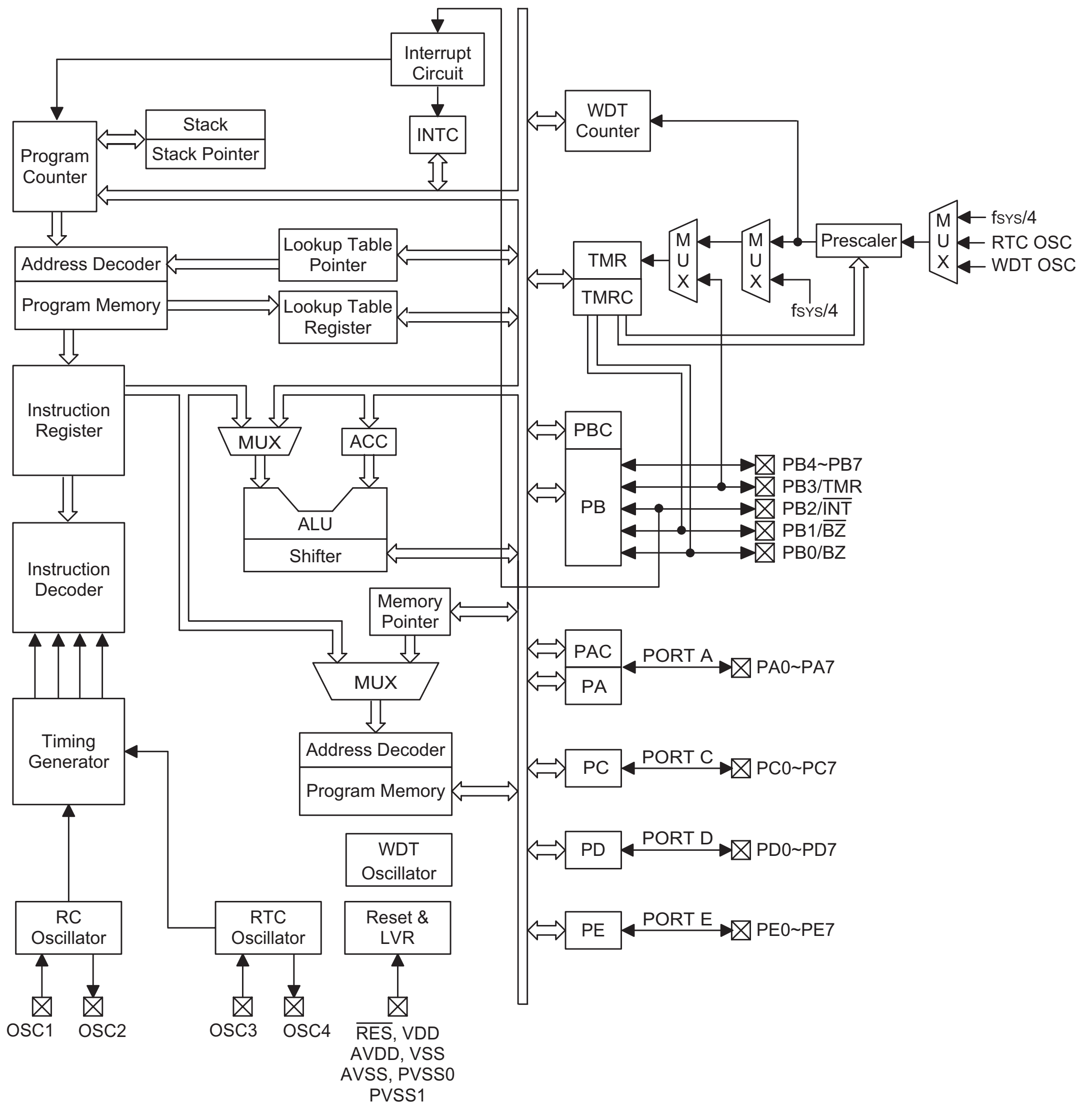
Features

- Operating voltage:
  f_{SYS}=4MHz: 2.2V~5.5V
  f_{SYS}=8MHz: 3.3V~5.5V
- Program memory:
  HT48R52: 2k×14
  HT48R53: 4k×15
- 88×8 data memory
- 40 bidirectional I/O lines
- One External interrupt input
- One Internal interrupt
- 8 bit programmable timer/event counter
- 4-level subroutine nesting
- Watchdog Timer (WDT)
- Buzzer output
- Low voltage reset (LVR)
- External RC and 32768Hz crystal oscillator
- Dual clock system offers three operating modes
  - Normal mode: Both RC and 32768Hz clock active
  - Slow mode: 32768Hz clock only
  - Idle mode: Periodical wake-up by watchdog timer overflow
- HALT function and wake-up feature reduce power consumption
- 14-bit table read instructions for HT48R52
- 15-bit table read instructions for HT48R53
- 63 powerful instructions
- One instruction cycle: 4 system clock periods
- All instructions in 1 or 2 instruction cycles
- Bit manipulation instructions
- Up to 0.5μs instruction cycle with 8MHz system clock
- 52-pin QFP package

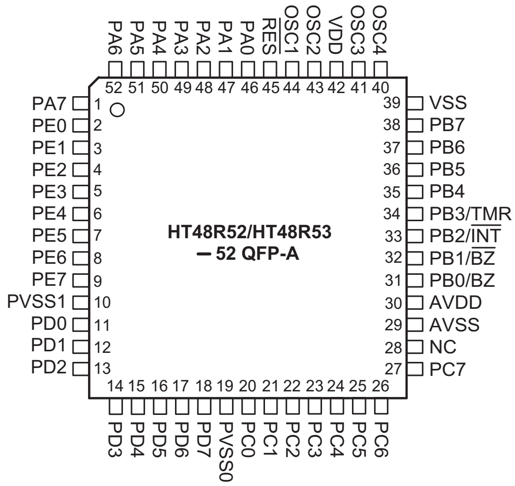
General Description

These devices are 8-bit high performance, RISC architecture microcontroller specifically designed for multiple I/O control product applications. The advantages of low power consumption, I/O flexibility, timer functions, oscillator options, HALT and wake-up functions, watchdog timer, buzzer driver, as well as low cost, enhance the versatility of these devices to suit a wide range of application possibilities such as industrial control, consumer products, subsystem controllers, etc.

Block Diagram



Pin Assignment



Pin Description

Pin Name	I/O	Options	Description
PA0~PA7	I/O	Pull-high Wake-up CMOS/Schmitt Trigger Input	Bidirectional 8-bit input/output port. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options). Each bit can be configured as a wake-up input by options.
PB0/BZ PB1/$\overline{\text{BZ}}$ PB2/$\overline{\text{INT}}$ PB3/TMR PB4~PB7	I/O	Pull-high CMOS/Schmitt PB0 or BZ PB1 or $\overline{\text{BZ}}$	Bidirectional 8-bit input/output port. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options). The PB0 and PB1 are pin-shared with the BZ and $\overline{\text{BZ}}$, respectively. Once the PB0 or PB1 is selected as buzzer driving outputs, the output signal range can be selected from $f_S/2$~$f_S/16$ by option. This external interrupt input is pin-shared with PB2.The external interrupt input is activated on a high to low transition. The timer input is pin-shared with PB3.
PC0~PC7	I/O	Pull-high	PC0~PC7 constitute an 8-bit input/output port. PC0~PC7 are configured as NMOS input/output pins with or without pull-high resistor by options.
PD0~PD7	I/O	Pull-high	PD0~PD7 constitute an 8-bit input/output port. PD0~PD7 are configured as NMOS input/output pins with or without pull-high resistor by options.
PE0~PE7	I/O	Pull-high	PE0~PE7 constitute an 8-bit input/output port. PE0~PE7 are configured as NMOS input/output pins with or without pull-high resistor by options.
$\overline{\text{RES}}$	I	—	Schmitt trigger reset input. Active low.
OSC1 OSC2	I O	RC	OSC1 and OSC2 are connected to an RC oscillator for the internal system clock. In the case of RC operation, OSC2 will be hold high by the MCU.
OSC3 OSC4	I O	RTC	Real time clock oscillators. OSC3 and OSC4 are connected to a 32768Hz crystal oscillator for timing purposes.
VDD	—	—	Positive power supply
AVDD	—	—	Positive power supply
VSS	—	—	Negative Power supply, ground
AVSS	—	—	Negative power supply, ground
PVSS0	—	—	Negative power supply for PC, PD7~PD4 port, ground
PVSS1	—	—	Negative power supply for PD3~PD0, PE port, ground

Note: "*" The pull-high resistors of each I/O port (PA, PB, PC, PD, PE) are all controlled by port option.

Absolute Maximum Ratings

Supply Voltage ....V_{SS}–0.3V to V_{SS}+6.0V

Input Voltage.............................V_{SS}–0.3V to V_{DD}+0.3V

Storage Temperature ....–50°C to 125°C

Operating Temperature..........................–40°C to 85°C

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the device. Functional operation of this device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=4MHz	2.2	—	5.5	V
		—	f_{SYS}=8MHz	3.3	—	5.5	
		—	f_{SYS}=32768Hz	—	—	—	
I_{DD1}	Operating Current (RC OSC)	3V	No load, f_{SYS}=4MHz	—	1.2	2	mA
		5V		—	2.5	5	
I_{DD2}	Operating Current (RC OSC)	5V	No load, f_{SYS}=8MHz	—	4	8	mA
I_{DD3}	Operating Current (32768Hz OSC, RC off)	3V	No load, f_{SYS}=32768Hz	—	0.3	0.6	mA
		5V		—	0.6	1	
I_{STB1}	Standby Current (WDT and 32768Hz Enabled)	3V	No load, system HALT	—	—	5	μA
		5V		—	—	10	
I_{STB2}	Standby Current (WDT Disabled, 32768Hz Enabled)	3V	No load, system HALT	—	0.8	1.5	μA
		5V		—	1.5	2.5	
V_{IL1}	Input Low Voltage for I/O Ports	—	—	0	—	0.3V_{DD}	V
V_{IH1}	Input High Voltage for I/O Ports	—	—	0.7V_{DD}	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	0.4V_{DD}	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	0.9V_{DD}	—	V_{DD}	V
V_{LVR}	Low Voltage Reset	—	3.3V option	2.7	3	3.3	V
I_{OL}	I/O Port Sink Current	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V		10	20	—	
I_{OH}	I/O Port Source Current	3V	V_{OH}=0.9V_{DD}	–2	–4	—	mA
		5V		–5	–10	—	
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V		10	30	50	

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock (RC OSC)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	
f_{TIMER}	Timer I/P Frequency	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{FSP1}	f_{SP} Time-out Period Clock Source from WDT	3V	With prescaler (f_S/4096)	184	369	737	ms
		5V		131	266	532	
t_{FSP2}	f_{SP} Time-out Period Clock Source from 32.768kHz	3V	With prescaler (f_S/4096)	—	125	—	ms
		5V		—	125	—	
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

Functional Description

Execution Flow

The system clock for the microcontroller is derived from either an RC oscillator or a 32768Hz crystal. 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 takes the next instruction cycle. However, the pipelining scheme causes each instruction to be effectively executed in a cycle. If an instruction changes the program counter, two cycles are required to complete the instruction.

Program Counter – PC

The program counter (PC) controls the sequence in which the instructions stored in the program ROM are executed and its contents specify a full range of program memory.

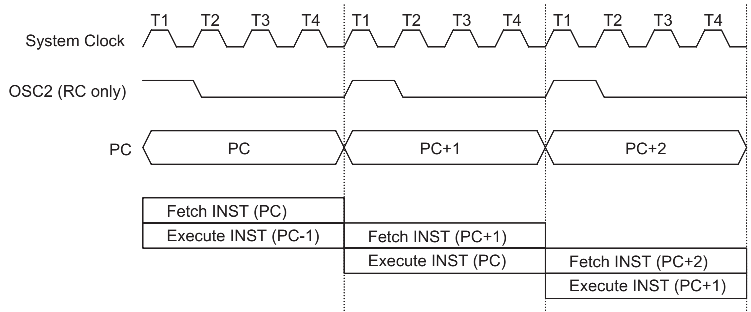
After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by one. The program counter then points to the memory word containing the next instruction code.

When executing a jump instruction, conditional skip execution, loading PCL register, subroutine call or return from subroutine, initial reset, internal interrupt, external interrupt or return from interrupt, 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 writeable register (06H). Moving data into the PCL performs a short jump. The destination will be within 256 locations.

When a control transfer takes place, an additional dummy cycle is required.



Execution Flow

Mode	Program Counter											
	***11**	***10**	***9**	***8**	***7**	***6**	***5**	***4**	***3**	***2**	***1**	***0**
Initial Reset	0	0	0	0	0	0	0	0	0	0	0	0
External Interrupt	0	0	0	0	0	0	0	0	0	1	0	0
Timer/Event Counter Overflow	0	0	0	0	0	0	0	0	1	0	0	0
Skip	Program Counter+2											
Loading PCL	*11	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

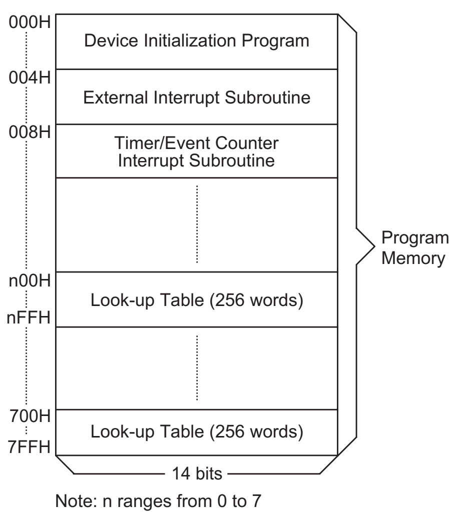
Note: *11~*0: Program counter bits
S11~S0: Stack register bits
#11~#0: Instruction code bits
@7~@0: PCL bits
For the HT48R52, the Program Counter is 11 bits wide, i.e. from *10~*0
For the HT48R53, the Program Counter is 12 bits wide, i.e. from *11~*0.

Program Memory − ROM

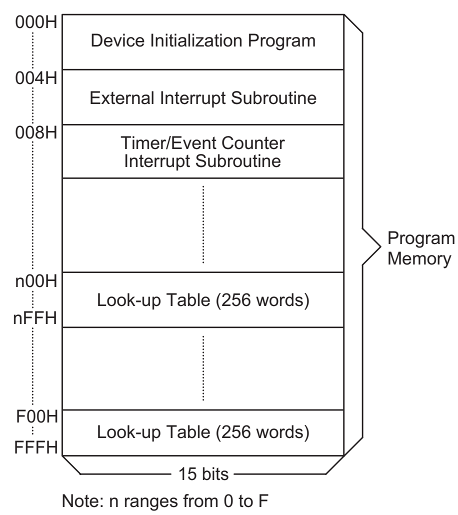
The program memory is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized into 2048×14 bits for HT48R52 or 4096×15 bits for HT48R53, addressed by the program counter and table pointer.

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

- Location 000H
  This area is reserved for program initialization. After a chip reset, the program always begins execution at location 000H.
- Location 004H
  This area is reserved for the external interrupt service program. If the $\overline{\text{INT}}$ 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 program memory space can be used as look-up tables. The instructions ″TABRDC [m]″ (the current page, one 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 transferred to the lower portion of the TBLH, and the remaining 2-bit words 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 the 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 can occur. In other words, using the table read instruction



Program Memory for the HT48R52



Program Memory for the HT48R53

Instruction	Table Location											
	***11**	***10**	***9**	***8**	***7**	***6**	***5**	***4**	***3**	***2**	***1**	***0**
TABRDC [m]	P11	P10	P9	P8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: *11~*0: Table location bits

P11~P8: Current program counter bits

@7~@0: Table pointer bits

For the HT48R52, the table location is 11 bits, i.e. from *10~*0

For the HT48R53, the table location is 12 bits, i.e. from *11~*0.

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 is supposed to be disabled prior to the table read instruction. It will not be enabled until the TBLH has been backed up. All table related instructions require two cycles to complete the operation. These areas may function as normal program memory depending upon the requirements.

Stack Register – STACK

This is a special part of the memory which is used to save the contents of the program counter only. The stack is organized into 4 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the stack pointer (SP) and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, 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 acknowledge signal will be inhibited. When the stack pointer is (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 4 return addresses are stored).

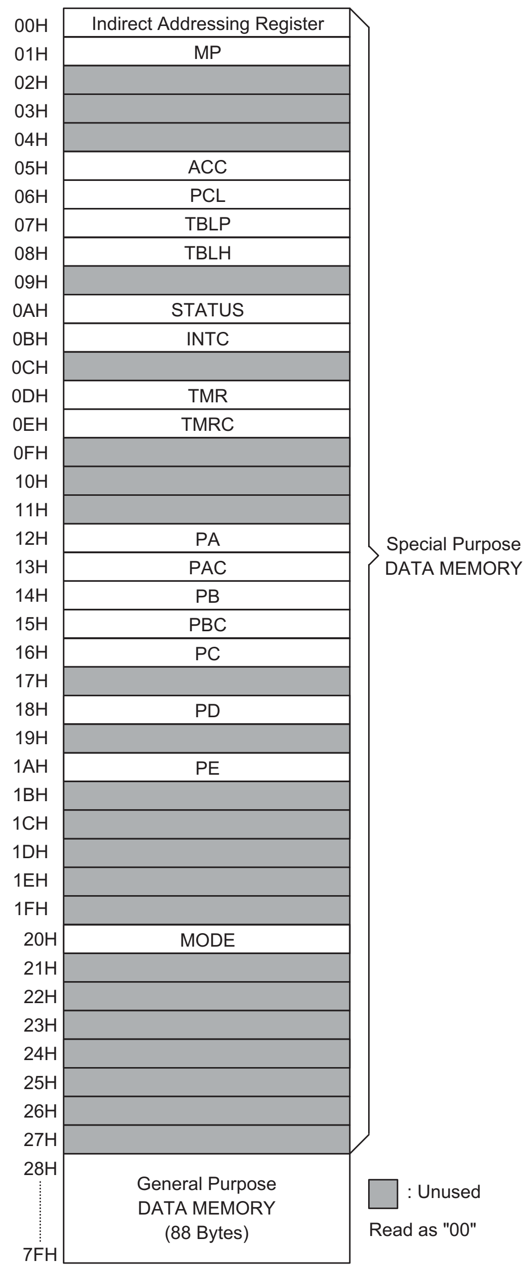
Data Memory – RAM

The data memory is designed with 106×8 bits. The data memory is divided into two functional groups, namely, function registers and general purpose data memory (88×8). Most are read/write, but some are read only.

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 reset by ″SET [m].i″ and ″CLR [m].i″. They are also indirectly accessible through memory pointer registers (MP).

Indirect Addressing Register

Location 00H is indirect addressing register that is not physically implemented. Any read/write operation of [00H] will access data memory pointed to by MP. Reading location 00H itself indirectly will return the result 00H. Writing indirectly results in no operation. The memory pointer register (MP) is 7-bit registers.



RAM Mapping

Accumulator

The accumulator is closely related to ALU operations. It is also mapped to location 05H of the data memory and can carry out immediate data operations. The data movement between two data memory locations must pass 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 and 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 (PDF), and watchdog time-out flag (TO). It also records the status information and controls the operation sequence.

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

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

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

Interrupt

The device 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 or 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 occur during this interval but only the interrupt request flag is recorded. If a certain inter-

Bit No.	Label	Function
0	C	C is set if an 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. C is also affected by a rotate through carry instruction.
1	AC	AC is set if an 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.
2	Z	Z is set if the result of an arithmetic or logic operation is zero, otherwise Z is cleared.
3	OV	OV is set if an 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.
4	PDF	PDF is cleared by system power-up or executing the ″CLR WDT″ instruction. PDF is set by executing the ″HALT″ instruction.
5	TO	TO is cleared by a system power-up or executing the ″CLR WDT″ or ″HALT″ instruction. TO is set by a WDT time-out.
6, 7	—	Unused bit, read as ″0″

STATUS (0AH) Register

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1=enable; 0=disable)
1	EEI	Controls the external interrupt (1=enable; 0=disable)
2	ETI	Controls the Timer/Event Counter 0 interrupt (1=enable; 0=disable)
3	—	Unused bit, read as ″0″
4	EIF	External interrupt request flag (1=active; 0=inactive)
5	TF	Internal Timer/Event Counter 0 request flag (1=active; 0=inactive)
6, 7	—	Unused bit, read as ″0″

INTC (0BH) Register

rupt requires servicing within the service routine, the EMI bit and the corresponding bit of the INTC may be set 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 a wake-up capability. As an interrupt is serviced, a control transfer occurs by pushing the program counter onto the stack, followed by a branch to a subroutine at specified location 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 contents should be saved in advance.

External interrupts are triggered by a high to low transition of the $\overline{\text{INT}}$ and the related interrupt request flag (EIF; bit 4 of the 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 interrupt request flag (TF; bit 5 of the 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 acknowledge signals are held until the ″RETI″ instruction is executed or the EMI bit and the related interrupt control bit are set to 1 (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.

Interrupts, occurring 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 following table shows the priority that is applied. These can be masked by resetting the EMI bit.

Interrupt Source	Priority	Vector
External Interrupt	1	04H
Timer/Event Counter Overflow	2	08H

The timer/event counter interrupt request flag (TF), external interrupt request flag (EIF), enable timer/event counter interrupt 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 recommended that a program does not use the ″CALL subroutine″ within the interrupt subroutine. 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.

Oscillator Configuration

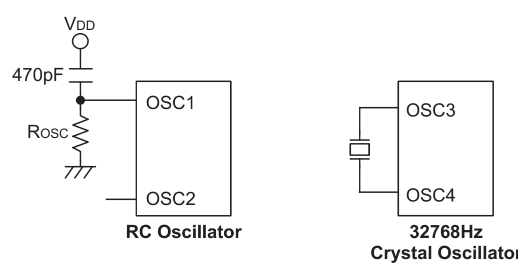
These devices provide two oscillator circuits for system clocks, i.e., RC oscillator and 32768Hz crystal oscillator, determined by software. No matter what type of oscillator is selected, the signal is used for the system clock.

If an RC oscillator is used, an external resistor between OSC1 and VSS is required and the resistance must range from 130kΩ to 2.4MΩ. OSC2 will be held in a permanent high state by the MCU. 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.

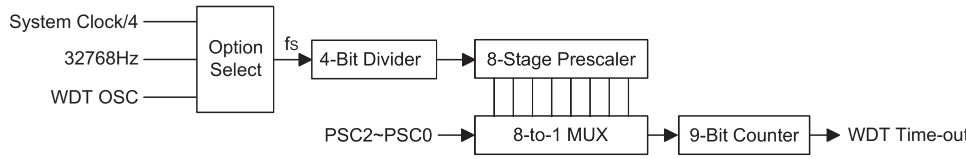
There is another oscillator circuit designed for the real time clock. In this case, only the 32768Hz crystal oscillator can be applied. The crystal should be connected between OSC3 and OSC4.

The RTC oscillator circuit can be controlled to oscillate quickly by setting the ″QOSC″ bit (bit 4 of mode). It is recommended to turn on the quick oscillating function upon power on, until the RTC oscillator is stable and then turn it off after 2 seconds to reduce power consumption.

The WDT oscillator is a free running on-chip RC oscillator, and no external components are required. Although the system enters the power down mode, the system clock stops, and the WDT oscillator still works within a period of approximately 65μs@5V. The WDT oscillator can be disabled by options to conserve power.



System Oscillator



Watchdog Timer

Watchdog Timer – WDT

The WDT clock source is implemented by a WDT OSC, external 32768Hz (f_{RTC}) or an instruction clock (system clock divided by 4), determined by option. This timer is designed to prevent a software malfunction or sequence from jumping to an unknown location with unpredictable results. The Watchdog can be disabled by option. If the Watchdog Timer is disabled, all the execution related to WDT results in no operation.

If the device operates in a noisy environment, using the on-chip WDT OSC or 32768Hz crystal oscillator is strongly recommended.

When the WDT clock source is selected, it will be first divided by 16 (4-stage), and then divided by TMRC prescaler (8-stage), after that, divided by 512 (9-stage) to get the nominal time-out period. By using the TMRC prescaler, longer time-out periods can be realized. Writing data to PSC2, PSC1, PSC0 can give different time-out periods. The WDT OSC period is 65μs. This time-out period may vary with temperature, VDD and process variations. The WDT OSC always works for any operation mode.

If the instruction clock (system clock/4) is selected as the WDT clock source, the WDT operates in the same manner.

If the WDT clock source is the 32768Hz, the WDT also operates in the same manner.

The WDT time-out under normal mode or slow mode will initialize a ″chip reset″ and set the status bit ″TO″. But in the idle mode (HALT instruction is executed) the time-out will initialize a ″warm reset″ and only the program counter and stack pointer are reset to 0.

To clear the WDT contents (not including the 4-bit divider and the 8-stage prescaler), three methods are adopted; external reset (a low level to $\overline{\text{RES}}$ pin), software instruction and a ″HALT″ instruction.

The software instruction 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 mask option ″WDT″ instruction. If the ″CLR WDT″ is selected (i.e. One clear instruction), any execution of the CLR WDT instruction will clear the WDT. In the case that ″CLR WDT1″ and ″CLR WDT2″ are chosen (i.e. two clear instructions), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip as a result of time-out.

Buzzer Output

The PB0 and PB1 are pin-shared with BZ and $\overline{\text{BZ}}$ signal, respectively. If the BZ & $\overline{\text{BZ}}$ option is selected, the output signal in output mode of PB0/PB1 will be the generated buzzer signal . The input mode always remain in its original functions. Once the BZ & $\overline{\text{BZ}}$ option is selected, the buzzer output signals are controlled by the PB0 data register only. The I/O functions of PB0/PB1 are shown below.

PBC Register PBC.0	PBC Register PBC.1	PB data Register PB.0	PB data Register PB.1	Output Function
0	0	1	X	PB0=BZ, PB1=$\overline{\text{BZ}}$
0	0	0	X	PB0=0, PB1=0
0	1	1	X	PB0=BZ, PB1=Input
0	1	0	X	PB0=0, PB1=Input
1	0	X	D	PB0=Input, PB1=D
1	1	X	X	PB0=Input, PB1=Input

PB0/PB1 Pin Function Control

Note: ″X″ stands for don′t care
″D″ stands for data ″0″ or ″1″

Operation Mode

The device support two system clock and three operation modes. The system clock could be RC oscillator or 32768Hz and operation mode could be Normal, Slow, or Idle mode. These are all selected by software.

Bit No.	Label	Function
0	MODS	System clock high/low mode select bit 0: High system clock (RC) 1: Low system clock (32768Hz), and RC oscillator stop Low system clock mode can only be used. When the WDT clock source option is 32768Hz or the function will be not predictable.
1, 2, 3	—	Unused bit, read as ″0″
4	QOSC	32768Hz OSC quick start-up oscillating 0=quick start; 1=slow start
5, 6, 7	—	Unused bit, read as ″0″

MODE (20H) Register

Mode	System Clock	HALT Instruction	MODS	RC Oscillator	32768Hz
Normal	RC oscillator	No Executed	0	On	On
Slow	32768Hz	No Executed	1	Off	On
Idle	HALT	Be executed	x	Off	On

Operation Mode

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 remains 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 WDT clock is from the WDT oscillator).
- All of the I/O ports maintain their original status.
- The PDF 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 PDF flags are examined, the cause for a chip reset can be determined. The PDF flag is cleared by a 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 program counter and SP, the other circuits remain in 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 options. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If it awakens from an interrupt, two sequence may occur. If the related interrupt is disabled or the interrupt 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 a wake-up event occurs, it takes 1024 (system clock period) to resume normal operation. In other words, a dummy period will be inserted after a wake-up. If the wake-up results from an interrupt acknowledge signal, 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. The RTC oscillator still runs in the HALT mode (if the RTC oscillator is enabled).

Reset

There are three 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 time-out during HALT is different from other chip reset conditions, since it can perform a ″warm reset″ that resets only the program counter and SP, leaving the other circuits in their original 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 PDF and TO flags, the program can distinguish between different ″chip resets″.

TO	PDF	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″ stands for ″unchanged″

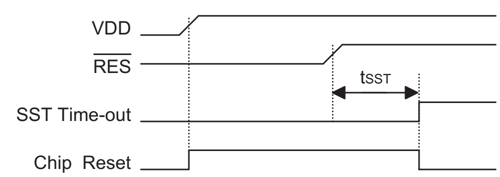
To guarantee that the system oscillator is started and stabilized, the SST (System Start-up Timer) provides an extra delay of 1024 system clock pulses when the system reset (power-up, WDT time-out or $\overline{\text{RES}}$ reset) or the system awakes from the HALT state.

When a system reset occurs, the SST delay is added during the reset period. Any wake-up from HALT will enable the SST delay.

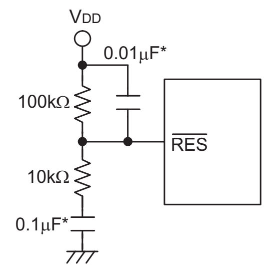
An extra option load time delay is added during a system reset (power-up, WDT time-out at normal mode or $\overline{\text{RES}}$ reset).

The functional unit chip reset status are shown below.

Program Counter	000H
Interrupt	Disable
Prescaler	Clear
WDT	Clear. After master reset, WDT begins counting
Timer/Event Counter	Off
Input/Output Ports	Input mode
SP	Points to the top of the stack

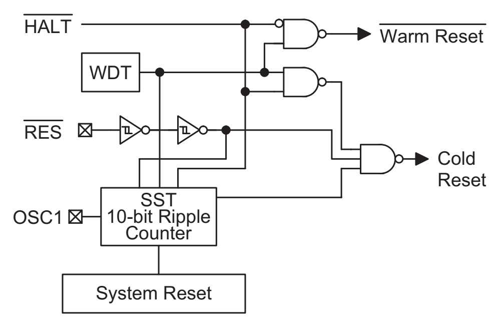


Reset Timing Chart



Reset Circuit

Note: ″*″ Make the length of the wiring, which is connected to the $\overline{\text{RES}}$ pin as short as possible, to avoid noise interference.



Reset Configuration

The states of the registers is summarized in the table.

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
MP	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH (HT48R52)	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
TBLH (HT48R53)	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u 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	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PE	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
MODE	---0 ---0	---u ---u	---u ---u	---u ---u	---u ---u

Note: ″*″ stands for ″warm reset″
″u″ stands for ″unchanged″
″x″ stands for ″unknown″

Timer/Event Counter

Timer/event counter (TMR) is implemented in the microcontroller. The timer/event counter contains an 8-bit programmable count-up counter and the clock may come from an external source or from the system clock/4 or f_{SP}.

The f_{SP} clock source is implemented by a WDT OSC, an external 32768Hz (f_{RTC}) or an instruction clock (system clock divided by 4), determined by options, if one of these three source is selected, it will be first divided by 16 (4-stage), and then divided by TMRC prescaler (8-stage) to get an f_{SP} output period. By using the TMRC prescaler, longer time-out periods can be realized. Writing data to PSC2, PSC1, PSC0 can give different f_{SP} output periods.

Using internal clock sources, there are 2 reference time-bases for the timer/event counter. The internal clock source can be selected as coming from $f_{SYS}/4$ or f_{SP} by options. Using an external clock input allows the user to count external events, measure time intervals or pulse widths, or generate an accurate time base. While using the internal clock allows the user to generate an accurate time base.

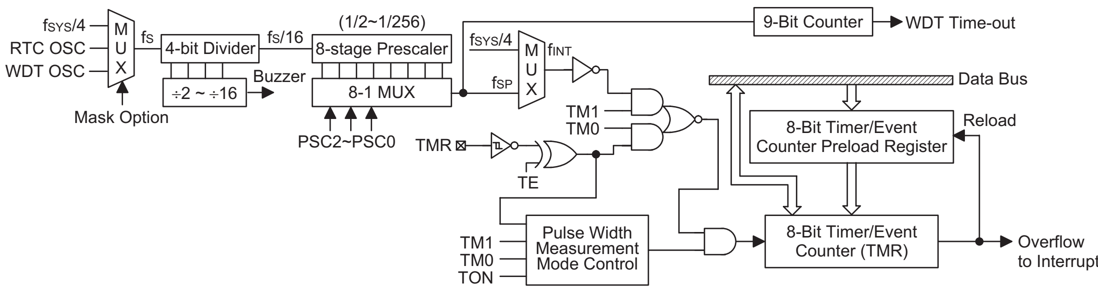
There are two registers related to the timer/event counter; TMR ([0DH]) and TMRC ([0EH]). Two physical registers are mapped to TMR location, writing TMR makes the starting value be placed in the timer/event counter preload register and reading TMR retrieves the contents of the timer/event counter. The 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 f_{INT} clock or RTC clock. The pulse width measurement mode can be used to count the high or low-level duration of the external signal. The counting is based on the f_{INT} clock or RTC 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 at the same time generates the interrupt request flag (TF; bit 5 of the INTC).

In the pulse width measurement mode with the TON and TE bits equal to one, once it goes 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 the 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 corresponding interrupt services.

In the case of timer/event counter OFF condition, writing data to the timer/event counter preload register will also reload that data to the timer/event counter. But if the timer/event counter is turned on, data written to it will only be kept in the timer/event counter preload register. The timer/event counter will still operate until overflow occurs (a timer/event counter reloading will occur at the same time). When the timer/event counter (reading TMR) is read, the clock will be blocked to avoid errors. As clock blocking may result in a counting error, this must be taken into consideration by the programmer. The bit2~bit0 of the TMRC can be used to define the pre-scaling stages of the internal clock sources of the timer/event counter. The definitions are as shown.



Timer/Event Counter and WDT

Bit No.	Label	Function
0 1 2	PSC0 PSC1 PSC2	Define the prescaler stages, PSC2, PSC1, PSC0= 000: $f_{SP}=f_S/32$ 001: $f_{SP}=f_S/64$ 010: $f_{SP}=f_S/128$ 011: $f_{SP}=f_S/256$ 100: $f_{SP}=f_S/512$ 101: $f_{SP}=f_S/1024$ 110: $f_{SP}=f_S/2048$ 111: $f_{SP}=f_S/4096$
3	TE	Defines the TMR active edge of the timer/event counter 0 (0=active on low to high; 1=active on high to low)
4	TON	To enable or disable timer 0 counting (0=disable; 1=enable)
5	—	Unused bit, read as "0"
6 7	TM0 TM1	Define the operating mode, TM1, TM0= 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMRC (0EH) Register

Input/Output Ports

There are 16 bidirectional input/output lines and 24 NMOS input/output in the microcontroller, labeled from PA to PE, which are mapped to the data memory of [12H], [14H], [16H], [18H] and [1AH] respectively. All pins on PA~PE can be used for both input and output operations.

For the 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, 16H, 18H, 1AH). For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

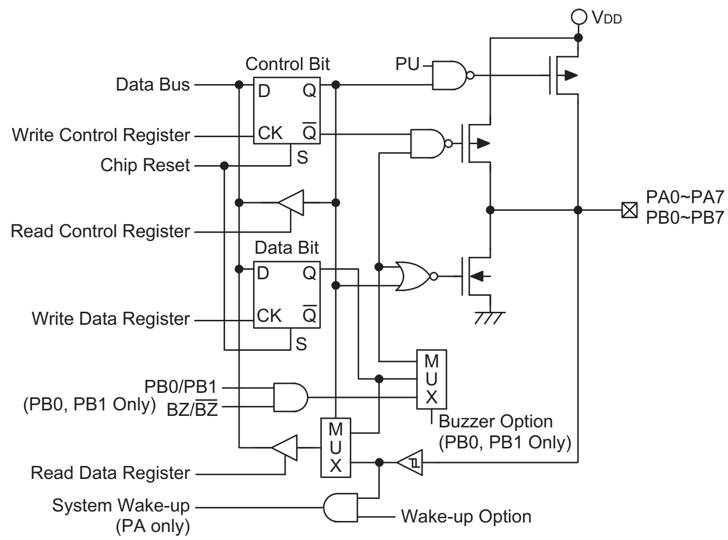
The PA and PB have their own control registers (PAC, PBC) to control the input/output configuration. These two control registers are mapped to locations 13H and 15H. CMOS output or Schmitt trigger input with structures can be reconfigured dynamically under software control. The control registers specifies which pin are set as input and which are set as outputs. To setup a pin as an input the corresponding bit of the control register must be set high, for an output it must be set low. The PC, PD and PE can also be used for both input and output operations, these ports are not achieved through the use of port control registers. Setting up an I/O pin as input is achieved by first setting its output high which effectively places its NMOS output transistor in high impedance state allowing the pin to be now used as an input.

After a chip reset, these input/output lines remain at high levels or floating state (depending on the pull-high options). Each bit of these input/output latches can be set or cleared by ″SET [m].i″ and ″CLR [m].i″ (m=12H, 14H, 16H, 18H or 1AH) instructions.

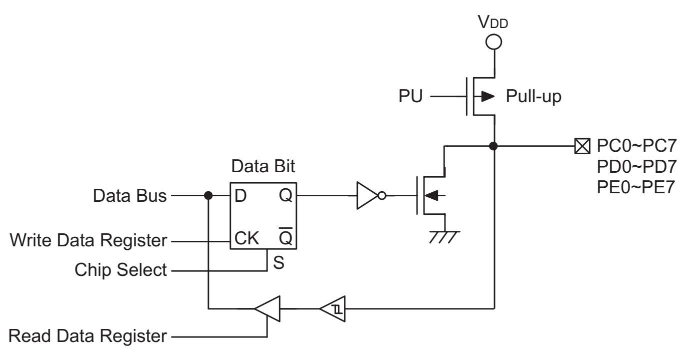
Some instructions first input data and then follow the output operations. For example, ″SET [m].i″, ″LR [m].i″, ″CPL [m]″, ″CPLA [m]″ read the entire port states into the CPU, execute the defined operations (bit-operation), and then write the results back to the latches or the accumulator.

Each line of port A has the capability of waking-up the device.

There is a pull-high option available for all I/O lines (port option). Once the pull-high option of an I/O line is selected, the I/O line have pull-high resistor. Otherwise,



PA, PB Input/Output Ports



PC, PD, PE Input/Output Ports

the pull-high resistor is absent. It should be noted that a non-pull-high I/O line operating in input mode will cause a floating state.

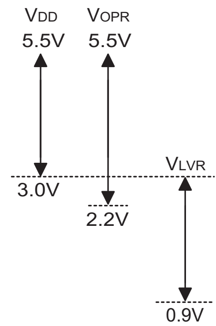
Low Voltage Reset – LVR

The microcontroller provides a low voltage reset circuit in order to monitor the supply voltage of the device. If the supply voltage of the device is within the range 0.9V~V_{LVR}, such as changing a battery, the LVR will automatically reset the device internally.

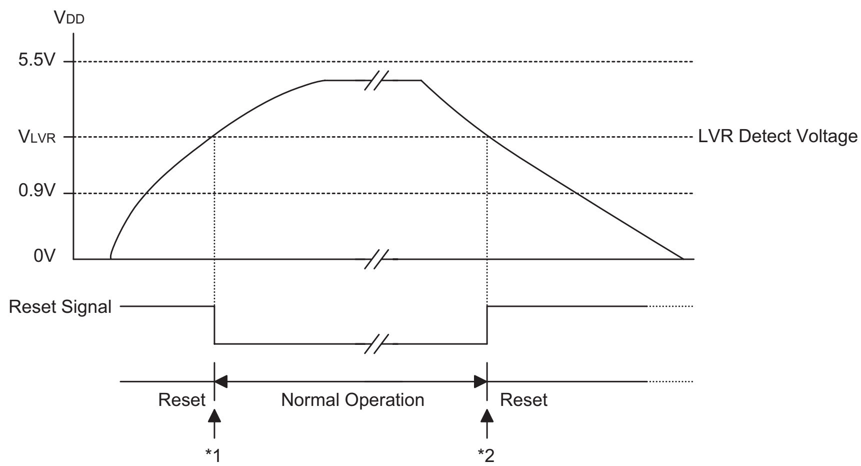
The LVR includes the following specifications:

- The low voltage (0.9V~V_{LVR}) has to remain in their original state for more than 1ms. If the low voltage state does not exceed 1ms, the LVR will ignore it and do not perform a reset function.
- The LVR uses the ″OR″ function with the external $\overline{RES}$ signal to perform a chip reset.

The relationship between V_{DD} and V_{LVR} is shown below.



Note: V_{OPR} is the voltage range for proper chip operation at 4MHz system clock.



Low Voltage Reset

Note: *1: To make sure that the system oscillator has stabilized, the SST provides an extra delay of 1024 system clock pulses before entering the normal operation.

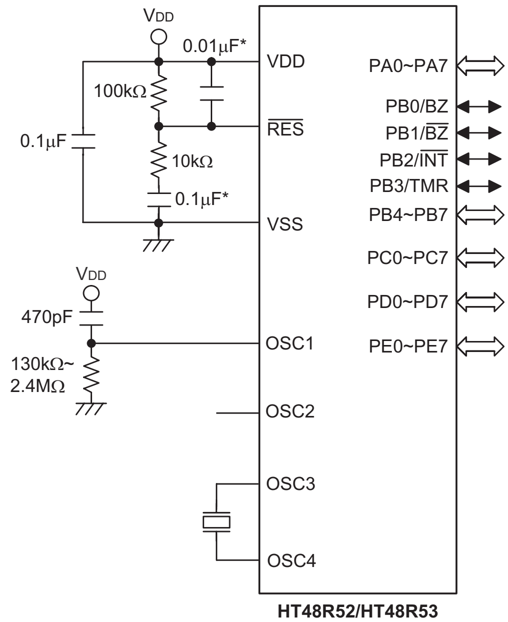
*2: Since low voltage has to be maintained for more than 1ms, therefore a 1ms delay is provided before entering the reset mode.

Options

The following table shows all kinds of options in the microcontroller. All of the options must be defined to ensure having a proper functioning system.

Items	Options
1	PA7~PA0 bit wake-up enable or disable (by bit)
2	PA, PB, PC, PD, PE pull-high enable or disable (by port)
3	WDT clock source: WDT oscillator or f_{SYS}/4 or 32768Hz oscillator
4	WDT enable or disable
5	CLRWDT instructions: 1 or 2 instructions
6	Timer/event counter clock sources: f_{SYS}/4 or f_{SP}
7	PB0, PB1: normal I/O or Buzzer function
8	Buzzer frequency: f_S/2, f_S/4, f_S/8, f_S/16
9	LVR enable or disable

Application Circuits



Note: The resistance and capacitance for reset circuit should be designed in such a way as to ensure that the VDD is stable and remains within a valid operating voltage range before bringing RES to high.

"*" Make the length of the wiring, which is connected to the RES pin as short as possible, to avoid noise interference.

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 data memory 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 and 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
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

Mnemonic	Description	Instruction Cycle	Flag Affected
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
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,PDF
CLR WDT1	Pre-clear Watchdog Timer	1	$TO^{(4)}$,$PDF^{(4)}$
CLR WDT2	Pre-clear Watchdog Timer	1	$TO^{(4)}$,$PDF^{(4)}$
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,PDF

Note: x: Immediate data

m: Data memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

√: Flag is affected

–: Flag is not affected

$^{(1)}$: If a loading to the PCL register occurs, the execution cycle of instructions will be delayed for one more cycle (four system clocks).

$^{(2)}$: If a skipping to the next instruction occurs, the execution cycle of instructions will be delayed for one more cycle (four system clocks). Otherwise the original instruction cycle is unchanged.

$^{(3)}$: $^{(1)}$ and $^{(2)}$

$^{(4)}$: The flags may be affected by the execution status. If the Watchdog Timer is cleared by executing the ″CLR WDT1″ or ″CLR WDT2″ instruction, the TO and PDF are cleared.
Otherwise the TO and PDF 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)

TO	PDF	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)

TO	PDF	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)

TO	PDF	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)

TO	PDF	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)

TO	PDF	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)

TO	PDF	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)

TO	PDF	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)

TO	PDF	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 ← Program Counter+1
Program Counter ← addr

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m] Clear data memory

Description: The contents of the specified data memory are cleared to 0.

Operation: [m] ← 00H

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m].i Clear bit of data memory

Description: The bit i of the specified data memory is cleared to 0.

Operation: [m].i ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR WDT Clear Watchdog Timer

Description: The WDT is cleared (clears the WDT). The power down bit (PDF) and time-out bit (TO) are cleared.

Operation: WDT ← 00H
PDF and TO ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
0	0	—	—	—	—

CLR WDT1 Preclear Watchdog Timer

Description: Together with CLR WDT2, clears the WDT. PDF and TO are also cleared. 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 PDF flags remain unchanged.

Operation: WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CLR WDT2 Preclear Watchdog Timer

Description: Together with CLR WDT1, clears the WDT. PDF and TO are also cleared. 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 PDF flags remain unchanged.

Operation: WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

TO	PDF	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 1 are changed to 0 and vice-versa.

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

Description
Each bit of the specified data memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice-versa. The complemented result is stored in the accumulator and the contents of the data memory remain unchanged.

Operation
ACC ← $\overline{[m]}$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

DAA [m] Decimal-Adjust accumulator for addition

Description
The accumulator value is adjusted to the BCD (Binary Coded 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

DEC [m] Decrement data memory

Description
Data in the specified data memory is decremented by 1.

Operation
[m] ← [m]–1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

Description
Data in the specified data memory is decremented by 1, leaving the result in the accumulator. The contents of the data memory remain unchanged.

Operation
ACC ← [m]–1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

HALT Enter power down mode

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

Operation: Program Counter ← Program Counter+1
PDF ← 1
TO ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
0	1	—	—	—	—

INC [m] Increment data memory

Description: Data in the specified data memory is incremented by 1

Operation: [m] ← [m]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

Description: Data in the specified data memory is incremented by 1, leaving the result in the accumulator. The contents of the data memory remain unchanged.

Operation: ACC ← [m]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

JMP addr Directly jump

Description: The program counter are replaced with the directly-specified address unconditionally, and control is passed to this destination.

Operation: Program Counter ←addr

Affected flag(s)

TO	PDF	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)

TO	PDF	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)

TO	PDF	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 memories).

Operation: [m] ←ACC

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

NOP No operation

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

Operation: Program Counter ← Program Counter+1

Affected flag(s)

TO	PDF	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 memories) perform a bitwise logical_OR operation. The result is stored in the accumulator.

Operation: ACC ← ACC ″OR″ [m]

Affected flag(s)

TO	PDF	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)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

Description: Data in the data memory (one of the data memories) 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)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

RET Return from subroutine

Description: The program counter is restored from the stack. This is a 2-cycle instruction.

Operation: Program Counter ← Stack

Affected flag(s)

TO	PDF	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: Program Counter ← Stack
ACC ← x

Affected flag(s)

TO	PDF	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.

Operation: Program Counter ← Stack
EMI ← 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RL [m] Rotate data memory left

Description: The contents of the specified data memory are rotated 1 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

Description: Data in the specified data memory is rotated 1 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)

TO	PDF	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 1 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

Description: Data in the specified data memory and the carry flag are rotated 1 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

RR [m] Rotate data memory right

Description: The contents of the specified data memory are rotated 1 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

Description: Data in the specified data memory is rotated 1 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)

TO	PDF	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 1 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

Description: Data of the specified data memory and the carry flag are rotated 1 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)

TO	PDF	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)

TO	PDF	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] ← ACC+$\overline{[m]}$+C

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SDZ [m] Skip if decrement data memory is 0

Description: The contents of the specified data memory are decremented by 1. If the result is 0, the next instruction is skipped. If the result is 0, 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SDZA [m] Decrement data memory and place result in ACC, skip if 0

Description: The contents of the specified data memory are decremented by 1. If the result is 0, the next instruction is skipped. The result is stored in the accumulator but the data memory remains unchanged. If the result is 0, 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m] Set data memory

Description: Each bit of the specified data memory is set to 1.

Operation: [m] ← FFH

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m]. i Set bit of data memory

Description: Bit i of the specified data memory is set to 1.

Operation: [m].i ← 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SIZ [m] Skip if increment data memory is 0

Description: The contents of the specified data memory are incremented by 1. If the result is 0, 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SIZA [m] Increment data memory and place result in ACC, skip if 0

Description: The contents of the specified data memory are incremented by 1. If the result is 0, the next instruction is skipped and the result is stored in the accumulator. The data memory remains unchanged. If the result is 0, 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SNZ [m].i Skip if bit i of the data memory is not 0

Description: If bit i of the specified data memory is not 0, the next instruction is skipped. If bit i of the data memory is not 0, 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)

TO	PDF	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)

TO	PDF	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)

TO	PDF	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)

TO	PDF	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 (1 of the data memories) are interchanged.

Operation: [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

Description: 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SZ [m] Skip if data memory is 0

Description: If the contents of the specified data memory are 0, 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SZA [m] Move data memory to ACC, skip if 0

Description: The contents of the specified data memory are copied to the accumulator. If the contents is 0, 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SZ [m].i Skip if bit i of the data memory is 0

Description: If bit i of the specified data memory is 0, 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)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

TABRDC [m] Move the ROM code (current page) to TBLH and 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 transferred to TBLH directly.

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

TABRDL [m] Move the ROM code (last page) to TBLH and 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 transferred to TBLH directly.

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

Affected flag(s)

TO	PDF	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)

TO	PDF	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 0 flag is affected.

Operation: [m] ← ACC ″XOR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

XOR A,x Logical XOR immediate data to the accumulator

Description: Data in the accumulator and the specified data perform a bitwise logical Exclusive_OR operation. The result is stored in the accumulator. The 0 flag is affected.

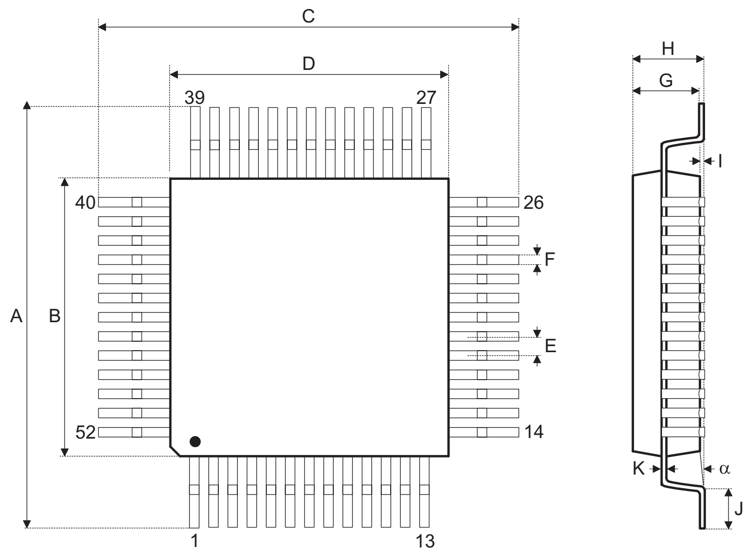
Operation: ACC ← ACC ″XOR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

Package Information

52-pin QFP (14×14) Outline Dimensions



Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	17.3	—	17.5
B	13.9	—	14.1
C	17.3	—	17.5
D	13.9	—	14.1
E	—	1	—
F	—	0.4	—
G	2.5	—	3.1
H	—	—	3.4
I	—	0.1	—
J	0.73	—	1.03
K	0.1	—	0.2
α	0°	—	7°

Holtek Semiconductor Inc. (Headquarters)
No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan
Tel: 886-3-563-1999
Fax: 886-3-563-1189
http://www.holtek.com.tw

Holtek Semiconductor Inc. (Taipei Sales Office)
4F-2, No. 3-2, YuanQu St., Nankang Software Park, Taipei 115, Taiwan
Tel: 886-2-2655-7070
Fax: 886-2-2655-7373
Fax: 886-2-2655-7383 (International sales hotline)

Holtek Semiconductor Inc. (Shanghai Sales Office)
7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China 200233
Tel: 021-6485-5560
Fax: 021-6485-0313
http://www.holtek.com.cn

Holtek Semiconductor Inc. (Shenzhen Sales Office)
43F, SEG Plaza, Shen Nan Zhong Road, Shenzhen, China 518031
Tel:0755-83465589
Fax:0755-83465590
ISDN : 0755-8346559

Holtek Semiconductor Inc. (Beijing Sales Office)
Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031
Tel:010-66410030, 66417751, 66417752
Fax:010-66410125

Holmate Semiconductor, Inc. (North America Sales Office)
46712 Fremont Blvd., Fremont, CA 94538
Tel: 510-252-9880
Fax: 510-252-9885
http://www.holmate.com