EI²OS is used for one of the two, EI²OS and ordinary interrupt are not both available for the other (without stop request).
×: EI²OS is not supported.

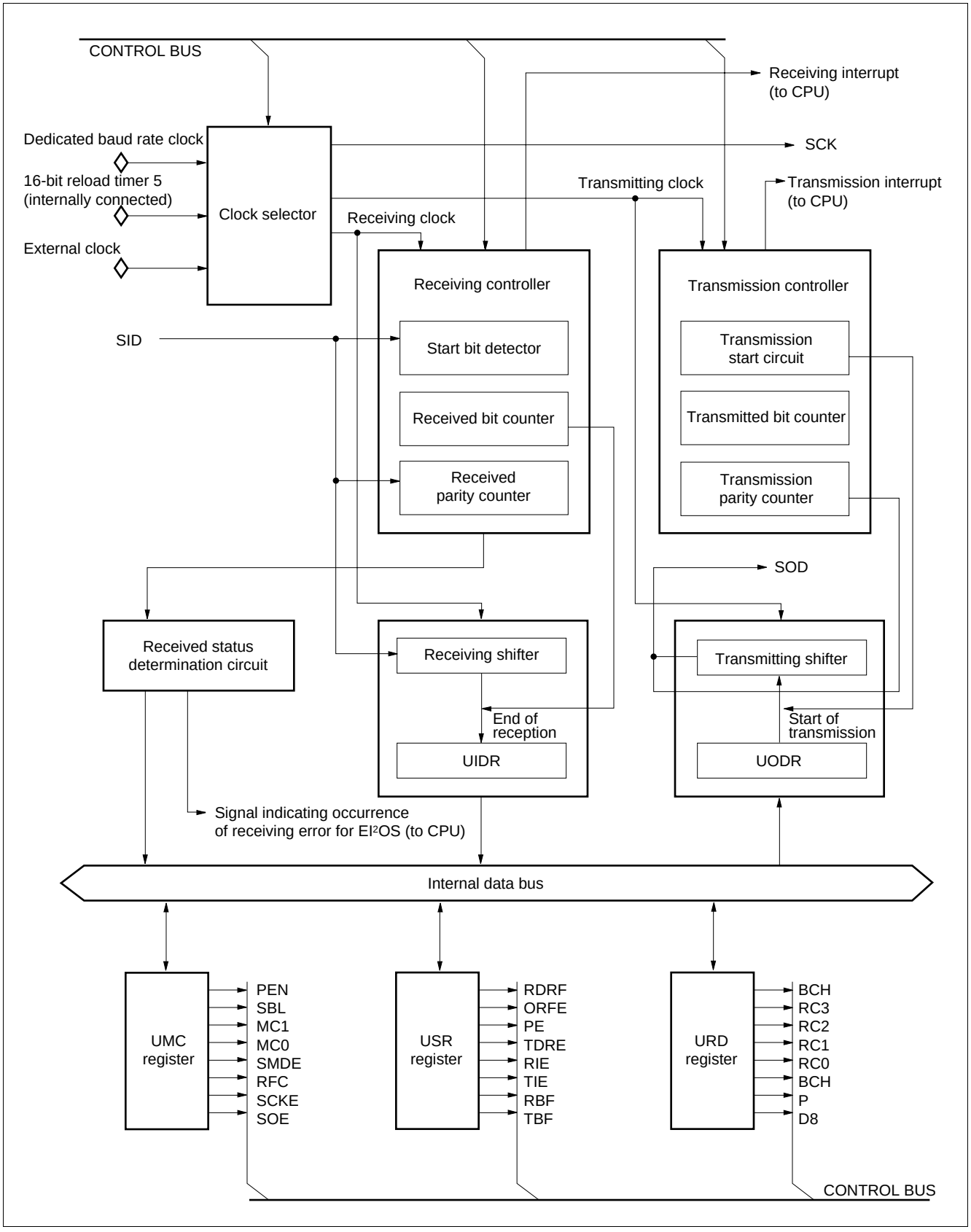
Note: Since the interrupt sources having interrupt vector Nos. 15 to 18, 20, and 25 to 28 are OR'ed, respectively, select them by means of the interrupt enable bits of each resource.

If EI²OS is used with the above-mentioned interrupt sources OR'ed with the interrupt vector Nos. 15 to 18, 20, and 25 to 28, be sure to activate one of the interrupt sources.

Also in this case, a request flag in the same series as the one interrupt source is likely to be cleared automatically by EI²OS.

Assume for example that an interrupt for compare 4 of the interrupt vector No. 25 is activated at this time by ICR07, so that the compare 6 is disabled. If EI²OS is activated at this time by ICR07, so that the compare 6 interrupt takes place during generation of or simultaneously with the compare 4 interrupt, not only the interrupt flag for the compare 4 but also that for the compare 6 will be automatically cleared after EI²OS is automatically transferred due to the compare 4 interrupt.

(2) Block Diagram



5. 10-bit A/D Converter

The 10-bit A/D converter converts analog input voltage into a digital value. The features of this module are described below:

- Conversion time: 6.125 µs/channel (min.) (with machine clock running at 16 MHz)
- Uses RC-type sequential comparison and conversion method with built-in sample and hold circuit
- 10-bit resolution
- Analog input can be selected by software from among 16 channels
  Single-conversion mode: Selects and converts one channel.
  Scan conversion mode: Converts several consecutive channels (up to 16 can be programmed).
  One-shot mode: Converts the specified channel once and terminates.
  Continuous conversion mode: Repeatedly converts the specified channel.
  Stop conversion mode: Pauses after converting one channel and waits until the next startup (permits synchronization of start of conversion).
- When A/D conversion is completed, an "A/D conversion complete" interrupt request can be issued to the CPU. Because the generation of this interrupt can be used to start up the EI^{2}OS and transfer the A/D conversion results to memory, this function is suitable for continuous processing.
- Startup triggers can be selected from among software, an external trigger (falling edge), and a timer (rising edge).

(1) Register Configuration

- **A/D Channel Setting Register (ADCH)**

This register specfies the A/D converter conversion channel.

Register name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ADCH	000032$_H$	ANS3	ANS2	ANS1	ANS0	ANE3	ANE2	ANE1	ANE0	00000000$_B$
		(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	

- **A/D Mode Register (ADMD)**

This register specfies the A/D converter operation mode and the startup source.

Register name	Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ADMD	000033$_H$	—	—	—	Reserved	MOD1	MOD0	STS1	STS0	- - - X0000$_B$
		(—)	(—)	(—)	(W)	(R/W)	(R/W)	(R/W)	(R/W)	

Note: Program "0" to bit 12 when write. Read value is indeterminated.

- **A/D Control Status Register (ADCS)**

This register is the A/D converter control and status register.

Register name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ADCS	000034$_H$	BUSY	INT	INTE	PAUS	—	—	STRT	Reserved	0000 - - 00$_B$
		(R/W)	(R/W)	(R/W)	(R/W)	(—)	(—)	(W)	(R/W)	

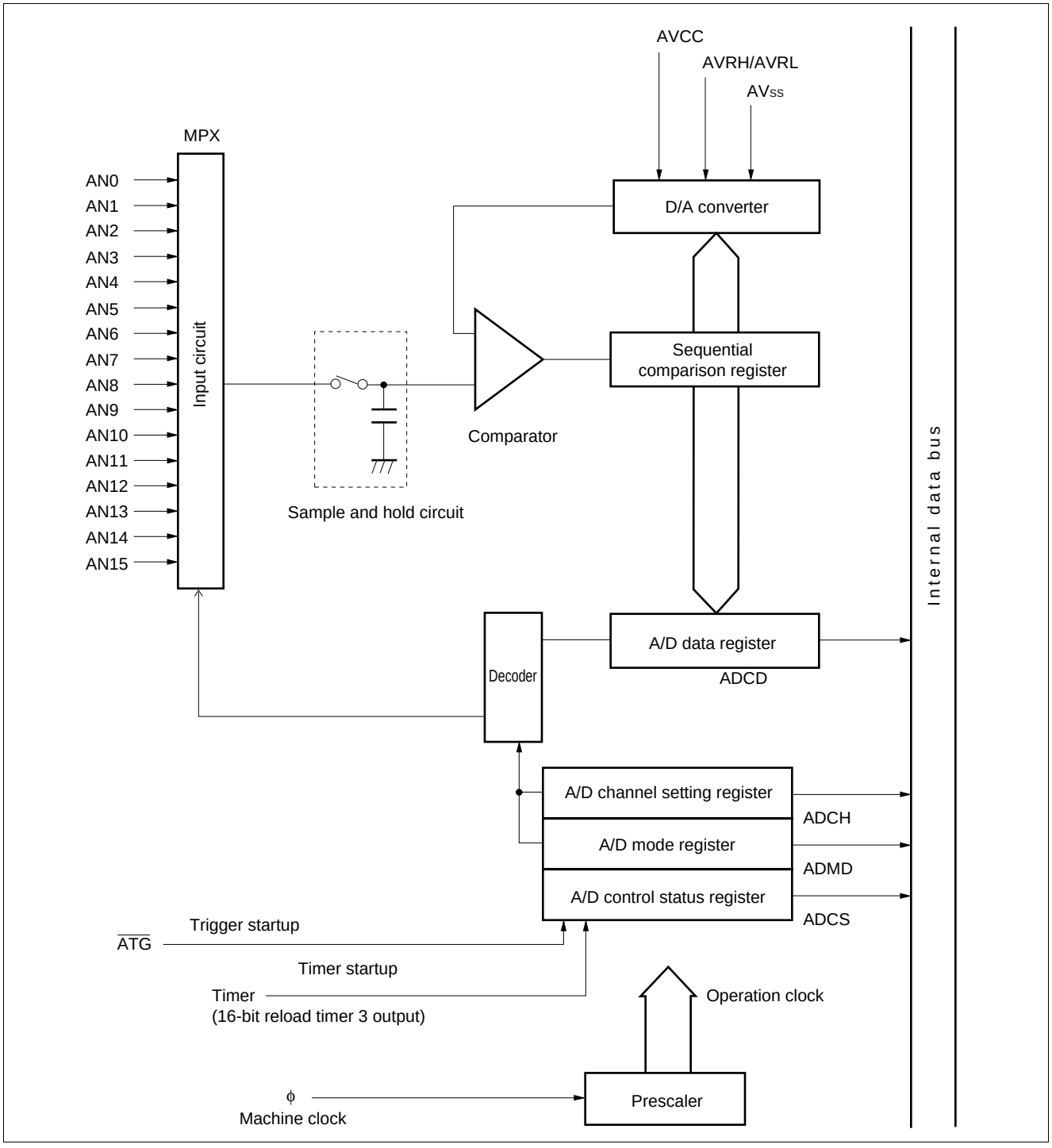
- **A/D Data Register (ADCD)**

This register stores the A/D converter conversion data.

Register name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
ADCD	000036$_H$	D7	D6	D5	D4	D3	D2	D1	D0	XXXXXXXX$_B$
		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	

Register name	Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
ADCD	000037H	—	—	—	—	—	—	D9	D8	000000XXB
		(R)	(R)	(R)	(R)	(R)	(R)	(R)	(R)	

(2) Block Diagram



6. PWC (Pulse Width Count) Timer

The PWC (pulse width count) timer is a 16-bit multifunction up-count timer with an input-signal pulse-width count function and a reload timer function. The hardware configuration of this module is a 16-bit up-count timer, an input pulse divider with divide ratio control register, four count input pins, and a 16-bit control register. Using these components, the PWC timer provides the following features:

- Timer functions: An interrupt request can be generated at set time intervals.
  Pulse signals synchronized with the timer cycle can be output.
  The reference internal clock can be selected from among three internal clocks.
- Pulse-width count functions: The time between arbitrary pulse input events can be counted.
  The reference internal clock can be selected from among three internal clocks.
  Various count modes:
  "H" pulse width (↑ to ↓)/"L" pulse width (↓ to ↑)
  Rising-edge cycle (↑ to ↑/Falling-edge cycle (↓ to ↓)
  Count between edges (↑ or ↓ to ↓ or ↑)
  Cycle count can be performed by 2^{2n} division (n = 1, 2, 3, 4) of the input pulse, with an 8 bit input divider.
  An interrupt request can be generated once counting has been performed.
  The number of times counting is to be performed (once or subsequently) can be selected.

The MB90220 series has four channels for this module.

(1) Register Configuration

- **PWC Control Status Register 0 to 3 (PWCSR0 to PWCSR3)**

Register name	Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
PWCSR0	000051H	STRT	STOP	EDIR	EDIE	OVIR	OVIE	ERR	POUT	00000000B
PWCSR1	000053H									
PWCSR2	000055H									
PWCSR3	000057H	(R/W)	(R/W)	(R)	(R/W)	(R/W)	(R/W)	(R)	(R/W)	

Register name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
PWCSR0	000050H	CKS1	CKS0	PIS1	PIS0	S/C	MOD1	MOD1	MOD0	00000000B
PWCSR1	000052H									
PWCSR2	000054H									
PWCSR3	000056H	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	

- **PWC Data Buffer Register 0 to 3 (PWCR0 to PWCR3)**

Register name	Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
PWCR0	001F01H									00000000B
PWCR1	001F03H									
PWCR2	001F05H									
PWCR3	001F07H	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	

Register name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
PWCR0	001F00H									00000000B
PWCR1	001F02H									
PWCR2	001F04H									
PWCR3	001F06H	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	(R/W)	

12. Watchdog Timer and Timebase Timer Functions

The watchdog timer consists of a 2-bit watchdog counter using carry from an 18-bit timebase timer as the clock source, a control register, and a watchdog reset control section. The timebase timer consists of an 18-bit timer and an interval interrupt control circuit.

(1) Register Configuration

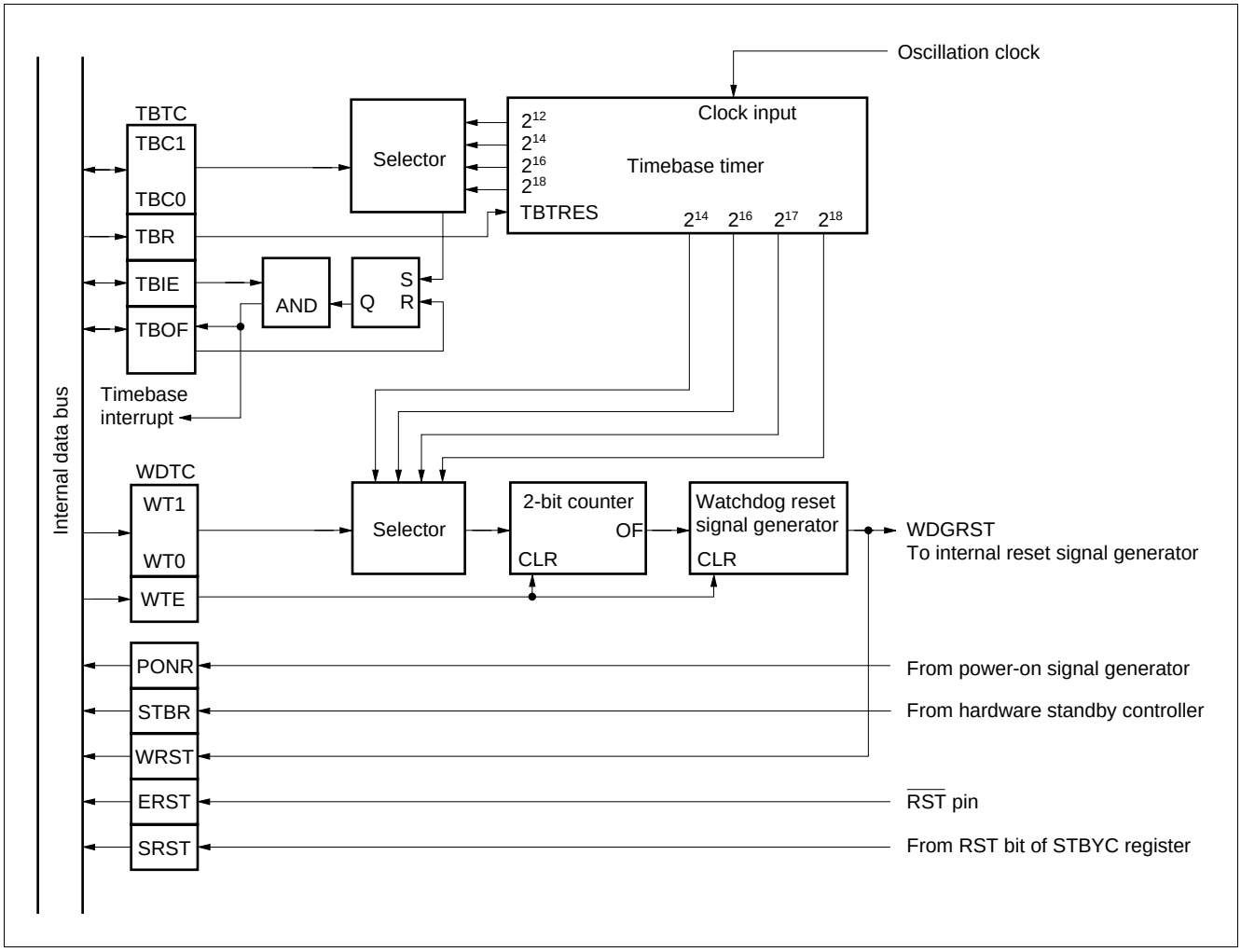
- **Watchdog Timer Control Register (WDTC)**

Register name	Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
WDTC	0000A8H	PONR	STBR	WRST	ERST	SRST	WTE	WT1	WT0	XXXXXXXX
		(R)	(R)	(R)	(R)	(R)	(W)	(W)	(W)	

- **Timebase Timer Control Register (TBTC)**

Register name	Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
TBTC	0000A9H	—	—	—	TBIE	TBOF	TBR	TBC1	TBC0	- - - XXXXX
		(—)	(—)	(—)	(R/W)	(R/W)	(R)	(R/W)	(R/W)	

(2) Block Diagram



13. Delay Interruupt Generation Module

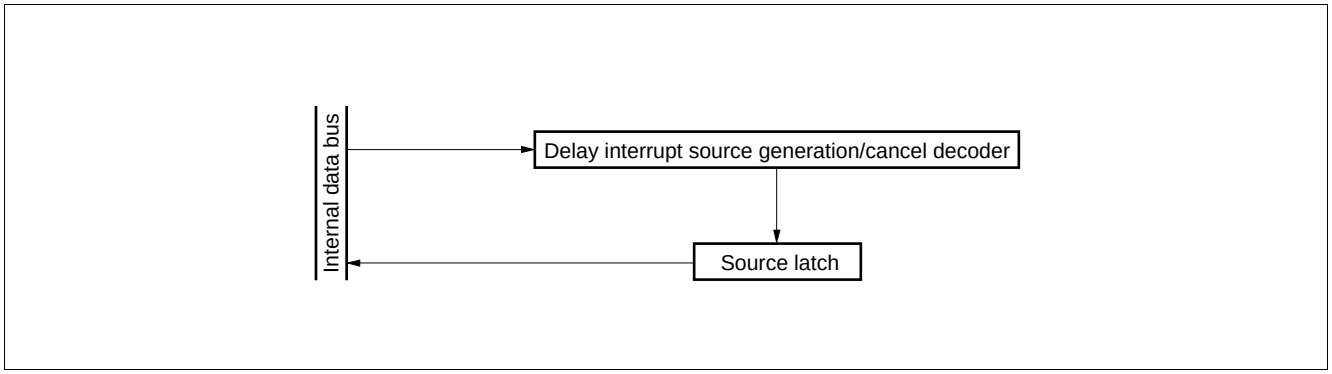
The delayed interrupt generation module is used to generate an interrupt task switching. Using this module allows an interrupt request to the F²MC-16F CPU to generated or cancel by software.

(1) Register Configuration

- **Delay Interrupt Source Generation/Cancel Register (DIRR)**

Register name	Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	
DIRR	00009F H	—	—	—	—	—	—	—	R0	Initial value 0
		(—)	(—)	(—)	(—)	(—)	(—)	(—)	(R/W)	

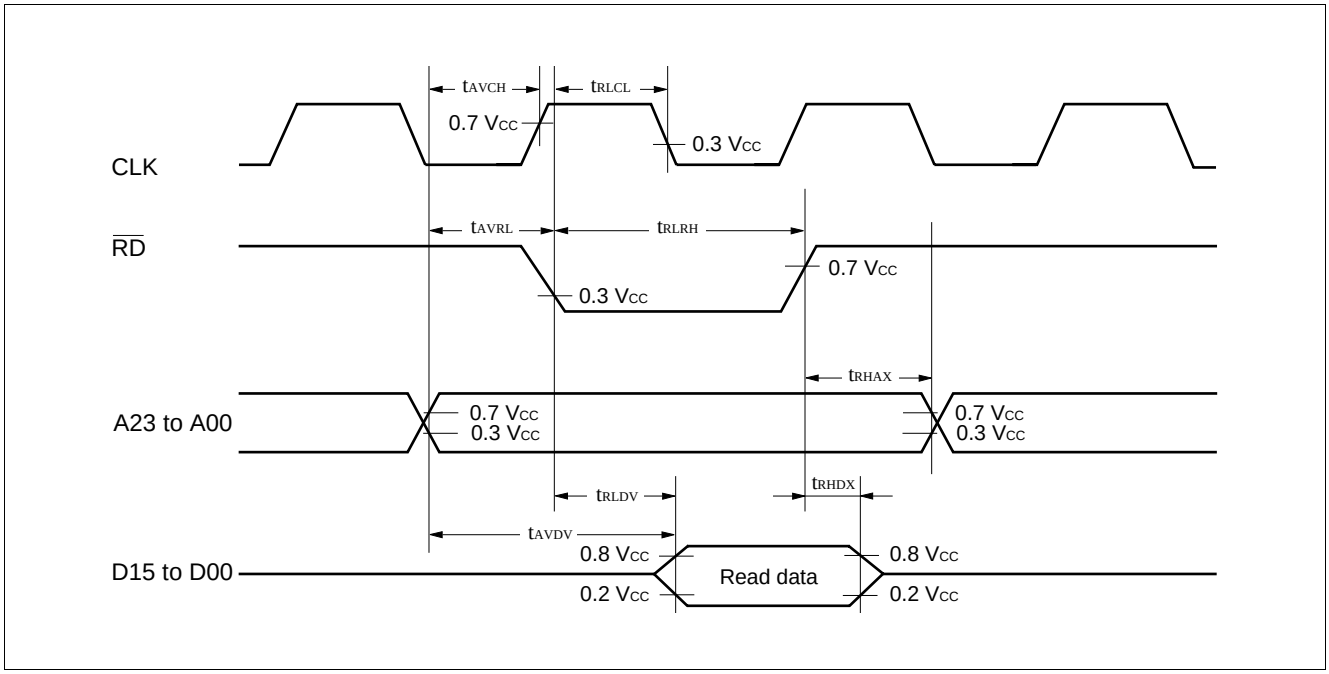
(2) Block Diagram



(5) Bus Read Timing

(V_{CC} = +4.5 V to +5.5 V, V_{SS} = 0.0 V, T_A = –40°C to +70°C)

Parameter	Symbol	Pin name	Condition	Value		Unit	Remarks
				Min.	Max.		
Valid address → $\overline{RD}$ ↓ time	t_{AVRL}	A23 to A00	Load condition: 80 pF	$t_{CYC}/2 - 20$	—	ns	
$\overline{RD}$ pulse width	t_{RLRH}	$\overline{RD}$		$t_{CYC} - 25$	—	ns	
$\overline{RD}$ ↓ → Valid data input	t_{RLDV}	D15 to D00		—	$t_{CYC} - 30$	ns	
$\overline{RD}$ ↑ → Data hold time	t_{RHDX}			0	—	ns	
Valid address → Valid data input	t_{AVDV}			—	$3t_{CYC}/2 - 40$	ns	
$\overline{RD}$ ↑ → Address valid time	t_{RHAX}	A23 to A00		$t_{CYC}/2 - 20$	—	ns	
Valid address → CLK ↑ time	t_{AVCH}	A23 to A00 CLK		$t_{CYC}/2 - 25$	—	ns	
$\overline{RD}$ ↓ → CLK ↓ time	t_{RLCL}	$\overline{RD}$, CLK		$t_{CYC}/2 - 25$	—	ns	



(Continued)

Symbol	Explanation
#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 value
vct4 vct8	Vector number (0 to 15) Vector number (0 to 255)
( )b	Bit address
rel ear eam	Branch specification relative to PC Effective addressing (codes 00 to 07) Effective addressing (codes 08 to 1F)
rlst	Register list

Table 6 Transfer Instructions (Byte) [50 Instructions]

Mnemonic	#	cycles	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOV A, dir	2	2	(b)	byte (A) ← (dir)	Z	*	–	–	–	*	*	–	–	–
MOV A, addr16	3	2	(b)	byte (A) ← (addr16)	Z	*	–	–	–	*	*	–	–	–
MOV A, Ri	1	1	0	byte (A) ← (Ri)	Z	*	–	–	–	*	*	–	–	–
MOV A, ear	2	1	0	byte (A) ← (ear)	Z	*	–	–	–	*	*	–	–	–
MOV A, eam	2+	2+ (a)	(b)	byte (A) ← (eam)	Z	*	–	–	–	*	*	–	–	–
MOV A, io	2	2	(b)	byte (A) ← (io)	Z	*	–	–	–	*	*	–	–	–
MOV A, #imm8	2	2	0	byte (A) ← imm8	Z	*	–	–	–	*	*	–	–	–
MOV A, @A	2	2	(b)	byte (A) ← ((A))	Z	–	–	–	–	*	*	–	–	–
MOV A, @RLi+disp8	3	6	(b)	byte (A) ← ((RLi))+disp8)	Z	*	–	–	–	*	*	–	–	–
MOV A, @SP+disp8	3	3	(b)	byte (A) ← ((SP)+disp8)	Z	*	–	–	–	*	*	–	–	–
MOVP A, addr24	5	3	(b)	byte (A) ←(addr24)	Z	*	–	–	–	*	*	–	–	–
MOVP A, @A	2	2	(b)	byte (A) ← ((A))	Z	–	–	–	–	*	*	–	–	–
MOVN A, #imm4	1	1	0	byte (A) ← imm4	Z	*	–	–	–	R	*	–	–	–
MOVX A, dir	2	2	(b)	byte (A) ← (dir)	X	*	–	–	–	*	*	–	–	–
MOVX A, addr16	3	2	(b)	byte (A) ← (addr16)	X	*	–	–	–	*	*	–	–	–
MOVX A, Ri	2	1	0	byte (A) ← (Ri)	X	*	–	–	–	*	*	–	–	–
MOVX A, ear	2	1	0	byte (A) ← (ear)	X	*	–	–	–	*	*	–	–	–
MOVX A, eam	2+	2+ (a)	(b)	byte (A) ← (eam)	X	*	–	–	–	*	*	–	–	–
MOVX A, io	2	2	(b)	byte (A) ← (io)	X	*	–	–	–	*	*	–	–	–
MOVX A, #imm8	2	2	0	byte (A) ← imm8	X	*	–	–	–	*	*	–	–	–
MOVX A, @A	2	2	(b)	byte (A) ← ((A))	X	–	–	–	–	*	*	–	–	–
MOVX A,@RWi+disp8	2	3	(b)	byte (A) ← ((RWi))+disp8)	X	*	–	–	–	*	*	–	–	–
MOVX A, @RLi+disp8	3	6	(b)	byte (A) ← ((RLi))+disp8)	X	*	–	–	–	*	*	–	–	–
MOVX A, @SP+disp8	3	3	(b)	byte (A) ← ((SP)+disp8)	X	*	–	–	–	*	*	–	–	–
MOVPX A, addr24	5	3	(b)	byte (A) ←(addr24)	X	*	–	–	–	*	*	–	–	–
MOVPX A, @A	2	2	(b)	byte (A) ← ((A))	X	–	–	–	–	*	*	–	–	–
MOV dir, A	2	2	(b)	byte (dir) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV addr16, A	3	2	(b)	byte (addr16) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV Ri, A	1	1	0	byte (Ri) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV ear, A	2	2	0	byte (ear) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV eam, A	2+	2+ (a)	(b)	byte (eam) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV io, A	2	2	(b)	byte (io) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV @RLi+disp8, A	3	6	(b)	byte ((RLi)) +disp8) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV @SP+disp8, A	3	3	(b)	byte ((SP)+disp8) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVP addr24, A	5	3	(b)	byte (addr24) ← (A)	–	–	–	–	–	*	*	–	–	–
MOV Ri, ear	2	2	0	byte (Ri) ← (ear)	–	–	–	–	–	*	*	–	–	–
MOV Ri, eam	2+	3+ (a)	(b)	byte (Ri) ← (eam)	–	–	–	–	–	*	*	–	–	–
MOVP @A, Ri	2	3	(b)	byte ((A)) ← (Ri)	–	–	–	–	–	*	*	–	–	–
MOV ear, Ri	2	3	0	byte (ear) ← (Ri)	–	–	–	–	–	*	*	–	–	–
MOV eam, Ri	2+	3+ (a)	(b)	byte (eam) ← (Ri)	–	–	–	–	–	*	*	–	–	–
MOV Ri, #imm8	2	2	0	byte (Ri) ← imm8	–	–	–	–	–	*	*	–	–	–
MOV io, #imm8	3	3	(b)	byte (io) ← imm8	–	–	–	–	–	–	–	–	–	–
MOV dir, #imm8	3	3	(b)	byte (dir) ← imm8	–	–	–	–	–	–	–	–	–	–
MOV ear, #imm8	3	2	0	byte (ear) ← imm8	–	–	–	–	–	*	*	–	–	–
MOV eam, #imm8	3+	2+ (a)	(b)	byte (eam) ← imm8	–	–	–	–	–	–	–	–	–	–
MOV @AL, AH	2	2	(b)	byte ((A)) ← (AH)	–	–	–	–	–	*	*	–	–	–

(Continued)

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

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

For an explanation of “(b)” and “(c), refer to Table 5, “Correction Values for Number of Cycle Used to Calculate Number of Actual Cycles.”

*1: 3 when dividing into zero, 6 when an overflow occurs, and 14 normally.
*2: 3 when dividing into zero, 5 when an overflow occurs, and 13 normally.
*3: 5 + (a) when dividing into zero, 7 + (a) when an overflow occurs, and 17 + (a) normally.
*4: 3 when dividing into zero, 5 when an overflow occurs, and 21 normally.
*5: 4 + (a) when dividing into zero, 7 + (a) when an overflow occurs, and 25 + (a) normally.
*6: (b) when dividing into zero or when an overflow occurs, and 2 × (b) normally.
*7: (c) when dividing into zero or when an overflow occurs, and 2 × (c) normally.
*8: 3 when byte (AH) is zero, and 7 when byte (AH) is not 0.
*9: 3 when byte (ear) is zero, and 7 when byte (ear) is not 0.
*10: 4 + (a) when byte (eam) is zero, and 8 + (a) when byte (eam) is not 0.
*11: 3 when word (AH) is zero, and 11 when word (AH) is not 0.
*12: 3 when word (ear) is zero, and 11 when word (ear) is not 0.
*13: 4 + (a) when word (eam) is zero, and 12 + (a) when word (eam) is not 0.

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

Mnemonic	#	cycles	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ANDL A, ear	2	5	0	long (A) ← (A) and (ear)	–	–	–	–	–	*	*	R	–	–
ANDL A, eam	2+	6+ (a)	(d)	long (A) ← (A) and (eam)	–	–	–	–	–	*	*	R	–	–
ORL A, ear	2	5	0	long (A) ← (A) or (ear)	–	–	–	–	–	*	*	R	–	–
ORL A, eam	2+	6+ (a)	(d)	long (A) ← (A) or (eam)	–	–	–	–	–	*	*	R	–	–
XORL A, ear	2	5	0	long (A) ← (A) xor (ear)	–	–	–	–	–	*	*	R	–	–
XORL A, eam	2+	6+ (a)	(d)	long (A) ← (A) xor (eam)	–	–	–	–	–	*	*	R	–	–

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

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

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

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

Table 17 Absolute Value Instructions (Byte/Word/Long Word) [3 Insturctions]

Mnemonic	#	cycles	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
ABS A	2	2	0	byte (A) ← absolute value (A)	Z	–	–	–	–	*	*	*	–	–
ABSW A	2	2	0	word (A) ← absolute value (A)	–	–	–	–	–	*	*	*	–	–
ABSL A	2	4	0	long (A) ← absolute value (A)	–	–	–	–	–	*	*	*	–	–

Table 18 Normalize Instructions (Long Word) [1 Instruction]

Mnemonic	#	cycles	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
NRML A, R0	2	*	0	long (A) ← Shifts to the position at which “1” was set first byte (R0) ← current shift count	–	–	–	–	*	–	–	–	–	–

* : 5 when the contents of the accumulator are all zeroes, 5 + (R0) in all other cases.

Table 19 Shift Instructions (Byte/Word/Long Word) [27 Instructions]

Mnemonic	#	cycles	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
RORC A	2	2	0	byte (A) ← Right rotation with carry	–	–	–	–	–	*	*	–	*	–
ROLC A	2	2	0	byte (A) ← Left rotation with carry	–	–	–	–	–	*	*	–	*	–
RORC ear	2	2	0	byte (ear) ← Right rotation with carry	–	–	–	–	–	*	*	–	*	*
RORC eam	2+	3+ (a)	2× (b)	byte (eam) ← Right rotation with carry	–	–	–	–	–	*	*	–	*	*
ROLC ear	2	2	0	byte (ear) ← Left rotation with carry	–	–	–	–	–	*	*	–	*	*
ROLC eam	2+	3+ (a)	2× (b)	byte (eam) ← Left rotation with carry	–	–	–	–	–	*	*	–	*	*
ASR A, R0	2	*1	0	byte (A) ← Arithmetic right barrel shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSR A, R0	2	*1	0	byte (A) ← Logical right barrel shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSL A, R0	2	*1	0	byte (A) ← Logical left barrel shift (A, R0)	–	–	–	–	–	*	*	–	*	–
ASR A, #imm8	3	*3	0	byte (A) ← Arithmetic right barrel shift (A, imm8)	–	–	–	–	*	*	*	–	*	–
LSR A, #imm8	3	*3	0	byte (A) ← Logical right barrel shift (A, imm8)	–	–	–	–	*	*	*	–	*	–
LSL A, #imm8	3	*3	0	byte (A) ← Logical left barrel shift (A, imm8)	–	–	–	–	–	*	*	–	*	–
ASRW A	1	2	0	word (A) ← Arithmetic right shift (A, 1 bit)	–	–	–	–	*	*	*	–	*	–
LSRW A/SHRW A	1	2	0	word (A) ← Logical right shift (A, 1 bit)	–	–	–	–	*	R	*	–	*	–
LSLW A/SHLW A	1	2	0	word (A) ← Logical left shift (A, 1 bit)	–	–	–	–	–	*	*	–	*	–
ASRW A, R0	2	*1	0	word (A) ← Arithmetic right barrel shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSRW A, R0	2	*1	0	word (A) ← Logical right barrel shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSLW A, R0	2	*1	0	word (A) ← Logical left barrel shift (A, R0)	–	–	–	–	–	*	*	–	*	–
ASRW A, #imm8	3	*3	0	word (A) ← Arithmetic right barrel shift (A, imm8)	–	–	–	–	*	*	*	–	*	–
LSRW A, #imm8	3	*3	0	word (A) ← Logical right barrel shift (A, imm8)	–	–	–	–	*	*	*	–	*	–
LSLW A, #imm8	3	*3	0	word (A) ← Logical left barrel shift (A, imm8)	–	–	–	–	–	*	*	–	*	–
ASRL A, R0	2	*2	0	long (A) ← Arithmetic right shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSRL A, R0	2	*2	0	long (A) ← Logical right barrel shift (A, R0)	–	–	–	–	*	*	*	–	*	–
LSLL A, R0	2	*2	0	long (A) ← Logical left barrel shift (A, R0)	–	–	–	–	–	*	*	–	*	–
ASRL A, #imm8	3	*4	0	long (A) ← Arithmetic right shift (A, imm8)	–	–	–	–	*	*	*	–	*	–
LSRL A, #imm8	3	*4	0	long (A) ← Logical right barrel shift (A, imm8)	–	–	–	–	*	*	*	–	*	–
LSLL A, #imm8	3	*4	0	long (A) ← Logical left barrel shift (A, imm8)	–	–	–	–	–	*	*	–	*	–

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

*1: 3 when R0 is 0, 3 + (R0) in all other cases.
*2: 3 when R0 is 0, 4 + (R0) in all other cases.
*3: 3 when imm8 is 0, 3 + (imm8) in all other cases.
*4: 3 when imm8 is 0, 4 + (imm8) in all other cases.

■ ORDERING INFORMATION

Part number	Type	Package	Remarks
MB90224 MB90223 MB90P224A MB90P224B	MB90224PF MB90223PF MB90P224PF MB90P224BPF	120-pin Plastic QFP (FPT-120P-M03)	
MB90W224A MB90W224B	MB90W224ZF MB90W224BZF	120-pin Ceramic QFP (FPT-120C-C02)	ES level only
MB90V220	MB90V220CR	256-pin Ceramic PGA (PGA-256C-A02)	For evaluation