HT45R34
C/R to F Type 8-Bit OTP MCU

Technical Document

- Tools Information
- FAQs
- Application Note
  - HA0075E MCU Reset and Oscillator Circuits Application Note

Features

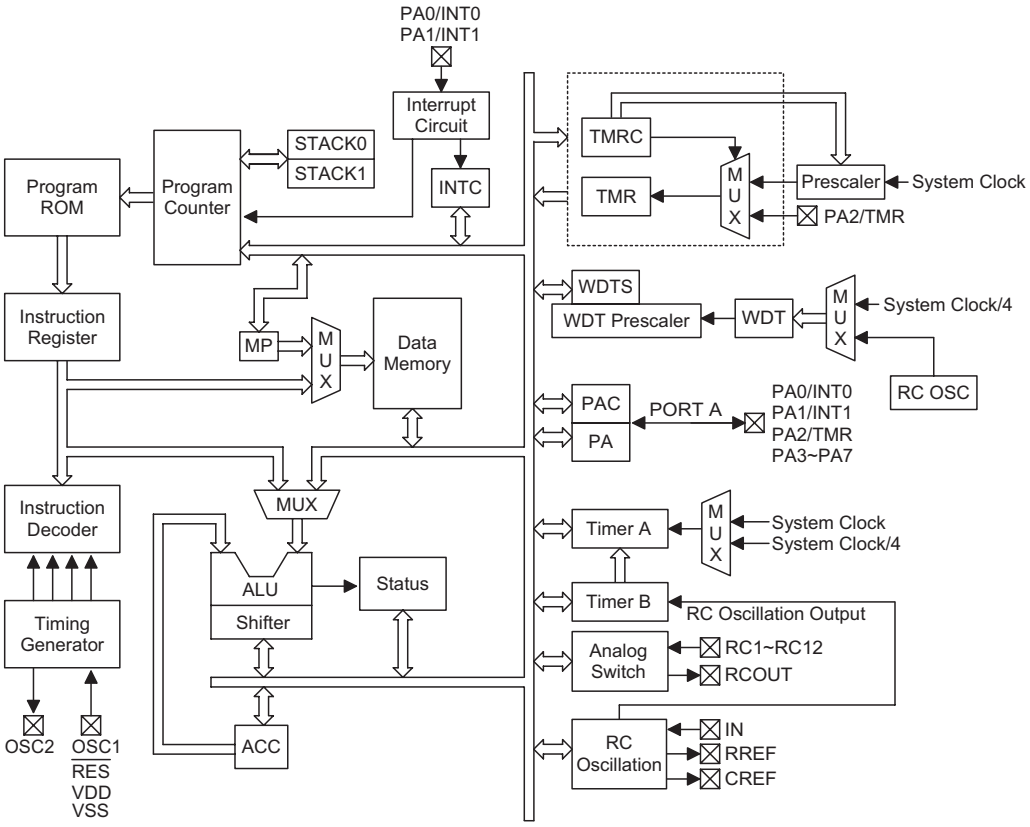
- Operating voltage:
  f_{SYS}=4MHz: 2.2V~5.5V
  f_{SYS}=8MHz: 3.3V~5.5V
- 8 bidirectional I/O lines
- Two external interrupt inputs shared with I/O lines
- 8-bit programmable timer/event counter with overflow interrupt and 7-stage prescaler
- External RC oscillation converter
- On-chip crystal and RC oscillator
- Watchdog Timer
- Multi-pin capacitor/resistor sensor input
- 1024×14 program memory
- 88×8 data memory RAM
- Power Down and Wake-up function reduce power consumption
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD}=5V
- All instructions executed in one or two machine cycles
- 14-bit table read instruction
- Four-level subroutine nesting
- Bit manipulation instruction
- 63 powerful instructions
- Low voltage reset function
- 20-pin DIP/SOP packages
  24-pin SKDIP/SOP packages

General Description

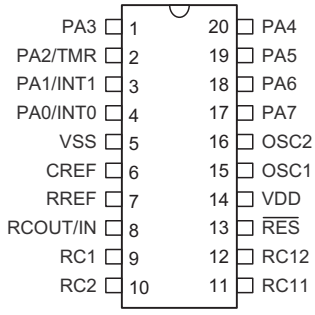
The HT45R34 is an 8-bit high performance, RISC architecture microcontroller device specifically designed for cost-effective multiple I/O control product applications.

The advantages of low power consumption, I/O flexibility, timer functions, oscillator options, Power Down and wake-up functions, Watchdog Timer, enhance the versatility of these devices to suit a wide range of application possibilities such as industrial control, consumer products, subsystem controllers, etc.

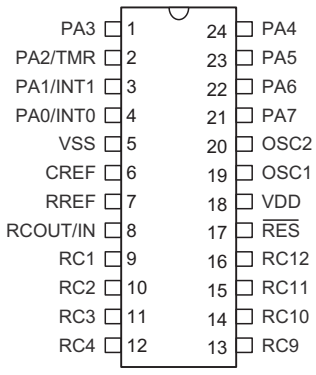
Block Diagram



Pin Assignment



HT45R34
— 20 DIP-A/SOP-A



HT45R34
— 24 SKDIP/SOP-A

Pin Description

Pin Name	I/O	Options	Description
PA0/INT0 PA1/INT1 PA2/TMR PA3~PA7	I/O	Pull-high* Wake-up	Bidirectional 8-bit I/O port. Each pin can be configured as a wake-up input via configuration options. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. Pull-high resistors can be added to the each pin via a configuration option. Pins PA0 and PA1 are pin-shared with external interrupt input pins INT0 and INT1, respectively. Configuration options determine the interrupt enable/disable and the interrupt low/high trigger type. Pins PA2 is pin-shared with the external timer input pins TMR.
RC1~RC12	I	—	Capacitor or resistor connection pins
RCOUT	I	—	Capacitor or resistor connection pin to RC OSC
IN	I	—	Oscillation input pin
RREF	O	—	Reference resistor connection pin
CREF	O	—	Reference capacitor connection pin
$\overline{RES}$	I	—	Schmitt trigger reset input. Active low
VSS	—	—	Negative power supply, ground
VDD	—	—	Positive power supply
OSC1 OSC2	I O	Crystal or RC	OSC1, OSC2 are connected to an RC network or Crystal determined by a configuration option, for the internal system clock. In the case of the RC oscillator, OSC2 can be used to monitor the system clock. Its frequency is 1/4 system clock.

Note: *All pull-high resistors are controlled by an option bit.

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

I_{OL} Total 300mA

Total Power Dissipation 500mW

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

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

I_{OH} Total ..... –200mA

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	V
I_{DD1}	Operating Current (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=4MHz	—	1	2	mA
		5V		—	3	5	mA
I_{DD2}	Operating Current (Crystal OSC, RC OSC)	5V	No load, f_{SYS}=8MHz	—	4	8	mA
I_{STB1}	Standby Current (WDT Enabled)	3V	No load, system HALT	—	—	5	μA
		5V		—	—	10	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{STB2}	Standby Current (WDT Disabled)	3V	No load, system HALT	—	—	1	μA
		5V		—	—	2	μA
V_{IL1}	Input Low Voltage for I/O Ports, TMR, INT0 and INT1	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, TMR, INT0 and INT1	—	—	$0.7V_{DD}$	—	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	—	LVR enabled	2.7	3.0	3.3	V
I_{OL}	PA, RREF and CREF Sink Current	3V	$V_{OL}=0.1V_{DD}$	4	8	—	mA
		5V		10	20	—	mA
I_{OH}	PA, RREF and CREF Source Current	3V	$V_{OH}=0.9V_{DD}$	–2	–4	—	mA
		5V		–5	–10	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock (Crystal OSC, RC OSC)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
f_{TIMER}	Timer I/P Frequency	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{WDT1}	Watchdog Time-out Period (WDT RC OSC)	3V	Without WDT prescaler	11	23	46	ms
		5V		8	17	33	ms
t_{WDT2}	Watchdog Time-out Period (System Clock/4)	—	Without WDT prescaler	—	1024	—	t_{SYS}
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	t_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{LVR}	Low Voltage Reset Time	—	—	0.25	1	2	ms

Note: *$t_{SYS}=1/f_{SYS}$

Functional Description

Execution Flow

The system clock for the microcontroller 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 takes 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 program counter, PC controls the sequence in which the instructions stored in program ROM are executed and its contents specify full range of program memory.

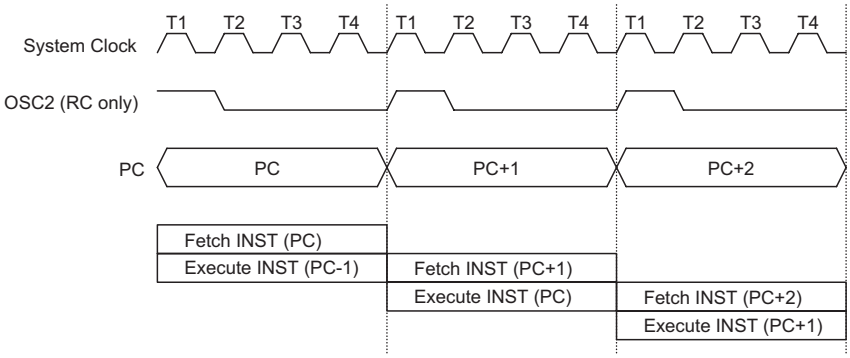
After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by one. The program counter then points to the memory word containing the next instruction code.

When executing a jump instruction, a conditional skip execution, loading the PCL register, a subroutine call, an initial reset, an internal interrupt, an external interrupt or return from a 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 the program will proceed with the next instruction.

The lower byte of the program counter, PCL is a readable and writable register. Moving data into the PCL performs a short jump. The destination must be within the current Program Memory Page.

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



Execution Flow

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	0	1	0	0
External Interrupt 1	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter Overflow	0	0	0	0	0	0	1	1	0	0
External RC Oscillation Converter Interrupt	0	0	0	0	0	1	0	0	0	0
Skip	Program Counter+2									
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

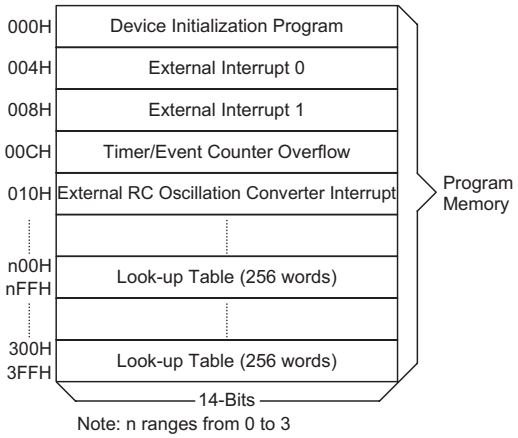
Note: *9~*0: Program Counter bits
#9~#0: Instruction code bits
S9~S0: Stack register bits
@7~@0: PCL bits

Program Memory

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 1024×14 bits, 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 initialisation. After a device reset, the program always begins execution at location 000H.
- Location 004H
  This location is reserved for the external interrupt 0 service program. If the INT0 input pin is activated, the interrupt is enabled and the stack is not full, the program begins execution at this location.
- Location 008H
  This location is reserved for the external interrupt 1 service program. If the INT1 input pin is activated, the interrupt is enabled and the stack is not full, the program begins execution at this location.
- Location 00CH
  This location is reserved for the Timer/Event Counter interrupt service program. If a Timer interrupt results from a Timer/Event Counter overflow, and the interrupt is enabled and the stack is not full, the program begins execution at this location.



Note: n ranges from 0 to 3

Program Memory

- Location 010H
  This location is reserved for the external RC oscillation converter interrupt service program. If an interrupt results from an external RC oscillation converter, and if the interrupt is enabled and the stack is not full, the program begins execution at this location.
- Table location
  Any location in the program memory can be used as a look-up table. The instructions "TABRDC [m]" (the current page, 1 page=256 words) and "TABRDL [m]" transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH. 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 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, which indicates the table location. Before accessing the table, the location must be placed in TBLP. The TBLH register 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 and errors may occur. Therefore, using the table read instruction in the main routine and also in the ISR should be avoided. However, if the table read instruction has to be used in both the main routine and in the ISR, the interrupt should be disabled prior to the table read instruction execution. The interrupt should not be re-enabled until 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 organised into 4-levels and is neither part of the data nor part of the 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 acknowledgment, the contents of the program counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled

Instruction	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

Note: *9~*0: Table location bits

P9~P8: Current program counter bits

@7~@0: Table pointer bits

by a return instruction, RET or RETI, the program counter is restored to its previous value from the stack. After a device reset, the stack pointer 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 acknowledgment 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 as only the most recent 4 return addresses are stored.

Data Memory – RAM

The data memory has a capacity of 111×8 bits. The data memory is divided into two functional groups: special function registers and general purpose data memory (88×8). Most are read/write, but some are read only.

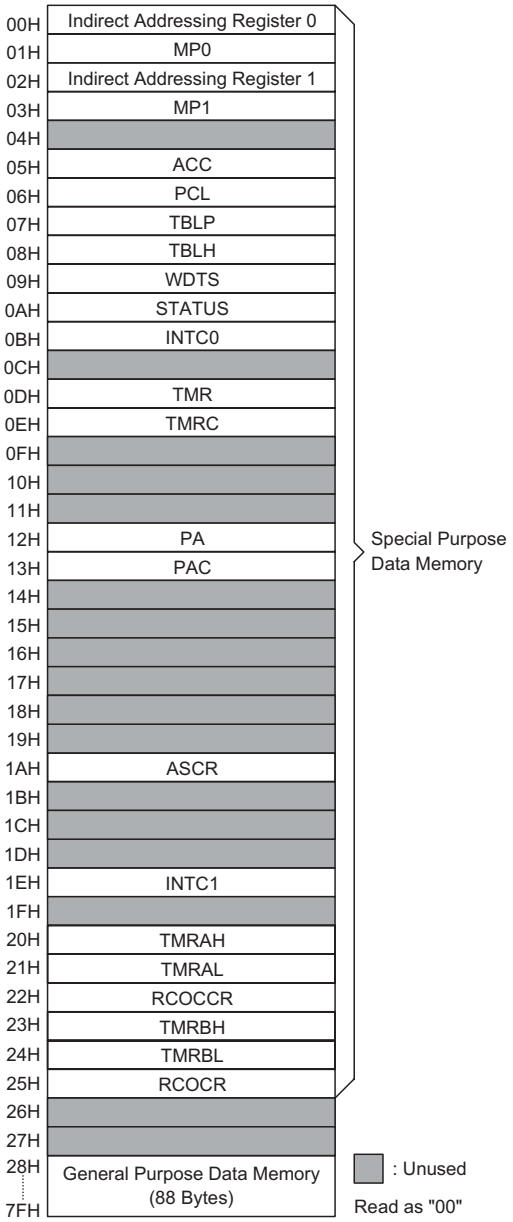
The general purpose data memory, addressed from 28H to 7FH, is used for data and control information under instruction commands.

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 reset by the ″SET [m].i″ and ″CLR [m].i″ bit manipulation instructions. They are also indirectly accessible through the memory pointer registers (MP0;01H, MP1;02H).

Indirect Addressing Register

The method of indirect addressing allows data manipulation using memory pointers instead of the usual direct memory addressing method where the actual memory address is defined. Any action on the indirect addressing registers will result in corresponding read/write operations to the memory location specified by the corresponding memory pointers. This device contains two indirect addressing registers known as IAR0 and IAR1 and two memory pointers MP0 and MP1. Note that these indirect addressing registers are not physically implemented and that reading the indirect addressing registers indirectly will return a result of ″00H″ and writing to the registers indirectly will result in no operation.

The two memory pointers, MP0 and MP1, are physically implemented in the data memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant indirect addressing registers is carried out, the actual address that the microcontroller is directed to is the address specified by the related memory pointer.



RAM Mapping

Bit 7 of the memory pointers are not implemented. However, it must be noted that when the memory pointers in this device is read, a value of ″1″ will be read.

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 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 a ″HALT″ or ″CLR WDT″ instruction or 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 pushed onto the stack automatically. 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 devices provides two external interrupts, one internal 8-bit timer/event counter interrupt and one external RC oscillation converter interrupt. The interrupt control register 0, INTC0, and interrupt control register 1, INTC1, both contain the interrupt control bits that are used to set the enable/disable and interrupt request flags.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the EMI bit will be cleared automatically. However this scheme may prevent 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 EMI bit and the corresponding bit of the INTC0 and INTC1 registers 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 Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full.

All 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 a specified location in the program memory. Only the program counter is pushed onto the stack. If the contents of the accumulator or status register are altered by the interrupt service program, this may corrupt the desired control sequence, therefore their contents should be saved in advance.

External interrupts are triggered by an edge transition on pins INT0 or INT1. A configuration option enables these pins as interrupts and selects if they are active on high to low or low to high transitions. If active their related interrupt request flag, EIF0; bit 4 in INTC0, and EIF1; bit 5 in INTC0, will be set. After the interrupt is en-

Bit No.	Label	Function
0	C	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. C is also affected by a rotate through carry instruction.
1	AC	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.
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 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.
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 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

abled, the stack is not full, and the external interrupt is active, a subroutine call to location ″04H″ or ″08H″ will occur. The interrupt request flags, EIF0 or EIF1, and the EMI bit will all be cleared to disable other interrupts.

The internal Timer/Event Counter interrupt is initialised by setting the Timer/Event Counter interrupt request flag, TF; bit 6 in INTC0. A timer interrupt will be generated when the timer overflows. After the interrupt is enabled, and the stack is not full, and the TF bit is set, a subroutine call to location ″0CH″ will occur. The related interrupt request flag, TF, is reset, and the EMI bit is cleared to disable other interrupts.

The external RC oscillation converter interrupt is initialized by setting the external RC oscillation converter interrupt request flag, RCOCF; bit 4 of INTC1. This is caused by a Timer A or Timer B overflow. When the interrupt is enabled, and the stack is not full and the RCOCF bit is set, a subroutine call to location ″10H″ will occur. The related interrupt request flag, RCOCF, will be reset and the EMI bit cleared to disable further interrupts.

During the execution of an interrupt subroutine, other interrupt acknowledgments 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, a ″RET″ or ″RETI″ instruction 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 0	1	04H
External Interrupt 1	2	08H
Timer/Event Counter Overflow	3	0CH
External RC Oscillation Converter Interrupt	4	10H

Interrupt Priority

The EMI, EEI0, EEI1, ETI and ERCOCI bits are all used to control the enable/disable status of interrupts. These bits prevent the requested interrupt from being serviced. Once the interrupt request flags, TF, RCOCF, EIF1 and EIF0, are all set, they remain in the INTC1 or INTC0 registers respectively 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 may be damaged once the ″CALL″ is executed in the interrupt subroutine.

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1= enabled; 0= disabled)
1	EEI0	Controls the external interrupt 0 (1= enabled; 0= disabled)
2	EEI1	Controls the external interrupt 1 (1= enabled; 0= disabled)
3	ETI	Controls the Timer/Event Counter interrupt (1= enabled; 0= disabled)
4	EIF0	External interrupt 0 request flag (1= active; 0= inactive)
5	EIF1	External interrupt 1 request flag (1= active; 0= inactive)
6	TF	Internal Timer/Event Counter request flag (1= active; 0= inactive)
7	—	Unused bit, read as ″0″

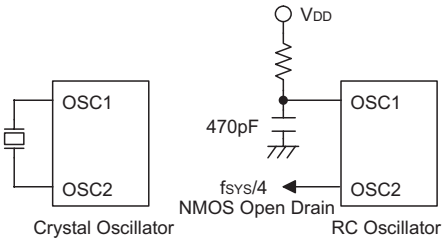
INTC0 (0BH) Register

Bit No.	Label	Function
0	ERCOCI	Controls the external RC oscillation converter interrupt (1= enabled; 0= disabled)
1~3, 5~7	—	Unused bit, read as ″0″
4	RCOCF	External RC oscillation converter request flag (1= active; 0= inactive)

INTC1 (1EH) Register

Oscillator Configuration

There are two oscillator circuits in the microcontroller.



System Oscillator

Both are designed for system clocks, namely the RC oscillator and the Crystal oscillator, the choice of which is determined by a configuration option. When the device enters the Power-down Mode, the system oscillator will stop running and will ignore external signals to conserve power.

If an RC oscillator is used, an external resistor between OSC1 and VDD is required to produce oscillation. The resistance must range from 24kΩ 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 device 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. No other external components are required. Instead of a crystal, a resonator can also be connected between OSC1 and OSC2 to get a frequency reference, but two external capacitors connected between OSC1, OSC2 and ground are required, if the oscillator frequency is less than 1MHz.

The WDT oscillator is a free running on-chip RC oscillator which requires no external components. Even if the system enters the Power Down Mode, where the system clock is stopped, the WDT oscillator will continue to operate with a period of approximately 65μs at 5V. The WDT oscillator can be disabled by a configuration option to conserve power.

Watchdog Timer – WDT

The WDT clock can be sourced from its own dedicated internal oscillator (WDT oscillator), or from the or instruction clock, which is the system clock divided by 4. The choice is determined via a configuration option. The WDT timer is designed to prevent a software malfunction or sequence from jumping to an unknown location with unpredictable results. The Watchdog Timer can be disabled by a configuration option. If the Watchdog Timer is disabled, any executions related to the WDT result in no operation.

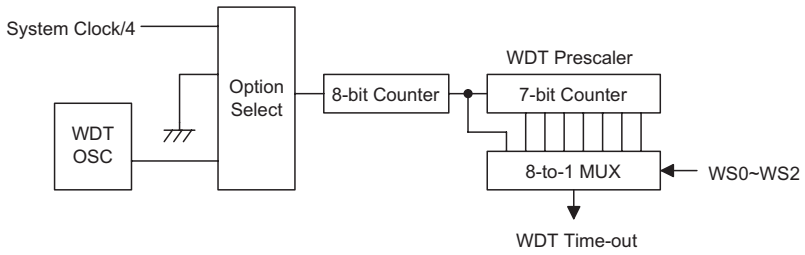
The WDT clock source is first divided by 256. If the internal WDT oscillator is used ,this gives a nominal time-out period of approximately 17ms at 5V. This time-out period may vary with temperatures, VDD and process variations. By using the WDT prescaler, longer time-out periods can be realised. Writing data to the WS2, WS1, WS0 bits in the WDTS register, can give different time-out periods. If WS2, WS1 and WS0 are all equal to 1, the division ratio will be 1:128, and the maximum time-out period will be 2.1s at 5V. If the internal WDT oscillator is disabled, the WDT clock may still come from the instruction clock and operate in the same manner except that in the Power Down state the WDT will stop counting and lose its protecting purpose. The high nibble and bit 3 of the WDTS can be used for user defined flags.

If the device operates in a noisy environment, using the internal WDT oscillator is the recommended choice, since the HALT instruction 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 (09H) Register

The WDT overflow under normal operation will generate a "chip reset" and set the status bit "TO". But in the Power Down mode, the overflow will generate a "warm reset", where only the Program Counter and Stack Pointer are reset to zero. To clear the contents of the WDT, including the WDT prescaler, three methods can be used; an external reset (a low level to $\overline{RES}$), a software instruction and a "HALT" instruction. The software instruction includes "CLR WDT" instruction and the instruction pair – "CLR WDT1" and "CLR WDT2". Of these two types of instruction, only one can be active depending on the configuration option – "CLR WDT times selection option". If the "CLR WDT" is selected, i.e. 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, i.e. 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 a time-out.



Watchdog Timer

Power Down Operation

The Power Down 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 internal WDT oscillator has been selected as the WDT source clock.
- The contents of the on chip RAM and registers remain unchanged.
- The WDT and WDT prescaler will be cleared and will resume counting, if the internal WDT oscillator has been selected as the WDT source clock
- All of the I/O ports will maintain their original status.
- The PDF flag is set and the TO flag is cleared.

The system can leave the Power Down 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 initialisation and the WDT overflow performs a ″warm reset″. After the TO and PDF flags are examined, the reason for the device reset can be determined. The PDF flag is cleared by a system power-up or executing the ″CLR WDT″ instruction and is set when a ″HALT″ instruction is executed. The TO flag is set if a WDT time-out occurs, and causes a wake-up that only resets the program counter and stack pointer; the other registers maintain their their original status.

The port A and interrupt methods of wake-up can be considered as a continuation of normal execution. Each bit in port A can be independently selected by configuration options to wake-up the device. When awakened from an I/O port stimulus, the program will resume execution at the next instruction. If it is awakened due to an interrupt, two sequences may happen. 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 Power Down Mode, the wake-up function of the related interrupt will be disabled. Once a wake-up event occurs, it takes 1024 t_{SYS} (system clock periods) to resume normal operation. A dummy period is therefore inserted after wake-up. If the wake-up results from an interrupt acknowledgment, 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 minimise power consumption, all the I/O pins should be carefully managed before entering the Power Down mode.

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

A WDT time-out, when the device is in the Power Down mode, is different from other device reset conditions, in that it can perform a ″warm reset″ that resets only the Program Counter and the Stack Poiner, leaving the other circuits in their original state. Some registers remain unchanged during other reset conditions. Most registers are reset to their ″initial condition″ when the reset conditions are met. By examining the PDF and TO flags, the program can distinguish between the different device reset types.

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″ means ″unchanged″

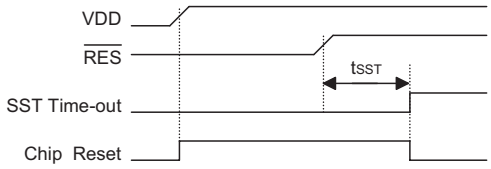
To guarantee that the system oscillator is started and stabilised, the SST or System Start-up Timer, provides an extra-delay of 1024 system clock pulses when the system is reset (power-up, WDT time-out or $\overline{RES}$ reset) or when the system awakens from a Power Down state.

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

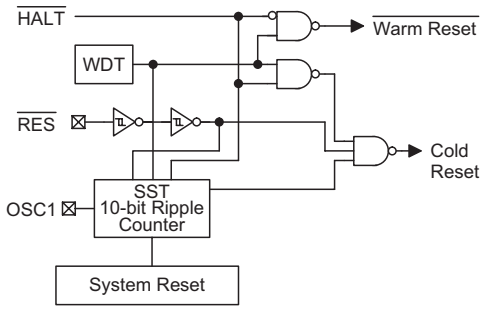
An extra option load time delay is added during a system reset (power-up, WDT time-out during normal mode or a $\overline{RES}$ reset).

The functional unit device reset status are shown below.

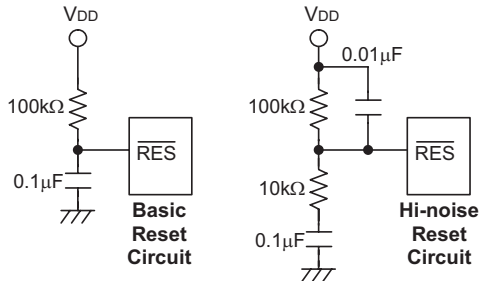
Program Counter	000H
Interrupt	Disable
Prescaler	Clear
WDT	Clear. After master reset, WDT begins counting
Timer/Event Counter	Off
Input/Output Ports	Input mode
Stack Pointer	Points to the top of the stack



Reset Timing Chart



Reset Configuration



Reset Circuit

Note: Most applications can use the Basic Reset Circuit as shown, however for applications with extensive noise, it is recommended to use the Hi-noise Reset Circuit.

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)*
MP0	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
MP1	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
Program Counter	000H	000H	000H	000H	000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu

Register	Reset (Power-on)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
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
ASCR	---1 1111	---1 1111	---1 1111	---1 1111	---u uuuu
INTC1	---0 ---0	---0 ---0	---0 ---0	---0 ---0	---u ---u
TMRAH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRAL	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RCOCCR	0000 1---	0000 1---	0000 1---	0000 1---	uuuu u---
TMRBH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRBL	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RCOCR	1xxx --00	1xxx --00	1xxx --00	1xxx --00	uuuu --uu

Note: ″*″ means ″warm reset″
″u″ means ″unchanged″
″x″ means ″unknown″

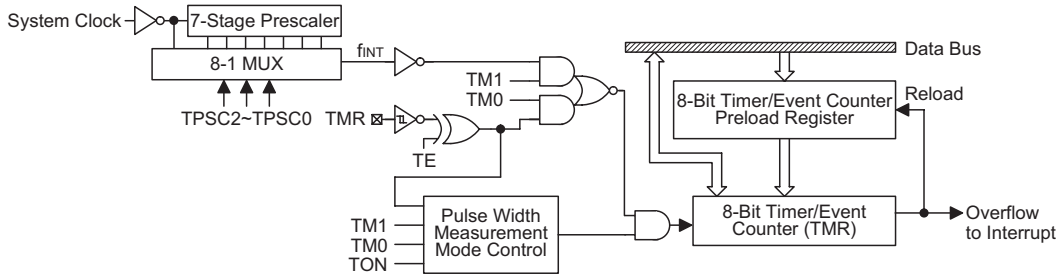
Timer/Event Counter

An 8-bit timer/event counter, known as Timer/Event Counter, is implemented in the microcontroller. The Timer/Event Counter contains an 8-bit programmable count-up counter whose clock may come from an external source or from the system clock. Using the external clock input allows the user to count external events, measure time internals or pulse widths, or generate an accurate time base. Using the internal clock allows the user to generate an accurate time base.

There are 2 registers related to the Timer/Event Counter, TMR and TMRC. Two physical registers are mapped to the TMR location; writing to TMR places the start value of the Timer/Event Counter in a preload register while reading TMR retrieves the contents of the Timer/Event Counter. The TMRC is a timer/event counter control register, which defines the timer operating conditions.

Bit No.	Label	Function
0~2	TPSC0~TPSC2	To define the prescaler stages, TPSC2, TPSC1, TPSC0= 000: $f_{INT}=f_{SYS}$ 001: $f_{INT}=f_{SYS}/2$ 010: $f_{INT}=f_{SYS}/4$ 011: $f_{INT}=f_{SYS}/8$ 100: $f_{INT}=f_{SYS}/16$ 101: $f_{INT}=f_{SYS}/32$ 110: $f_{INT}=f_{SYS}/64$ 111: $f_{INT}=f_{SYS}/128$
3	TE	To define the TMR active edge of the timer/event counter (0=active on low to high; 1=active on high to low)
4	TON	To enable or disable timer counting (0=disabled; 1=enabled)
5	—	Unused bit, read as ″0″
6 7	TM0 TM1	To 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



Timer/Event Counter

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. The pulse width measurement mode can be used to measure the high or low level duration of an external signal on the TMR pin. The counting is based on the f_{INT} clock source. In the event counting 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 an interrupt request flag TF; bit 5 of INTC0, is generated at the same time.

In the pulse width measurement mode, with the TON and TE bits equal to one, once the TMR pin has received a transient from low to high, or high to low if the TE bit is 0, it will start counting until the TMR pin returns to its original level and resets the TON bit. The measured result will remain in the Timer/Event Counter even if the activated transient occurs again. Therefore, only a single shot measurement can be made. The TON bit must be set again by software for further measurements to be made. 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 a counter overflow, the counter is reloaded from the Timer/Event Counter preload register and issues an interrupt request just like the other two modes.

To enable a 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 Timer/Event Counter overflow is one of the wake-up sources. No matter what the operation mode is, writing a 0 to ETI can disable the interrupt service.

If the Timer/Event Counter is switched off, then writing data to the Timer/Event Counter preload register will also directly reload that data to the Timer/Event Counter. But if the Timer/Event Counter is already running, data written to it will only be loaded into the Timer/Event Counter preload register. The Timer/Event Counter will continue to operate until an overflow occurs. When the Timer/Event Counter is read, the clock will be blocked to avoid errors. As clock blocking may results in a counting error, this must be taken into consideration by the programmer. Bit0~Bit2 of the TMRC register can be used to define the pre-scaling stages of the internal clock source of the Timer/Event Counter.

External RC Oscillation Converter

An external RC oscillation mode is implemented in the device. The RC oscillation converter contains two 16-bit programmable count-up counters.

The RC oscillation converter is comprised of the TMRAL, TMRAH, TMRBL, TMRBH registers when the RCO bit, bit 1 of RCOCR register, is ″1″. The RC oscillation converter Timer B clock source may come from an external RC oscillator. The Timer A clock source comes from the system clock or from the system clock/4, determined by the RCOCCR register.

There are six registers related to the RC oscillation converter, i.e., TMRAH, TMRAL, RCOCCR, TMRBH, TMRBL and RCOCR. The internal timer clock is the input to TMRAH and TMRAL, the external RC oscillation is the input to TMRBH and TMRBL. The OVB bit, bit 0 of the RCOCR register, decides whether Timer A overflows or Timer B overflows, then the RCOCF bit is set and an external RC oscillation converter interrupt occurs. When the RC oscillation converter mode Timer A or Timer B overflows, the RCOCON bit is reset to ″0″ and stops counting. Writing to TMRAH/TMRBH places the start value in Timer A/Timer B while reading TMRAH/TMRBH obtains the contents of Timer A/Timer B. Writing to TMRAL/TMRBL only writes the data into a low byte buffer. However writing to TMRAH/TMRBH will write the data and the contents of the low byte buffer into the Timer A/Timer B (16-bit) simultaneously. Timer A/Timer B is changed by writing to TMRAH/TMRBH but writing to TMRAL/TMRBL will keep the Timer A/Timer B unchanged.

Reading TMRAH/TMRBH will also latch the TMRAL/TMRBL into the low byte buffer to avoid false timing problem. Reading TMRAL/TMRBL returns the contents of the low byte buffer. Therefore, the low byte

of Timer A/Timer B can not be read directly. It must read TMRAH/TMRBH first to ensure that the low byte contents of Timer A/Timer B are latched into the buffer.

The resistor and capacitor form an oscillation circuit and input to TMRBH and TMRBL. The RCOM0, RCOM1 and RCOM2 bits of RCOCCR define the clock source of Timer A. It is recommended that the clock source of Timer A uses the system clock or the instruction clock.

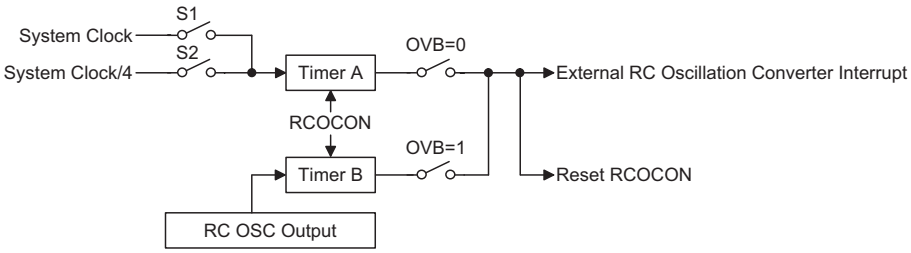
If the RCOCON bit, bit 4 of RCOCCR, is set to "1", Timer A and Timer B will start counting until Timer A or Timer B overflows, the timer/event counter will then generate an interrupt request flag which is RCOCF; bit 4 of INTC1. The Timer A and Timer B will stop counting and will reset the RCOCON bit to "0" at the same time. If the RCOCON bit is "1", TMRAH, TMRAL, TMRBH and TMRBL cannot be read or written.

Bit No.	Label	Function
0~2	—	Unused bit, read as "0"
3	—	Undefined bit, this bit can read/write
4	RCOCON	Enable or disable external RC oscillation converter counting (0= disabled; 1= enabled)
5 6 7	RCOM0 RCOM1 RCOM2	Define the Timer A clock source, RCOM2, RCOM1, RCOM0= 000= System clock 001= System clock/4 010= Unused 011= Unused 100= Unused 101= Unused 110= Unused 111= Unused

RCOCCR (22H) Register

Bit No.	Label	Function
0	OVB	In the RC oscillation converter mode, this bit is used to define the timer/event counter interrupt, which comes from Timer A overflow or Timer B overflow. (0= Timer A overflow; 1= Timer B overflow)
1	RCO	Define RC oscillation converter mode. (0= Disable RC oscillation converter mode; 1= Enable RC oscillation converter mode)
2~3	—	Unused bit, read as "0"
4~7	RW	4-bit read/write registers for user defined.

RCOCR (25H) Register



External RC Oscillation Converter

External RC oscillation converter mode example program - Timer A overflow:

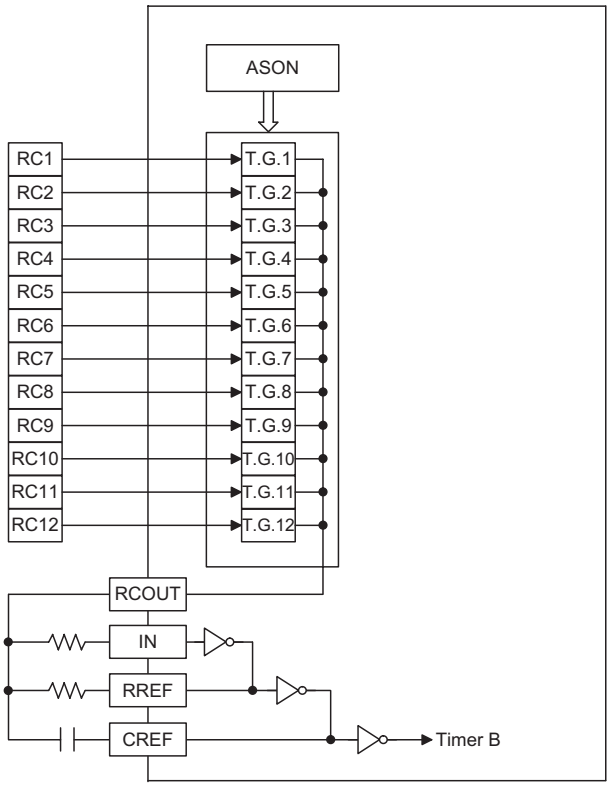
```
clr RCOCCR
mov a, 00000010b            ; Enable External RC oscillation mode and set Timer A overflow
mov RCOCR,a
clr intc1.4                 ; Clear External RC Oscillation Converter interrupt request flag
mov a, low (65536-1000)     ; Give timer A initial value
mov tmral, a                ; Timer A count 1000 time and then overflow
mov a, high (65536-1000)
mov tmrah, a
mov a, 00h                  ; Give timer B initial value
mov tmrbl, a
mov a, 00h
mov tmrbh, a
mov a, 00110000b            ; Timer A clock source=fSYS/4 and timer on
mov RCOCCR, a
p10:
clr wdt
snz intc1.4                 ; Polling External RC Oscillation Converter interrupt request flag
jmp p10
clr intc1.4                 ; Clear External RC Oscillation Converter interrupt request flag
                            ; Program continue
```

Analog Switch

There are 12 analog switch lines in the microcontroller for RC1~RC12, and a corresponding Analog Switch control register, which is mapped to the data memory of ″1AH″.

Bit No.	Label	Function
0~4	ASON	Defines the analog switch for RC1~RC12 which is on. ASON= 00000b= Analog switch 1 on, other analog switch off 00001b= Analog switch 2 on, other analog switch off 00010b= Analog switch 3 on, other analog switch off 00011b= Analog switch 4 on, other analog switch off 00100b= Analog switch 5 on, other analog switch off 00101b= Analog switch 6 on, other analog switch off 00110b= Analog switch 7 on, other analog switch off 00111b= Analog switch 8 on, other analog switch off 01000b= Analog switch 9 on, other analog switch off 01001b= Analog switch 10 on, other analog switch off 01010b= Analog switch 11 on, other analog switch off 01011b= Analog switch 12 on, other analog switch off 01100b= All analog switch off 01101b= All analog switch off 01110b= All analog switch off 01111b= All analog switch off 1xxxxb= All analog switch off and RC OSC always off.
5~7	—	Unused bit, read as ″0″

ASCR (1AH) Register



Analog Switch

Input/Output Ports

There are 8 bidirectional input/output lines in the microcontroller, all located within port PA. 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 the "MOV A,[m]" instruction. 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, known as PAC, to control the input/output configuration. With this control register, the pin status is either a CMOS output or a Schmitt trigger input, but can be reconfigured dynamically, under software control. To function as an input, the corresponding bit in the control register must be written with a "1". The input source also depends 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.

When setup as an output the output types are CMOS.

After a device reset, the I/O ports will be initially all setup as inputs, and will therefore be in a high state if the configuration options have selected pull-high resistors, otherwise they will be in a floating condition. Each bit of these input/output latches can be set or cleared by the "SET [m].i" and "CLR [m].i" 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.

Each line of port A has a pull-high option. Once the pull-high option is selected, the I/O line will have a pull-high resistor connected. Otherwise, the pull-high resistors are absent. It should be noted that a non-pull-high I/O line operating in an input mode will be in a floating state.

The PA0, PA1 and PA2 are pin-shared with INT0, INT1 and TMR pins, respectively.

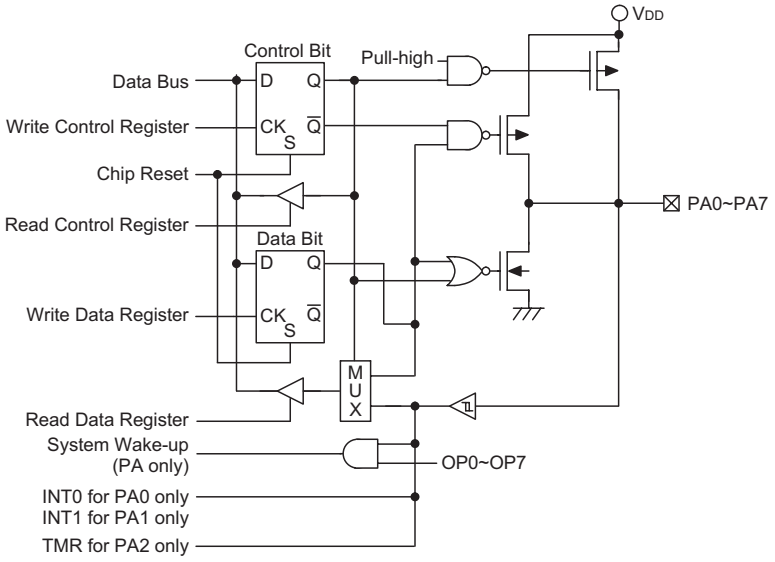
It is recommended that unused or not bonded out I/O lines should be set as output pins using software instruction to avoid consuming power under input floating state.

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 when changing a battery, the LVR will automatically reset the device internally.

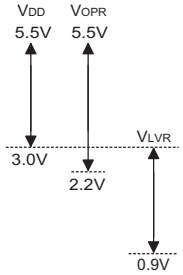
The LVR includes the following specifications:

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

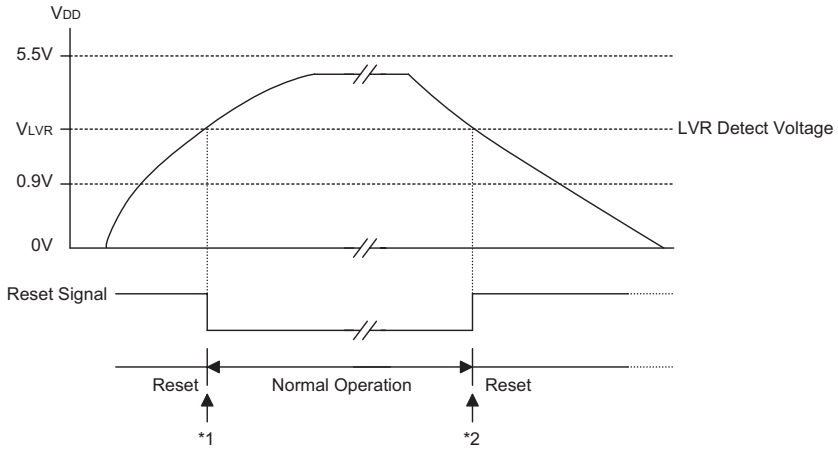


Input/Output Ports

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 starting the normal operation.

*2: Since low voltage has to be maintained its original state for longer than t_{LVR}, therefore a t_{LVR} delay enters the reset mode.

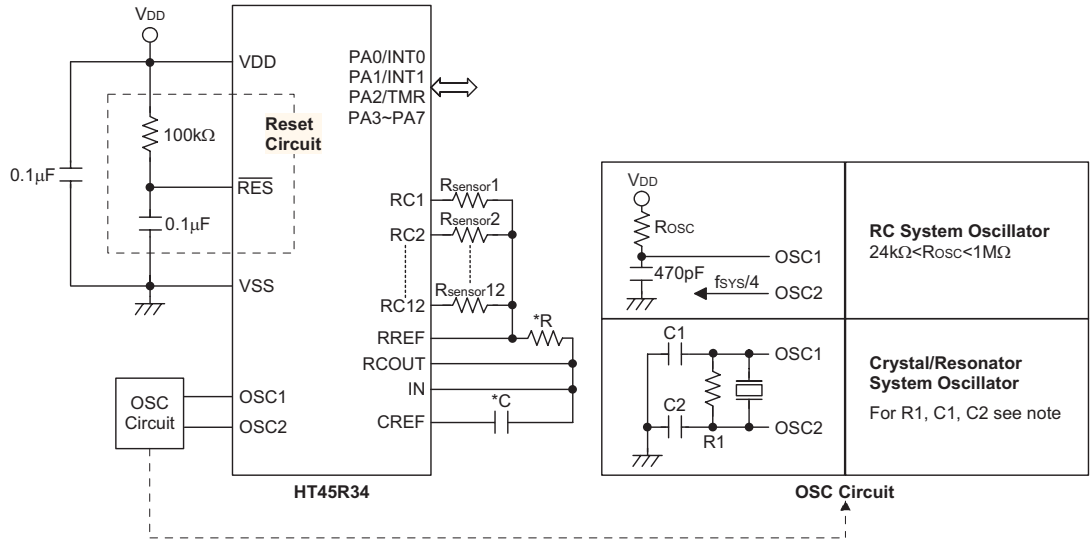
Options

The following table shows the various options within the microcontroller. All of the options must be defined to ensure proper system functioning.

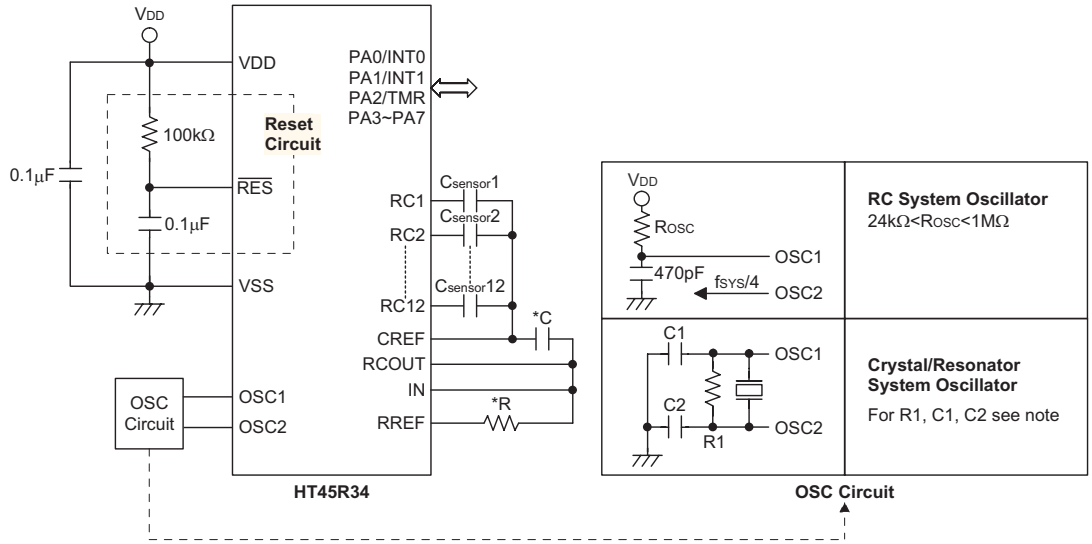
No.	Function	Description
1	Wake up PA0~PA7 (bit option)	None wake-up or wake-up
2	Pull high PA0~PA7 (bit option)	None pull-high or pull-high
3	WDT clock source	WDTOSC or $f_{SYS}/4$
4	WDT	Enable or disable
5	CLRWDT	1 or 2 instructions
6	LVR	Enable or disable
7	OSC	X′tal mode or RC mode
8	INT0 trigger edge	Disable, rising edge, falling edge or double edge
9	INT1 trigger edge	Disable, rising edge, falling edge or double edge

Application Circuits

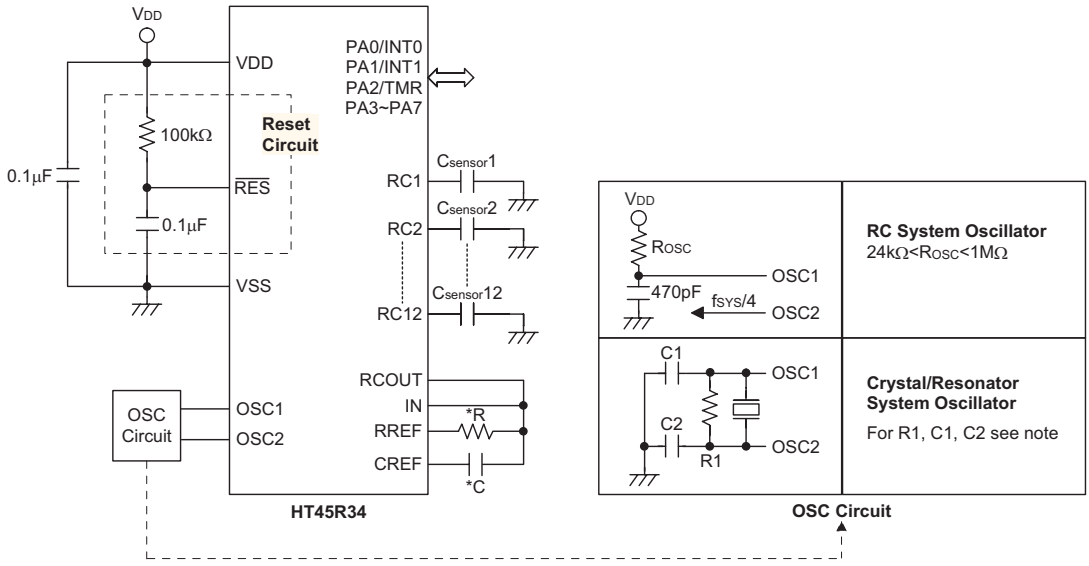
R to F Application Circuit



C to F Application Circuit 1



C to F Application Circuit 2



Note: 1. Crystal/resonator system oscillators
For crystal oscillators, C1 and C2 are only required for some crystal frequencies to ensure oscillation. For resonator applications C1 and C2 are normally required for oscillation to occur. For most applications it is not necessary to add R1. However if the LVR function is disabled, and if it is required to stop the oscillator when VDD falls below its operating range, it is recommended that R1 is added. The values of C1 and C2 should be selected in consultation with the crystal/resonator manufacturer specifications.

2. Reset circuit
The reset circuit resistance and capacitance values should be chosen to ensure that VDD is stable and remains within its operating voltage range before the $\overline{RES}$ pin reaches a high level. Ensure that the length of the wiring connected to the $\overline{RES}$ pin is kept as short as possible, to avoid noise interference.

3. For applications where noise may interfere with the reset circuit and for details on the oscillator external components, refer to Application Note HA0075E for more information.

4. The "*R" resistance and "*C" capacitance should be consideration for the frequency of RC OSC.

5. $R_{sensor}1$~$R_{sensor}12$ are the resistance sensors.

6. $C_{sensor}1$~$C_{sensor}12$ are the capacitance sensors.

Instruction Set

Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek microcontrollers, a comprehensive and flexible set of over 60 instructions is provided to enable programmers to implement their application with the minimum of programming overheads.

For easier understanding of the various instruction codes, they have been subdivided into several functional groupings.

Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5μs and branch or call instructions would be implemented within 1μs. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be ″CLR PCL″ or ″MOV PCL, A″. For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of three kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

Logical and Rotate Operations

The standard logical operations such as AND, OR, XOR and CPL all have their own instruction within the Holtek microcontroller instruction set. As with the case of most instructions involving data manipulation, data must pass through the Accumulator which may involve additional programming steps. In all logical data operations, the zero flag may be set if the result of the operation is zero. Another form of logical data manipulation comes from the rotate instructions such as RR, RL, RRC and RLC which provide a simple means of rotating one bit right or left. Different rotate instructions exist depending on program requirements. Rotate instructions are useful for serial port programming applications where data can be rotated from an internal register into the Carry bit from where it can be examined and the necessary serial bit set high or low. Another application where rotate data operations are used is to implement multiplication and division calculations.

Branches and Control Transfer

Program branching takes the form of either jumps to specified locations using the JMP instruction or to a subroutine using the CALL instruction. They differ in the sense that in the case of a subroutine call, the program must return to the instruction immediately when the subroutine has been carried out. This is done by placing a return instruction RET in the subroutine which will cause the program to jump back to the address right after the CALL instruction. In the case of a JMP instruction, the program simply jumps to the desired location. There is no requirement to jump back to the original jumping off point as in the case of the CALL instruction. One special and extremely useful set of branch instructions are the conditional branches. Here a decision is first made regarding the condition of a certain data memory or individual bits. Depending upon the conditions, the program will continue with the next instruction or skip over it and jump to the following instruction. These instructions are the key to decision making and branching within the program perhaps determined by the condition of certain input switches or by the condition of internal data bits.

Bit Operations

The ability to provide single bit operations on Data Memory is an extremely flexible feature of all Holtek microcontrollers. This feature is especially useful for output port bit programming where individual bits or port pins can be directly set high or low using either the "SET [m].i" or "CLR [m].i" instructions respectively. The feature removes the need for programmers to first read the 8-bit output port, manipulate the input data to ensure that other bits are not changed and then output the port with the correct new data. This read-modify-write process is taken care of automatically when these bit operation instructions are used.

Table Read Operations

Data storage is normally implemented by using registers. However, when working with large amounts of fixed data, the volume involved often makes it inconvenient to store the fixed data in the Data Memory. To overcome this problem, Holtek microcontrollers allow an area of Program Memory to be setup as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

Other Operations

In addition to the above functional instructions, a range of other instructions also exist such as the "HALT" instruction for Power-down operations and instructions to control the operation of the Watchdog Timer for reliable program operations under extreme electric or electromagnetic environments. For their relevant operations, refer to the functional related sections.

Instruction Set Summary

The following table depicts a summary of the instruction set categorised according to function and can be consulted as a basic instruction reference using the following listed conventions.

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1Note	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	1Note	Z, C, AC, OV
SUB A,x	Subtract immediate data from the 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	1Note	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, result in Data Memory	1Note	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	C
Logic Operation			
AND A,[m]	Logical AND Data Memory to ACC	1	Z
OR A,[m]	Logical OR Data Memory to ACC	1	Z
XOR A,[m]	Logical XOR Data Memory to ACC	1	Z
ANDM A,[m]	Logical AND ACC to Data Memory	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	1Note	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			

Mnemonic	Description	Cycles	Flag Affected
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1Note	C
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1Note	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1Note	None
SET [m].i	Set bit of Data Memory	1Note	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1Note	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1note	None
SZ [m].i	Skip if bit i of Data Memory is zero	1Note	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1Note	None
SIZ [m]	Skip if increment Data Memory is zero	1Note	None
SDZ [m]	Skip if decrement Data Memory is zero	1Note	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1Note	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1Note	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 table (current page) to TBLH and Data Memory	2Note	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2Note	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1Note	None
SET [m]	Set Data Memory	1Note	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1Note	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.
2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.
3. For the ″CLR WDT1″ and ″CLR WDT2″ instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both ″CLR WDT1″ and ″CLR WDT2″ instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

ADC A,[m]	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C
ADCM A,[m]	Add ACC to Data Memory with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C
ADD A,[m]	Add Data Memory to ACC
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)	OV, Z, AC, C
ADD A,x	Add immediate data to ACC
Description	The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + x
Affected flag(s)	OV, Z, AC, C
ADDM A,[m]	Add ACC to Data Memory
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m]
Affected flag(s)	OV, Z, AC, C
AND A,[m]	Logical AND Data Memory to ACC
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)	Z
AND A,x	Logical AND immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical AND operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″AND″ x
Affected flag(s)	Z
ANDM A,[m]	Logical AND ACC to Data Memory
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)	Z

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None
CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None
CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None
CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

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) Z

CPLA [m] Complement Data Memory with result in ACC

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

Affected flag(s) Z

DAA [m] Decimal-Adjust ACC for addition with result in Data Memory

Description Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.

Operation $[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

Affected flag(s) C

DEC [m] Decrement Data Memory

Description Data in the specified Data Memory is decremented by 1.

Operation $[m] \leftarrow [m] - 1$

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

Description Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

HALT Enter power down mode

Description This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation $TO \leftarrow 0$
$PDF \leftarrow 1$

Affected flag(s) TO, PDF

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z
INCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] + 1
Affected flag(s)	Z
JMP addr	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter ← addr
Affected flag(s)	None
MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None
MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC ← x
Affected flag(s)	None
MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None
NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None
OR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ [m]
Affected flag(s)	Z

OR A,x Logical OR immediate data to ACC

Description: Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation: ACC ← ACC "OR" x

Affected flag(s): Z

ORM A,[m] Logical OR ACC to Data Memory

Description: Data in the specified 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): Z

RET Return from subroutine

Description: The Program Counter is restored from the stack. Program execution continues at the restored address.

Operation: Program Counter ← Stack

Affected flag(s): None

RET A,x Return from subroutine and load immediate data to ACC

Description: The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.

Operation: Program Counter ← Stack
ACC ← x

Affected flag(s): None

RETI Return from interrupt

Description: The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.

Operation: Program Counter ← Stack
EMI ← 1

Affected flag(s): None

RL [m] Rotate Data Memory left

Description: The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.

Operation: [m].(i+1) ← [m].i; (i = 0~6)
[m].0 ← [m].7

Affected flag(s): None

RLA [m] Rotate Data Memory left with result in ACC

Description: The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation: ACC.(i+1) ← [m].i; (i = 0~6)
ACC.0 ← [m].7

Affected flag(s): None

RLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation
[m].(i+1) ← [m].i; (i = 0~6)
[m].0 ← C
C ← [m].7

Affected flag(s) C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation
ACC.(i+1) ← [m].i; (i = 0~6)
ACC.0 ← C
C ← [m].7

Affected flag(s) C

RR [m] Rotate Data Memory right

Description The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.

Operation
[m].i ← [m].(i+1); (i = 0~6)
[m].7 ← [m].0

Affected flag(s) None

RRA [m] Rotate Data Memory right with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation
ACC.i ← [m].(i+1); (i = 0~6)
ACC.7 ← [m].0

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation
[m].i ← [m].(i+1); (i = 0~6)
[m].7 ← C
C ← [m].0

Affected flag(s) C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation
ACC.i ← [m].(i+1); (i = 0~6)
ACC.7 ← C
C ← [m].0

Affected flag(s) C

SBC A,[m] Subtract Data Memory from ACC with Carry

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

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

Affected flag(s) OV, Z, AC, C

SBCM A,[m] Subtract Data Memory from ACC with Carry and result in Data Memory

Description The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

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

Affected flag(s) OV, Z, AC, C

SDZ [m] Skip if decrement Data Memory is 0

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation $[m] \leftarrow [m] - 1$
Skip if [m] = 0

Affected flag(s) None

SDZA [m] Skip if decrement Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation $ACC \leftarrow [m] - 1$
Skip if ACC = 0

Affected flag(s) None

SET [m] Set Data Memory

Description Each bit of the specified Data Memory is set to 1.

Operation $[m] \leftarrow FFH$

Affected flag(s) None

SET [m].i Set bit of Data Memory

Description Bit i of the specified Data Memory is set to 1.

Operation $[m].i \leftarrow 1$

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation [m] ← [m] + 1
Skip if [m] = 0

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

Description The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

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

Affected flag(s) None

SNZ [m].i Skip if bit i of Data Memory is not 0

Description If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC – [m]

Affected flag(s) OV, Z, AC, C

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

Description The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation [m] ← ACC – [m]

Affected flag(s) OV, Z, AC, C

SUB A,x Subtract immediate data from ACC

Description The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation ACC ← ACC – x

Affected flag(s) OV, Z, AC, C

SWAP [m] Swap nibbles of Data Memory

Description The low-order and high-order nibbles of the specified Data Memory are interchanged.

Operation [m].3~[m].0 ↔ [m].7 ~ [m].4

Affected flag(s) None

SWAPA [m] Swap nibbles of Data Memory with result in ACC

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in 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) None

SZ [m] Skip if Data Memory is 0

Description If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m] = 0

Affected flag(s) None

SZA [m] Skip if Data Memory is 0 with data movement to ACC

Description The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation ACC ← [m]
Skip if [m] = 0

Affected flag(s) None

SZ [m].i Skip if bit i of Data Memory is 0

Description If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.

Operation Skip if [m].i = 0

Affected flag(s) None

TABRDC [m] Read table (current page) to TBLH and Data Memory

Description The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

TABRDL [m] Read table (last page) to TBLH and Data Memory

Description The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

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

Affected flag(s) None

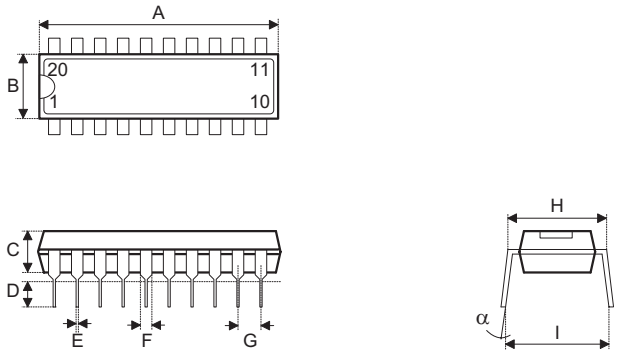
XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

XORM A,[m]	Logical XOR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″XOR″ [m]
Affected flag(s)	Z

XOR A,x	Logical XOR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ x
Affected flag(s)	Z

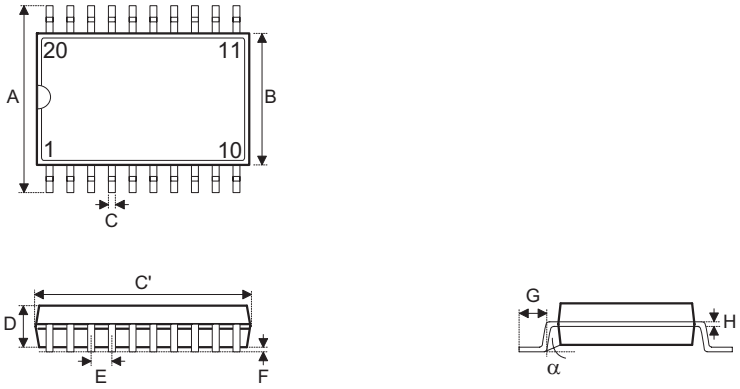
Package Information

20-pin DIP (300mil) Outline Dimensions



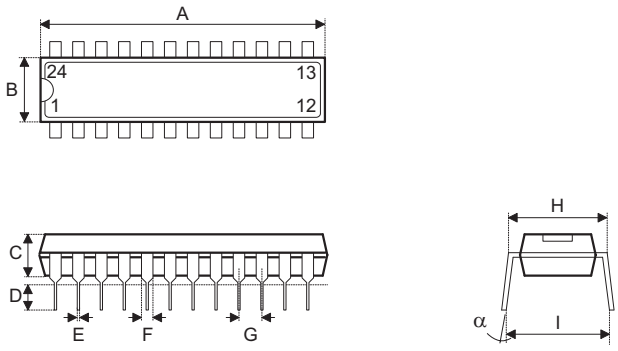
Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1020	—	1045
B	240	—	260
C	125	—	135
D	125	—	145
E	16	—	20
F	50	—	70
G	—	100	—
H	295	—	315
I	335	—	375
α	0°	—	15°

20-pin SOP (300mil) Outline Dimensions



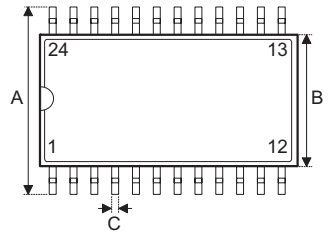
Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	394	—	419
B	290	—	300
C	14	—	20
C′	490	—	510
D	92	—	104
E	—	50	—
F	4	—	—
G	32	—	38
H	4	—	12
α	0°	—	10°

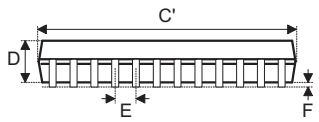
24-pin SKDIP (300mil) Outline Dimensions

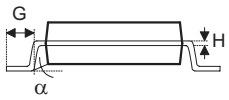


Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1235	—	1265
B	255	—	265
C	125	—	135
D	125	—	145
E	16	—	20
F	50	—	70
G	—	100	—
H	295	—	315
I	345	—	360
α	0°	—	15°

24-pin SOP (300mil) Outline Dimensions



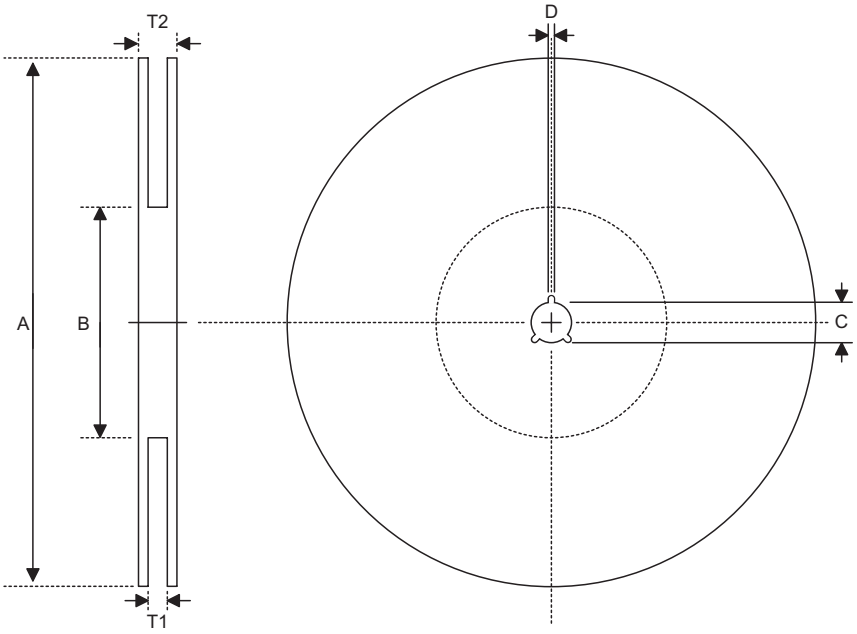




Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	394	—	419
B	290	—	300
C	14	—	20
C′	590	—	614
D	92	—	104
E	—	50	—
F	4	—	—
G	32	—	38
H	4	—	12
α	0°	—	10°

Product Tape and Reel Specifications

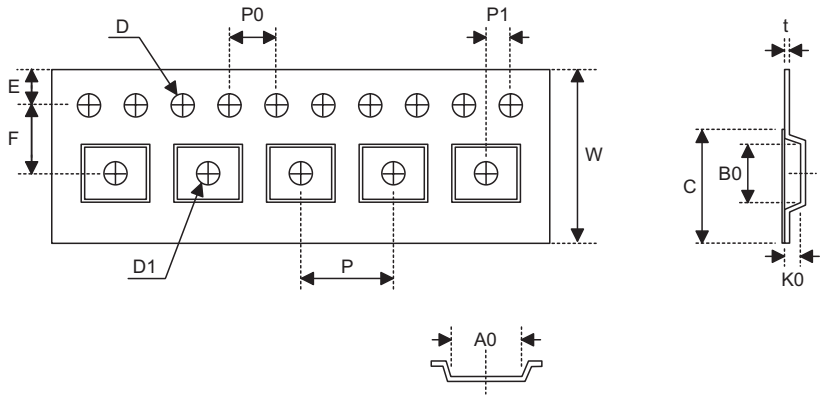
Reel Dimensions



SOP 20W, SOP 24W

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1
B	Reel Inner Diameter	62±1.5
C	Spindle Hole Diameter	13+0.5 −0.2
D	Key Slit Width	2±0.5
T1	Space Between Flange	24.8+0.3 −0.2
T2	Reel Thickness	30.2±0.2

Carrier Tape Dimensions



SOP 20W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24+0.3 −0.1
P	Cavity Pitch	12±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5+0.1
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4±0.1
P1	Cavity to Perforation (Length Direction)	2±0.1
A0	Cavity Length	10.8±0.1
B0	Cavity Width	13.3±0.1
K0	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.3±0.05
C	Cover Tape Width	21.3

SOP 24W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24±0.3
P	Cavity Pitch	12±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.55+0.1
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4±0.1
P1	Cavity to Perforation (Length Direction)	2±0.1
A0	Cavity Length	10.9±0.1
B0	Cavity Width	15.9±0.1
K0	Cavity Depth	3.1±0.1
t	Carrier Tape Thickness	0.35±0.05
C	Cover Tape Width	21.3

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: 86-21-6485-5560
Fax: 86-21-6485-0313
http://www.holtek.com.cn

Holtek Semiconductor Inc. (Shenzhen Sales Office)
5/F, Unit A, Productivity Building, Cross of Science M 3rd Road and Gaoxin M 2nd Road, Science Park, Nanshan District, Shenzhen, China 518057
Tel: 86-755-8616-9908, 86-755-8616-9308
Fax: 86-755-8616-9722

Holtek Semiconductor Inc. (Beijing Sales Office)
Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031
Tel: 86-10-6641-0030, 86-10-6641-7751, 86-10-6641-7752
Fax: 86-10-6641-0125

Holtek Semiconductor Inc. (Chengdu Sales Office)
709, Building 3, Champagne Plaza, No.97 Dongda Street, Chengdu, Sichuan, China 610016
Tel: 86-28-6653-6590
Fax: 86-28-6653-6591

Holtek Semiconductor (USA), Inc. (North America Sales Office)
46729 Fremont Blvd., Fremont, CA 94538
Tel: 1-510-252-9880
Fax: 1-510-252-9885
http://www.holtek.com