HT48R12 SPECIFICATION

Features

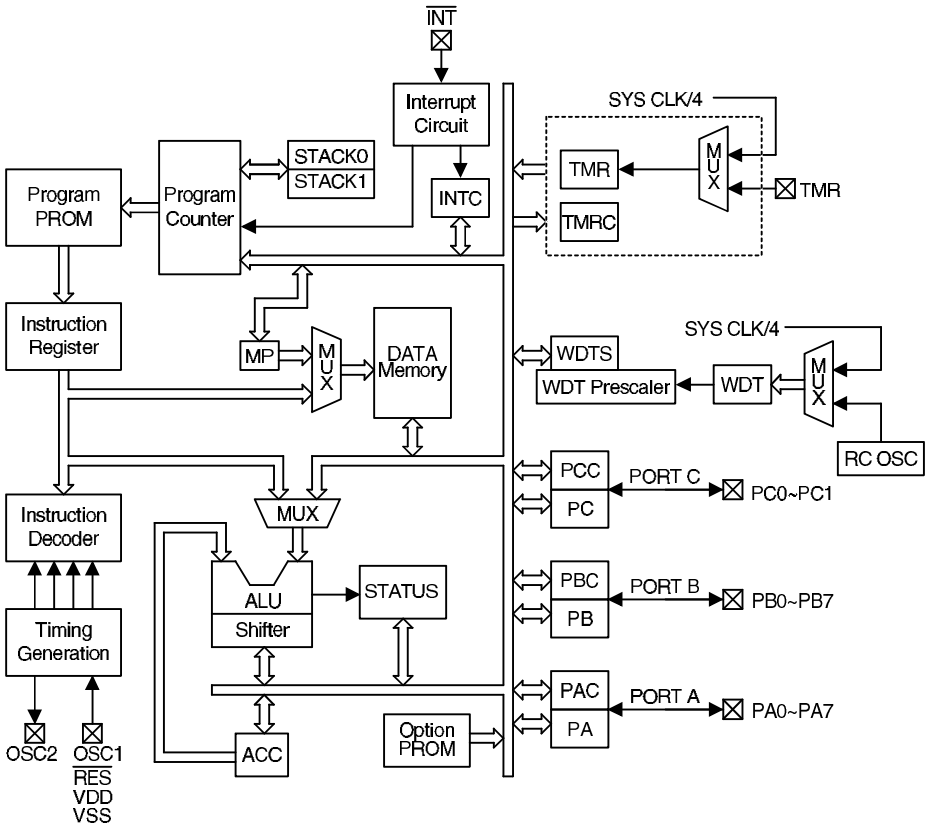
- 18 bidirectional I/O lines
- An interrupt input
- An 8-bit programmable Timer/event counter with overflow interrupt
- On-chip crystal and RC oscillator
- Watch dog timer
- 1K × 14 program memory PROM
- 64 × 8 data memory RAM
- Operating voltage: 3.0V~5.2V
- 63 powerful instructions
- Halt function to reduce power consumption and wake-up feature
- Up to 1μs instruction cycle with 4MHz system clock at V_{DD}=5V
- All instructions in 1 or 2 machine cycles
- 14-bit table read instruction
- Two-level subroutine nesting
- Bit manipulation instruction
- 1 × 14 option memory PROM

General Description

The HT48R12 is an 8-bit high performance RISC-like microcontroller specifically designed for multiple I/O product applications. The device is particularly suitable for use in products such as remote controllers, fan/light controllers, washing machine controllers, scales, toys and various subsystem controllers. A halt feature is included to reduce power consumption.

The program and option PROM can be electrically programmed. The PROM type program and option memory make the HT48R12 suitably use in product development.

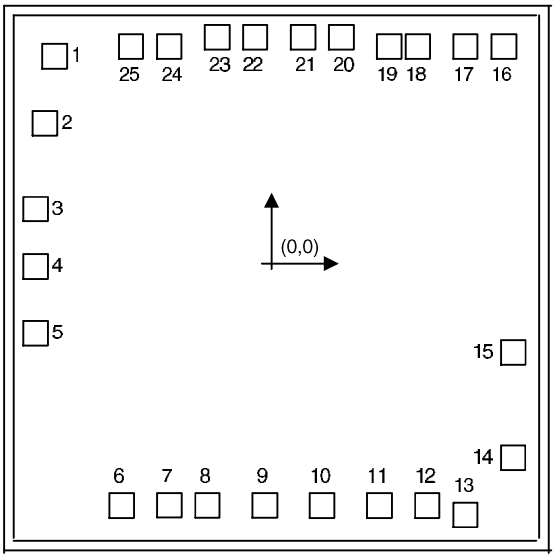
Block Diagram



Pad Description

Pad No.	Pad Name	I/O	ROM Code Option	Function
2, 1 25~24 19~16	PA0~PA7	I/O	Wake-up	Bidirectional 8-bit Input/Output port. Each bit can be configured as a wake-up input by a ROM code option. Software instructions determine the CMOS output or schmitt trigger input with a pull-high resistor.
6~3 23~20	PB0~PB7	I/O	—	Bidirectional 8-bit Input/Output port. Software instructions determine the CMOS output or schmitt trigger input with a pull-high resistor.
7	VSS	—	—	Negative power supply, GND
8	$\overline{\text{INT}}$	I	—	External interrupt schmitt trigger input with a pull-high resistor. Edge triggered is activated on a high to low transition.
9	TMR	I	—	Schmitt trigger input for Timer/Event Counter.
10~11	PC0~PC1	I/O	—	Bdirectional 6-bit Input/Output port. Software instructions determine the CMOS output or schmitt trigger input with a pull high resistor.
12	$\overline{\text{RES}}$	I	—	Schmitt trigger reset input. Active low
13	VDD	—	—	Positive power supply
14 15	OSC1 OSC2	I O	Crystal or RC	OSC1, OSC2 are connected to the RC network or Crystal (determined by ROM code option) for the internal system clock. In the case of RC operation, OSC2 is the output terminal for 1/4 system clock.

Pad Position

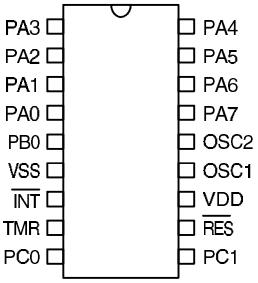


* The IC substrate should be connected to VSS in the PCB layout artwork.
* The TMR pad must be bonded to VDD or VSS if the TMR pad is not used.

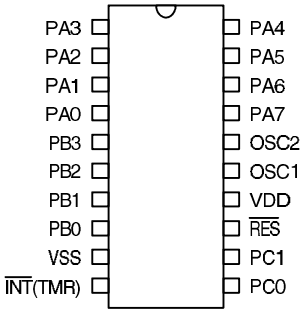
Pad Assignment

Pad No.	Pad Name	Pad No.	Pad Name
1	PA1	14	OSC1
2	PA0	15	OSC2
3	PB3	16	PA7
4	PB2	17	PA6
5	PB1	18	PA5
6	PB0	19	PA4
7	VSS	20	PB7
8	$\overline{\text{INT}}$	21	PB6
9	TMR	22	PB5
10	PC0	23	PB4
11	PC1	24	PA3
12	$\overline{\text{RES}}$	25	PA2
13	VDD		

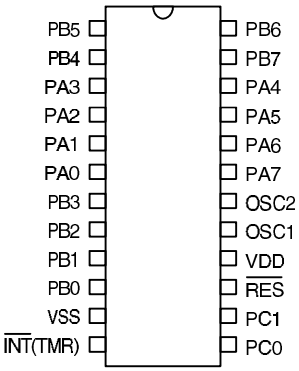
Package & Pin Assignment



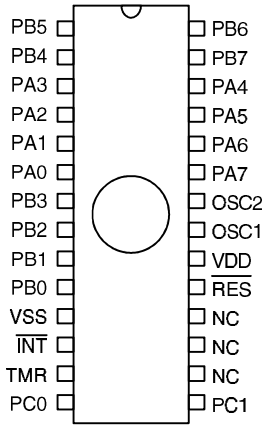
18DIP-F-0



20DIP-F-0



24SDIP-B-0



28CDIP-A-0

Note:

1. For the dice form, the TMR pad must be bonded to VDD or VSS if the TMR pad is not used.
2. The (TMR)$\overline{\text{INT}}$ indicates the TMR pad must be bonded to the $\overline{\text{INT}}$ pin.
3. For packaging of the HT48R12 microcontroller, the OSC2 pin (RC system oscillator only) had better not be bonded out unless necessary.

Absolute Maximum Ratings

Parameter	Symbol	Minimum	Maximum	Unit
Supply Voltage	V_{DD}	–0.3	5.5	V
Input Voltage	V_I	V_{SS}–0.3	V_{DD}+0.3	V
Storage Temperature	T_{STG}	–50	125	°C
Operating Temperature	T_{OP}	–25	70	°C

D.C. Characteristics

(Ta=25°C)

Symbol	Parameter	Test Condition		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
V_{DD}	Operating voltage	—	—	3.0	—	5.2	V
I_{DD1}	Operating current (Crystal OSC)	3V	No load, f_{SYS}=2MHz	—	0.7	1.5	mA
		5V		—	1.8	3	mA
I_{DD2}	Operating current (RC OSC)	3V	No load, f_{SYS}=2MHz	—	0.6	1	mA
		5V		—	1.6	3	mA
I_{STB1}	Stand-by current (WDT enabled)	3V	No load, System HALT	—	—	5	µA
		5V		—	—	10	µA
I_{STB2}	Stand-by current (WDT disabled)	3V	No load, System HALT	—	—	1	µA
		5V		—	—	2	µA
V_{IL1}	Input low voltage for I/O ports	3V	—	0	0.9	0.6	V
		5V	—	0	1.5	1.0	V
V_{IH1}	Input high voltage for I/O ports	3V	—	2.4	2.1	3	V
		5V	—	4.0	3.5	5	V
V_{IL2}	Input low voltage (TMR,$\overline{INT}$)	3V	—	0	0.7	0.6	V
		5V	—	0	1.3	1.0	V
V_{IH2}	Input high voltage (TMR,$\overline{INT}$)	3V	—	2.4	2.3	3	V
		5V	—	4.0	3.7	5	V
V_{IL3}	Input low voltage ($\overline{RES}$)	3V	—	—	1.5	—	V
		5V	—	—	2.5	—	V
V_{IH3}	Input high voltage ($\overline{RES}$)	3V	—	—	2.4	—	V
		5V	—	—	4.0	—	V

Symbol	Parameter	Test Condition		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
I_{OL}	I/O ports sink current	3V	V_{DD}=3V, V_{OL}=0.3V	1.5	2.5	—	mA
		5V	V_{DD}=5V, V_{OL}=0.5V	4	6	—	mA
I_{OH}	I/O ports source current	3V	V_{DD}=3V, V_{OH}=2.7V	–1	–1.5	—	mA
		5V	V_{DD}=5V, V_{OH}=4.5V	–2	–3	—	mA
R_{PH}	Pull-high resistance of I/O ports & $\overline{INT}$	3V	—	40	60	80	KΩ
		5V	—	10	30	50	KΩ

A.C Characteristics

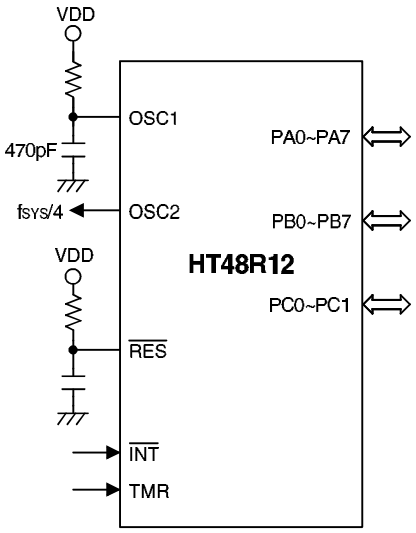
(Ta=25°C)

Symbol	Parameter	Test Condition		Min.	Typ.	Max.	Unit
		V_{DD}	Condition				
f_{SYS1}	System clock (Crystal OSC)	3V	—	400	—	2000	KHz
		5V	—	400	—	4000	KHz
f_{SYS2}	System clock (RC OSC)	3V	—	400	—	2000	KHz
		5V	—	400	—	3000	KHz
f_{TIMER}	Timer I/P frequency (TMR)	3V	—	0	—	2000	KHz
		5V	—	0	—	4000	KHz
t_{WDTOSC}	Watchdog oscillator	3V	—	45	90	180	µs
		5V	—	35	65	130	µs
t_{WDT1}	Watchdog time-out period (RC)	3V	Without WDT prescaler	12	23	45	ms
		5V		9	17	35	ms
t_{WDT2}	Watchdog time-out period (System clock)	—	Without WDT prescaler	—	1024	—	t_{SYS}
t_{RES}	External reset low pulse width	—	—	1	—	—	µs
t_{SST}	System start-up timer period	—	Power-up or wake-up from halt	—	1024	—	t_{SYS}
t_{INT}	Interrupt pulse width	—	—	1	—	—	µs

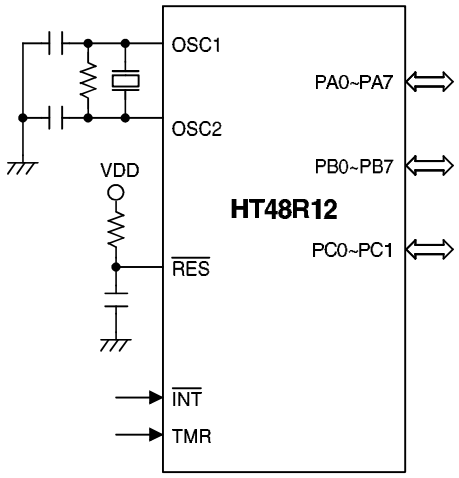
Note: t_{SYS}=1/(f_{SYS})

Application Circuit

RC oscillator for multiple I/O applications



Crystal oscillator for multiple I/O applications



SYSTEM ARCHITECTURE

Execution Flow

The system clock for the HT48R12 is derived from either a crystal or an RC oscillator. The system clock is internally divided into four non-overlapping clocks. One instruction cycle consists of four system clock cycles.

Instruction fetching and execution are pipelined in such a way that a fetch takes an instruction cycle while decoding and execution take the next instruction cycle. However, the pipelining scheme causes each instruction to effectively execute in a cycle. If an instruction changes the program counter, two cycles are required to complete the instruction.

Program Counter - PC

The 10-bit program counter (PC) controls the sequence in which the instructions stored in program PROM are executed and its contents specify a maximum of 1024 addresses.

After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by one. The program counter then points to the memory word containing the next instruction code.

When executing a jump instruction, conditional skip execution, loading PCL register, subroutine call, initial reset, internal interrupt, external interrupt or return from subroutine, the PC manipulates the program transfer by loading the address corresponding to each instruction.

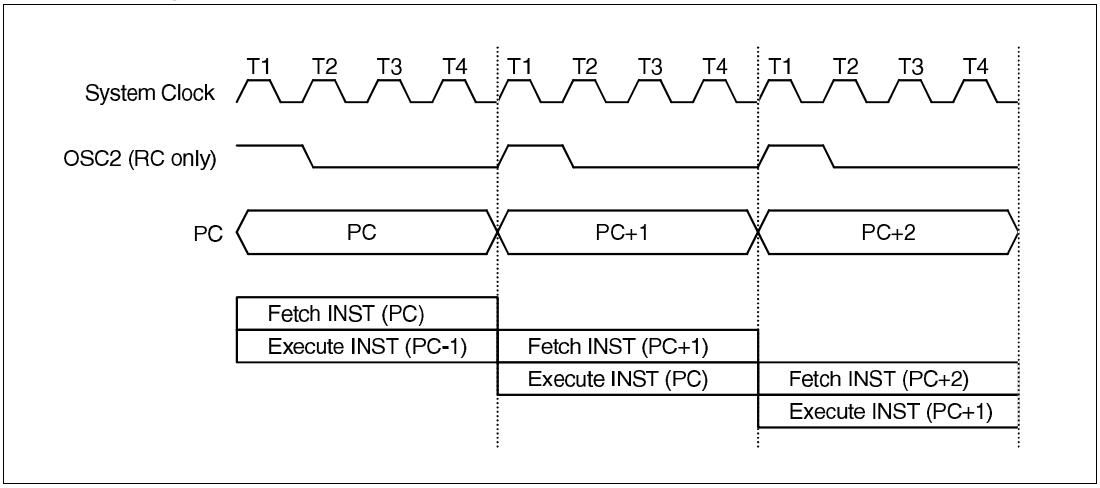
The conditional skip is activated by instructions. Once the condition is met, the next instruction, fetched during the current instruction execution, is discarded and a dummy cycle replaces it to get the proper instruction. Otherwise proceed with the next instruction.

The lower byte of the program counter (PCL) is a readable and writable register (06H). Moving data into the PCL performs a short jump. The destination will be within 256 locations.

Once the control transfer takes place, the execution suffers from having an additional dummy cycle.

Program Memory - PROM

The program memory is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized with 1024 × 14 bits, addressed by the program counter and table pointer.



Execution Flow

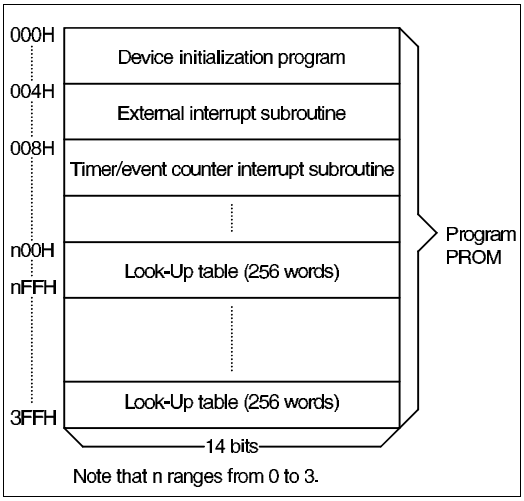
Certain locations in the program memory are reserved for special usages:

Location 000H:
This area is reserved for program initialization. After chip reset, the program always begins execution at location 000H.

Location 004H:
This area is reserved for the external interrupt service program. If the $\overline{INT}$ input pin is activated, the interrupt is enabled and the stack is not full, the program begins execution at location 004H.

Location 008H:
This area is reserved for the Timer/event Counter interrupt service program. If a timer interrupt results from a Timer/event Counter overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 008H.

Table location:
Any location in the PROM space can be used as look-up tables. The instructions "TABRDC [m]" (the current page, 1 page=256 words) and "TABRDL [m]" (the last page) transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH (08H). Only the destination of the lower-order byte in the table is well-defined, the other bits of the table word are transfered to the lower portion of TBLH, and the remaining 2 bits are read as "0". The Table Higher-order byte register (TBLH) is read only. The table pointer (TBLP) is a read/write register (07H), which indicates the table location. Before accessing the table, the location must be placed in TBLP. The TBLH is read only and cannot be restored. If the main routine and the ISR (Interrupt Service Routine) both employ the table read instruction, the contents of the TBLH in the main routine are likely to be changed by the table read instruction used in the ISR. Errors



Program Memory

Mode	Program Counter									
	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial reset	0	0	0	0	0	0	0	0	0	0
External interrupt	0	0	0	0	0	0	0	1	0	0
Timer/event Counter overflow	0	0	0	0	0	0	1	0	0	0
Skip										
Loading PCL	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, call branch	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from subroutine	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

Notes: *9~*0: Bits of Program Counter
S9~S0: Bits of Stack Register
#9~#0: Bits of Instruction Code
@7~@0: Bits of PCL

are thus brought about. Given this, using the table read instruction in the main routine and the ISR simultaneously should be avoided. However, if the table read instruction has to be applied in both the main routine and the ISR, the interrupt(s) is supposed to be disabled prior to the table read instruction. It (They) will not be enabled until the TBLH has been backup. All table related instructions require 2 cycles to complete the operation. These areas may function as normal program memory depending upon the user's requirements.

Stack Register - STACK

This is a special part of memory which is used to save the contents of the program counter (PC) only. The stack is organized into 2 levels and is neither part of the data nor program space, and is neither readable nor writable. The activated level is indexed by the stack pointer (SP) and is neither readable nor writeable. At a subroutine call or interrupt acknowledgement, the contents of the program counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction (RET or RETI), the program counter is restored to its previous value from the stack. After a chip reset, the SP will point to the top of the stack.

If the stack is full and a non-masked interrupt takes place, the interrupt request flag will be recorded but the acknowledgement will be inhibited. When the stack pointer is decremented (by RET or RETI), the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. In a similar case, if the stack is full and a "CALL" is subsequently executed, stack overflow occurs and the first entry will be lost (only the most recent two return addresses are stored).

Data Memory - RAM

The data memory is designed with 81×8 bits. The data memory is divided into two functional groups: special function registers and general purpose data memory (64×8). Most of them are read/write, but some are read only.

The special function registers include the Indirect addressing register (00H), Timer/event Counter (TMR;0DH), Timer/event Counter control register (TMRC;0EH), Program counter lower-order byte register (PCL;06H), Memory pointer register (MP;01H), Accumulator (ACC;05H), Table pointer (TBLP;07H), Table higher-order byte register (TBLH;08H), Status register (STATUS;0AH), Interrupt control register (INTC;0BH), Watch Dog Timer option setting register (WDTS;09H), I/O registers (PA;12H, PB;14H, PC;16H) and I/O control registers (PAC;13H, PBC;15H, PCC;17H). The remaining space before the 40H is reserved for future expanded usages and reading these locations will get "00H". The general purpose data memory, addressed from 40H to 7FH, is used for data and control information under instruction commands.

All of the data memory areas can handle arithmetic, logic, increment, decrement and rotate operations directly. Except for some dedicated bits, each bit in the data memory can be set and

Instruction(s)	Table Location									
	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
TABRDC [m]	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Notes: *9~*0: Bits of table location

@7~@0: Bits of table pointer

P9~P8: Bits of current program Counter

reset by “SET [m].i” and “CLR [m].i”. They are also indirectly accessable through Memory pointer register (MP;01H).

Indirect Addressing Register

Location 00H is an indirect addressing register that is not physically implemented. Any read/write operation of [00H] accesses data memory pointed to by MP (01H). Reading location 00H itself indirectly will return the result 00H. Writing indirectly results in no operation.

The memory pointer register MP (01H) is a 7-bit register. The bit 7 of MP is undefined and reading will return the result “1”. Any writing operation to MP will only transfer the lower 7-bit data to MP.

Accumulator

The accumulator closely relates to ALU operations. It is also mapped to location 05H of the data memory and is the one which can operate with immediate data. The data movement between two data memories must get through the accumulator.

Arithmetic and Logic Unit - ALU

This circuit performs 8-bit arithmetic and logic operations. The ALU provides the following functions:

Arithmetic operations (ADD, ADC, SUB, SBC, DAA)

Logic operations (AND, OR, XOR, CPL)

Rotation (RL, RR, RLC, RRC)

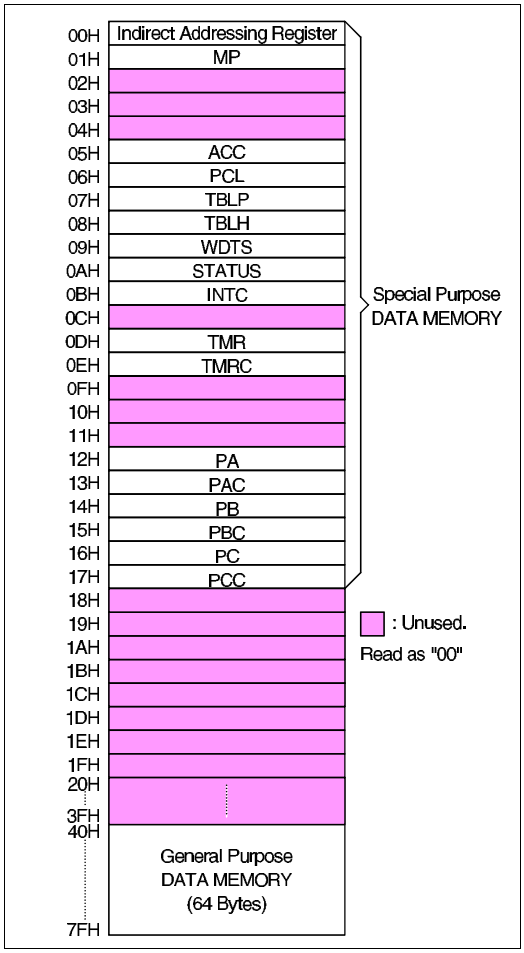
Increment & Decrement (INC, DEC)

Branch decision (SZ, SNZ, SIZ, SDZ ....)

The ALU not only saves the results of a data operation but also changes the status register.

Status Register - STATUS

This 8-bit register (0AH) contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PD), and watch dog time-out flag (TO). It also records the status information and controls the operation sequence.



RAM Mapping

With the exception of the TO and PD flags, bits in the status register can be altered by instructions like any other register. Any data written into the status register will not change the TO or PD flag. In addition operations related to the status register may give different results from those intended. The TO flag can be affected only by system power-up, a WDT time-out or executing the “CLR WDT” or “HALT” instruction. The PD flag can be affected only by executing the “HALT” or “CLR WDT” inctruction or a system power-up.

The Z, OV, AC and C flags generally reflect the statuses of the latest operations.

In addition, on entering the interrupt sequence or executing the subroutine call, the status register will not be pushed onto the stack automatically. If the contents of status are important and the subroutine can corrupt the status register, the programmer must take precautions to save it properly.

Interrupt

The HT48R12 provides an external interrupt and internal Timer/event Counter interrupts. The Interrupt Control Register (INTC;0BH) contains the interrupt control bits to set the enable/disable and the interrupt request flags.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked (by clearing the EMI bit). This scheme may prevent any further interrupt nesting. Other interrupt requests may happen during this interval but only the interrupt request flag is recorded. If a certain interrupt requires servicing within the service routine, the programmer may set the EMI bit and the corresponding bit of INTC to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the SP is decremented. If immediate service is desired, the stack must be prevented from becoming full.

All these kinds of interrupts have the wake-up capability. As an interrupt is serviced, a control transfer occurs by pushing the program counter onto the stack and then branching to subroutines at specified location(s) in the program memory. Only the program counter is pushed onto the stack. If the contents of the register or status register (STATUS) are altered by the interrupt service program which corrupts the desired control sequence, the programmer must save these contents first.

External interrupts are triggered by a high to low transition of $\overline{\text{INT}}$ and the related interrupt request flag (EIF; bit 4 of INTC) will be set. When the interrupt is enabled, the stack is not full and the external interrupt is active, a subroutine call to location 04H will occur. The interrupt request flag (EIF) and EMI bits will be cleared to disable other interrupts.

The internal Timer/event Counter interrupt is initialized by setting the Timer/event Counter

Labels	Bits	Function
C	0	C is set if the operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. Also it is affected by a rotate through carry instruction.
AC	1	AC is set if the operation results in a carry out of the low nibbles in addition or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
Z	2	Z is set if the result of an arithmetic or logic operation is zero; otherwise Z is cleared.
OV	3	OV is set if the operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
PD	4	PD is cleared by system power-up or executing the “CLR WDT” instruction. PD is set by executing the “HALT” instruction.
TO	5	TO is cleared by system power-up or executing the “CLR WDT” or “HALT” instruction. TO is set by a WDT time-out.
–	6	Undefined, read as “0”.
–	7	Undefined, read as “0”.

STATUS Register

interrupt request flag (TF; bit 5 of INTC), caused by a timer overflow. When the interrupt is enabled, the stack is not full and the TF bit is set, a subroutine call to location 08H will occur. The related interrupt request flag (TF) will be reset and the EMI bit cleared to disable further interrupts.

During the execution of an interrupt subroutine, other interrupt acknowledgements are held until the "RETI" instruction is executed or the EMI bit and the related interrupt control bit are set to 1 (of course, if the stack is not full). To return from the interrupt subroutine, "RET" or "RETI" may be invoked. RETI will set the EMI bit to enable an interrupt service, but RET will not.

No.	Interrupt Source	Priority	Vector
a	External interrupt	1	04H
b	Timer/event Counter overflow	2	08H

Interrupts, occuring in the interval between the rising edges of two consecutive T2 pulses, will be serviced on the latter of the two T2 pulses, if the corresponding interrupts are enabled. In the case of simultaneous requests the priorities in the following table apply. These can be masked by resetting the EMI bit.

The Timer/event Counter interrupt request flag (TF), External interrupt request flag (EIF), Enable Timer/event Counter bit (ETI), Enable external interrupt bit (EEI) and Enable master interrupt bit (EMI) constitute an interrupt control register (INTC) which is located at 0BH in the data memory. EMI, EEI, ETI are used to control the enabling/disabling of interrupts. These bits prevent the requested interrupt from being serviced. Once the interrupt request flags (TF, EIF) are set, they will remain in the INTC register until the interrupts are serviced or cleared by a software instruction.

It is suggested that a program does not use the "CALL subroutine" within the interrupt subroutine. It because interrupts often occur in an unpredictable manner or need to be serviced immediately in some applications. If only one stack is left and enabling the interrupt is not well controlled, the original control sequence will be damaged once the "CALL" operates in the interrupt subroutine.

Register	Bit No.	Label	Function
INTC (0BH)	0	EMI	Controls the Master (Global) interrupt (1=enabled; 0=disabled)
	1	EEI	Controls the external interrupt (1=enabled; 0=disabled)
	2	ETI	Controls the Timer/event Counter interrupt (1=enabled; 0=disabled)
	3	–	Unused bit, read as "0".
	4	EIF	External interrupt request flag (1=active; 0=inactive)
	5	TF	Internal Timer/event Counter request flag (1=active; 0=inactive)
	6	–	Unused bit, read as "0".
	7	–	Unused bit, read as "0".

INTC Register

Oscillator Configuration

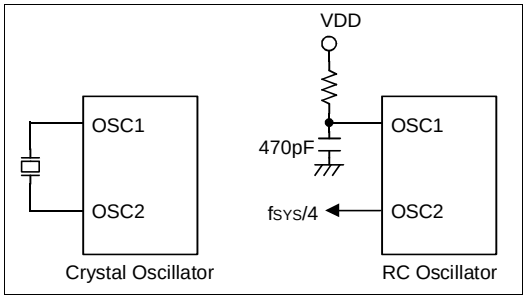
There are 2 oscillator circuits in the HT48R12.

Both of them are designed for system clocks, namely the RC oscillator and the Crystal oscillator, which are determined by the ROM code option. No matter what oscillator type is selected, the signal provides the system clock. The HALT mode stops the system oscillator and resists the external signal to conserve power.

If an RC oscillator is used, an external resistor between OSC1 and VDD is required and the resistance must range from 51KΩ to 1MΩ. The system clock, divided by 4, is available on OSC2, which can be used to synchronize external logic. The RC oscillator provides the most cost effective solution. However, the frequency of oscillation may vary with VDD, temperatures and the chip itself due to process variations. It is, therefore, not suitable for timing sensitive operations where an accurate oscillator frequency is desired.

If the Crystal oscillator is used, a crystal across OSC1 and OSC2 is needed to provide the feedback and phase shift required for the oscillator, and no other external components are demanded. Instead of a crystal, the resonator can also be connected between OSC1 and OSC2 to get a frequency reference, but two external capacitors in OSC1 and OSC2 are required.

The WDT oscillator is a free running on-chip RC oscillator, and no external components are required. Even if the system enters the power down mode, the system clock is stopped, but the WDT oscillator still works with a period of approximately 78μs. The WDT oscillator can be disabled by ROM code option to conserve power.

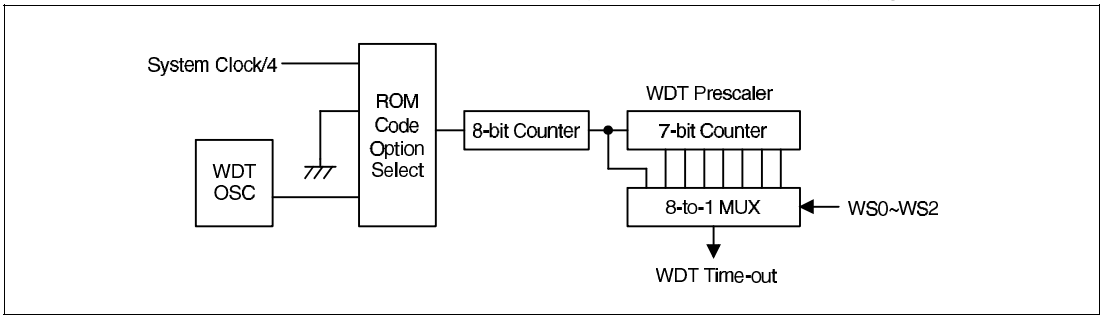


System Oscillator

Watch Dog Timer - WDT

The clock source of WDT is implemented by an dedicated RC oscillator (WDT oscillator) or instruction clock (system clock divided by 4), decided by ROM code option. This timer is designed to prevent a software malfunction or sequence from jumping to an unknown location with unpredictable results. The Watch Dog Timer can be disabled by a ROM code option. If the Watch Dog Timer is disabled, all the executions related to the WDT result in no operation.

Once the internal WDT oscillator (RC oscillator with a period of 78μs normally) is selected, it is first divided by 256 (8-stages) to get the nominal time-out period of approximately 20ms. This time-out period may vary with temperatures, VDD and process variations. By invoking the WDT prescaler, longer time-out periods can be realized. Writing data to WS2, WS1, WS0 (bit 2,1,0 of the WDTS) can give different time-



Watch Dog Timer

out periods. If WS2, WS1, and WS0 all equal to 1, the division ratio is up to 1:128, and the maximum time-out period is 2.6 seconds.

If the WDT oscillator is disabled, the WDT clock may still come from the instruction clock and operate in the same manner except that in the HALT state the WDT may stop counting and lose its protecting purpose. In this situation the logic can only be restarted by external logic. The high nibble and bit 3 of the WDTS are reserved for user's defined flags, and the programmer may use these flags to indicate some specified status.

If the device operates in a noisy enviroment, using the on-chip RC oscillator (WDT OSC) is strongly recommended, since the HALT will stop the system clock.

WS2	WS1	WS0	Division Ratio
0	0	0	1:1
0	0	1	1:2
0	1	0	1:4
0	1	1	1:8
1	0	0	1:16
1	0	1	1:32
1	1	0	1:64
1	1	1	1:128

WDTS Register

The overflow of WDT under normal operation will initialize "chip reset" and set the status bit "TO". But in the HALT mode, the overflow will initialize a "warm reset", and only the PC and SP are reset to zero. To clear the contents of WDT (including the WDT prescaler), 3 methods are adopted; external reset (a low level to $\overline{RES}$), software instruction(s) and a "HALT" instruction. The software instruction(s) include "CLR WDT" and the other set – "CLR WDT1" and "CLR WDT2". Of these two types of instruction, only one can be active depending on the ROM code option – "CLR WDT times selection option". If the "CLR WDT" is selected (ie. CLRWDT times equal one), any execution of the "CLR WDT" instruction will clear the WDT. In the case that "CLR WDT1" and "CLR WDT2" are chosen (ie. CLRWDT times equal two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip as a result of time-out.

Power Down Operation - HALT

The HALT mode is initialized by the "HALT" instruction and results in the following...

- The system oscillator will be turned off but the WDT oscillator keeps running (If the WDT oscillator is selected).
- The contents of the on–chip RAM and registers remain unchanged.
- WDT and WDT prescaler will be cleared and recounted again (If the clock of WDT is from the WDT oscillator).
- All of the I/O ports maintain their original status.
- The PD flag is set and the TO flag is cleared.

The system can leave the HALT mode by means of an external reset, an interrupt, an external falling edge signal on port A or a WDT overflow. An external reset causes a device initialization and the WDT overflow performs a "warm reset". After the TO and PD flags are examined, the reason for chip reset can be determined. The PD flag is cleared by system power-up or executing the "CLR WDT" instruction and is set when executing the "HALT" instruction. The TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the PC and SP; the others keep their original status.

The port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each bit in port A can be independently selected to wake up the device by the ROM code option. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If it is awakening from an interrupt, two sequences may happen. If the related interrupt(s) is (are) disabled or the interrupt(s) is enabled but the stack is full, the program will resume execution at the next instruction. If the interrupt is enabled and the stack is not full,

the regular interrupt response takes place. If an interrupt request flag is set to "1" before entering the HALT mode, the wake-up function of the related interrupt will be disabled.

Once the wake-up event(s) occurs, it takes 1024 t_{SYS} (system clock period) to resume normal operation. In other words, a dummy period will be inserted after wake-up. If the wake-up results from an interrupt acknowledgement, the actual interrupt subroutine execution will be delayed by one or more cycles. If the wake-up results in the next instruction execution, this will be executed immediately after the dummy period is finished.

To minimize power consumption, all the I/O pins should be carefully managed before entering the HALT status.

Reset

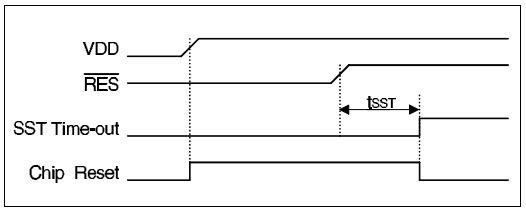
There are 3 ways in which a reset can occur:

- $\overline{\text{RES}}$ reset during normal operation
- $\overline{\text{RES}}$ reset during HALT
- WDT time-out reset during normal operation

The WDT time-out during HALT is different from other chip reset conditions, since it can perform a "warm reset" that resets only PC and SP, leaving the other circuits keep their state. Some registers remain unchanged during other reset conditions. Most registers are reset to the "initial condition" when the reset conditions are met. By examining the PD flag and TO flag, the program can distinguish between different "chip resets".

TO	PD	RESET Conditions
0	0	$\overline{\text{RES}}$ reset during power-up
u	u	$\overline{\text{RES}}$ reset during normal operation
0	1	$\overline{\text{RES}}$ wake-up HALT
1	u	WDT time-out during normal operation
1	1	WDT wake-up HALT

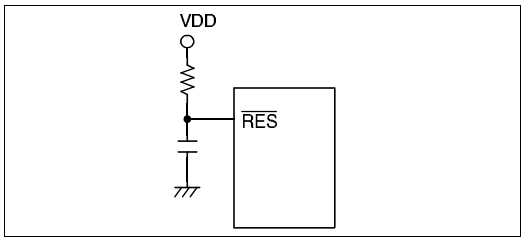
Note: "u" means that "unchange".



Reset Timing Chart

To guarantee that the system oscillator is started and stabilized, the SST (System Start-up Timer) provides an extra-delay to delay 1024 system clock pulses when system power-up or the system awakes from the HALT state.

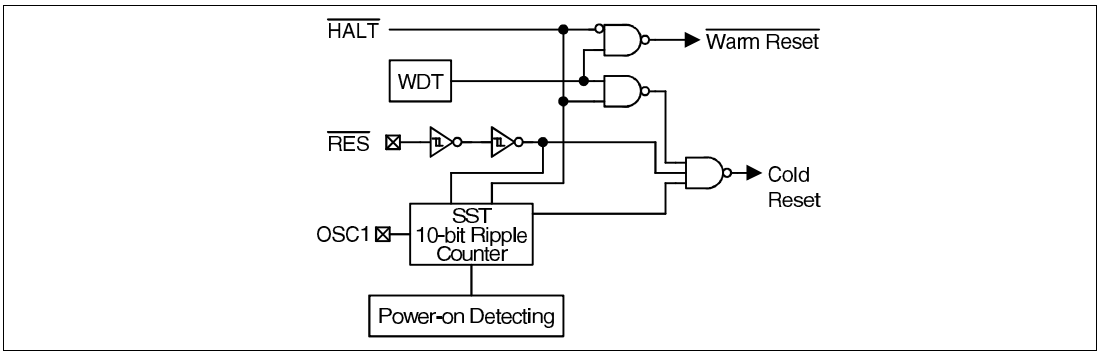
When the system power-up occurs, the SST delay is added during the reset period. But when the reset comes from the $\overline{\text{RES}}$ pin, the SST delay is disabled. Any wake-up from HALT will enable the SST delay.



Reset Circuit

The chip reset statuses of the functional units are as shown.

PC	000H
Interrrupt	Disable
Prescaler	Clear
WDT	Clear. After master reset, WDT begins counting
Timer/event Counter	Off
Input/output Ports	Input mode
SP	Point to the top of stack



Reset Configuration

The states of the registers is summarized in the table.

Register	Reset (power on)	WDT time-out (normal operation)	$\overline{\text{RES}}$ Reset (normal operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (HALT)
TMR	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMRC	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---
PC	000H	000H	000H	000H	000H*
MP	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	--00 -000	--00 -000	--00 -000	--00 -000	--uu -uuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	------11	------11	------11	------11	------uu
PCC	------11	------11	------11	------11	------uu

Note: "*" means "warm reset".
"u" means "unchanged".
"x" means "unknown".

Timer/Event Counter

A Timer/event Counter (TMR) is implemented in the HT48R12. The Timer/event Counter contains an 8-bit programmable count-up counter and the clock may come from an external source or the system clock divided by 4.

Using the internal instruction clock, there is only one reference time-base. The external clock input allows the user to count external events, measure time intervals or pulse widths, or to generate an accurate time base.

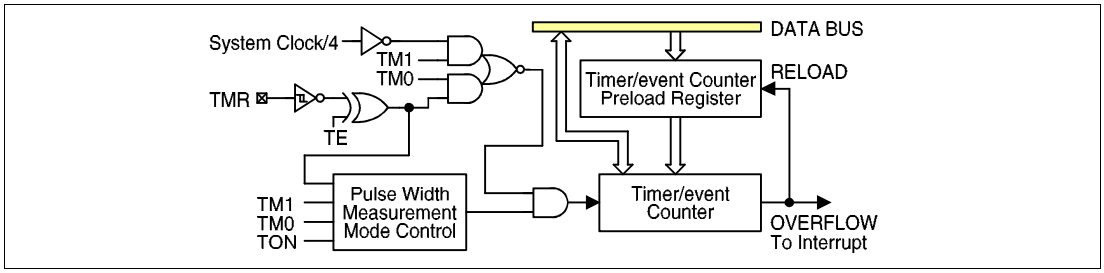
There are 2 registers related to Timer/event Counter; TMR ([0DH]), TMRC ([0EH]). Two physical registers are mapped to TMR location; writing TMR makes the starting value put in the Timer/event Counter Preload register and reading TMR gets the content of the Timer/event Counter. The TMRC is a Timer/event Counter control register, which defines some options.

The TM0, TM1 bits define the operating mode. The Event count mode is used to count external events, which means the clock source comes from an external (TMR) pin. The Timer mode functions as a normal timer with the clock source coming from the instruction clock. The pulse width measurement mode can be used to count the high or low level duration of the external signal (TMR). The counting is based on the instruction clock.

In the Event count or Timer mode, once the Timer/event Counter starts counting, it will count from the current contents in the Timer/event Counter to FFH. Once overflow occurs, the counter is reloaded from the Timer/event Counter Preload register and generates the interrupt request flag (TF; bit 5 of INTC) at the same time.

Label (TMRC)	Bits	Function
—	0-2	Unused bits, read as "0".
TE	3	To define the TMR active edge of Timer/event Counter (0=active on low to high; 1=active on high to low)
TON	4	To enable/disable timer counting (0=disabled; 1=enabled)
—	5	Unused bits, read as "0".
TM0 TM1	6 7	To define the operating mode 01=Event count mode (External clock) 10=Timer mode (Internal clock) 11=Pulse width measurement mode 00=Unused

TMRC Register



Timer/Event Counter

In the pulse width measurement mode with the TON and TE bits are equal to one, once the TMR has received a transient from low to high (or high to low if the TE bits is "0") it will start counting until the TMR returns to the original level and resets the TON. The measured result will remain in the Timer/event Counter even if the activated transient occurs again. In other words, only one cycle measurement can be done. Until setting the TON, the cycle measurement will function again as long as it receives further transient pulse. Note that, in this operating mode, the Timer/event Counter starts counting not according to the logic level but according to the transient edges. In the case of counter overflows, the counter is reloaded from the Timer/event Counter Preload register and issues the interrupt request just like the other two modes.

To enable the counting operation, the Timer ON bit (TON; bit 4 of TMRC) should be set to 1. In the pulse width measurement mode, the TON will be cleared automatically after the measurement cycle is completed. But in the other two modes the TON can only be reset by instructions. The overflow of the Timer/event counter is one of the wake-up sources. No matter what the operation mode is, writing a 0 to ETI can disable the interrupt service.

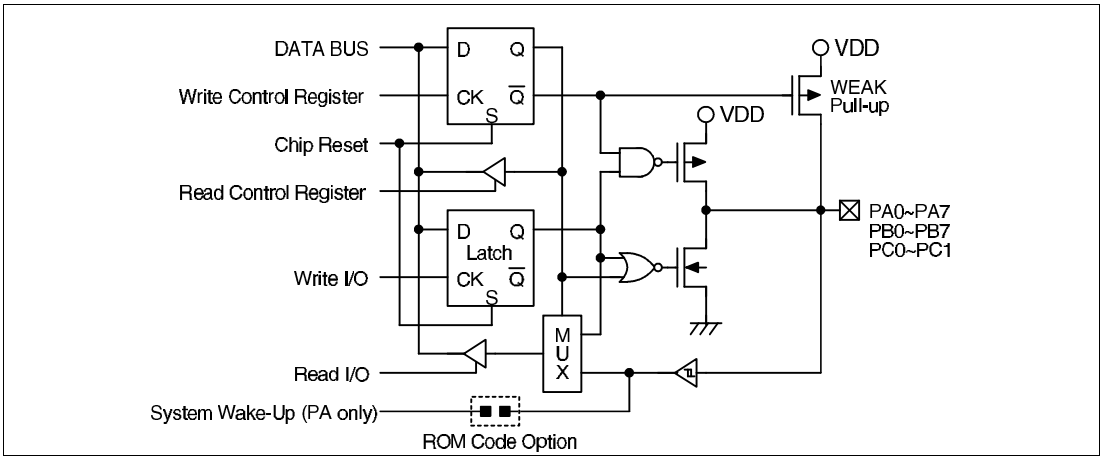
In the case of Timer/event Counter OFF condition, writing data to the Timer/event Counter Preload register will also reload that data to Timer/event Counter. But if the Timer/event Counter is turned on, data written to the Timer/event Counter will only be kept in the Timer/event Counter Preload register. The Timer/event Counter will still operate until overflow occurs.

When the Timer/event Counter (reading TMR) is read, the clock will be blocked to avoid errors. As clock blocking may results in a counting error, blocking of the clock must be taken into consideration by the programmer.

Input/Output Ports

There are 18 bi-directional input/output lines in the HT48R12, labelled from PA to PC, which are mapped to the data memory of [12H], [14H] and [16H] respectively. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, that is, the inputs must be ready at the T2 rising edge of instruction "MOV A,[m]" (m=12H, 14H or 16H). For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Each I/O line has its own control register (PAC, PBC, PCC) to control the input/output configuration. With this control register, CMOS output or schmitt trigger input with pull-high resistor structures can be reconfigured dynamically



Input/Output Ports

(i.e., on-the-fly) under software control. To function as an input, the corresponding latch of the control register must write "1". The pull-high resistance will be exhibited automatically. The input source(s) also depend(s) on the control register. If the control register bit is "1", the input will read the pad state. If the control register bit is "0", the contents of the latches will move to the internal bus. The latter is possible in the "read-modify-write" instruction. For output function, CMOS is the only configuration. These control registers are mapped to locations 13H, 15H and 17H.

After a chip reset, these input/output lines stay at high levels. Each bit of these intput/output latches can be set or cleared by "SET [m].i" and "CLR [m].i" (m=12H, 14H or 16H) instructions.

Some instructions first input data and then follow the output operations. For example, "SET [m].i", "CLR [m].i", "CPL [m]", "CPLA [m]" read the entire port states into the CPU, execute the defined operations (bit-operation), and then write the results back to the latches or the accumulator.

Each line of port A has the capability of waking-up the device. The highest six bits of port C are not physically implemented; on reading them a "0" is returned whereas writing then results in a no-operation. Note that the pull-high options are not available for all I/O lines.

ROM Code Option

The following table shows 7 kinds of ROM code option in the HT48R12. All of the ROM code options must be defined to ensure proper system functioning.

Items	Option	Description
1	OP[7:0]	OP0~OP7 → PA0~PA7 Bit=0 Without wake up Bit=1 With wake up
2	OP8	Bit=0 RC mode Bit=1 Crystal mode
3	OP9	Bit=0 Two cycle CLRWDT Bit=1 One cycle CLRWDT
4	OP10	Unused bit This bit must be set as "0" by default.
5	OP11	Unused bit This bit must be set as "0" by default.
6	OP12	Bit=0 RC clock for WDT source Bit=1 System clock for WDT source
7	OP13	Bit=0 Enable WDT Bit=1 Disable WDT

PROM Programming & Verification

The program memory used in the HT48R12 is arranged into a 1K×14 bits program PROM and a 1×14 bits option PROM. The program code and option code are stored in the program PROM and option PROM. The programming of PROM can be summarized in 9 steps as described below:

- Power on
- Set VPP($\overline{RES}$) to 12.5V
- Set $\overline{CS}$(PA5) to low

Let PA3~PA0(AD3~AD0) be the address and data bus and the PA4(CLK) be the clock input. The data on the AD3~AD0 pins will be clocked into or out the HT48R12 on the falling edge of PA4(CLK) for PROM programming and verification.

The address data contains the code address (11 bits) and two option bits. A complete write cycle will contain 4 CLK cycles. The first cycle, bit 0~3 of the address are latched into the HT48R12. The second and third cycles, bit 4~7 and bit 8~9 are latched respectively. The fourth cycle, bit 2 is the TSEL option bit and bit 3 is the OSEL option bit. Bit 2~3 in third cycle and bit 0~1 in the fourth cycle are undefined. If the TSEL is "1" and the OSEL is "0", the TEST memory will be read. If the TSEL is "0" and the OSEL is "1", the option PROM will be accessed. If both the TSEL and OSEL are "0", the program PROM will be managed.

The code data is 14 bits wide. A complete read/write cycle contains 4 CLK cycles. In the first cycle, bit 0~3 of the code data are accessed. In the second and third, bit 4~7 and bit 8~11 are accessed respectively. In the fourth cycle, bit 12~13 are accessed. Bit 14~15 are undefined. During code verification, reading will return the result "00".

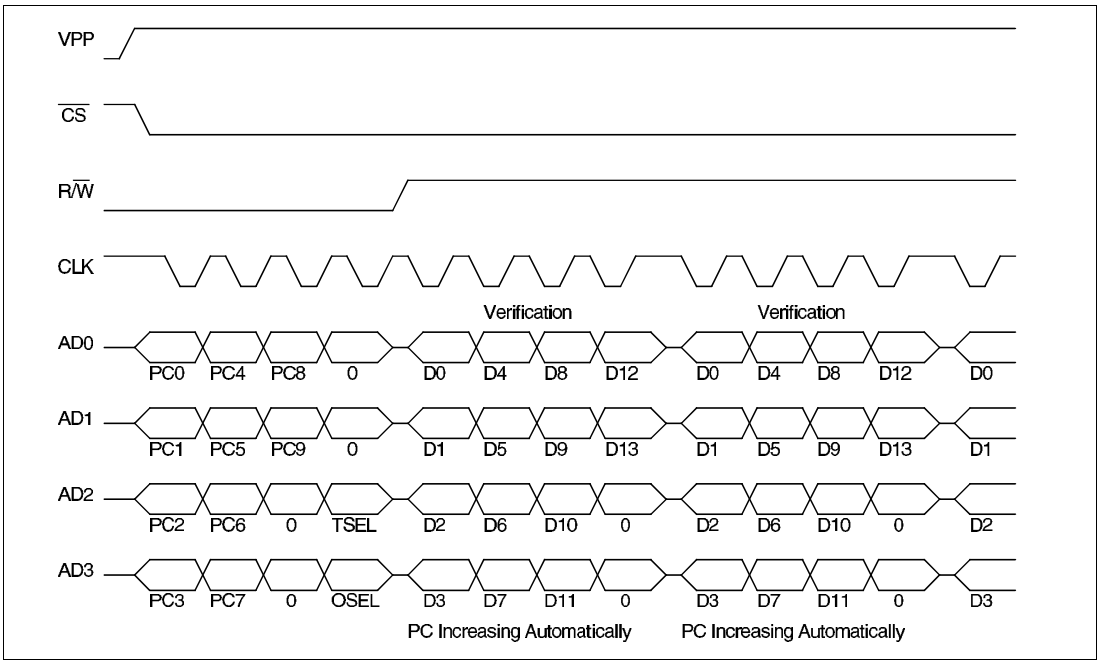
Select the TSEL and OSEL to program and verify the program PROM and option PROM. Use the R/$\overline{W}$(PA6) to select the programming or verification

The address is incremented by one automatically after a code verification cycle. If the uncontinued address programming or verification is accomplished, the automatic addressing increment is disabled. For the uncontinued address programming and verification, the $\overline{CS}$ pin must return to high level for a programming or verification cycle, that is, if an uncontinued address is managed, the programming or verification cycle must be interrupted and restarted as well.

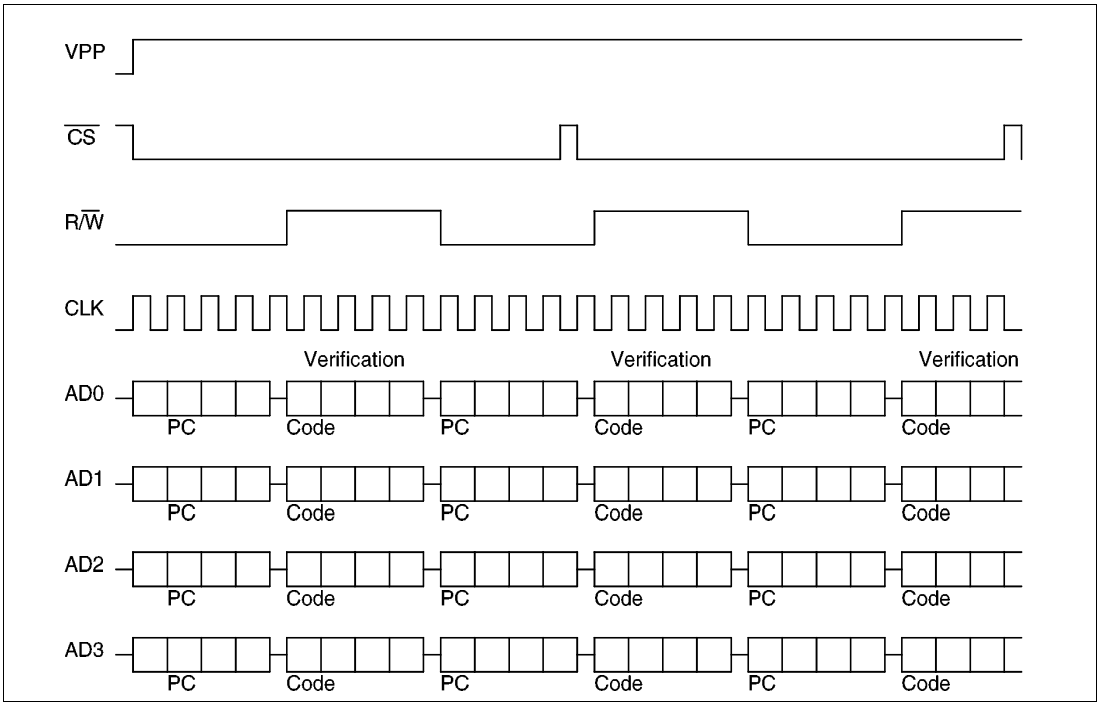
The related pins of PROM programming and verification are listed in the following table.

Pin Name	Function	Description
PA0	AD0	Bit 0 of address/data bus
PA1	AD1	Bit 1 of address/data bus
PA2	AD2	Bit 2 of address/data bus
PA3	AD3	Bit 3 of address/data bus
PA4	CLK	Serial clock input for address and data
PA5	$\overline{CS}$	Chip select, active low
PA6	R/$\overline{W}$	Read/write control input
$\overline{RES}$	VPP	Programming power supply

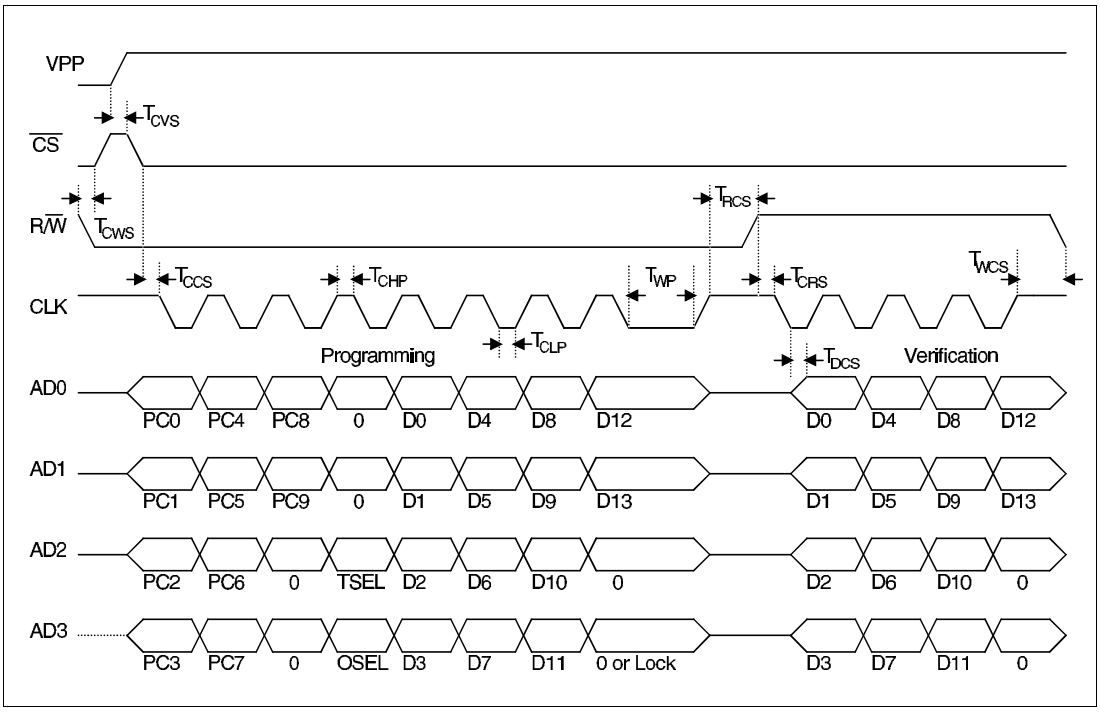
The timing charts of programming and verification are as shown. There is a LOCK signal for code protection. If the LOCK is "1", reading the code will return the result "1". However, if the LOCK is "0", the code protection is disabled and the code always can be read until the LOCK is programmed as "1".



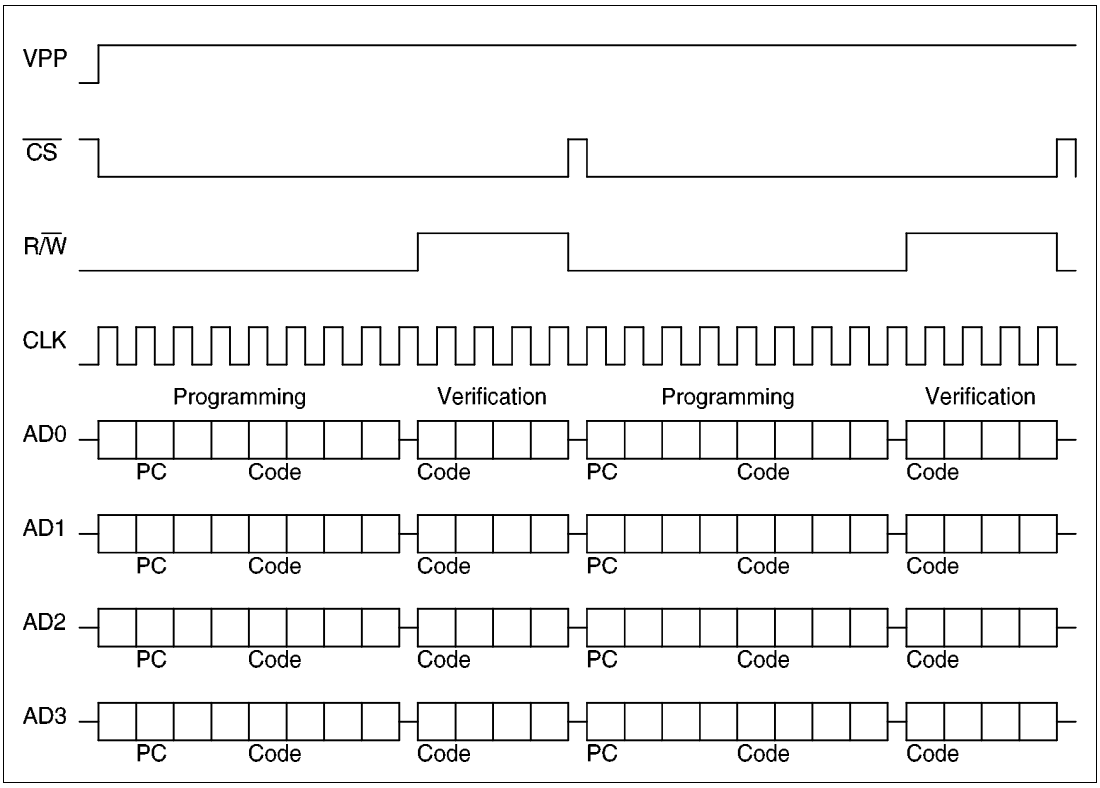
Successive Verification



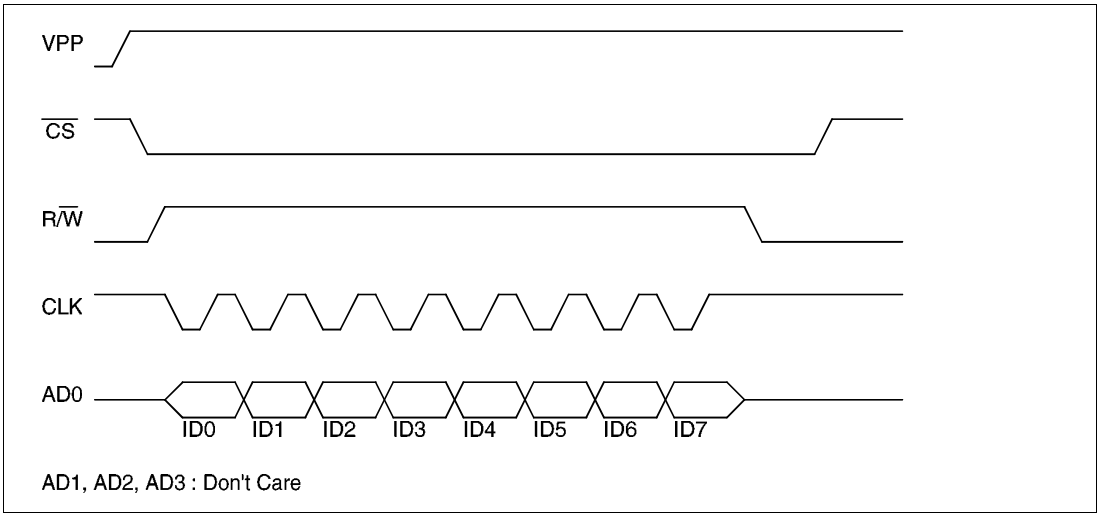
Non-successive Verification



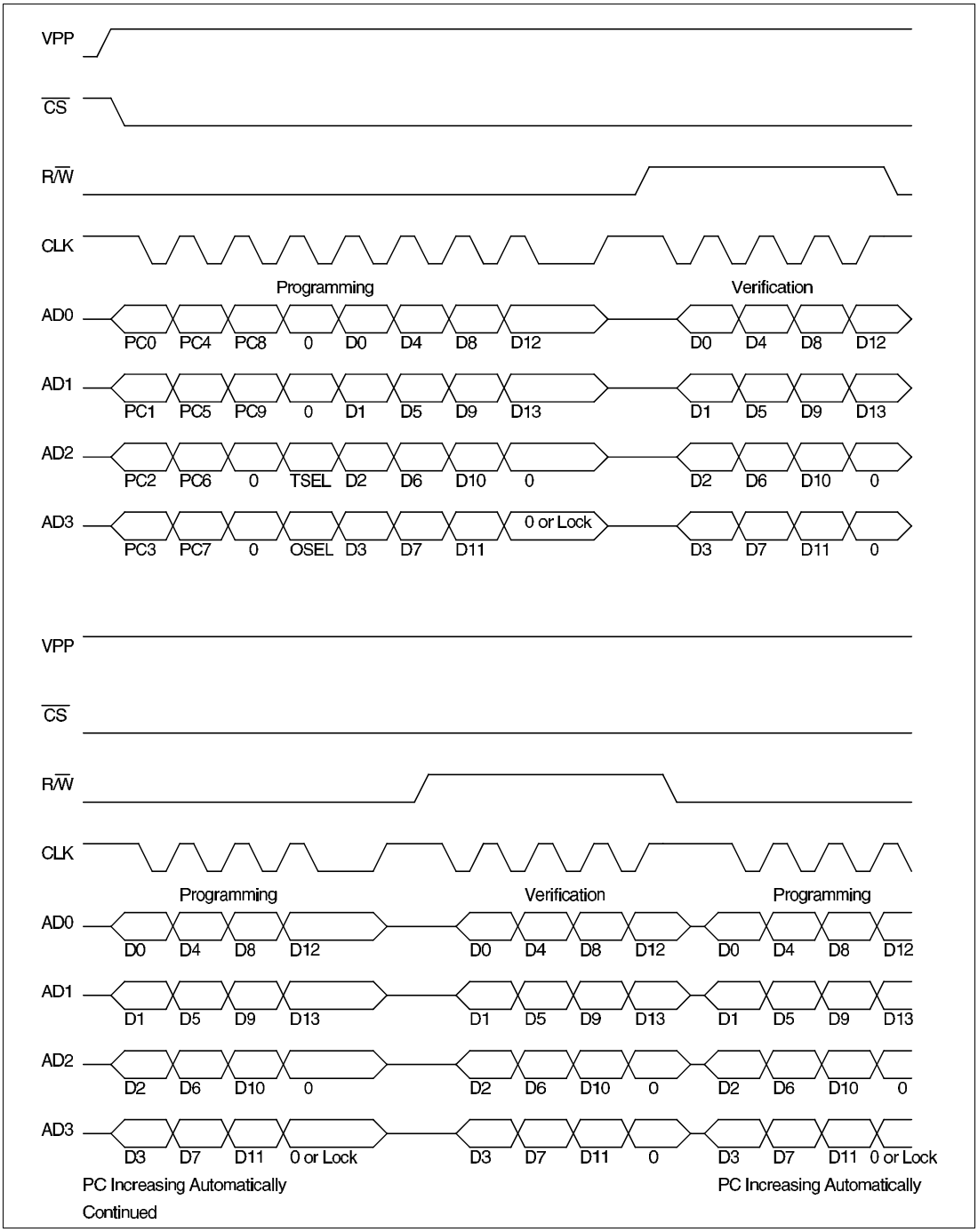
Code Programming and Verification



Non-successive Programming and Verification



Code Programming and Verification



Successive Programming and Verification

Instruction Set

Instruction Set Summary

Mnemonic	Description	Instruction Cycle	Flag Affected
Arithmetic			
ADD A,[m]	Add data memory to ACC	1	Z,C,AC,OV
ADDM A,[m]	Add ACC to data memory	1(1)	Z,C,AC,OV
ADD A,x	Add immediate data to ACC	1	Z,C,AC,OV
ADC A,[m]	Add data memory to ACC with carry	1	Z,C,AC,OV
ADCM A,[m]	Add ACC to register with carry	1(1)	Z,C,AC,OV
SUB A,x	Subtract immediate data from ACC	1	Z,C,AC,OV
SUB A,[m]	Subtract data memory from ACC	1	Z,C,AC,OV
SUBM A,[m]	Subtract data memory from ACC with result in data memory	1(1)	Z,C,AC,OV
SBC A,[m]	Subtract data memory from ACC with carry	1	Z,C,AC,OV
SBCM A,[m]	Subtract data memory from ACC with carry with result in data memory	1(1)	Z,C,AC,OV
DAA [m]	Decimal adjust ACC for addition with result in data memory	1(1)	C
Logic Operation			
AND A,[m]	AND data memory to ACC	1	Z
OR A,[m]	OR data memory to ACC	1	Z
XOR A,[m]	Exclusive-OR data memory to ACC	1	Z
ANDM A,[m]	AND ACC to data memory	1(1)	Z
ORM A,[m]	OR ACC to data memory	1(1)	Z
XORM A,[m]	Exclusive-OR ACC to data memory	1(1)	Z
AND A,x	AND immediate data to ACC	1	Z
OR A,x	OR immediate data to ACC	1	Z
XOR A,x	Exclusive-OR immediate data to ACC	1	Z
CPL [m]	Complement data memory	1(1)	Z
CPLA [m]	Complement data memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment data memory with result in ACC	1	Z
INC [m]	Increment data memory	1(1)	Z
DECA [m]	Decrement data memory with result in ACC	1	Z
DEC [m]	Decrement data memory	1(1)	Z

Mnemonic	Description	Instruction Cycle	Flag Affected
Rotate			
RRA [m]	Rotate data memory right with result in ACC	1	None
RR [m]	Rotate data memory right	1(1)	None
RRCA [m]	Rotate data memory right through carry with result in ACC	1	C
RRC [m]	Rotate data memory right through carry	1(1)	C
RLA [m]	Rotate data memory left with result in ACC	1	None
RL [m]	Rotate data memory left	1(1)	None
RLCA [m]	Rotate data memory left through carry with result in ACC	1	C
RLC [m]	Rotate data memory left through carry	1(1)	C
Data Move			
MOV A,[m]	Move data memory to ACC	1	None
MOV [m],A	Move ACC to data memory	1(1)	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of data memory	1(1)	None
SET [m].i	Set bit of data memory	1(1)	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if data memory is zero	1(2)	None
SZA [m]	Skip if data memory is zero with data movement to ACC	1(2)	None
SZ [m].i	Skip if bit i of data memory is zero	1(2)	None
SNZ [m].i	Skip if bit i of data memory is not zero	1(2)	None
SIZ [m]	Skip if increment data memory is zero	1(3)	None
SDZ [m]	Skip if decrement data memory is zero	1(3)	None
SIZA [m]	Skip if increment data memory is zero with result in ACC	1(2)	None
SDZA [m]	Skip if decrement data memory is zero with result in ACC	1(2)	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None

Mnemonic	Description	Instruction Cycle	Flag Affected
Table Read			
TABRDC [m]	Read ROM code (current page) to data memory and TBLH	2(1)	None
TABRDL [m]	Read ROM code (last page) to data memory and TBLH	2(1)	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear data memory	1(1)	None
SET [m]	Set data memory	1(1)	None
CLR WDT	Clear Watchdog timer	1	TO,PD
CLR WDT1	Pre-clear Watchdog timer	1	TO(4),PD(4)
CLR WDT2	Pre-clear Watchdog timer	1	TO(4),PD(4)
SWAP [m]	Swap nibbles of data memory	1(1)	None
SWAPA [m]	Swap nibbles of data memory with result in ACC	1	None
HALT	Enter power down mode	1	TO,PD

Notes:

x = 8 bits immediate data

m = 7 bits data memory address

A = accumulator

i = 0...7 number of bits

addr = 11 bits program memory address

√ = Flag(s) is affected

– = Flag(s) is not affected

(1) = If a loading to the PCL register occurs, the execution cycle of instructions will be delayed one more cycle (4 system clocks).

(2) = If a skipping next instruction occurs, the execution cycle of instructions will be delayed one more cycle (4 system clocks). Otherwise the original instruction cycle(s) is unchanged.

(3) = (1) and (2)

(4) = The flags may be affected by execution status. If the watchdog timer is cleared by executing the CLR WDT1 or CLR WDT2 instruction, the TO is set and the PD is cleared. Otherwise the TO and PD flags remain unchanged.

Instruction Definition

ADC A,[m] Add data memory and carry to the accumulator

Description The contents of the specified data memory, accumulator and the carry flag are added simultaneously, leaving the result in the accumulator.

Operation ACC ← ACC+[m]+C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADCM A,[m] Add the accumulator and carry to data memory

Description The contents of the specified data memory, accumulator and the carry flag are added simultaneously, leaving the result in the specified data memory.

Operation [m] ← ACC+[m]+C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADD A,[m] Add data memory to the accumulator

Description The contents of the specified data memory and the accumulator are added. The result is stored in the accumulator.

Operation ACC ← ACC+[m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADD A,x Add immediate data to the accumulator

Description The contents of the accumulator and the specified data are added, leaving the result in the accumulator.

Operation ACC ← ACC+x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

ADDM A,[m] Add the accumulator to the data memory

Description The contents of the specified data memory and the accumulator are added. The result is stored in the data memory.

Operation [m] ← ACC+[m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

AND A,[m] Logical AND accumulator with data memory

Description Data in the accumulator and the specified data memory perform a bitwise logical_AND operation. The result is stored in the accumulator.

Operation ACC ← ACC "AND" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

AND A,x Logical AND immediate data to the accumulator

Description Data in the accumulator and the specified data perform a bitwise logical_AND operation. The result is stored in the accumulator.

Operation ACC ← ACC "AND" x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

ANDM A,[m] Logical AND data memory with the accumulator

Description Data in the specified data memory and the accumulator perform a bitwise logical_AND operation. The result is stored in the data memory.

Operation [m] ← ACC "AND" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

CALL addr Subroutine call

Description The instruction unconditionally calls a subroutine located at the indicated address. The program counter increments once to obtain the address of the next instruction, and pushes this onto the stack. The indicated address is then loaded. Program execution continues with the instruction at this address.

Operation Stack ← PC+1
PC ← addr

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

CLR [m] Clear data memory

Description The contents of the specified data memory are cleared to zero.

Operation [m] ← 00H

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

CLR [m].i Clear bit of data memory

Description The bit i of the specified data memory is cleared to zero.

Operation [m].i ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

CLR WDT Clear watch dog timer

Description The WDT and the WDT Prescaler are cleared (re-counting from zero). The power down bit (PD) and time-out bit (TO) are cleared.

Operation WDT & WDT Prescaler ← 00H
PD & TO ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0	0	–	–	–	–

CLR WDT1 Preclear watch dog timer

Description The TD, PD flags, WDT and the WDT Prescaler has cleared (re-counting from zero), if the other preclear WDT instruction has been executed. Only execution of this instruction without the other preclear instruction just sets the indicated flag which implies this instruction has been executed and the TO and PD flags remain unchanged.

Operation WDT & WDT Prescaler ← 00H*
PD & TO ← 0*

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0*	0*	–	–	–	–

CLR WDT2 Preclear watch dog timer

Description The TO, PD flags, WDT and the WDT Prescaler are cleared (re-counting from zero), if the other preclear WDT instruction has been executed. Only execution of this instruction without the other preclear instruction, sets the indicated flag which implies this instruction has been executed and the TO and PD flags remain unchanged.

Operation WDT & WDT Prescaler ← 00H*
PD & TO ← 0*

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0*	0*	–	–	–	–

CPL [m] Complement data memory

Description Each bit of the specified data memory is logically complemented (1's complement). Bits which previously contained a one are changed to zero and vice-versa.

Operation $[m] \leftarrow [\overline{m}]$

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

CPLA [m] Complement data memory-place result in the accumulator

Description Each bit of the specified data memory is logically complemented (1's complement). Bits which previously contained a one are changed to zero and vice-versa. The complemented result is stored in the accumulator and the contents of the data memory remain unchanged.

Operation $ACC \leftarrow [\overline{m}]$

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

DAA [m] Decimal-Adjust accumulator for addition

Description The accumulator value is adjusted to the BCD (Binary Code Decimal) code. The accumulator is divided into two nibbles. Each nibble is adjusted to the BCD code and an internal carry (AC1) will be done if the low nibble of the accumulator is greater than 9. The BCD adjustment is done by adding 6 to the original value if the original value is greater than 9 or a carry (AC or C) is set; otherwise the original value remains unchanged. The result is stored in the data memory and only the carry flag (C) may be affected.

Operation If ACC.3~ACC.0 >9 or AC=1
then [m].3~[m].0 ← (ACC.3~ACC.0)+6, AC1=$\overline{AC}$
else [m].3~[m].0) ← (ACC.3~ACC.0), AC1=0
and
If ACC.7~ACC.4+AC1 >9 or C=1
then [m].7~[m].4 ← ACC.7~ACC.4+6+AC1,C=1
else [m].7~[m].4 ← ACC.7~ACC.4+AC1,C=C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

DEC [m] Decrement data memory

Description Data in the specified data memory is decremented by one.

Operation [m] ← [m]–1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

DECA [m] Decrement data memory-place result in the accumulator

Description Data in the specified data memory is decremented by one, leaving the result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC ← [m]–1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

HALT Enter power down mode

Description This instruction stops program execution and turn off the system clock. The contents of the RAM and registers are retained. The WDT and prescaler are cleared. The power down bit (PD) is set and the WDT time-out bit (TO) is cleared.

Operation PC ← PC+1
PD ← 1
TO ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	0	1	–	–	–	–

INC [m] Increment data memory

Description Data in the specified data memory is incremented by one.

Operation [m] ← [m]+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

INCA [m] Increment data memory-place result in the accumulator

Description Data in the specified data memory is incremented by one, leaving the result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC ← [m]+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

JMP addr Directly jump

Description Bits 0~10 of the program counter are replaced with the directly-specified address unconditionally, and control is passed to this destination.

Operation PC ← addr

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

MOV A,[m] Move data memory to the accumulator

Description The contents of the specified data memory are copied to the accumulator.

Operation ACC ← [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

MOV A,x Move immediate data to the accumulator

Description The 8-bit data specified by the code is loaded into the accumulator.

Operation ACC ← x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

MOV [m],A Move the accumulator to data memory

Description The contents of the accumulator are copied to the specified data memory (one of the data memory).

Operation [m] ← ACC

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

NOP No operation

Description No operation is performed. Execution continues with the next instruction.

Operation PC ← PC+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

OR A,[m] Logical OR accumulator with data memory

Description Data in the accumulator and the specified data memory (one of the data memory) perform a bitwise logical_OR operation. The result is stored in the accumulator.

Operation ACC ← ACC "OR" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

OR A,x Logical OR immediate data to the accumulator

Description Data in the accumulator and the specified data perform a bitwise logical_OR operation. The result is stored in the accumulator.

Operation ACC ← ACC "OR" x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

ORM A,[m] Logical OR data memory with the accumulator

Description Data in the data memory (one of the data memory) and the accumulator perform a bitwise logical_OR operation. The result is stored in the data memory.

Operation [m] ← ACC "OR" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

RET Return from subroutine

Description The program counter is restored from the stack. This is a two cycle instruction.

Operation PC ← Stack

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RET A,x Return and place immediate data in the accumulator

Description The program counter is restored from the stack and the accumulator loaded with the specified 8-bit immediate data.

Operation PC ← Stack
ACC ← x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RETI Return from interrupt

Description The program counter is restored from the stack, and interrupts are enabled by setting the EMI bit. EMI is the enable master(global) interrupt bit (bit 0; register INTC).

Operation PC ← Stack
EMI ← 1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RL [m] Rotate data memory left

Description The contents of the specified data memory are rotated one bit left with bit 7 rotated into bit 0.

Operation [m].(i+1) ← [m].i; [m].i:bit i of the data memory (i=0-6)
[m].0 ← [m].7

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RLA [m] Rotate data memory left-place result in the accumulator

Description Data in the specified data memory is rotated one bit left with bit 7 rotated into bit 0, leaving the rotated result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC.(i+1) ← [m].i; [m].i:bit i of the data memory (i=0-6)
ACC.0 ← [m].7

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RLC [m] Rotate data memory left through carry

Description The contents of the specified data memory and the carry flag are rotated one bit left. Bit 7 replaces the carry bit; the original carry flag is rotated into the bit 0 position.

Operation [m].(i+1) ← [m].i; [m].i:bit i of the data memory (i=0-6)
[m].0 ← C
C ← [m].7

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

RLCA [m] Rotate left through carry-place result in the accumulator

Description Data in the specified data memory and the carry flag are rotated one bit left. Bit 7 replaces the carry bit and the original carry flag is rotated into bit 0 position. The rotated result is stored in the accumulator but the contents of the data memory remain unchanged.

Operation ACC.(i+1) ← [m].i; [m].i:bit i of the data memory (i=0-6)
ACC.0 ← C
C ← [m].7

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

RR [m] Rotate data memory right

Description The contents of the specified data memory are rotated one bit right with bit 0 rotated to bit 7.

Operation [m].i ← [m].(i+1); [m].i:bit i of the data memory (i=0-6)
[m].7 ← [m].0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RRA [m] Rotate right-place result in the accumulator

Description Data in the specified data memory is rotated one bit right with bit 0 rotated into bit 7, leaving the rotated result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC.(i) ← [m].(i+1); [m].i:bit i of the data memory (i=0-6)
ACC.7 ← [m].0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

RRC [m] Rotate data memory right through carry

Description The contents of the specified data memory and the carry flag are together rotated one bit right. Bit 0 replaces the carry bit; the original carry flag is rotated into the bit 7 position.

Operation [m].i ← [m].(i+1); [m].i:bit i of the data memory (i=0-6)
[m].7 ← C
C ← [m].0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

RRCA [m] Rotate right through carry-place result in the accumulator

Description Data of the specified data memory and the carry flag are rotated one bit right. Bit 0 replaces the carry bit and the original carry flag is rotated into the bit 7 position. The rotated result is stored in the accumulator. The contents of the data memory remain unchanged.

Operation ACC.i ← [m].(i+1); [m].i:bit i of the data memory (i=0-6)
ACC.7 ← C
C ← [m].0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	√

SBC A,[m] Subtract data memory and carry from the accumulator

Description The contents of the specified data memory and the complement of the carry flag are subtracted from the accumulator, leaving the result in the accumulator.

Operation ACC ← ACC+$\overline{[m]}$+C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SBCM A,[m] Subtract data memory and carry from the accumulator

Description The contents of the specified data memory and the complement of the carry flag are subtracted from the accumulator, leaving the result in the data memory.

Operation $[m] \leftarrow ACC+\overline{[m]}+C$

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SDZ [m] Skip if decrement data memory is zero

Description The contents of the specified data memory are decremented by one. If the result is zero, the next instruction is skipped. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if ([m]–1)=0, [m] ← ([m]–1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SDZA [m] Decrement data memory-place result in ACC, skip if zero

Description The contents of the specified data memory are decremented by one. If the result is zero, the next instruction is skipped. The result is stored in the accumulator but the data memory remains unchanged. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if ([m]–1)=0, ACC ← ([m]–1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SET [m] Set data memory

Description Each bit of the specified data memory is set to one.

Operation [m] ← FFH

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SET [m].i Set bit of data memory

Description Bit "i" of the specified data memory is set to one.

Operation [m].i ← 1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SIZ [m] Skip if increment data memory is zero

Description The contents of the specified data memory are incremented by one. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if ([m]+1)=0, [m] ← ([m]+1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SIZA [m] Increment data memory-place result in ACC, skip if zero

Description The contents of the specified data memory are incremented by one. If the result is zero, the next instruction is skipped and the result is stored in the accumulator. The data memory remains unchanged. If the result is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if ([m]+1)=0, ACC ← ([m]+1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SNZ [m].i Skip if bit "i" of the data memory is not zero

Description If bit "i" of the specified data memory is not zero, the next instruction is skipped. If bit "i" of the data memory is not zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if [m].i≠0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SUB A,[m] Subtract data memory from the accumulator

Description The specified data memory is subtracted from the contents of the accumulator, leaving the result in the accumulator.

Operation ACC ← ACC+$\overline{[m]}$+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SUBM A,[m] Subtract data memory from the accumulator

Description The specified data memory is subtracted from the contents of the accumulator, leaving the result in the data memory.

Operation [m] ← ACC+$\overline{[m]}$+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SUB A,x Subtract immediate data from the accumulator

Description The immediate data specified by the code is subtracted from the contents of the accumulator, leaving the result in the accumulator.

Operation ACC ← ACC+$\overline{x}$+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	√	√	√	√

SWAP [m] Swap nibbles within the data memory

Description The low-order and high-order nibbles of the specified data memory (one of the data memories) are interchanged.

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SWAPA [m] Swap data memory-place result in the accumulator

escription The low-order and high-order nibbles of the specified data memory are interchanged, writing the result to the accumulator. The contents of the data memory remain unchanged.

Operation ACC.3~ACC.0 ← [m].7~[m].4
ACC.7~ACC.4 ← [m].3~[m].0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SZ [m] Skip if data memory is zero

Description If the contents of the specified data memory are zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if [m]=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SZA [m] Move data memory to ACC, skip if zero

Description The contents of the specified data memory are copied to the accumulator. If the contents is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if [m]=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

SZ [m].i Skip if bit "i" of the data memory is zero

Description If bit "i" of the specified data memory is zero, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if [m].i=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

TABRDC [m] Move ROM code (current page) to TBLH & data memory

Description The low byte of ROM code (current page) addressed by the table pointer (TBLP) is moved to the specified data memory and the high byte transfered to TBLH directly.

Operation [m] ← ROM code (low byte)
TBLH ← ROM code (high byte)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

TABRDL [m] Move ROM code (last page) to TBLH & data memory

Description The low byte of ROM code (last page) addressed by the table pointer (TBLP) is moved to the data memory and the high byte transfered to TBLH directly.

Operation [m] ← ROM code (low byte)
TBLH ← ROM code (high byte)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	–	–	–

XOR A,[m] Logical XOR accumulator with data memory

Description Data in the accumulator and the indicated data memory perform a bitwise logical Exclusive_OR operation and the result is stored in the accumulator.

Operation ACC ← ACC "XOR" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

XORM A,[m] Logical XOR data memory with the accumulator

Description Data in the indicated data memory and the accumulator perform a bitwise logical Exclusive_OR operation. The result is stored in the data memory. The zero flag is affected.

Operation [m] ← ACC "XOR" [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–

XOR A,x Logical XOR immediate data to the accumulator

Description Data in the the accumulator and the specified data perform a bitwise logical Exclusive_OR operation. The result is stored in the accumulator. The zero flag is affected.

Operation ACC ← ACC "XOR" x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
–	–	–	–	–	√	–	–