Patent Number : 75201, 62630, 61007 (R.O.C)
Patent Pending : 83216083 (R.O.C)

GENERAL DESCRIPTION

EM73491 is an advanced single chip CMOS 4-bit micro-controller. It contains 4K-byte ROM, 244-nibble RAM, 4-bit ALU, 13-level subroutine nesting, 22-stage time base, two independent 12-bit timer/counter and one data pointer (DP) for the kernel function, and the EM73491 also contains 5 interrupt sources, 8 I/O ports (including 1 output port , 2 input ports and 5 bidirection ports) and it can also support user the internal reset.
For the application in the telecom product, EM73491 also supports user the beep tone and DTMF function.
Except low-power consumption and high speed, EM73491 also has sleep and hold mode operation for power saving function.
EM73491 is suitable for appliaction in telecom products and family appliances.

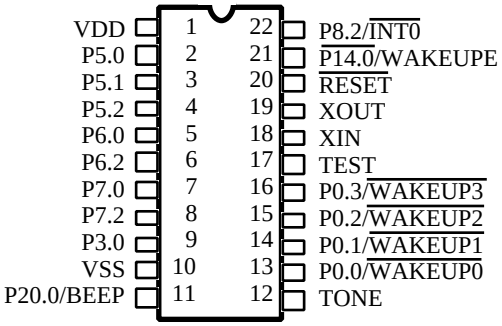
FEATURES

- Operation voltage : 2.2V to 5.5V (clock frequency : 480 KHz to 4 MHz)
- Clock source : Single clock system available for resonator or crystal and external clock source by mask option.
- Frequency selection : 480K/800K/960K/3.58M/3.84M/4M Hz decided by mask option.
- Instruction set : 110 powerful instructions.
- Instruction cycle time : Up to 2μs for 4 MHz.
- ROM capacity : 4K X 8 bits.
- RAM capacity : 244 X 4 bits.
- Input port : 2 ports (5-bit)
  Port 0 : Pull-up or pull-down resistor available by mask option for input port function.
  Pull-up resistor available by mask option for used as sleep releasing port.
  Port 14 : Pull-up resistor available by mask option.
- Output port : 1 port (1-bit) (push-up or open-drain type decided by mask option).
- Bidirection I/O port : 5 ports (16-bit) (push-pull or open-drain type decided by mask option).
- 12 bits timer/counter : Two 12-bit timer/counters are programmable for timer, event counter and pulse width measurement.
- Built-in time base counter : 22 stages.
- Subroutine nesting : Up to 13 levels.
- Interrupt : External . . . . . 2 input interrupt sources.
  Internal . . . . . 2 Timer overflow interrupts.
  1 Time base interrupt.
- Beep tone function.
- DTMF tone function.
- Power saving function : Sleep function, CPU hold internal state and stop oscillating.
  Hold function, CPU hold internal state and oscillator still working.

- Package type : EM73491BP DIP 22 pins.
  EM73491DP DIP 28 pins.

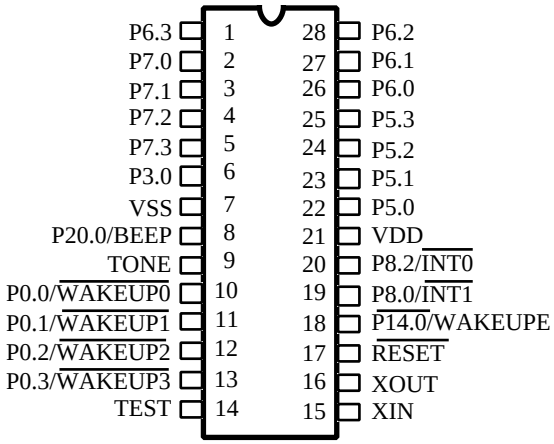
PIN CONFIGURATIONS

EM73491BP



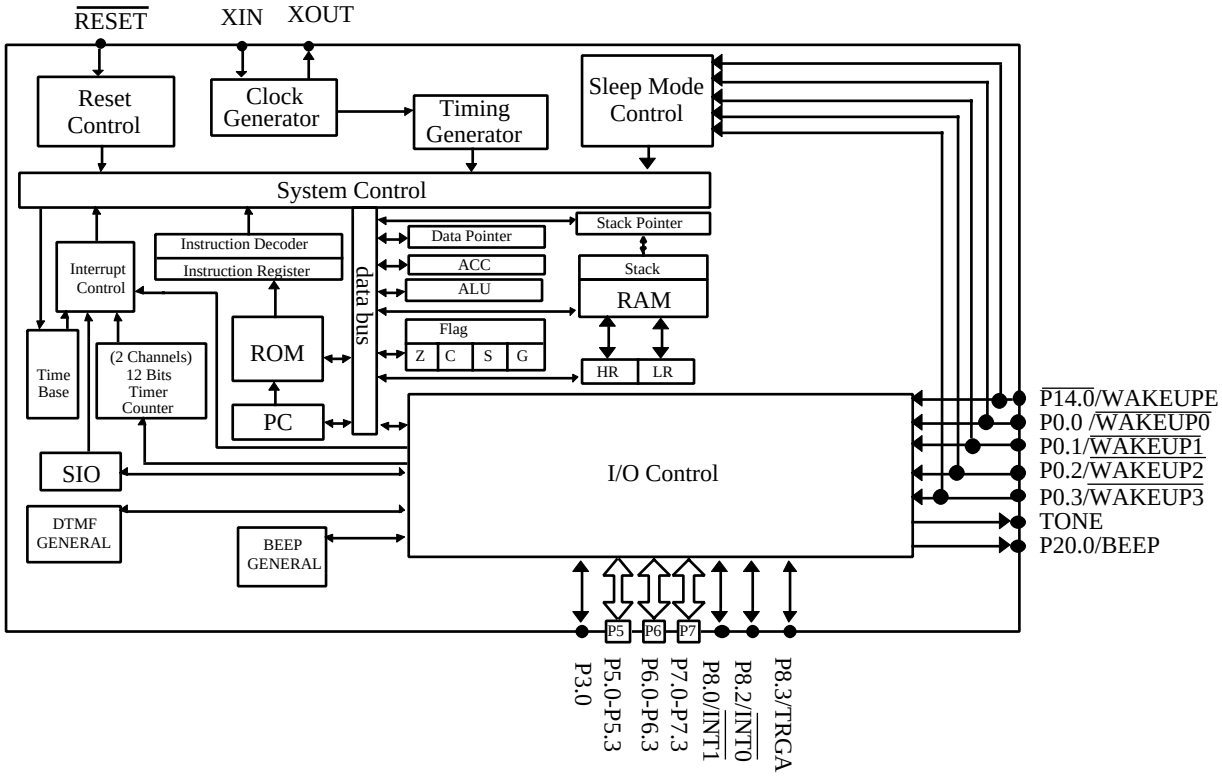
22 pin DIP

EM73491DP



28 pin DIP

FUNCTION BLOCK DIAGRAM



PIN DESCRIPTIONS

Symbol	Pin-type	Function
V_{DD}		Power supply (+)
V_{SS}		Power supply (-)
$\overline{RESET}$	RESET-A	External reset input pin or internal reset function by mask option mask option : none pull-up
XIN	OSC-A	Crystal or external clock source connecting pin
XOUT	OSC-A	Crystal connecting pin
P0.(0..3)/$\overline{WAKEUP0..3}$	INPUT-D	4 bits input port with Sleep/Hold releasing and R-option function mask option : R-option enable, wakeup disable, none R-option enable, wakeup disable, pull-up R-option disable, wakeup enable, none R-option disable, wakeup enable, pull-up R-option disable, wakeup disable, none R-option disable, wakeup disable, pull-up R-option disable, wakeup disable, pull-down
P3.0	I/O-G	1-bit bidirection I/O pin mask option : NMOS open-drain PMOS open-drain push-pull
P5(0..3),P6(0..3) P7(0..3)	I/O-F	4 bits bidirection I/O ports with R-option function mask option : R-option enable, push-pull R-option disable, open-drain R-option disable, push-pull
P8.0/$\overline{INT1}$ P8.2/$\overline{INT0}$	I/O-C	2 bit bidirection I/O port with external interrupt sources input mask option : open-drain push-pull
P8.3/TRGA	I/O-C	1 bit bidirection I/O pin with timer/counter A external input mask option : open-drain push-pull
$\overline{P14.0}$/WAKEUPE	INPUT-F	1 bit input pin with Sleep/Hold releasing function mask option : none pull-pull
P20.0/BEEP	OUTPUT-B	1 bit output pin is shared with BEEP output pin mask option : open-drain push-pull
TONE		DTMF output pin
TEST		Test pin must be connected to V_{SS}

FUNCTION DESCRIPTIONS

PROGRAM ROM (4K X 8 bits)

4 K x 8 bits program ROM contains user's program and some fixed data .
The basic structure of program ROM can be divided into 5 parts.

1. Address 000h: Reset start address.
2. Address 002h - 00Ch: 5 kinds of interrupt service rountine entry addresses .
3. Address 00Eh-086h : SCALL subroutine entry address, only available at 00Eh,016h,01Eh,026h, 02Eh, 036h, 03Eh, 046h, 04Eh, 056h, 05Eh, 066h, 06Eh, 076h ,07Eh, 086h .
4. Address 000h - 7FFh : LCALL subroutine entry address
5. Address FE0h - FFFh : The data region for 5-to-8 bits data conversion table .
6. Address 000h - FFFh : Except used as above function, the other region can be used as user's program region.

address	4096 x 8 bits
000h	Reset start address
002h	$\overline{INT0}$; External interrupt service routine entry address
004h	
006h	TRGA; Timer/counterA interrupt service routine entry address
008h	TRGB; Timer/counter B interrupt service routine entry address
00Ah	TBI; Time base interrupt service routine entry address
00Ch	$\overline{INT1}$; External interrupt service routine entry address
00Eh 086h	SCALL, subroutine call entry address
⋮	⋮
FE0h FFFh	Data conversion table for "OUT12" instruction

User's program and fixed data are stored in the program ROM. User's program is according the PC value to send next executed instruction code . Fixed data can be read out by two ways.

(1) Table-look-up instruction:

Table-look-up instruction is depended on the Data Pointer (DP) to indicate to ROM address, then to get the ROM code data .

LDAX **Acc ← ROM[DP]$_L$**
LDAXI **Acc ← ROM[DP]$_H$,DP+1**

DP is a 12-bit data register which can store the program ROM address to be the pointer for the ROM code data . first, user load ROM address into DP by instruction "LDADPL, LDADPM, LDADPH". then user can get the lower nibble of ROM code data by instruction "LDAX" and higher nibble by instruction "LDAXI".

PROGRAM EXAMPLE: Read out the ROM code of address 777h by table-look-up instruction.

```
LDIA #07h;
STADPL    ; [DP]L ← 07h
STADPM    ; [DP]M ← 07h
STADPH    ; [DP]H ← 07h, Load DP=777h
```

```
LDL #00h;
LDH #03h;
LDAX        ; ACC ← 6h
STAMI       ; RAM[30] ← 6h
LDAXI       ; ACC ← 5h
STAM        ; RAM[31] ← 5h
;
ORG 777h
DATA 56h;
:
```

(2) 5-to-8 bits data conversion instruction:

OUT12 : IF CF=1 Port1= ROM[FF0h+RAM[HL]]$_L$; Port2= ROM [FF0h+RAM[HL]]$_H$
IF CF=0 Port1= ROM[FE0h+RAM[HL]]$_L$; Port2= ROM[FE0h+RAM[HL]]$_H$

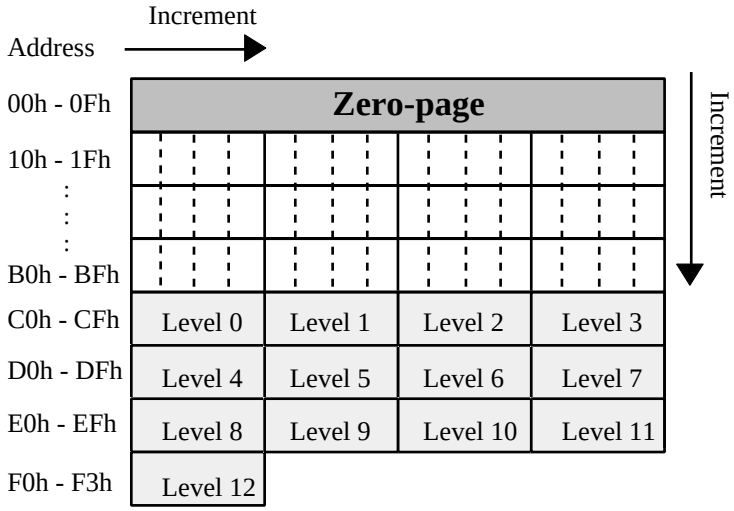
5-to-8 bits data conversion instruction can read fixed data from data conversion table (FE0-FFF) out to Port1 and Port2 synchronously, the 5-bit data is composed by CF and RAM data which specified by HL, when CF=1, the 8-bit data is located in address of FF0h+ RAM[HL] of ROM, in the other way, when CF=0, the 8-bit data is located in address FE0h + RAM[HL] of ROM.

PROGRAM EXAMPLE : To output port1, port2 data by 5-to-8 bits data conversion instruction.

```
LDL  #00h;
LDH  #03h;
LDIA  #00h;
STAM  ;  RAM[30] ← 00h
TTCFS;  CF ← 1
OUT12;
:
:
ORG     FF0h
DATA    40h;
        7Ch;
        12h;
        18h;
        2Ch;
        09h;
        01h;
        5Ch;
        00h;
        08h;
```

DATA RAM (244-nibble)

There is total 244 - nibble data RAM from address 00 to F3h
Data RAM includes 3 parts: zero page region, stacks and data area.



ZERO- PAGE:

From 00h to 0Fh is the location of zero-page . It is used as the pointer in zero -page addressing mode for the instruction of "STD #k,y; ADD #k,y; CLR y,b; CMP y,b".

PROGRAM EXAMPLE: To wirte immediate data "07h" to address "03h" of RAM and to clear bit 2 of RAM.
STD #07h, 03h ; RAM[03] ← 07h
CLR 0Eh,2 ; $RAM[0Eh]_2 \leftarrow 0$

STACK:

There are 13 - level (maximum) stack for user using for subroutine (including interrupt and CALL). User can assign any level be the starting stack by giving the level number to stack pointer(SP) .
When user using any instruction of CALL or subroutine, before entry the subroutine, the previous PC address will be saved into stack until return from those subroutines ,the PC value will be restored by the data saved in stack.

DATA AREA:

Except the special area used by user, the whole RAM can be used as data area for storing and loading general data.

ADDRESSING MODE

(1) Indirect addressing mode:
Indirect addressing mode indicates the RAM address by specified HL register .
For example: LDAM ; Acc ← RAM[HL]
STAM ; RAM[HL] ← Acc

(2) Direct addressing mode:
Direct addressing mode indicates the RAM address by immediate data .

For example: LDA x ; Acc← RAM[x]
STA x ; RAM[x] ← Acc

(3) Zero-page addressing mode
For zero-page region, user can using direct addressing to write or do any arithematic, comparsion or bit manupulated operation directly.
For example: STD #k,y ; RAM[y] ← #k
ADD #k,y; RAM[y] ← RAM[y] + #k

PROGRAM COUNTER (4K ROM)

Program counter (PC) is composed by a 12-bit counter, which indicates the next executed address for the instruction of program ROM.
For a 4K - byte size ROM, PC can indicate address form 000h - FFFh, for BRANCH and CALL instrcutions, PC is changed by instruction indicating.

(1) Branch instruction:

SBR a
Object code: 00aa aaaa
Condition: SF=1; PC ← $PC_{11\text{-}6.a}$ (branch condition satisified)

PC | Hold original PC value+1 | a | a | a | a | a | a |

SF=0; PC← PC +1(branch condition not satisified)

PC | Original PC value + 1 |

LBR a
Object code: 1100 aaaa aaaa aaaa
Condition: SF=1; PC ← a (branch condition satisified)

PC | a | a | a | a | a | a | a | a | a | a | a | a |

SF=0 ; PC ← PC + 2 (branch condition not satisified)

PC | Original PC value + 2 |

(2) Subroutine instruction:

SCALL a
Object code: 1110 nnnn
Condition : PC ← a ; a=8n+6 ; n=1..15 ; a=86h, n=0

PC | 0 | 0 | 0 | 0 | a | a | a | a | a | a | a | a |

LCALL a
Object code: 0100 0 aaa aaaa aaaa
Condition: PC ← a

PC | 0 | a | a | a | a | a | a | a | a | a | a | a |

RET
Object code: 0100 1111
Condition: PC ← STACK[SP]; SP + 1

PC | The return address stored in stack |

RT I
Object code: 0100 1101
Condition : FLAG. PC ← STACK[SP]; EI ← 1; SP + 1

PC | The return address stored in stack |

(3) Interrupt acceptance operation:

When an interrupt is accepted, the original PC is pushed into stack and interrupt vector will be loaded into PC,The interrupt vectors are as following:

$\overline{\textbf{INT0}}$ (External interrupt from P8.2)

| PC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

TRGA (Timer A overflow interrupt)

| PC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

TRGB (Time B overflow interrupt)

| PC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

TBI (Time base interrupt)

| PC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

$\overline{\textbf{INT1}}$ (External interrupt from P8.0)

| PC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |

(4) Reset operation:

| PC | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(5) Other operations:

For 1-byte instruction execution: PC + 1
For 2-byte instruction execution: PC + 2

ACCUMULATOR

Accumulator is a 4-bit data register for temporary data . For the arithematic, logic and comparative opertion .., ACC plays a role which holds the source data and result .

FLAGS

There are four kinds of flag, CF (Carry flag), ZF (Zero flag), SF (Status flag) and GF (General flag), these 4 1-bit flags are affected by the arithematic, logic and comparative .... operation .
All flags will be put into stack when an interrupt subroutine is served, and the flags will be restored after RTI instruction executed .

(1) Carry Flag (CF)

The carry flag is affected by following operation:
a. Addition : CF as a carry out indicator, when the addition operation has a carry-out, CF will be "1", in another word, if the operation has no carry-out, CF will be "0".

b. Subtraction : CF as a borrow-in indicator, when the subtraction operation must has a borrow, in the CF will be "0", in another word, if no borrow-in, CF will be "1".

c. Comparision: CF is as a borrow-in indicator for Comparision operation as the same as subtraction operation.

d. Rotation: CF shifts into the empty bit of accumulator for the rotation and holds the shift out data after rotation.

e. CF test instruction : For TFCFC instruction, the content of CF sends into SF then clear itself "0". For TTSFC instruction, the content of CF sends into SF then set itself "1".

(2) Zero Flag (ZF)

ZF is affected by the result of ALU, if the ALU operation generate a "0" result, the ZF will be "1", otherwise, the ZF will be "0".

(3) Status Flag (SF)

The SF is affected by instruction operation and system status .

a. SF is initiated to "1" for reset condition .

b. Branch instruction is decided by SF, when SF=1, branch condition will be satisified, otherwise, branch condition will not be satisified by SF = 0 .

(4) General Flag (GF)

GF is a one bit general purpose register which can be set, clear, test by instruction SGF, CGF and TGS.

PROGRAM EXAMPLE:
Check following arithematic operation for CF, ZF, SF

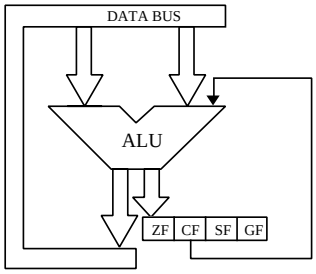
	CF	ZF	SF
LDIA #00h;	-	1	1
LDIA #03h;	-	0	1
ADDA #05h;	-	0	1
ADDA #0Dh;	-	0	0
ADDA #0Eh;	-	0	0

ALU

The arithematic operation of 4 - bit data is performed in ALU unit . There are 2 flags can be affected by the result of ALU operation, ZF and SF . The operation of ALU can be affected by CF only .

ALU STRUCTURE

ALU supported user arithematic operation function, including : addition, subtraction and rotaion.



ALU FUNCTION

(1) Addition:

For instruction ADDAM, ADCAM, ADDM #k, ADD #k,y .... ALU supports addition function. The addition operation can affect CF and ZF. For addition operation, if the result is "0", ZF will be "1", otherwise, not equal "0", ZF will be "0", When the addition operation has a carry-out. CF will be "1", otherwise, CF will be "0".

EXAMPLE:

Operation	Carry	Zero
3+4=7	0	0
7+F=6	1	0
0+0=0	0	1
8+8=0	1	1

(2) Subtraction:

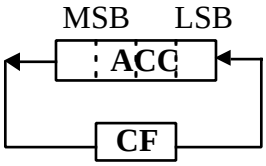
For instruction SUBM #k, SUBA #k, SBCAM, DECM... ALU supports user subtraction function . The subtraction operation can affect CF and ZF, For subtraction operation, if the result is negative, CF will be "0", it means a borrow out, otherwise, if the result is positive, CF will be "1". For ZF, if the result of subtraction operation is "0", the ZF will be "1", otherwise, ZF will be "1".

EXAMPLE:

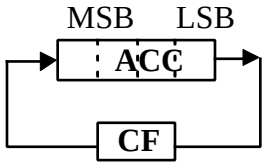
Operation	Carry	Zero
8-4=4	1	0
7-F= -8(1000)	0	0
9-9=0	1	1

(3) Rotation:

There are two kinds of rotation operation, one is rotation left, the other is rotation right.
RLCA instruction rotates Acc value to left, shift the CF value into the LSB bit of Acc and the shift out data will be hold in CF.



RRCA instruction operation rotates Acc value to right, shift the CF value into the MSB bit of Acc and the shift out data will be hold in CF.



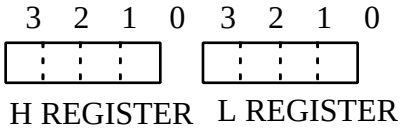
PROGRAM EXAMPLE: To rotate Acc right and shift a "1" into the MSB bit of Acc .

TTCFS; CF ← 1
RRCA; rotate Acc right and shift CF=1 into MSB.

HL REGISTER

HL register are two 4-bit registers, they are used as a pair of pointer for the address of RAM memory and also 2 independent temporary 4-bit data registers. For some instruction, L register can be a pointer to indicate the pin number (Port5 - Port7) .

HL REGISTER STRUCTURE



HL REGISTER FUNCTION

(1) For instruction : LDL #k, LDH #k, THA, THL, INCL, DECL, EXAL, EXAH, HL register used as a temporary register .

PROGRAM EXAMPLE: Load immediate data "5h" into L register, "Dh" into H register.
LDL #05h;
LDH #0Dh;

(2) For instruction LDAM, STAM, STAMI .., HL register used as a pointer for the address of RAM memory.

PROGRAM EXAMPLE: Store immediate data #Ah into RAM of address 35h.

```
LDL #5h;
LDH #3h;
STDMI #0Ah; RAM[35] ← Ah
```

(3) For instruction : SELP, CLPL, TFPL, L regieter be a pointer to indicate the bit of I/O port.

When LR = 4 - 7, indicate P5.0 - P5.3
LR = 8 - B, indicate P6.0 - P6.3
LR = C - F, indicate P7.0 - P7.3

PROGRAM EXAMPLE: To set bit 2 of Port6 to "1"

```
LDL #0Ah;
SEPL ; P6.2 ← 1
```

STACK POINTER (SP)

Stack pointer is a 4-bit register which stores the present stack level number.
Before using stack, user must set the SP value first, CPU will not initiate the SP value after reset condition . When a new subroutine is accepted, the SP will be decreased one automatically, in another word, if returning from a subroutine, the SP will be increased one .
The data transfer between ACC and SP is by instruction of "LDASP" and "STASP".

DATA POINTER (DP)

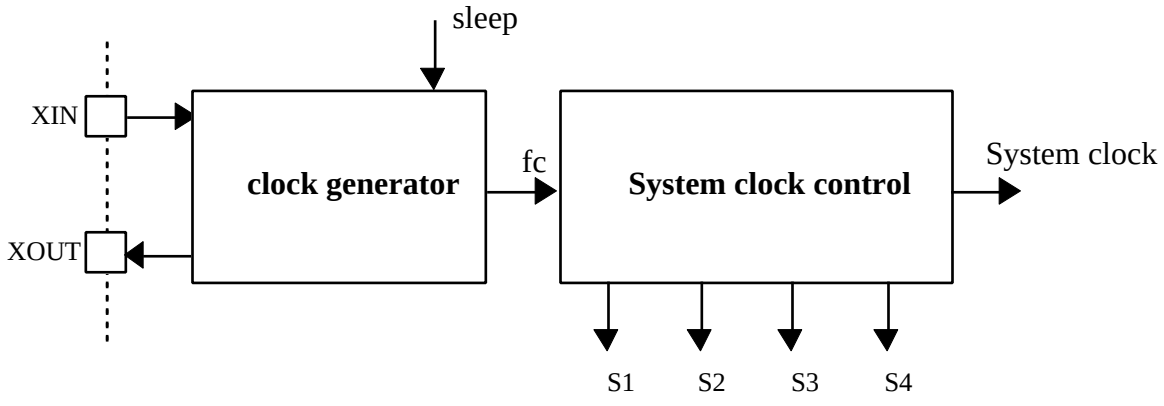
Data pointer is a 12-bit register which stores the address of ROM can indicate the ROM code data specified by user (refer to data ROM).

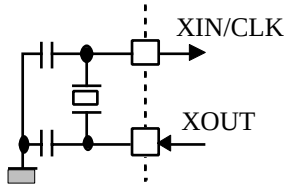
CLOCK AND TIMING GENERATOR

The clock generator is supported by a single clock system, the clock source comes from crystal (resonator),the working frequency range is 480 K Hz to 4 MHz depending on the working voltage.

CLOCK AND TIMING GENERATOR STRUCTURE

The clock generator connects outside compoments (crystal or resonator by XIN and XOUT pin for crystal osc type) the clock generator generates a basic system clock "fc".
When CPU sleeping, the clock generator will be stoped until the sleep condition released. The system clock control generates 4 basic phase signals (S1, S2, S3, S4) and system clock .





Crystal connection

CLOCK AND TIMING GENERATOR FUNCTION

The frequency of fc is the oscillation frequency for XIN, XOUT by crystal (resonator).
When CPU sleeps, the XOUT pin will be in "high" state .
The instruction cycle equal 8 basic clock fc.

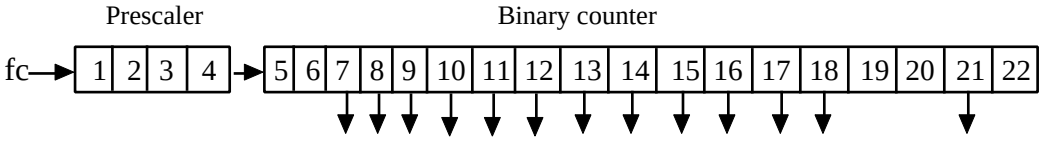
1 instructure cycle = 8 / fc

TIMING GENERATOR AND TIME BASE

The timing generator produces the system clock from basic clock pulse which can be normal mode or slow mode clock.

1 instruction cycle = 8 basic clock pulses

There are 22 stages time base .



When working in the single clock mode, the timebase clock source is come from fc.

Time base provides basic frequency for following function:
1. TBI (time base interrupt) .
2. Timer/counter, internal clock source.
3. Warm-up time for sleep - mode releasing.

TIME BASE INTERRUPT (TBI)

The time base can be used to generate a fixed frequency interrupt . There are 8 kinds of frequencies can be selected by setting "P25"

Single clock mode

P25	3	2	1	0

(initial value 0000)

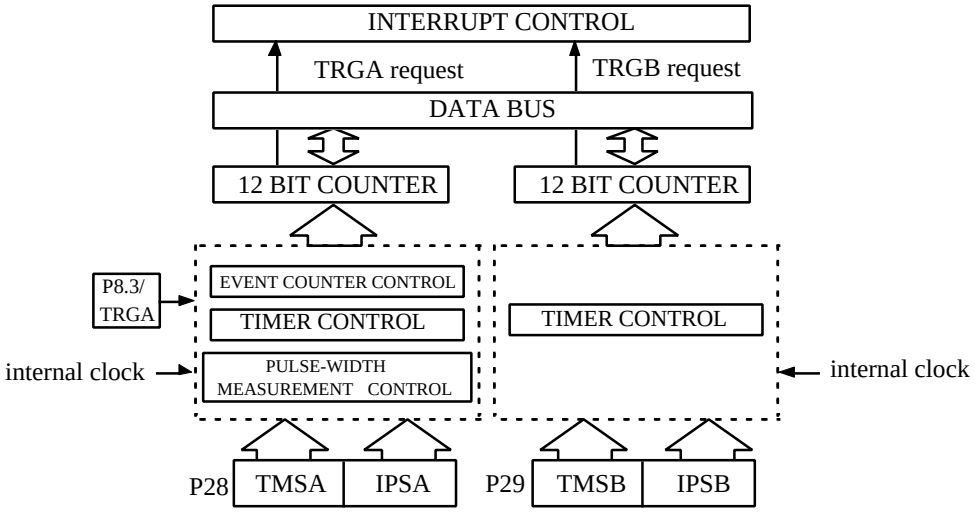
0 0 x x: Interrupt disable
0 1 0 0: Interrupt frequency XIN / 2^{10} Hz
0 1 0 1: Interrupt frequency XIN / 2^{11} Hz
0 1 1 0: Interrupt frequency XIN / 2^{12} Hz
0 1 1 1: Interrupt frequency XIN / 2^{13} Hz
1 1 0 0: Interrupt frequency XIN / 2^{9} Hz
1 1 0 1: Interrupt frequency XIN / 2^{8} Hz
1 1 1 0: Interrupt frequency XIN / 2^{15} Hz
1 1 1 1: Interrupt frequency XIN / 2^{17} Hz
1 0 x x: Reserved

TIMER / COUNTER (TIMERA, TIMERB)

Timer/counters can support user three special functions:
1. Even counter
2. Timer.
3. Pulse-width measurement.

These three functions can be executed by 2 timer/counter independently.
For timerA, the counter data is saved in timer register TAH, TAM, TAL, which user can set counter initial value and read the counter value by instruction "LDATAH(M,L), STATAH(M,L)" and timer register is TBH, TBM, TBL and W/R instruction "LDATBH (M,L), STATBH (M,L)".

The basic structure of timer/counter is composed by two same structure counter, these two counters can be set initial value and send counter value to timer register, P28 and P29 are the command ports for timerA and timer B, user can choose different operation mode and different internal clock rate by setting these two ports. When timer/counter overflow, it will generate a TRGA(B) interrupt request to interrupt control unit.



TIMER/COUNTER CONTROL

P8.3/TRGA is the external timer input for timerA, it is used in event counter and pulse-width measure ment mode.

Timer/counter command port: P28 is the command port for timer/counterA and P29 is for the timer/ counterB.

Port 28

3	2	1	0
TMSA		IPSA	

Initial state: 0000

Port 29

3	2	1	0
TMSB		IPSB	

Initial state: 0000

TIMER/COUNTER MODE SELECTION	
TMSA (B)	Function description
0 0	Stop
0 1	Event counter mode
1 0	Timer mode
1 1	Pulse width measurement mode

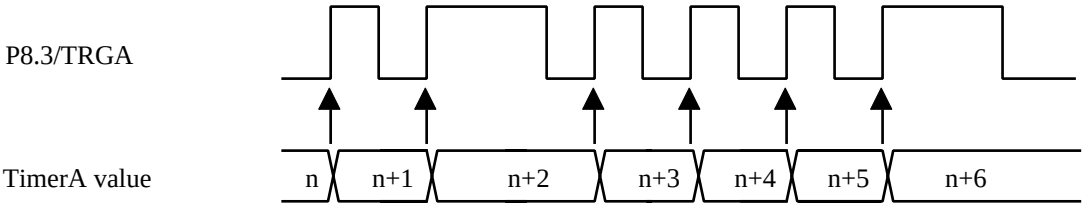
INTERNAL PULSE-RATE SELECTION	
IPSA(B)	Function description
0 0	XIN/2^{10} Hz
0 1	XIN/2^{14} Hz
1 0	XIN/2^{18} Hz
1 1	XIN/2^{22} Hz

TIMER/COUNTER FUNCTION

Timer/counterA can be programmable for timer, event counter and pulse width measurement. Timer/counter B only can be programmable for timer.

EVENT COUNTER MODE

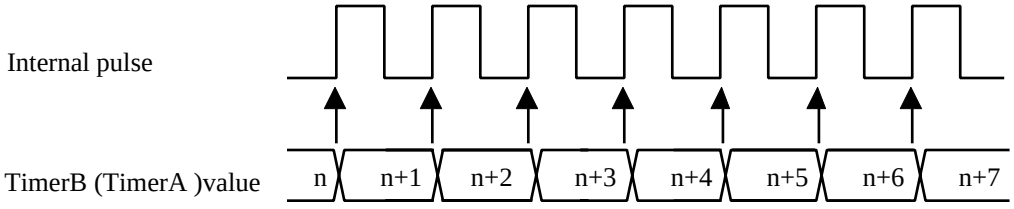
For event counter mode, timer/counter increases one at any rising edge of P8.3/TRGA for timerA. When timerA counts overflow, it will give interrupt control an interrupt request TRGA.



PROGRAM EXAMPLE: Enable timerA with P28.
LDIA #0100B;
OUTA P28; Enable timerA with event counter mode

TIMER MODE

For timer mode ,timer/counter increase one at any rising edge of internal pulse . User can choose 4 kinds of internal pulse rate by setting IPSB for timerB (IPSA for timerA).
When timer/counter counts overflow, TRGB (TRGA) will be generated to interrupt control unit.



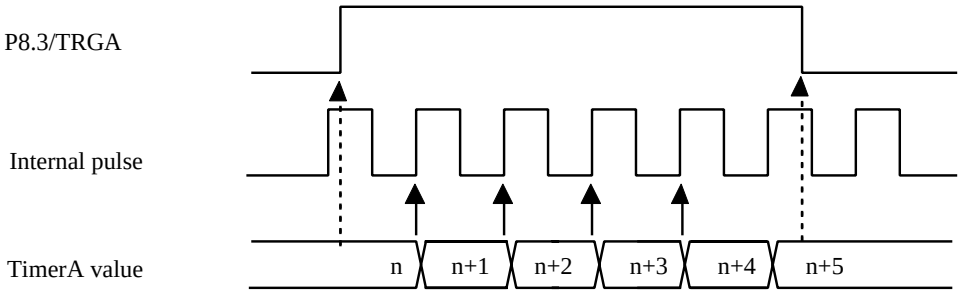
PROGRAM EXAMPLE: To generate TRGA interrupt request after 60 ms with system clock XIN=4MHz
LDIA #0100B;
EXAE; enable mask 2
EICIL 110111B; interrupt latch ←0, enable EI
LDIA #06H;

```
STATAL;
LDIA #01H;
STATAM;
LDIA #0FH;
STATAH;
LDIA #1000B;
OUTA P28; enable timerA with internal pulse rate: XIN/2^10 Hz
```

NOTE: The preset value of timer/counter register is calculated as following procedure.
Internal pulse rate: $XIN/2^{10}$; XIN = 4MHz
The time of timer counter count one = 2^{10} /XIN = 1024/4000=0.256ms
The number of internal pulse to get timer overflow = 60 ms/ 0.256ms = 234.375 = 0EAH
The preset value of timer/counter register = 1000H - 0EAH = 0F16H

PULSE WIDTH MEASUREMENT MODE

For the pulse width measurement mode, the counter only incresed by the rising edge of internal pulse rate as external timer/counter input (P8.3/TRGA), interrupt request will be generated as soon as timer/counter count overflow.



PROGRAM EXAMPLE: Enable timerA by pulse width measurement mode .

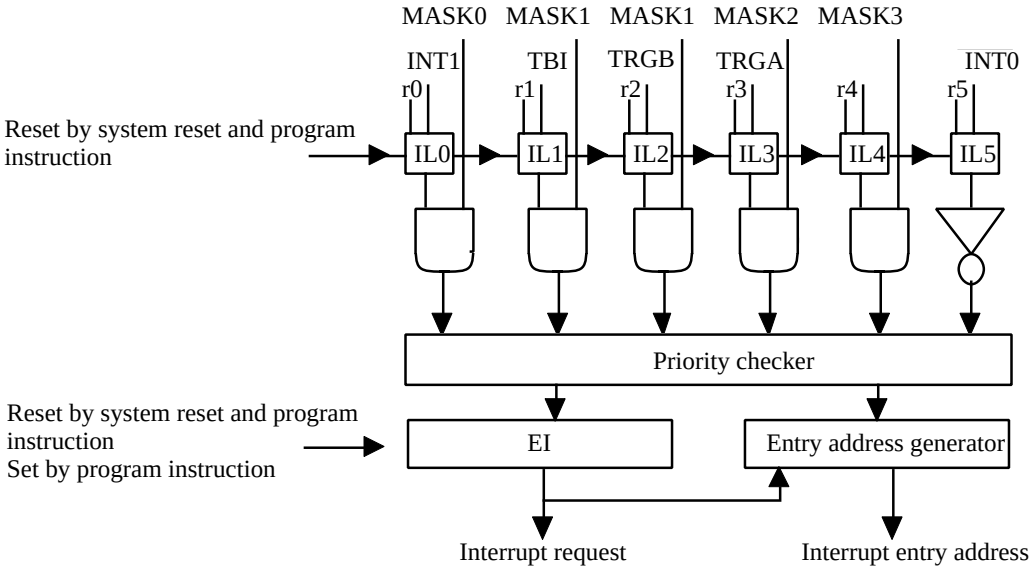
```
LDIA #1100B;
OUTA P28; Enable timerA with pulse width measurement mode.
```

INTERRUPT FUNCTION

There are 5 interrupt sources, 1 external interrupt sources, 4 internal interrupt sources . Multiple interrupts are admitted according the priority .

Type	Interrupt source	Priority	Interrupt Latch	Interrupt Enable condition	Program ROM entry address
External	External interrupt($\overline{INT0}$)	1	IL5	EI=1	002h
Internal	Reserved	2	IL4	EI=1, MASK3=1	004h
Internal	TimerA overflow interrupt (TRGA)	3	IL3	EI=1, MASK2=1	006h
Internal	TimerB overflow interrupt (TRGB)	4	IL2	EI=1, MASK1=1	008h
Internal	Time base interrupt(TBI)	5	IL1		00Ah
External	External interrupt($\overline{INT1}$)	6	IL0	EI=1,MASK0=1	00Ch

INTERRUPT STRUCTURE



Interrupt controller:

IL0-IL5 : Interrupt latch . Hold all interrupt requests from all interrupt sources. ILr can not be set by program, but can be reset by program or system reset, so IL only can decide which interrupt source can be accepted.

MASK0-MASK3 : Except $\overline{\text{INT0}}$,MASK register can promit or inhibit all interrupt sources.

EI : Enable interrupt Flip-Flop can promit or inhibit all interrupt sources, when interrupt happened, EI is cleared to "0" automatically, after RTI instruction happened, EI will be set to "1" again .

Priority checker: Check interrupt priority when multiple interrupts happened.

INTERRUPT FUNCTION

The procedure of interrupt operation:
1. Push PC and all flags to stack.
2. Set interrupt entry address into PC.
3. Set SF= 1.
4. Clear EI to inhibit other interrupts happened.
5. Clear the IL for which interrupt source has already be accepted.
6. To excute interrupt subroutine from the interrupt entry address.
7. CPU accept RTI, restore PC and flags from stack . Set EI to accept other interrupt requests.

PROGRAM EXAMPLE: To enable interrupt of "$\overline{\text{INT0}}$, TRGA"

```
LDIA #1100B;
EXAE; set mask register "1100B"
EICIL 111111B  ; enable interrupt F.F.
```

POWER SAVING FUNCTION (Sleep / Hold functlon)

During sleep and hold condition, CPU holds the system's internal status with a low power consumption, for the sleep mode, the system clock will be stoped in the sleep condition and system need a warm up time for the stability of system clock running after wakeup . In the other way, for the hold mode, the system clock does not stop at all and it does not need a warm-up time any way.
The sleep and hold mode is controlled by Port 16 and released by P0(0..3)/$\overline{\text{WAKEUP0..3}}$ or $\overline{\text{P14.0}}$/ WAKEUPE.

P16 3 2 1 0

WM	**SE**	**SWWT**

initial value :0000

WM	Set wake-up release mode
0	Wake-up in edge release mode
1	Wake-up in level release mode

SWWT	Set wake-up warm-up time
0 0	2^{18} /XIN
0 1	2^{14} /XIN
1 0	2^{16} /XIN
1 1	Hold mode

SE	Enable sleep/hold
0	Reserved
1	Enable sleep / hold rnode

Sleep and hold condition:

1. Osc stop (sleep only) and CPU internal status held .
2. Internal time base clear to "0".
3. CPU internal memory ,flags, register, I/O held original states.
4. Program counter hold the executed address after sleep release.

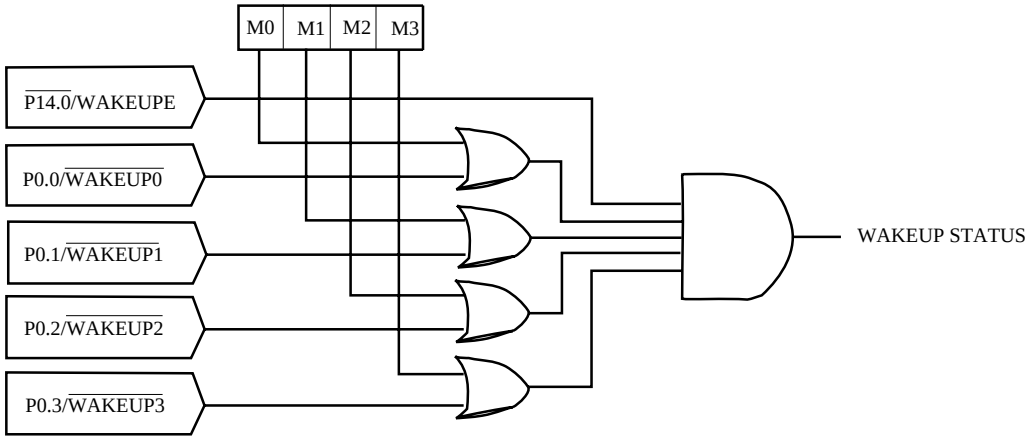
Release condition:

1. Osc start to oscillating.(sleep only).
2. Warm-up time passing (sleep only).
3. According PC to execute the following program.

There are two kinds of sleep/hold release mode .

1. Edge release mode:
   Release sleep/hold condition by the falling edge of any one of P0(0..3)/$\overline{\text{WAKEUP0..3}}$ or by the rising edge of $\overline{\text{P14.0}}$/WAKEUPE.
2. Level release mode:
   Release sleep/hold condition by the low level of any one of P0(0..3)/$\overline{\text{WAKEUP0..3}}$ or by the high level of $\overline{\text{P14.0}}$/WAKEUPE.

   The P14.0 responses the wakeup status and it's function as following :



WAKEUP STATUS = (P0.0+M0)(P0.1+M1)(P0.2+M2)(P0.3+M3)(P14.0)

The M0..3 are the mask options of P0(0..3)/$\overline{\text{WAKEUP0..3}}$ for wakeup function. When M0..3 is equal to 0, the wakeup mask option is enabled. Otherwise, the wakeup mask option is disabled.
If user want to start the sleep/hold operation with the WAKEUP STATUS being high, the sleep/hold function will be released immediately. So, to start the sleep/hold operation in level release mode, the WAKEUP STATUS must be recognized to low by program.

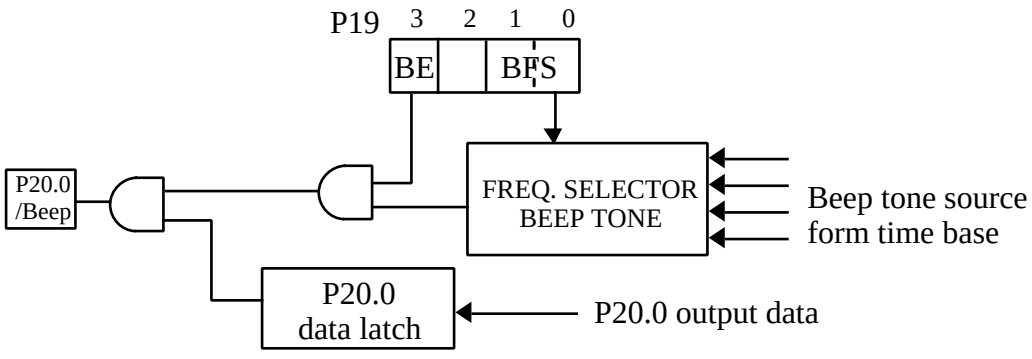
PROGRAM EXAMPLE: Detect the WAKEUP STATUS to start the sleep/hold operation in the level release mode.

```
WAKE:   TFP     P14.0   ;wait the WAKEUP STATUS goes to low
        B       WAKE
        LDIA    #1100B  ;enable sleep operation in level release mode, warm-up time is 2^18/XIN
        OUTA    P16
```

Note : There are 4 independent mask option (M0..3) for wakeup function in EM73491. So,the wakeup function of P0(0..3)/$\overline{\text{WAKEUP0..3}}$ and are enabled or disabled independently.

BEEP TONE GENERATOR

Beep tone generator can generate a series of square wave in audiable frequency .
The beep tone output pin is shared with P20.0 . when beep tone is disable, P20.0 is a general purpose output pin, in the other way, if beep tone enable, P20.0 is the beep tone output pin.



P19 is the command port for beep tone generator. The bit0,1 are called BFS, it is the selection bit for the beep tone output frequency . The bit 3 is called BE, it controls the enable and the disable of beep tone.

Port 19 3 2 1 0

BE		BFS

Initial state : 0x00

BE	Beep tone enable
0	Disable
1	Enable

According to the different clock frequency, XIN, beep tone has following selection .

BFS	Beep tone freq. selection
0 0	XIN / 2^7 KHz(3.75KHz)
0 1	XIN / 2^8 KHz (1.875 KHz)
1 0	XIN / 2^9 KHz (937.5 Hz)
1 1	XIN / 2^{10} KHz (486.75 Hz)

BFS	Beep tone freq. selection	800 KHz	960 KHz
0 0	XIN / 2^8 KHz	3.125 KHz	3.75 KHz
0 1	XIN / 2^9 KHz	1.563 KHz	1.875 KHz
1 0	XIN / 2^{10} KHz	781.3 Hz	937.5 Hz
1 1	XIN / 2^{11} KHz	390.6 Hz	468.8 Hz

XIN = 480 KHz XIN = 800K/960K Hz

BFS	Beep tone freq. selection	3.84 MHz	3.58 MHz	4 MHz
0 0	XIN / 2^{10} KHz	3.75 KHz	3.496 KHz	3.906 KHz
0 1	XIN /2^{11} KHz	1.875 KHz	1.748 KHz	1.953 KHz
1 0	XIN /2^{12} KHz	937.5 KHz	874 Hz	976.5 KHz
1 1	XIN /2^{13} KHz	468.8 KHz	437 Hz	488.2 KHz

XIN = 3.58 M /3.84 M /4 M Hz

PROGRAM EXAMPLE: To send 1.875KHz beep tone with system clock 480 KHz

```
LDIA    #1111B
OUTA    P20     ;  CLEAR BEEP TONE OUTPUT
LDIA    #1001B
OUTA    Pl9     ; SEND BEEP TONE
:
```

To send 874Hz beep tone with system clock 3.58 MHz

```
LDIA    #1111B
OUTA    P20     ; CLEAR BEEP TONE OUTPUT
LDIA    #1010B
OUTA    P19     ; SEND BEEP TONE
:
```

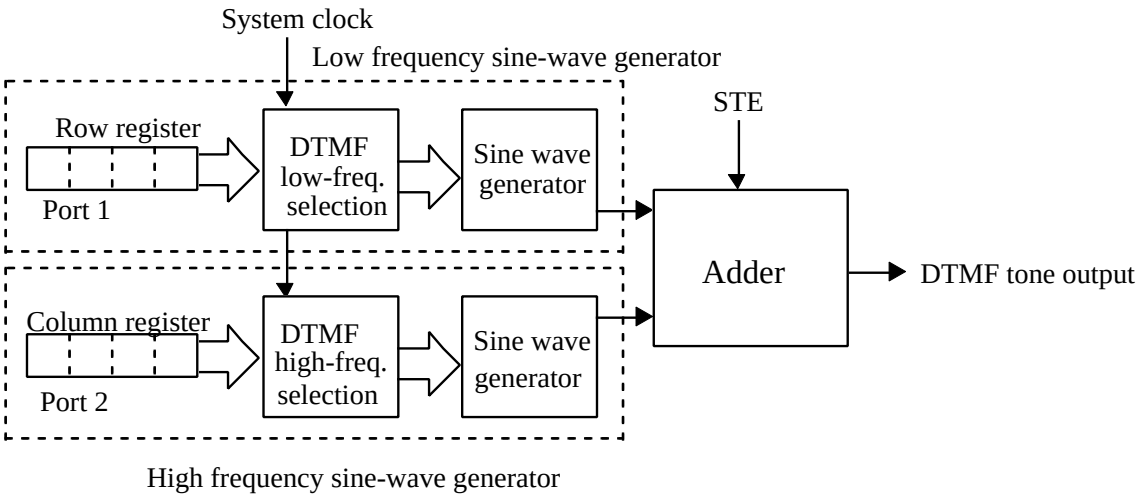
To disable beep tone

```
LDIA    #0000B;
OUTA    P19     ; DISABLE BEEP TONE
:
```

DTMF (Dual Tone Multi Frequency) TONE

Built-in DTMF generator can generate dialing tone signals for the telephone of dialing tone type. There are two kinds of DTMF tone . One is the group of low frequency, the other is the group of high frequency, each group has 4 kinds of frequency, user can choose single tone frequency or dual tone (combined both low and high frequencies) by setting P13,so user can get 16 kinds of DTMF frequency totally .

DTMF generator contains a low frequency sine - wave generator for generating the DTMF signal which selected by row register and a high frequency sine - wave generator for generating the DTMF signal which selected by column register. Adder is controlled by the bit3 of P13(STE) which controls the dual tone or single tone of DTMF signal.



The bit3 of P13 is the control bit of enable dual or signal tone mode of the DTMF signal.

Port13 3 2 1 0

STE				

Initial state:0xxx

STE	Single tone enable
0	Dual tone mode
1	Single tone mode

Port 1 and Port 2 is the register for low frequency (row) and high frequency (column) selection

	STANDARD	Tone output frequency							
ROW		800 KHz	Deviation	3.58 MHz	Deviation	480 KHz 960 KHz 3.84 MHz	Deviation	4 MHz	Deviation
0001	697 Hz	699.3 Hz	+.3%	699.2 Hz	+.3%	697.7 Hz	+.1%	698.0 HZ	+.1%
0010	770 Hz	796.2 Hz	- .1%	771.6 Hz	+.2%	769.2 Hz	- .1%	771.6 Hz	+.2%
0100	852 Hz	854.7 Hz	+.3%	854.0 Hz	+.2%	851.1 Hz	- .1%	850.3 Hz	- .2%
1000	941 Hz	943.4 Hz	+.2%	940.1 Hz	- .1%	937.5 Hz	- .4%	940.0 Hz	- .1%
COLUMN									
0001	1209 Hz	1204.8 Hz	- .3%	1203.0 Hz	- .5%	1212.1 Hz	+.3%	1213.6 Hz	+.4%
0010	1336 Hz	1333.3 Hz	- .2%	1331.8 Hz	- .3%	1333.3 Hz	- .2%	1329.8 Hz	- .5%
0100	1477 Hz	1470.4 Hz	- .4%	1472.0 Hz	- .3%	1481.5 Hz	+.3%	1470.6 Hz	- .4%
1000	1633 Hz	1639.3 Hz	+.4%	1645.2 Hz	+.7%	1621.6 Hz	- .7%	1623.4 Hz	- .6%

DTMF tone is enable by setting Port 1 and Port 2 and bit3 of P13 . When user sets in single tone mode,either row or column register must be set by an effective value, otherwise, if any ineffective value or both register are load effective value, tone output will be disable.
In the other way, for dual tone mode, both row and column register must be set by effective value, then the DTMF tone output will output in dual tone mode, otherwise, if either row or column register is set by ineffective value, the DTMF tone output will be disable and there is no tone output .

PROGRAM EXAMPLE:

(1) Out DTMF tone signal for single tone of 697.7Hz

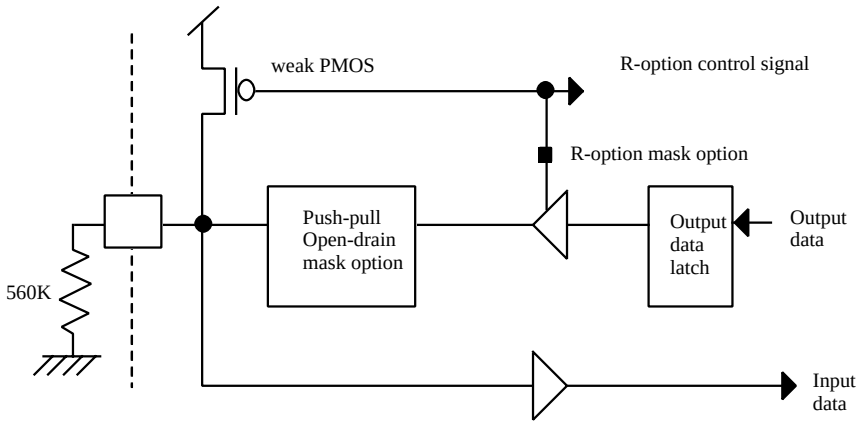
```
OUT #1000b,P13; Set in single tone rnode
OUT #0000b,P2  ; Set an ineffective value for column register
OUT #0001b,P1  ; Enable DTMF signal of 697.7Hz
```

(2) Out DTMF signal for dual tone by OUT12 instruction (5-to-8 bits data conversion)

```
LDL #00h;
LDH #03h;
STDMI #0lh; RAM[30] ← 01h
TFCFC,      CF ← 0
OUT12,      DTMF tone out key "1"
ORG FE0h
DATA 0001 0001b;  corresponding to telephone key "1"
DATA 0010 0001b;  corresponding to telephone key "4"
:
:
```

R OPTION FUNCTION FOR I/O PIN

R-option can provide more function for I/O pins. For example, it can be used in keyboard scan for telecom products. When the key is pressed, this I/O pin is a normal I/O pin. When the key in not pressed and the R-option function is enabled, this I/O pin can use as input pin to detect external status of this pad.



R-option can provide the Hi-Z function for output driver and use the weak PMOS to pull high. When the R-option control signal is high, the weak PMOS turns off and the output data latch is in the normal mode. In the case, there are two mask options (push-pull, open-drain) for this I/O pin. When the R-option control signal is low, the weak PMOS turns on and the output data latch is Hi-Z. In this case, the input data is low when this pad connects a 560KΩ external resistor to GND, and the input data is high when this pad is open outside.

CONTROL OF R OPTION

R-option control signal can be decided by P20.3. When the CPU resets, the initial value is high. in the sleep mode, P20.3 must be set to high and the output pins with R-option function must be clear to low for power saving.

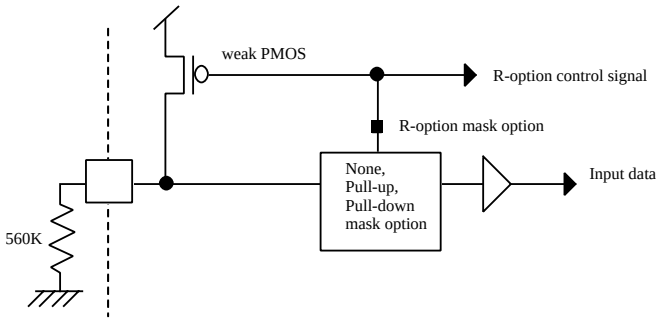
P20 3 2 1 0 Initial value : 1 * * *

0	*	*	*	:	R-option enable
1	*	*	*	:	R-option disable

```
ex   ; Normal program
     ; I/O operation
           :
           :
     ; R-option operation for P5, P6
      OUT     #0Fh, P5      ; let P5 pins pull high
      OUT     #0Fh, P6      ; let P6 pins pull high
     LDIA     #00h
     OUTA     P20           ; enable R-option function
     ; Wait 1 ms for deviation capacitor to charge up
      INA     P5            ; read P5 pins' external status
     STA      05h
     INA      P6            ; read P6 pins' external status
     STA      06h
     LDIA     #08h
     OUTA     P20      ; disable R-option function
           :
     ; Let pins low for sleep mode
      OUT     #00h, P5
      OUT     #00h, P6
     ; Enable sleep mode
           :
```

R OPTION FUNCTION FOR P0 (INPUT ONLY)

There is some difference between I/O pin's and input pin's R-option function. When input pin's R-option mask option is selected, there are two pin's mask options (none, pull-up) can be slected. When the R-option control signal is low, the weak PMOS will turn on and pin's pull-up resistor will turn off (this pin's mask option selects pull-up). In this case, the input data is low when this pad connects a 560KΩ external resistor to GND, and the input data is high when this pad is open outside. When the R-option control signal is high, the weak PMOS will trun off and pin's pull-down resistor will trun on. In sleep mode, user must disable R-option function for power saving. The R-option control signal is also decided by P20.3.




```
ex   ; Normal program
     ; I/O operation
           :
           :
     ; R-option operation for P0
     LDIA     #00h
     OUTA     P20           ; enable R-option function
     ; Wait 1 ms for deviation capacitor to charge up
      INA     P0            ; read P5 pins' external status
     STA      00h
     LDIA     #08h
     OUTA     P20           ; disable R-option function
```

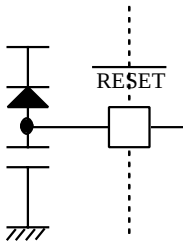
```
        :
        :
; Enable sleep mode
```

RESETTING FUNCTION

When CPU in normal working condition and $\overline{\text{RESET}}$ pin holds in low level for three instruction cycles at least, then CPU begins to initialize the whole internal states, and when $\overline{\text{RESET}}$ pin changes to high level, CPU begins to work in normal condition.
The CPU internal state during reset condition is as following table :

Hardware condition in RESET state	Initial value
Program counter	000h
Status flag	01h
Interrupt enable flip-flop (EI)	00h
MASK0 ,1, 2, 3	00h
Interrupt latch (IL)	00h
P1, 2, 13, 16, 25, 28, 29	00h
P3, 5, 6, 7, 8, 20	0Fh
XIN	Start oscillation

The $\overline{\text{RESET}}$ pin is a hysteresis input pin and it has a pull-up resistor available by mask option.
The simplest RESET circuit is connect $\overline{\text{RESET}}$ pin with a capacitor to V_{SS} and a diode to V_{DD}.



EM73491 I/O PORT DESCRIPTION :

Port	Input function		Output function		Note
0	E	Input port , wakeup function			
1		--	I	DTMF row register	
2		--	I	DTMF column register	
3	E	P3.0, input pin	E	P3.0, output pin	
4		--		--	
5	E	Input port, R-option	E	Output port, R-option	
6	E	Input port, R-option	E	Output port, R-option	
7	E	Input port, R-option	E	Output port, R-option	
8	E	Input port, external interrupt input	E	Output port	no P8.1
9		--		--	
10		--		--	
11		--		--	
12		--		--	
13		--	I	DTMF control register	
14	E	$\overline{\text{P14.0}}$, input pin, wakeup function		--	
15		--		--	
16			I	Sleep mode control register	
17				--	
18				--	
19			I	BEEP control register	
20			I	P20.3 : R-option control register	
21				--	
22				--	
23				--	
24				--	
25			I	Timebase control register	
26				--	
27				--	
28			I	Timer/counter A control register	
29			I	Timer/counter B control register	
30				--	
31				--	

ABSOLUTE MAXIMUM RATINGS

Items	Sym.	Ratings	Conditions
Supply Voltage	V_{DD}	-0.5V to 6V	
Input Voltage	V_{IN}	-0.5V to V_{DD}+0.5V	
Output Voltage	V_O	-0.5V to V_{DD}+0.5V	
Power Dissipation	P_D	300mW	T_{OPR}=50°C
Operating Temperature	T_{OPR}	0°C to 50°C	
Storage Temperature	T_{STG}	-55°C to 125°C	

RECOMMANDED OPERATING CONDITIONS

Items	Sym.	Ratings	Condition
Supply Voltage	V_{DD}	2.4V to 5.5V	
Input Voltage	V_{IH}	0.90xV_{DD} to V_{DD}	
	V_{IL}	0V to 0.10xV_{DD}	
Operating Frequency	F_C	480K to 4.19MHz	XIN,XOUT (crystal osc)

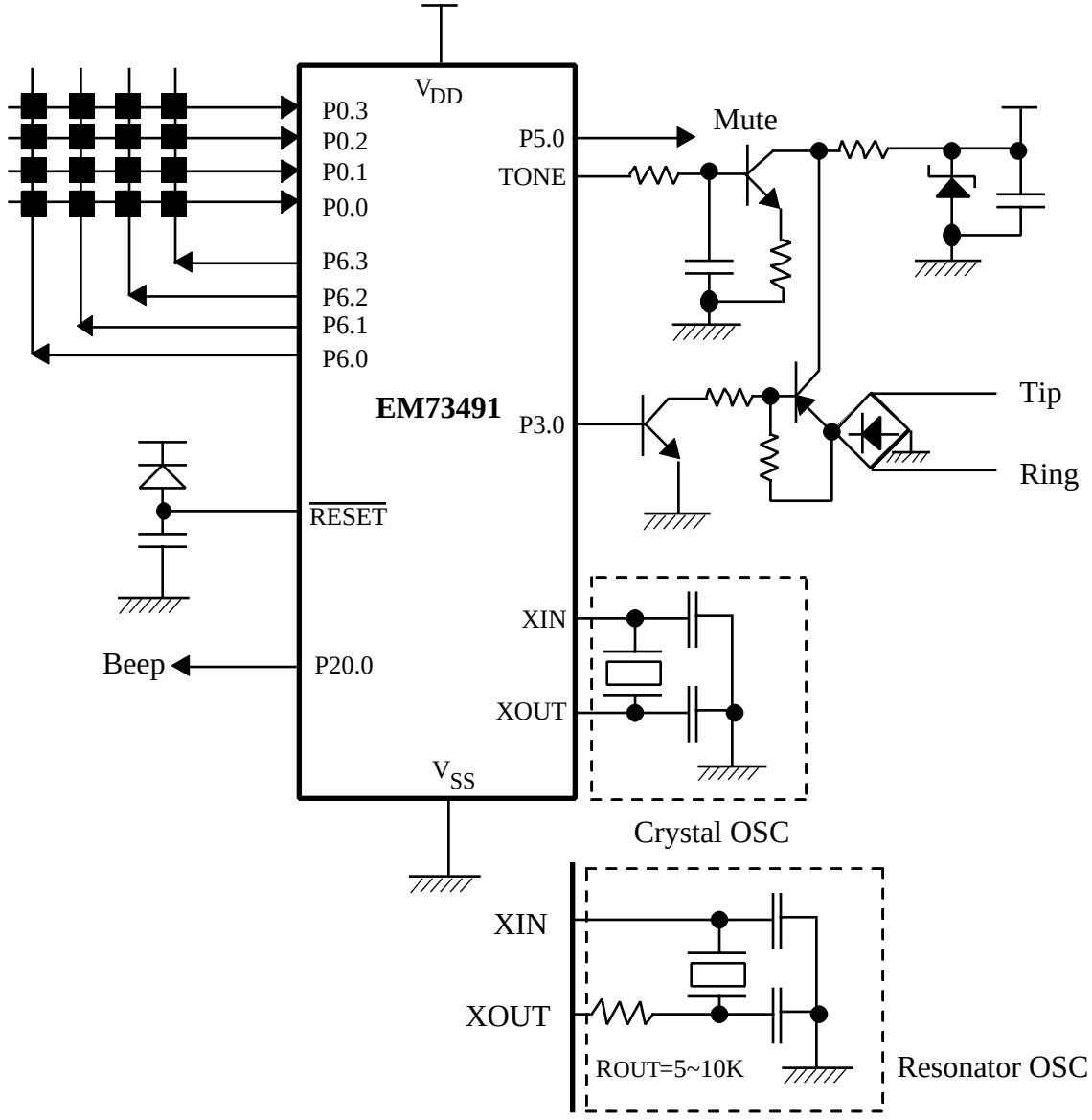
DC ELECTRICAL CHARACTERISTICS (V_{DD}=5±0.5V, V_{SS}=0V, T_{OPR}=25°C)

Parameters	Sym.	Min.	Typ.	Max.	Unit	Conditions
Supply current	I_{DD}	-	4.5	5.5	mA	V_{DD}=5.5V,no load Fc=4.19MHz(crystal osc)
		-	0.1	1	μA	V_{DD}=5.5V, Sleep mode without voltage detect
		-	2.5	-	μA	V_{DD}=2.5V, Sleep mode with voltage detect
Hysteresis voltage	V_{HYS+}	0.50V_{DD}	-	0.75V_{DD}	V	$\overline{RESET}$, P8, P9,P14, P0
	V_{HYS-}	0.20V_{DD}	-	0.40V_{DD}	V	
Input current	I_{IH}	-	-	±1	μA	$\overline{RESET}$, P0, V_{DD}=5.5V,V_{IH}=5.5/0V
		-	-	±1	μA	Open-drain,V_{DD}=5.5V,V_{IH}=5.5/0V
	I_{IL}	-	-	-1	mA	Push-pull,V_{DD}=5.5V ,V_{IL}=0.4V
Output voltage	V_{OH}	2.4	-	-	V	Push-pull,V_{DD}=4.5V,I_{OH}=-250μA
	V_{OL}	-	-	0.4	V	V_{DD}=4.5V,I_{OL}=2mA
Leakage current	I_{LO}	-	-	1	μA	Open-drain, V_{DD}=5.5V, V_O=5.5V
Input resistor	R_{IN}	30	90	150	KΩ	P0
		100	300	450	KΩ	$\overline{RESET}$
R-option threshold point	V_{TH}	-	0.45V_{DD}	-	V	V_{DD}=3~5.5V
R-option weak PMOS resistance	R_{WP}	-	2	-	MΩ	V_{DD}=3~5.5V

DTMF ELECTRICAL CHARACTERISTICS (V_{DD}=5±0.5V, V_{SS}=0V, T_{OPR}=25°C)

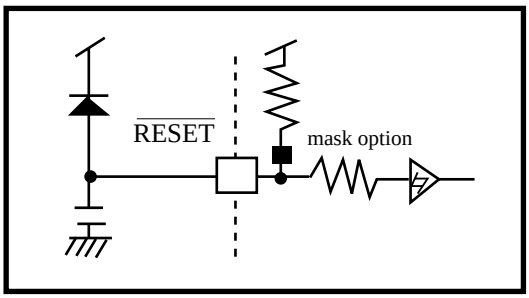
Sym.	Min.	Typ.	Max.	Unit	Conditions
PEHB	1	2	3	db	PEHB=20LOG(COL/ROW)
V_{TONE}	120	150	180	mVrms	RL>10KΩ,V_{DD}=2.7V(ROW)
DIS	-	-	10	%	Distortion
F_{TONE}	-	-	0.7	%	Stability, except error of OSC

APPLICTION CIRCUIT



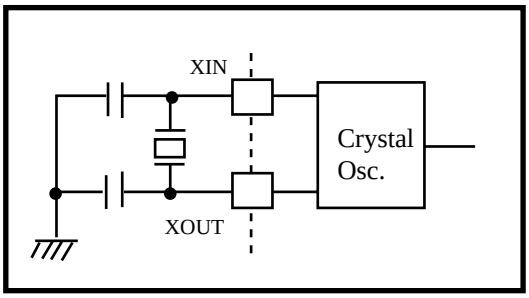
$\overline{\text{RESET}}$ PIN TYPE

TYPE RESET-A



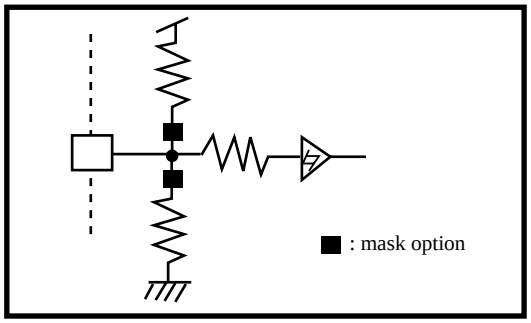
OSCILLATION PIN TYPE

TYPE OSC-A

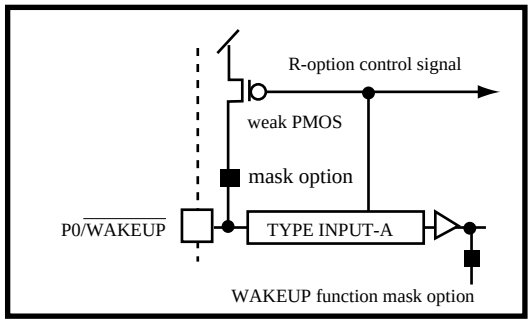


INPUT PIN TYPE

TYPE INPUT-A

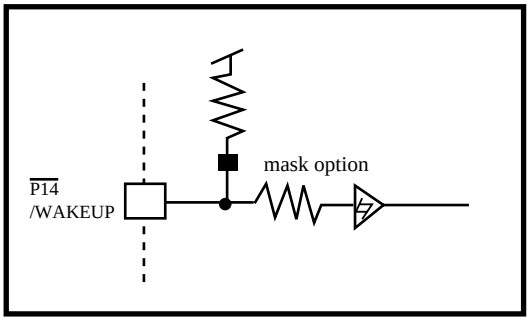


TYPE INPUT-D



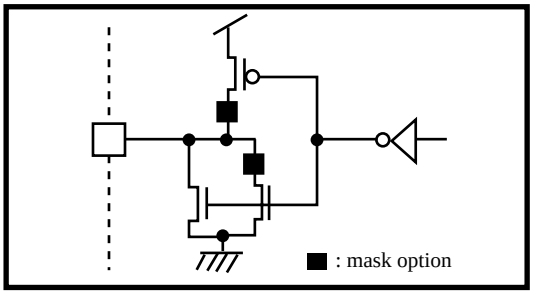
OUTPUT PIN TYPE

TYPE INPUT-F

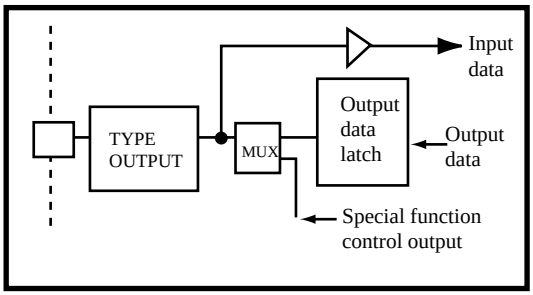


OUTPUT PIN TYPE

TYPE OUTPUT

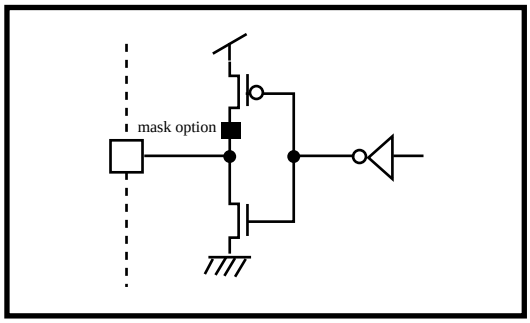


TYPE OUTPUT-B

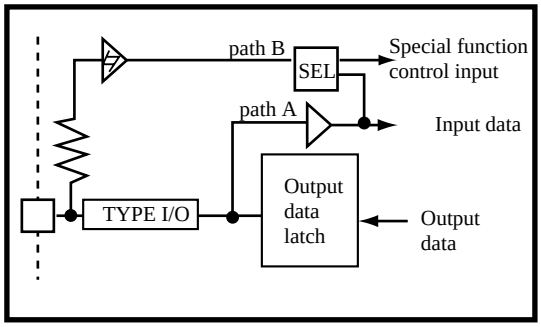


I/O PIN TYPE

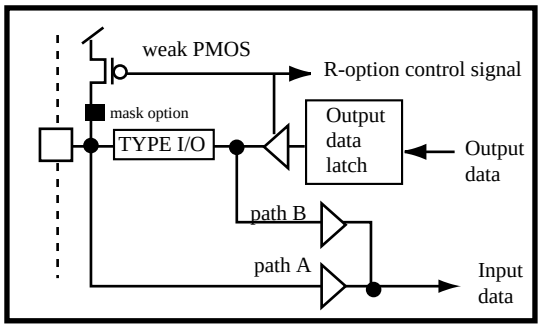
TYPE I/O



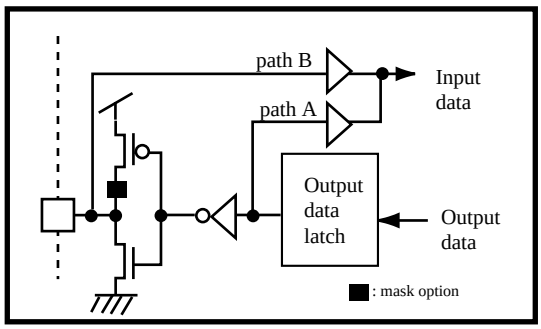
TYPE I/O-C



TYPE I/O-F



TYPE I/O-G



Path A : For set and clear bit of port instructions, data goes through path A from output data latch to CPU.

Path B : For input and test instructions, data from output pin go through path B to CPU and the output data latch will be set to high.

INSTRUCTION TABLE

(1) Data Transfer

Mnemonic	Object code (binary)	Operation description	Byte	Cycle	Flag		
					C	Z	S
LDA x	0110 1010 xxxx xxxx	Acc←RAM[x]	2	2	-	Z	1
LDAM	0101 1010	Acc ←RAM[HL]	1	1	-	Z	1
LDAX	0110 0101	Acc←ROM[DP]$_L$	1	2	-	Z	1
LDAXI	0110 0111	Acc←ROM[DP]$_H$,DP++	1	2	-	Z	1
LDH #k	1001 kkkk	HR←k	1	1	-	-	1
LDHL x	0100 1110 xxxx xx00	LR←RAM[x],HR←RAM[x+1]	2	2	-	-	1
LDIA #k	1101 kkkk	Acc←k	1	1	-	Z	1
LDL #k	1000 kkkk	LR←k	1	1	-	-	1
STA x	0110 1001 xxxx xxxx	RAM[x]←Acc	2	2	-	-	1
STAM	0101 1001	RAM[HL]←Acc	1	1	-	-	1
STAMD	0111 1101	RAM[HL]←Acc, LR--	1	1	-	Z	C
STAMI	0111 1111	RAM[HL]←Acc, LR++	1	1	-	Z	C'
STD #k,y	0100 1000 kkkk yyyy	RAM[y]←k	2	2	-	-	1
STDMI #k	1010 kkkk	RAM[HL]←k, LR++	1	1	-	Z	C'
THA	0111 0110	Acc←HR	1	1	-	Z	1
TLA	0111 0100	Acc←LR	1	1	-	Z	1

(2) Rotate

Mnemonic	Object code (binary)	Operation description	Byte	Cycle	Flag		
					C	Z	S
RLCA	0101 0000	←CF←Acc←	1	1	C	Z	C'
RRCA	0101 0001	→CF→Acc→	1	1	C	Z	C'

(3) Arithmetic operation

Mnemonic	Object code (binary)	Operation description	Byte	Cycle	Flag		
					C	Z	S
ADCAM	0111 0000	Acc←Acc + RAM[HL] + CF	1	1	C	Z	C'
ADD #k,y	0100 1001 kkkk yyyy	RAM[y]←RAM[y] +k	2	2	-	Z	C'
ADDA #k	0110 1110 0101 kkkk	Acc←Acc+k	2	2	-	Z	C'
ADDAM	0111 0001	Acc←Acc + RAM[HL]	1	1	-	Z	C'
ADDH #k	0110 1110 1001 kkkk	HR←HR+k	2	2	-	Z	C'
ADDL #k	0110 1110 0001 kkkk	LR←LR+k	2	2	-	Z	C'
ADDM #k	0110 1110 1101 kkkk	RAM[HL]←RAM[HL] +k	2	2	-	Z	C'
DECA	0101 1100	Acc←Acc-1	1	1	-	Z	C
DECL	0111 1100	LR←LR-1	1	1	-	Z	C
DECM	0101 1101	RAM[HL]←RAM[HL]-1	1	1	-	Z	C
INCA	0101 1110	Acc←Acc + 1	1	1	-	Z	C'

INCL	0111 1110	LR←LR + 1	1	1	-	Z	C'
INCM	0101 1111	RAM[HL]←RAM[HL]+1	1	1	-	Z	C'
SUBA #k	0110 1110 0111 kkkk	Acc←k-Acc	2	2	-	Z	C
SBCAM	0111 0010	Acc←RAM[HL]- Acc - CF'	1	1	C	Z	C
SUBM #k	0110 1110 1111 kkkk	RAM[HL]←k - RAM[HL]	2	2	-	Z	C

(4) Logical operation

Mnemonic	Object code (binary)	Operation description	Byte	Cycle	Flag		
					C	Z	S
ANDA #k	0110 1110 0110 kkkk	Acc←Acc&k	2	2	-	Z	Z'
ANDAM	0111 1011	Acc←Acc & RAM[HL]	1	1	-	Z	Z'
ANDM #k	0110 1110 1110 kkkk	RAM[HL]←RAM[HL]&k	2	2	-	Z	Z'
ORA #k	0110 1110 0100 kkkk	Acc←Acc ¦k	2	2	-	Z	Z'
ORAM	0111 1000	Acc ←Acc ¦ RAM[HL]	1	1	-	Z	Z'
ORM #k	0110 1110 1100 kkkk	RAM[HL]←RAM[HL] ¦k	2	2	-	Z	Z'
XORAM	0111 1001	Acc←Acc^RAM[HL]	1	1	-	Z	Z'

(5) Exchange

Mnemonic	Object code (binary)	Operation description	Byte	Cycle	Flag		
					C	Z	S
EXA x	0110 1000 xxxx xxxx	Acc↔RAM[x]	2	2	-	Z	1
EXAH	0110 0110	Acc↔HR	1	2	-	Z	1
EXAL	0110 0100	Acc↔LR	1	2	-	Z	1
EXAM	0101 1000	Acc↔RAM[HL]	1	1	-	Z	1
EXHL x	0100 1100 xxxx xx00	LR↔RAM[x], HR↔RAM[x+1]	2	2	-	-	1

(6) Branch

Mnemonic	Object code (binary)	Operation description	Byte	Cycle	Flag		
					C	Z	S
SBR a	00aa aaaa	If SF=1 then PC←PC_{11-6}.a_{6-0} else null	1	1	-	-	1
LBR a	1100 aaaa aaaa aaaa	If SF= 1 then PC←a else null	2	2	-	-	1

(7) Compare

Mnemonic	Object code (binary)	Operation description	Byte	Cycle	Flag		
					C	Z	S
CMP #k,y	0100 1011 kkkk yyyy	k-RAM[y]	2	2	C	Z	Z'
CMPA x	0110 1011 xxxx xxxx	RAM[x]-Acc	2	2	C	Z	Z'
CMPAM	0111 0011	RAM[HL] - Acc	1	1	C	Z	Z'
CMPH #k	0110 1110 1011 kkkk	k - HR	2	2	-	Z	C
CMPIA #k	1011 kkkk	k - Acc	1	1	C	Z	Z'
CMPL #k	0110 1110 0011 kkkk	k-LR	2	2	-	Z	C

(8) Bit manipulation

Mnemonic	Object code (binary)	Operation description	Byte	Cycle	Flag		
					C	Z	S
CLM b	1111 00bb	$RAM[HL]_b \leftarrow 0$	1	1	-	-	1
CLP p,b	0110 1101 11bb pppp	$PORT[p]_b \leftarrow 0$	2	2	-	-	1
CLPL	0110 0000	$PORT[LR_{3-2}+4]LR_{1-0} \leftarrow 0$	1	2	-	-	1
CLR y,b	0110 1100 11bb yyyy	$RAM[y]_b \leftarrow 0$	2	2	-	-	1
SEM b	1111 01bb	$RAM[HL]_b \leftarrow 1$	1	1	-	-	1
SEP p,b	0110 1101 01bb pppp	$PORT[p]_b \leftarrow 1$	2	2	-	-	1
SEPL	0110 0010	$PORT[LR_{3-2}+4]LR_{1-0} \leftarrow 1$	1	2	-	-	1
SET y,b	0110 1100 01bb yyyy	$RAM[y]_b \leftarrow 1$	2	2	-	-	1
TF y,b	0110 1100 00bb yyyy	$SF \leftarrow RAM[y]_b'$	2	2	-	-	*
TFA b	1111 10bb	$SF \leftarrow Acc_b'$	1	1	-	-	*
TFM b	1111 11bb	$SF \leftarrow RAM[HL]_b'$	1	1	-	-	*
TFP p,b	0110 1101 00bb pppp	$SF \leftarrow PORT[p]_b'$	2	2	-	-	*
TFPL	0110 0001	$SF \leftarrow PORT[LR_{3-2}+4]LR_{1-0}'$	1	2	-	-	*
TT y,b	0110 1100 10bb yyyy	$SF \leftarrow RAM[y]_b$	2	2	-	-	*
TTP p,b	0110 1101 10bb pppp	$SF \leftarrow PORT[p]_b$	2	2	-	-	*

(9) Subroutine

Mnemonic	Object code (binary)	Operation description	Byte	Cycle	Flag		
					C	Z	S
LCALL a	0100 0aaa aaaa aaaa	STACK[SP]←PC, SP←SP -1, PC←a	2	2	-	-	-
SCALL a	1110 nnnn	STACK[SP]←PC, SP←SP - 1, PC←a, a = 8n + 6 (n ≠0),0086h (n = 0)	1	2	-	-	-
RET	0100 1111	SP←SP + 1, PC←STACK[SP]	1	2	-	-	-

(10) Input/output

Mnemonic	Object code (binary)	Operation description	Byte	Cycle	Flag		
					C	Z	S
INA p	0110 1111 0100 pppp	Acc←PORT[p]	2	2	-	Z	Z'
INM p	0110 1111 1100 pppp	RAM[HL]←PORT[p]	2	2	-	-	Z'
OUT #k,p	0100 1010 kkkk pppp	PORT[p]←k	2	2	-	-	1
OUTA p	0110 1111 000p pppp	PORT[p]←Acc	2	2	-	-	1
OUTM p	0110 1111 100p pppp	PORT[p]←RAM[HL]	2	2	-	-	1
OUT12	0111 0111	PORT[2].PORT[1]← ROM[FE0h+CF.RAM[HL]]	1	2	-	-	1

(11) Flag manipulation

Mnemonic	Object code (binary)	Operation description	Byte	Cycle	Flag		
					C	Z	S
CGF	0101 0111	GF←0	1	1	-	-	1
SGF	0101 0101	GF←1	1	1	-	-	1

TFCFC	0101 0011	SF←CF', CF←0	1	1	0	-	*
TGS	0101 0100	SF←GF	1	1	-	-	*
TTCFS	0101 0010	SF←CF, CF←1	1	1	1	-	*
TZS	0101 1011	SF←ZF	1	1	-	-	*

(12) Interrupt control

Mnemonic	Object code (binary)	Operation description	Byte	Cycle	Flag		
					C	Z	S
CIL r	0110 0011 11rr rrrr	IL←IL & r	2	2	-	-	1
DICIL r	0110 0011 10rr rrrr	EIF←0,IL←IL&r	2	2	-	-	1
EICIL r	0110 0011 01rr rrrr	EIF←1,IL←IL&r	2	2	-	-	1
EXAE	0111 0101	MASK↔Acc	1	1	-	-	1
RTI	0100 1101	SP←SP+1,FLAG.PC ←STACK[SP],EIF ←1	1	2	*	*	*

(13) CPU control

Mnemonic	Object code (binary)	Operation description	Byte	Cycle	Flag		
					C	Z	S
NOP	0101 0110	no operation	1	1	-	-	-

(14) Timer/Counter & Data pointer & Stack pointer control

Mnemonic	Object code (binary)	Operation description	Byte	Cycle	Flag		
					C	Z	S
LDADPL	0110 1010 1111 1100	Acc←$[DP]_L$	2	2	-	Z	1
LDADPM	0110 1010 1111 1101	Acc←$[DP]_M$	2	2	-	Z	1
LDADPH	0110 1010 1111 1110	Acc←$[DP]_H$	2	2	-	Z	1
LDASP	0110 1010 1111 1111	Acc←SP	2	2	-	Z	1
LDATAL	0110 1010 1111 0100	Acc←$[TA]_L$	2	2	-	Z	1
LDATAM	0110 1010 1111 0101	Acc←$[TA]_M$	2	2	-	Z	1
LDATAH	0110 1010 1111 0110	Acc←$[TA]_H$	2	2	-	Z	1
LDATBL	0110 1010 1111 1000	Acc←$[TB]_L$	2	2	-	Z	1
LDATBM	0110 1010 1111 1001	Acc←$[TB]_M$	2	2	-	Z	1
LDATBH	0110 1010 1111 1010	Acc←$[TB]_H$	2	2	-	Z	1
STADPL	0110 1001 1111 1100	$[DP]_L$←Acc	2	2	-	-	1
STADPM	0110 1001 1111 1101	$[DP]_M$←Acc	2	2	-	-	1
STADPH	0110 1001 1111 1110	$[DP]_H$←Acc	2	2	-	-	1
STASP	0110 1001 1111 1111	SP←Acc	2	2	-	-	1
STATAL	0110 1001 1111 0100	$[TA]_L$←Acc	2	2	-	-	1
STATAM	0110 1001 1111 0101	$[TA]_M$←Acc	2	2	-	-	1
STATAH	0110 1001 1111 0110	$[TA]_H$←Acc	2	2	-	-	1
STATBL	0110 1001 1111 1000	$[TB]_L$←Acc	2	2	-	-	1
STATBM	0110 1001 1111 1001	$[TB]_M$←Acc	2	2	-	-	1
STATBH	0110 1001 1111 1010	$[TB]_H$←Acc	2	2	-	-	1

**** SYMBOL DESCRIPTION

Symbol	Description	Symbol	Description
HR	H register	LR	L register
PC	Program counter	DP	Data pointer
SP	Stack pointer	STACK[SP]	Stack specified by SP
A_{CC}	Accumulator	FLAG	All flags
CF	Carry flag	ZF	Zero flag
SF	Status flag	GF	General flag
EI	Enable interrupt register	IL	Interrupt latch
MASK	Interrupt mask	PORT[p]	Port (address : p)
TA	Timer/counter A	TB	Timer/counter B
RAM[HL]	Data memory (address : HL)	RAM[x]	Data memory (address : x)
$ROM[DP]_L$	Low 4-bit of program memory	$ROM[DP]_H$	High 4-bit of program memory
$[DP]_L$	Low 4-bit of data pointer register	$[DP]_M$	Middle 4-bit of data pointer register
$[DP]_H$	High 4-bit of data pointer register	$[TA]_L([TB]_L)$	Low 4-bit of timer/counter A (timer/counter B) register
$[TA]_M([TB]_M)$	Middle 4-bit of timer/counter A (timer/counter B) register	$[TA]_H([TB]_H)$	High 4-bit of timer/counter A (timer/counter B) register
←	Transfer	↔	Exchange
+	Addition	-	Substraction
&	Logic AND		Logic OR
^	Logic XOR	'	Inverse operation
.	Concatenation	#k	4-bit immediate data
x	8-bit RAM address	y	4-bit zero-page address
p	4-bit or 5-bit port address	b	Bit address
r	6-bit interrupt latch	PC_{11-6}	Bit 11 to 6 of program counter
LR_{1-0}	Contents of bit assigned by bit 1 to 0 of LR	a_{5-0}	Bit 5 to 0 of destination address for branch instruction
LR_{3-2}	Bit 3 to 2 of LR		