Infineon Technologies - MB90224PF-GT-370E1 Datasheet

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"

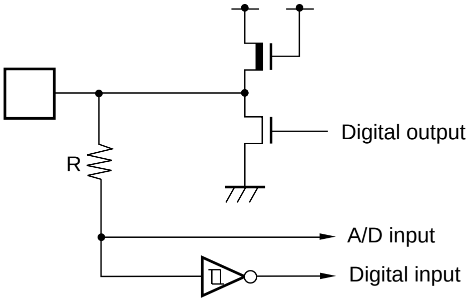
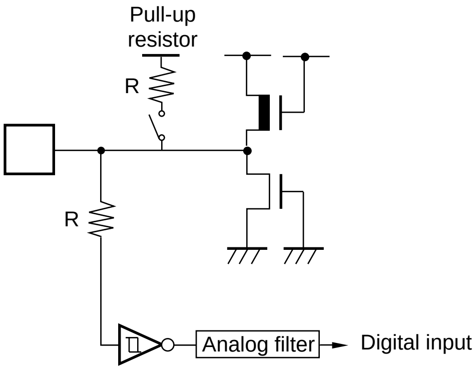
Details	
Product Status	Obsolete
Core Processor	F²MC-16F
Core Size	16-Bit
Speed	16MHz
Connectivity	EBI/EMI, UART/USART
Peripherals	POR, PWM, WDT
Number of I/O	102
Program Memory Size	96KB (96K x 8)
Program Memory Type	Mask ROM
EEPROM Size	-
RAM Size	4.5K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data 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/mb90224pf-gt-370e1

Pin no.	Pin name	Circuit type	Function
QFP*			
12 to 16	INT3 to INT7	C	External interrupt request input pins When external interrupts are enabled, these inputs may be used suddenly at any time; therefore, it is necessary to stop output by other functions on these pins, except when using them for output deliberately.
78	P50	C	General-purpose I/O port This function is valid in single-chip mode and when the CLK output specification is disabled.
	CLK		CLK output pin This function is valid in modes where the external bus is enabled and the CLK output specification is enabled.
79	P51	C	General-purpose I/O port This function is valid in single-chip mode or when the ready function is disabled.
	RDY		Ready input pin This function is valid in modes where the external bus is enabled and the ready function is enabled.
80	P52	C	General-purpose I/O port This function is valid in single-chip mode or when the hold function is disabled.
	$\overline{\text{HAK}}$		Hold acknowledge output pin This function is valid in modes where the external bus is enabled and the hold function is enabled.
81	P53	C	General-purpose I/O port This function is valid in single-chip mode or external bus mode and when the hold function is disabled.
	HRQ		Hold request input pin This function is valid in modes where the external bus is enabled and the hold function is enabled. During this operation, the input may be used suddenly at any time; therefore, it is necessary to stop output by other fuctions on this pin, except when using it for output deliberately.
82	P54	C	General-purpose I/O port This function is valid in single-chip mode, when the external bus is in 8-bit mode, or when $\overline{\text{WRH}}$ pin output is disabled.
	$\overline{\text{WRH}}$		Write strobe output pin for the high-order 8 bits of the data bus This function is valid in modes where the external bus is enabled, the external bus is in 16-bit mode, and $\overline{\text{WRH}}$ pin output is enabled.
83	P55	C	General-purpose I/O port This function is valid in single-chip mode or when $\overline{\text{WRL}}$ pin output is disabled.
	$\overline{\text{WRL}}$		Write strobe output pin for the low-order 8 bits of the data bus This function is valid in modes where the external bus is enabled and $\overline{\text{WRL}}$ pin output is enabled.

* : FPT-120P-M03, FPT-120C-C02

(Continued)

(Continued)

Type	Circuit	Remarks
F		• N-channel open-drain output • CMOS-level hysteresis input with A/D control and with standby control
G	 Pull-up resistor R R Analog filter Digital input	• CMOS-level hysteresis input with no standby control and with pull-up resistor • With input analog filter (40 ns Typ.) MB90223, MB90224: $\overline{\text{RST}}$ pin can be set to with or without a pull-up resistor by a mask option. MB90P224A: With pull-up resistor MB90W224A: With pull-up resistor MB90P224B: With no pull-up resistor MB90W224B: With no pull-up resistor

: P-type transistor
: N-type transistor

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

3. EPROM Programmer Socket Adapter and Recommended Programmer Manufacturer

Part No.			MB90P224B
Package			QFP-120
Compatible socket adapter **Sun Hayato Co., Ltd.**			ROM-120QF-32DP-16F
Recommended programmer manufacturer and programmer name	**Advantest corp.**	**R4945A** **(main unit)** **+** **R49451A** **(adapter)**	Recommended

Inquiry: Sun Hayato Co., Ltd.: TEL: (81)-3-3986-0403
FAX: (81)-3-5396-9106
Advantest Corp.: TEL: Except JAPAN (81)-3-3930-4111

4. Erase Procedure

Data written in the MB90W224A/W224B is erased (from “0” to “1”) by exposing the chip to ultraviolet rays with a wavelength of 2,537 Å through the translucent cover.

Recommended irradiation dosage for exposure is 10 Wsec/cm^2. This amount is reached in 15 to 20 minutes with a commercial ultraviolet lamp positioned 2 to 3 cm above the package (when the package surface illuminance is 1200 μW/cm^2).

If the ultraviolet lamp has a filter, remove the filter before exposure. Attaching a mirrored plate to the lamp increases the illuminance by a factor of 1.4 to 1.8, thus shortening the required erasure time. If the translucent part of the package is stained with oil or adhesive, transmission of ultraviolet rays is degraded, resulting in a longer erasure time. In that case, clean the translucent part using alcohol (or other solvent not affecting the package).

The above recommended dosage is a value which takes the guard band into consideration and is a multiple of the time in which all bits can be evaluated to have been erased. Observe the recommended dosage for erasure; the purpose of the guard band is to ensure erasure in all temperature and supply voltage ranges. In addition, check the life span of the lamp and control the illuminance appropriately.

Data in the MB90W224A/W224B is erased by exposure to light with a wavelength of 4,000 Å or less.

Data in the device is also erased even by exposure to fluorescent lamp light or sunlight although the exposure results in a much lower erasure rate than exposure to 2,537 Å ultraviolet rays. Note that exposure to such lights for an extended period will therefore affect system reliability. If the chip is used where it is exposed to any light with a wavelength of 4,000 Å or less, cover the translucent part, for example, with a protective seal to prevent the chip from being exposed to the light.

Exposure to light with a wavelength of 4,000 to 5,000 Å or more will not erase data in the device. If the light applied to the chip has a very high illuminance, however, the device may cause malfunction in the circuit for reasons of general semiconductor characteristics. Although the circuit will recover normal operation when exposure is stopped, the device requires proper countermeasures for use in a place exposed continuously to such light even though the wavelength is 4,000 Å or more.

Address	Register	Register name	Access	Resouce name	Initial value
001F28$_H$	OCU compare lower-order data register 06	CPR06L	R/W	Output compare 12	0 0 0 0 0 0 0 0
001F29$_H$					0 0 0 0 0 0 0 0
001F2A$_H$	OCU compare higher-order data register 06	CPR06			0 0 0 0 0 0 0 0
001F2B$_H$					0 0 0 0 0 0 0 0
001F2C$_H$	OCU compare lower-order data register 07	CPR07L	R/W	Output compare 13	0 0 0 0 0 0 0 0
001F2D$_H$					0 0 0 0 0 0 0 0
001F2E$_H$	OCU compare higher-order data register 07	CPR07			0 0 0 0 0 0 0 0
001F2F$_H$					0 0 0 0 0 0 0 0
001F30$_H$	16-bit timer register 0	TMR0	R	16-bit reload timer 0	X X X X X X X X
001F31$_H$					X X X X X X X X
001F32$_H$	16-bit reload register 0	TMRLR0	W		X X X X X X X X
001F33$_H$					X X X X X X X X
001F34$_H$	16-bit timer register 1	TMR1	R	16-bit reload timer 1	X X X X X X X X
001F35$_H$					X X X X X X X X
001F36$_H$	16-bit timer reload register 1	TMRLR1	W		X X X X X X X X
001F37$_H$					X X X X X X X X
001F38$_H$	16-bit timer register 2	TMR2	R	16-bit reload timer 2	X X X X X X X X
001F39$_H$					X X X X X X X X
001F3A$_H$	16-bit timer reload register 2	TMRLR2	W		X X X X X X X X
001F3B$_H$					X X X X X X X X
001F3C$_H$	16-bit timer register 3	TMR3	R	16-bit reload timer 3	X X X X X X X X
001F3D$_H$					X X X X X X X X
001F3E$_H$	16-bit timer reload register 3	TMRLR3	W		X X X X X X X X
001F3F$_H$					X X X X X X X X
001F40$_H$	16-bit timer register 4	TMR4	R	16-bit reload timer 4	X X X X X X X X
001F41$_H$					X X X X X X X X
001F42$_H$	16-bit timer reload register 4	TMRLR4	W		X X X X X X X X
001F43$_H$					X X X X X X X X
001F44$_H$	16-bit timer register 5	TMR5	R	16-bit reload timer 0	X X X X X X X X
001F45$_H$					X X X X X X X X
001F46$_H$	16-bit timer reload register 5	TMRLR5	W		X X X X X X X X
001F47$_H$					X X X X X X X X

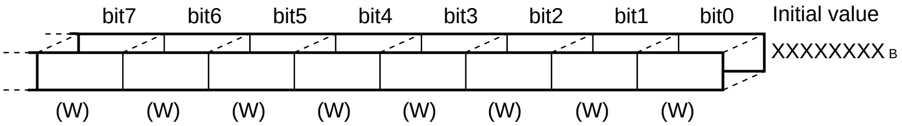
(Continued)

■ INTERRUPT SOURCES AND INTERRUPT VECTORS/INTERRUPT CONTROL REGISTERS

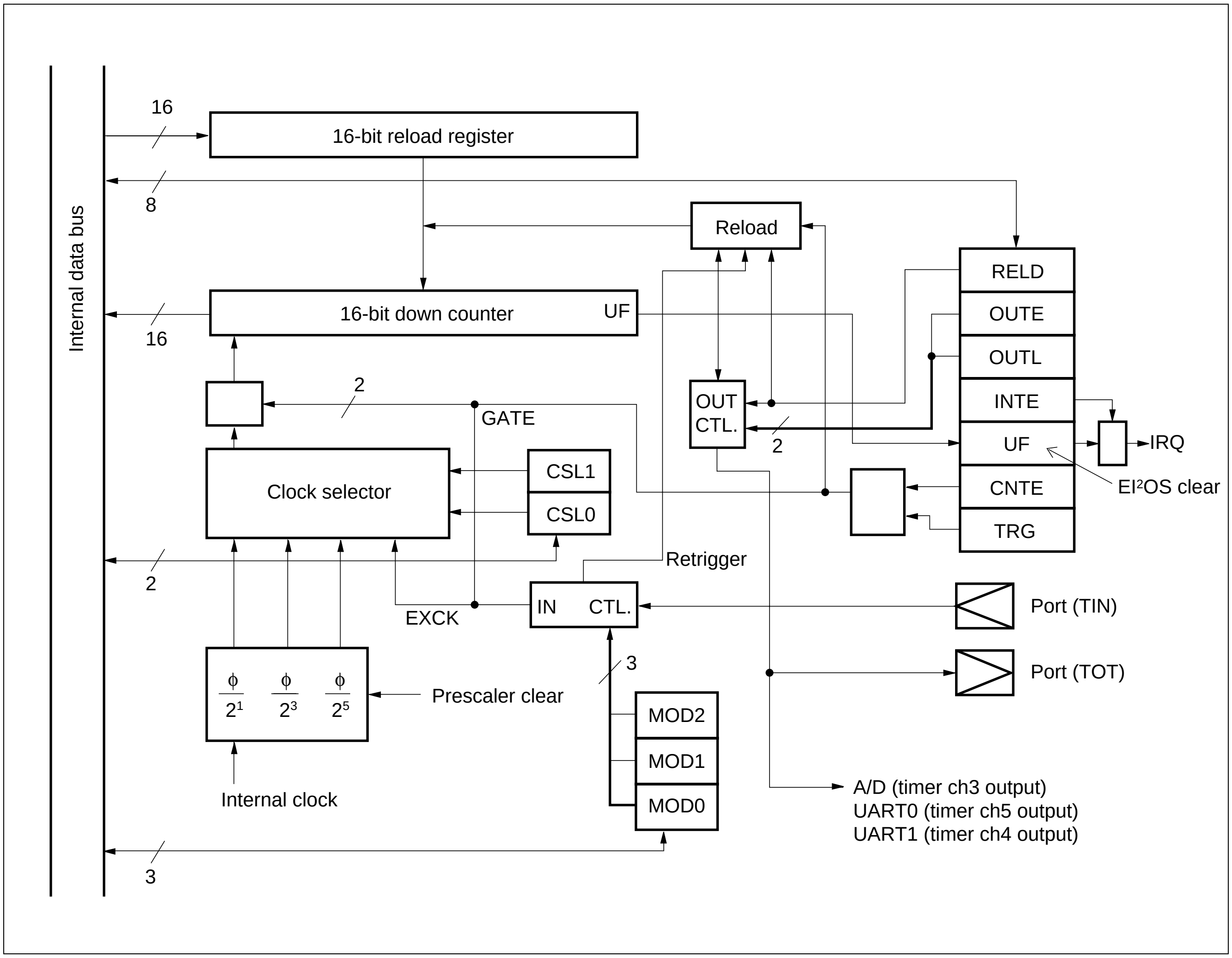
Interrupt source	EI^{2}OS support	Interrupt vector			Interrupt control register	
		No.		Address	ICR	Address
Reset	×	#08	08$_H$	FFFFDC$_H$	—	—
INT9 instruction	×	#09	09$_H$	FFFFD8$_H$	—	—
Exception	×	#10	0A$_H$	FFFFD4$_H$	—	—
External interrupt #0	△	#11	0B$_H$	FFFFD0$_H$	ICR00	0000B0$_H$
External interrupt #1	△	#12	0C$_H$	FFFFCC$_H$		
External interrupt #2	△	#13	0D$_H$	FFFFC8$_H$	ICR01	0000B1$_H$
Input capture 0	△	#14	0E$_H$	FFFFC4$_H$		
PWC0 count completed/overflow	△	#15	0F$_H$	FFFFC0$_H$	ICR02	0000B2$_H$
PWC1 count completed/overflow/input capture 1	△	#16	10$_H$	FFFFBC$_H$		
PWC2 count completed/overflow/input capture 2	△	#17	11$_H$	FFFFB8$_H$	ICR03	0000B3$_H$
PWC3 count completed/overflow/input capture 3	△	#18	12$_H$	FFFFB4$_H$		
24-bit timer, overflow	△	#19	13$_H$	FFFFB0$_H$	ICR04	0000B4$_H$
24-bit timer, intermediate bit/timebase timer, interval interrupt	△	#20	14$_H$	FFFFAC$_H$		
Compare 0	△	#21	15$_H$	FFFFA8$_H$	ICR05	0000B5$_H$
Compare 1	△	#22	16$_H$	FFFFA4$_H$		
Compare 2	△	#23	17$_H$	FFFFA0$_H$	ICR06	0000B6$_H$
Compare 3	△	#24	18$_H$	FFFF9C$_H$		
Compare 4/6	△	#25	19$_H$	FFFF98$_H$	ICR07	0000B7$_H$
Compare 5/7	△	#26	1A$_H$	FFFF94$_H$		
16-bit timer 0/1/2, overflow/PPG0	△	#27	1B$_H$	FFFF90$_H$	ICR08	0000B8$_H$
16-bit timer 3/4/5, overflow/PPG1	△	#28	1C$_H$	FFFF8C$_H$		
10-bit A/D converter count completed	□	#29	1D$_H$	FFFF88$_H$	ICR09	0000B9$_H$
UART1 transmission completed	△	#31	1F$_H$	FFFF80$_H$	ICR10	0000BA$_H$
UART1 reception completed	△	#32	20$_H$	FFFF7C$_H$		
UART0 (ch.1) transmission completed	△	#33	21$_H$	FFFF78$_H$	ICR11	0000BB$_H$
UART0 (ch.2) transmission completed	△	#34	22$_H$	FFFF74$_H$		
UART0 (ch.1) reception completed	○	#35	23$_H$	FFFF70$_H$	ICR12	0000BC$_H$
UART0 (ch.2) reception completed	△	#36	24$_H$	FFFF6C$_H$		
UART0 (ch.0) transmission completed	◎	#37	25$_H$	FFFF68$_H$	ICR13	0000BD$_H$

(Continued)

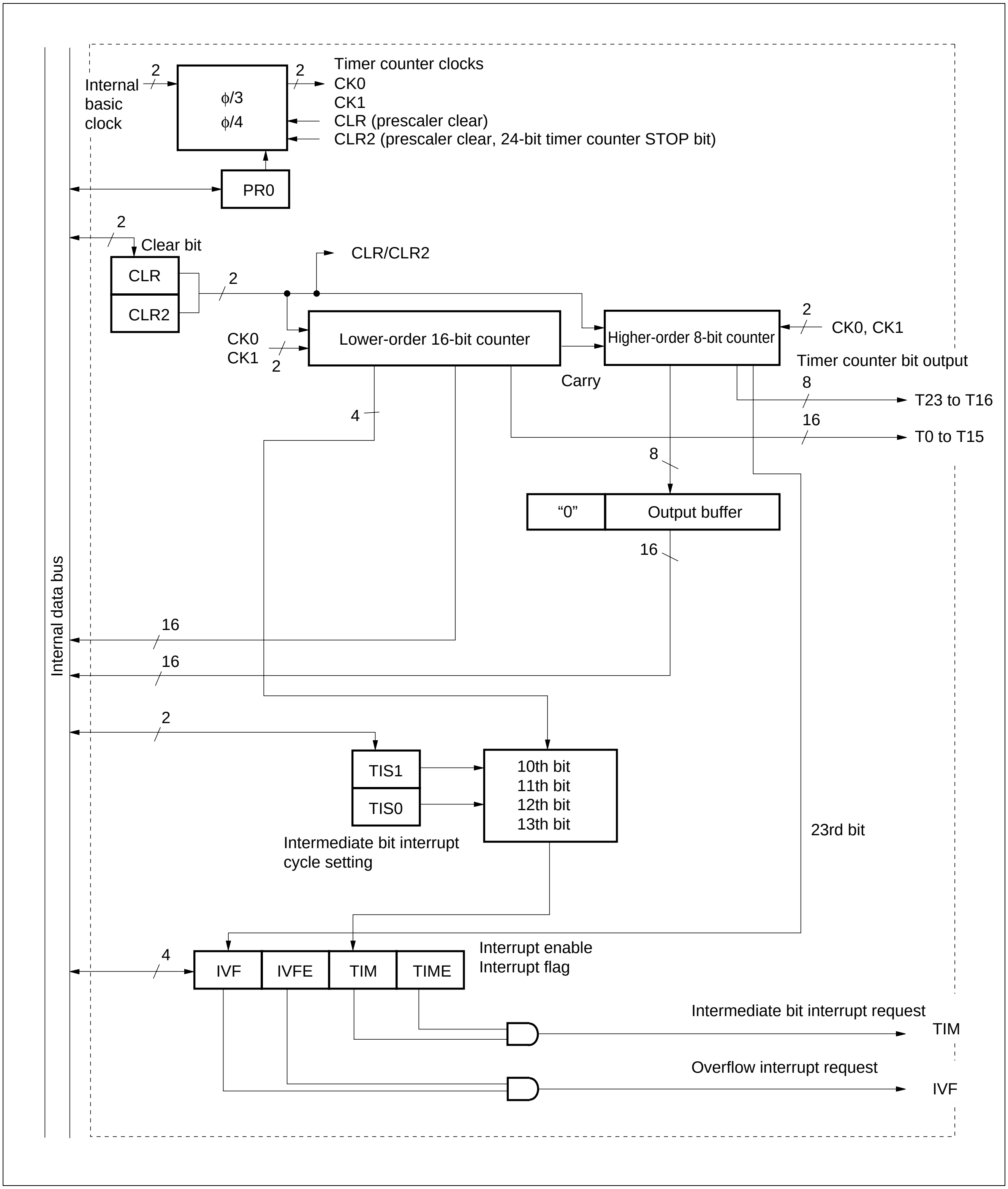
Register name	Address
TMRLR0	001F32 H
TMRLR1	001F36 H
TMRLR2	001F3A H
TMRLR3	001F3E H
TMRLR4	001F42 H
TMRLR5	001F46 H



(2) Block Diagram



(2) Block Diagram



(Continued)

Parameter	Symbol	Pin name	Condition	Value			Unit	Remarks
				Min.	Typ.	Max.		
Analog power supply voltage	I_A	AV_{CC}	f_C = 16 MHz*9	—	3	7	mA	
	I_{AH}		—	—	—	5*6	µA	At stop mode
Input capacitance	C_{IN}	*7	—	—	10	—	pF	

*1: Hysteresis input pins
$\overline{\text{RST}}$, $\overline{\text{HST}}$, P00 to P07, P10 to P17, P20 to P27, P30 to P37, P40 to P47, P50 to P57, P60 to P67, P80 to P87, P90 to P97, PA0 to PA7, PB0 to PB7, PC0 to PC5

*2: Ouput pins
P00 to P07, P10 to P17, P20 to P27, P30 to P37, P40 to P47, P50 to P57, P70 to P77, P80 to P87, PA0 to PA7, PB0 to PB7, PC0 to PC5

*3: Output pins
P00 to P07, P10 to P17, P20 to P27, P30 to P37, P40 to P47, P50 to P57, P60 to P67, P70 to P77, P80 to P87, P90 to P97, PA0 to PA7, PB0 to PB7, PC0 to PC5

*4: A list of availabilities of pull-up/pull-down resistors

Pin name	MB90223/224	MB90P224A/W224A	MB90P224B/W224B
$\overline{\text{RST}}$	Availability of pull-up resistors is optionally defined.	Pull-up resistors available	Unavailable
MD1	Pull-up resistors available	Unavailable	Unavailable
MD0, MD2	Pull-up resistors available	Unavailable	Unavailable

*5: V_{CC} = +5.0 V, V_{SS} = 0.0 V, T_A = +25°C, F_C = 16 MHz

*6: The current value applies to the CPU stop mode with A/D converter inactive (V_{CC} = AV_{CC} = AVRH = +5.5 V).

*7: Other than V_{CC}, V_{SS}, AV_{CC} and AV_{SS}

*8: Measurement condition of power supply current; external clock pin and output pin are open.
Measurement condition of V_{CC}; see the table above mentioned.

*9: F_C = 12 MHz for MB90223

4. AC Characteristics

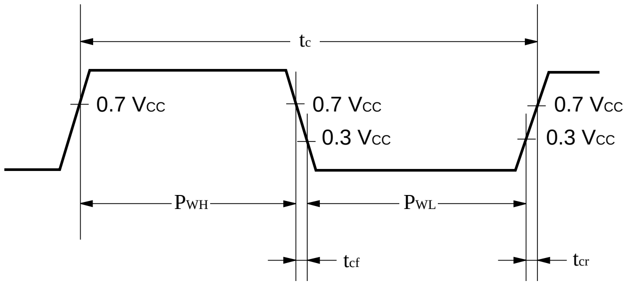
(1) Clock Timing Standards

Single-chip mode MB90223/224/P224B/W224B : (V_{CC} = +4.5 to +5.5 V, V_{SS} = 0.0 V, T_A = –40°C to +105°C)
MB90P224A/W224A : (V_{CC} = +4.5 to +5.5 V, V_{SS} = 0.0 V, T_A = –40°C to +85°C)
External bus mode : (V_{CC} = +4.5 to +5.5 V, V_{SS} = 0.0 V, T_A = –40°C to +70°C)

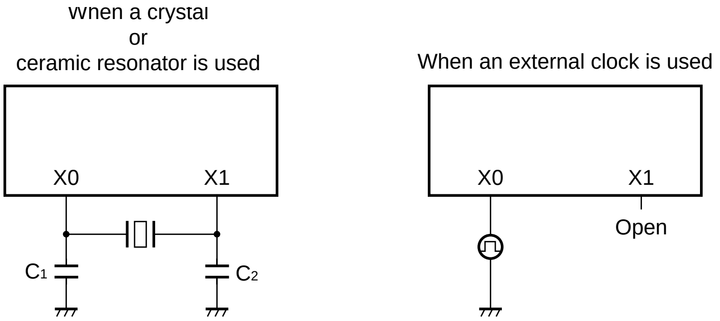
Parameter	Symbol	Pin name	Condition	Value			Unit	Remarks
				Min.	Typ.	Max.		
Clock frequency	F_C	X0, X1	—	10	—	16	MHz	MB90224/ P224A/P224B MB90W224A/ W224B
				10	—	12	MHz	MB90223
Clock cycle time	t_C	X0, X1	—	62.5	—	100	ns	MB90224/ P224A/P224B MB90W224A/ W224B
				83.4	—	100	ns	MB90223
Input clock pulse width	P_{WH} P_{WL}	X0	—	0.4 t_c	—	0.6 t_c	ns	Equivalent to 60% duty ratio
Input clock rising/falling times	t_{cr} t_{cf}	X0	—	—	—	8	ns	t_{cr} + t_{cf}

t_C = 1/f_C

- **Clock Input Timings**



- **Clock Conditions**

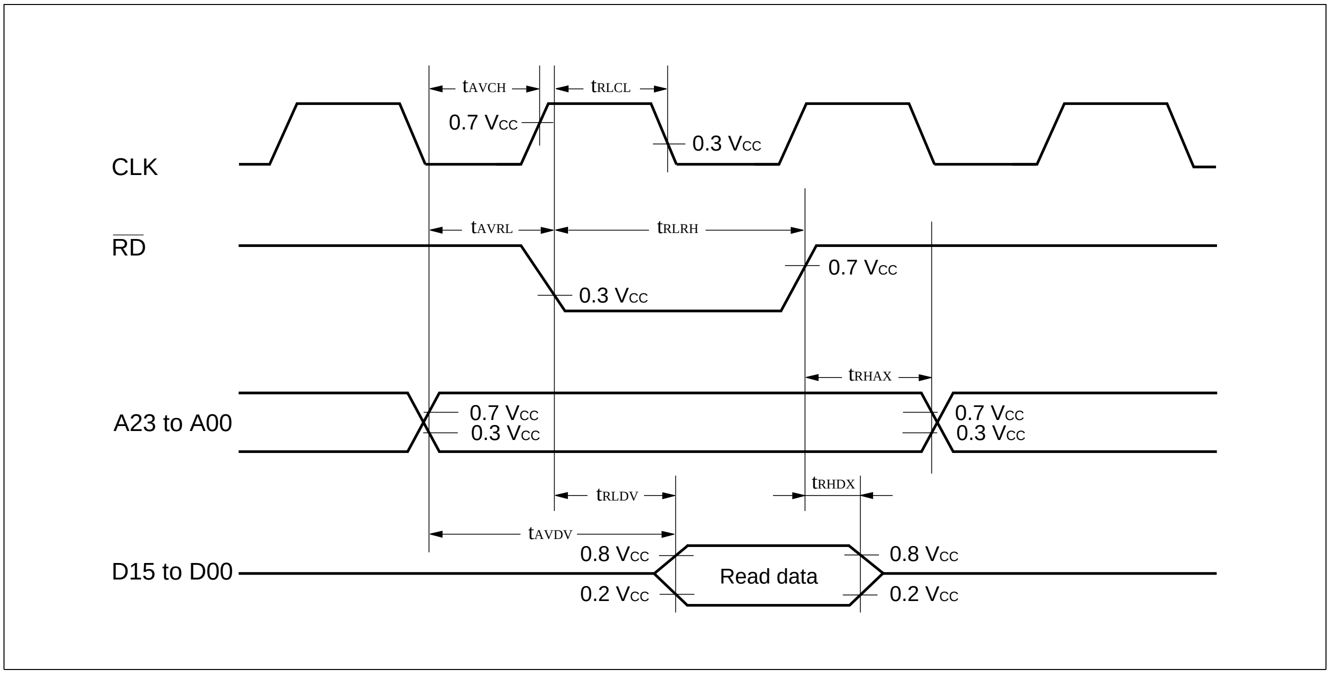


C_1 = C_2 = 10 pF
Select the optimum capacity value for the resonator

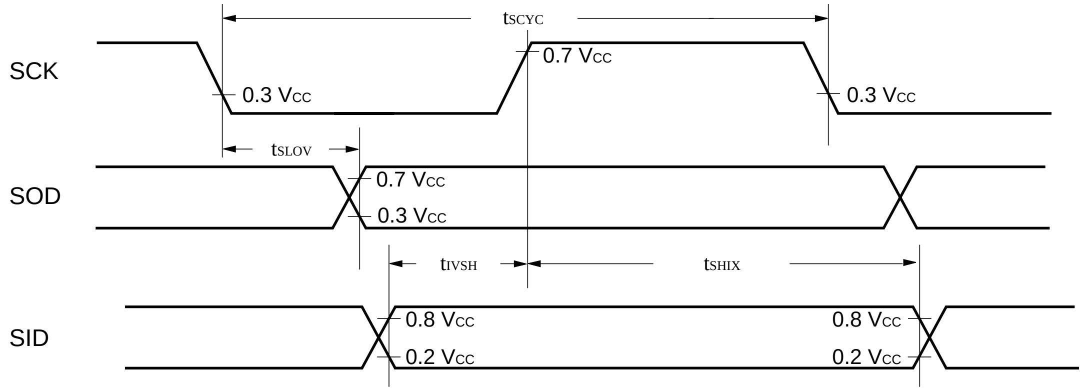
(5) Bus Read Timing

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

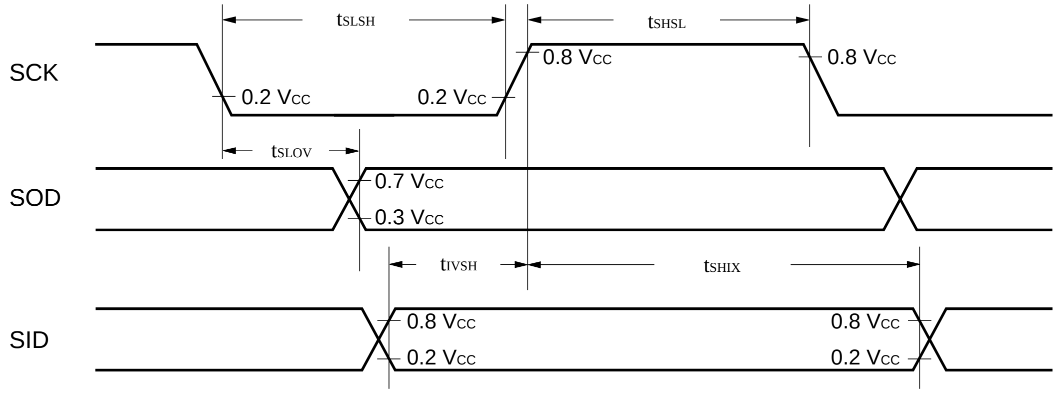
Parameter	Symbol	Pin name	Condition	Value		Unit	Remarks
				Min.	Max.		
Valid address → $\overline{RD}$ ↓ time	t_{AVRL}	A23 to A00	Load condition: 80 pF	$t_{CYC}/2 - 20$	—	ns	
$\overline{RD}$ pulse width	t_{RLRH}	$\overline{RD}$		$t_{CYC} - 25$	—	ns	
$\overline{RD}$ ↓ → Valid data input	t_{RLDV}	D15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



- **Internal Shift Clock Mode**



- **External Shift Clock Input Mode**



■ INSTRUCTION SET (412 INSTRUCTIONS)

Table 1 Explanation of Items in Table of Instructions

Item	Explanation
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.
#	Indicates the number of bytes.
~	Indicates the number of cycles. See Table 4 for details about meanings of letters in items.
B	Indicates the correction value for calculating the number of actual cycles during execution of instruction. The number of actual cycles during execution of instruction is summed with the value in the “cycles” column.
Operation	Indicates operation of instruction.
LH	Indicates special operations involving the bits 15 through 08 of the accumulator. Z: Transfers “0”. X: Extends before transferring. —: Transfers nothing.
AH	Indicates special operations involving the high-order 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 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: Cannot be used for addresses that have different meanings depending on whether they are read or written.

Table 3　Effective Address Fields

Code	Notation			Address format	Number of bytes in address extemsion*
00	R0	RW0	RL0	Register direct “ea” corresponds to byte, word, and long-word types, starting from the left	—
01	R1	RW1	(RL0)		
02	R2	RW2	RL1		
03	R3	RW3	(RL1)		
04	R4	RW4	RL2		
05	R5	RW5	(RL2)		
06	R6	RW6	RL3		
07	R7	RW7	(RL3)		
08	@RW0			Register indirect	0
09	@RW1				
0A	@RW2				
0B	@RW3				
0C	@RW0 +			Register indirect with post-increment	0
0D	@RW1 +				
0E	@RW2 +				
0F	@RW3 +				
10	@RW0 + disp8			Register indirect with 8-bit displacement	1
11	@RW1 + disp8				
12	@RW2 + disp8				
13	@RW3 + disp8				
14	@RW4 + disp8				
15	@RW5 + disp8				
16	@RW6 + disp8				
17	@RW7 + disp8				
18	@RW0 + disp16			Register indirect with 16-bit displacemen	2
19	@RW1 + disp16				
1A	@RW2 + disp16				
1B	@RW3 + disp16				
1C	@RW0 + RW7			Register indirect with index	0
1D	@RW1 + RW7			Register indirect with index	0
1E	@PC + dip16			PC indirect with 16-bit displacement	2
1F	addr16			Direct address	2

* : The number of bytes for address extension is indicated by the “+” symbol in the “#” (number of bytes) column in the Table of Instructions.

Table 4 Number of Execution Cycles for Each Form of Addressing

Code	Operand	(a)*
		Number of execution cycles for each from of addressing
00 to 07	Ri RWi RLi	Listed in Table of Instructions
08 to 0B	@RWj	1
0C to 0F	@RWj +	4
10 to 17	@RWi + disp8	1
18 to 1B	@RWj + disp16	1
1C 1D 1E 1F	@RW0 + RW7 @RW1 + RW7 @PC + dip16 @addr16	2 2 2 1

* : “(a)” is used in the “cycles” (number of cycles) column and column B (correction value) in the Table of Instructions.

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

Operand	(b)*	(c)*	(d)*
	byte	word	long
Internal register	+ 0	+ 0	+ 0
Internal RAM even address	+ 0	+ 0	+ 0
Internal RAM odd address	+ 0	+ 1	+ 2
Even address not in internal RAM	+ 1	+ 1	+ 2
Odd address not in internal RAM	+ 1	+ 3	+ 6
External data bus (8 bits)	+ 1	+ 3	+ 6

* : “(b)”, “(c)”, and “(d)” are used in the “cycles” (number of cycles) column and column B (correction value) in the Table of Instructions.

Table 7 Transfer Instructions (Word) [40 Instructions]

Mnemonic	#	cycles	B	Operation	LH	AH	I	S	T	N	Z	V	C	RMW
MOVW A, dir	2	2	(c)	word (A) ← (dir)	–	*	–	–	–	*	*	–	–	–
MOVW A, addr16	3	2	(c)	word (A) ← (addr16)	–	*	–	–	–	*	*	–	–	–
MOVW A, SP	1	2	0	word (A) ← (SP)	–	*	–	–	–	*	*	–	–	–
MOVW A, RWi	1	1	0	word (A) ← (RWi)	–	*	–	–	–	*	*	–	–	–
MOVW A, ear	2	1	0	word (A) ← (ear)	–	*	–	–	–	*	*	–	–	–
MOVW A, eam	2+	2+ (a)	(c)	word (A) ← (eam)	–	*	–	–	–	*	*	–	–	–
MOVW A, io	2	2	(c)	word (A) ← (io)	–	*	–	–	–	*	*	–	–	–
MOVW A, @A	2	2	(c)	word (A) ← ((A))	–	–	–	–	–	*	*	–	–	–
MOVW A, #imm16	3	2	0	word (A) ← imm16	–	*	–	–	–	*	*	–	–	–
MOVW A, @RWi+disp8	2	3	(c)	word (A) ← ((RWi) +disp8)	–	*	–	–	–	*	*	–	–	–
MOVW A, @RLi+disp8	3	6	(c)	word (A) ← ((RLi) +disp8)	–	*	–	–	–	*	*	–	–	–
MOVW A, @SP+disp8	3	3	(c)	word (A) ← ((SP) +disp8)	–	*	–	–	–	*	*	–	–	–
MOVPW A, addr24	5	3	(c)	word (A) ← (addr24)	–	*	–	–	–	*	*	–	–	–
MOVPW A, @A	2	2	(c)	word (A) ← ((A))	–	–	–	–	–	*	*	–	–	–
MOVW dir, A	2	2	(c)	word (dir) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW addr16, A	3	2	(c)	word (addr16) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW SP, # imm16	4	2	0	word (SP) ← imm16	–	–	–	–	–	*	*	–	–	–
MOVW SP, A	1	2	0	word (SP) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW RWi, A	1	1	0	word (RWi) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW ear, A	2	2	0	word (ear) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW eam, A	2+	2+ (a)	(c)	word (eam) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW io, A	2	2	(c)	word (io) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW @RWi+disp8, A	2	3	(c)	word ((RWi) +disp8) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW @RLi+disp8, A	3	6	(c)	word ((RLi) +disp8) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVW @SP+disp8, A	3	3	(c)	word ((SP) +disp8) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVPW addr24, A	5	3	(c)	word (addr24) ← (A)	–	–	–	–	–	*	*	–	–	–
MOVPW @A, RWi	2	3	(c)	word ((A)) ← (RWi)	–	–	–	–	–	*	*	–	–	–
MOVW RWi, ear	2	2	0	word (RWi) ← (ear)	–	–	–	–	–	*	*	–	–	–
MOVW RWi, eam	2+	3+ (a)	(c)	word (RWi) ← (eam)	–	–	–	–	–	*	*	–	–	–
MOVW ear, RWi	2	3	0	word (ear) ← (RWi)	–	–	–	–	–	*	*	–	–	–
MOVW eam, RWi	2+	3+ (a)	(c)	word (eam) ← (RWi)	–	–	–	–	–	*	*	–	–	–
MOVW RWi, #imm16	3	2	0	word (RWi) ← imm16	–	–	–	–	–	*	*	–	–	–
MOVW io, #imm16	4	3	(c)	word (io) ← imm16	–	–	–	–	–	–	–	–	–	–
MOVW ear, #imm16	4	2	0	word (ear) ← imm16	–	–	–	–	–	*	*	–	–	–
MOVW eam, #imm16	4+	2+ (a)	(c)	word (eam) ← imm16	–	–	–	–	–	–	–	–	–	–
MOVW @AL, AH	2	2	(c)	word ((A)) ← (AH)	–	–	–	–	–	*	*	–	–	–
XCHW A, ear	2	3	0	word (A) ↔ (ear)	–	–	–	–	–	–	–	–	–	–
XCHW A, eam	2+	3+ (a)	2× (c)	word (A) ↔ (eam)	–	–	–	–	–	–	–	–	–	–
XCHW RWi, ear	2	4	0	word (RWi) ↔ (ear)	–	–	–	–	–	–	–	–	–	–
XCHW RWi, eam	2+	5+ (a)	2× (c)	word (RWi) ↔ (eam)	–	–	–	–	–	–	–	–	–	–

Note: For an explanation of “(a)” and “(c)”, 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 9 Addition and Subtraction Instructions (Byte/Word/Long Word) [42 Instructions]

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

For an explanation of “(a)”, “(b)”, “(c)” 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 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 14 Logical 1 Instructions (Byte, Word) [39 Instructions]

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

For an explanation of “(a)”, “(b)”, “(c)” 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 23 Bit Manipulation Instructions [21 Instructions]

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

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

*1: 5 when branching, 4 when not branching
*2: 7 when condition is satisfied, 6 when not satisfied
*3: Undefined count
*4: Until condition is satisfied

Table 24 Accumulator Manipulation Instructions (Byte/Word) [6 Instructions]

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

Table 25 String Instructions [10 Instructions]

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

m: RW0 value (counter value)

*1: 3 when RW0 is 0, 2 + 6 × (RW0) for count out, and 6n + 4 when match occurs
*2: 4 when RW0 is 0, 2 + 6 × (RW0) in any other case
*3: (b) × (RW0)
*4: (b) × n
*5: (b) × (RW0)
*6: (c) × (RW0)
*7: (c) × n
*8: (c) × (RW0)