

**Features**

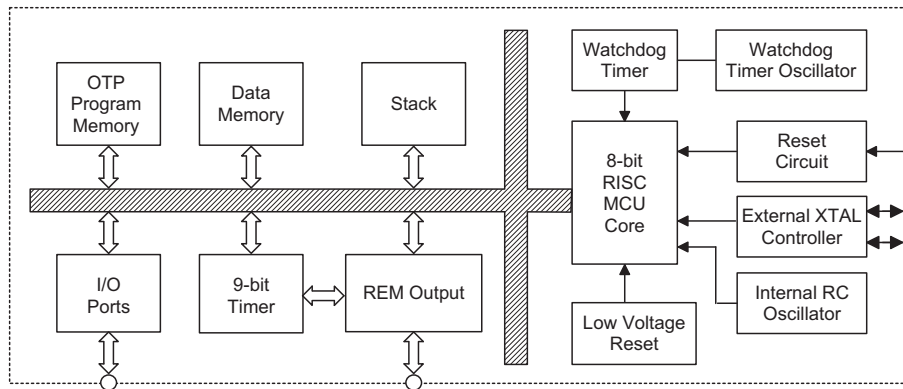
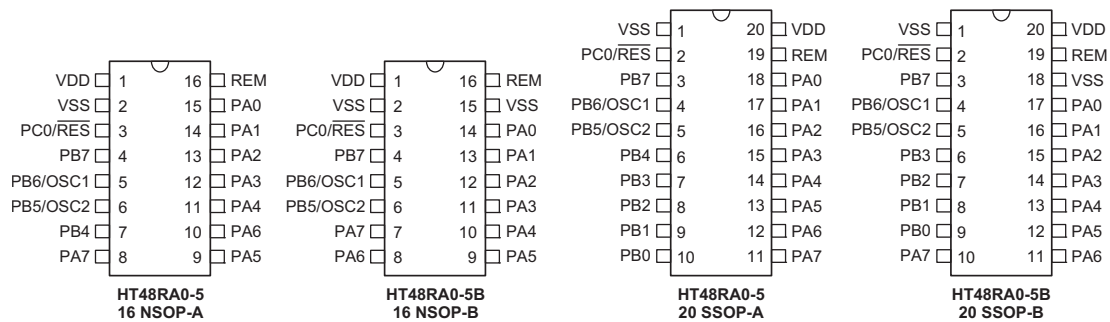
- Operating voltage:  
f<sub>SYS</sub>=4MHz at V<sub>DD</sub>=2.0V~3.6V (LVR enabled)  
f<sub>SYS</sub>=4MHz at V<sub>DD</sub>=1.8V~3.6V (LVR disabled)
- Oscillator types:  
External high frequency Crystal -- HXT  
Internal high frequency RC -- HIRC
- 1K×14 program memory
- 32×8 data RAM
- One-level subroutine nesting
- 17 bidirectional I/O lines
- Fully integrated internal 4095kHz oscillator requires no external components
- One programmable carrier output - using 9-bit timer
- Carrier output pin (REM)
- Build-in IR Driver (350mA@3.0V)
- Watchdog Timer
- Low voltage reset function
- Power-down and wake-up features reduce power consumption
- 14-bit table read instructions
- Up to 1μs instruction cycle with 4MHz system clock
- 62 powerful instructions
- All instructions executed in 1 or 2 machine cycles
- Bit manipulation instructions
- HT48RA0-5 use in external IR driver
- HT48RA0-5B use in internal IR driver
- 16-pin NSOP and 20-pin SSOP packages

**General Description**

The HT48RA0-5 is 8-bit high performance, RISC architecture microcontroller device specifically designed for multiple I/O control product applications.

The advantages of low power consumption, I/O flexibility, timer functions, watchdog timer, HALT and wake-up

functions, as well as low cost, enhance the versatility of this device to suit a wide range of application possibilities such as industrial control, consumer products, and particularly suitable for use in products such as infrared remote controllers and various subsystem controllers.

**Block Diagram**

**Pin Assignment**

**Pin Description**

Pin Name	I/O	Configuration Option	Description
PA0~PA7	I/O	—	Bidirectional 8-bit input/output port with pull-high resistors. Software instructions determine if the pin is an NMOS output or Schmitt Trigger input. Each pin can have a wake-up function if configured as an input pin.
PB0~PB4 PB5/OSC2 PB6/OSC1 PB7	I/O	OSC	Bidirectional 8-bit input/output port with pull-high resistors. Software instructions determine if the pin is an NMOS output or Schmitt Trigger input. Each pin can have a wake-up function if configured as an input pin. PB5 and PB6 are pin-shared with the external crystal pins named OSC2 and OSC1 respectively determined by a configuration option.
PC0/ $\overline{\text{RES}}$	I/O	$\overline{\text{RES}}$	Bidirectional 1-bit input/output port without a pull-high resistor. Software instructions determine if the pin is an NMOS output or Schmitt Trigger input. This pin has the capability of wake-up when it is configured as an input pin. PC0 is pin-shared with the external reset pin named RES determined by a configuration option.
REM	O	REMDRV	Carrier output dual function pin. REM is a CMOS carrier output pin with an initial low level after a reset. REM pin is a high sink current NMOS open drain carrier output pin which will be in a floating condition after a reset. The selection of REM or REMDRV is determined by a configuration option.
VDD	—	—	Positive power supply
VSS	—	—	Negative power supply, ground

**Absolute Maximum Ratings**

Supply Voltage .....	$V_{SS}-0.3V$ to $V_{SS}+4.0V$	Storage Temperature .....	$-50^{\circ}C$ to $125^{\circ}C$
Input Voltage .....	$V_{SS}-0.3V$ to $V_{DD}+0.3V$	Operating Temperature .....	$-20^{\circ}C$ to $70^{\circ}C$
$I_{OL}$ Total .....	150mA	$I_{OH}$ Total .....	$-100mA$
Total Power Dissipation .....	500mW		

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**
 $T_a=25^{\circ}C$ 

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		$V_{DD}$	Conditions				
$V_{DD}$	Operating Voltage	—	LVR disable	1.8	—	3.6	V
			LVR enable	2.0	—	3.6	V
$I_{DD}$	Operating Current	3V	No load, $f_{SYS}=4MHz$	—	0.5	1.2	mA
$I_{STB1}$	Standby Current	3V	No load, system HALT, WDT disable	—	0.1	1.0	$\mu A$
$I_{STB2}$	Standby Current	3V	No load, system HALT, WDT enable	—	—	5.0	$\mu A$
$V_{IL}$	Input Low Voltage for I/O Ports	—	—	0	—	$0.2V_{DD}$	V
$V_{IH}$	Input High Voltage for I/O Ports	—	—	$0.8V_{DD}$	—	$V_{DD}$	V
$V_{IL}$	Input Low Voltage for $\overline{RES}$ Ports	—	—	0	—	$0.4V_{DD}$	V
$V_{IH}$	Input High Voltage for $\overline{RES}$ Ports	—	—	$0.9V_{DD}$	—	$V_{DD}$	V
$I_{OH}$	REM Output Source Current	3V	$V_{OH}=0.9V_{DD}$	-5	-7	—	mA
$I_{OL1}$	PA, PB, REM Output Sink Current	3V	$V_{OL}=0.1V_{DD}$	6	12	—	mA
$I_{OL2}$	PC0 Sink Current	3V	$V_{OL}=0.1V_{DD}$	0.8	1.2	—	mA
$I_{OL3}$	REM Output Sink Current	3V	$V_{OL}=0.2V_{DD}$	300	350	—	mA
$R_{PH}$	Pull-high Resistance of Port A, Port B	3V	—	100	150	200	$k\Omega$
$V_{LVR}$	Low Voltage Reset Voltage	—	—	1.8	1.9	2.0	V
$V_{POR}$	$V_{DD}$ Start Voltage to ensure Power-on Reset	—	—	—	—	100	mV
$R_{POR}$	$V_{DD}$ Rise Rate to ensure Power-on Reset	—	—	0.035	—	—	V/ms

**A.C. Characteristics**

Ta=25°C

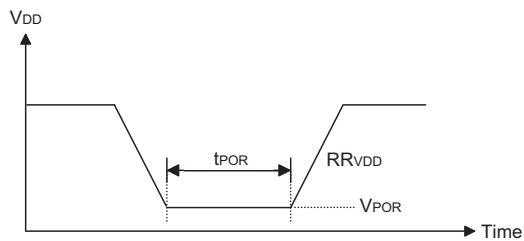
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
f <sub>SYS</sub>	System Clock	1.8V~3.6V	Ta= -20°C ~ 70°C	400	—	4000	kHz
f <sub>HIRC</sub>	System Clock (HIRC)	2.2V~3.6V	Ta= -20°C ~ 50°C	4013	4095	4179	kHz
t <sub>SST</sub>	System Start-up Timer Period	—	Power-up or wake-up from HALT	—	1024	—	t <sub>SYS</sub>
t <sub>WDTOSC</sub>	Watchdog Oscillator	3V	—	45	90	180	μs
t <sub>LVR</sub>	Low Voltage Width to Reset	—	—	0.25	1.00	2.00	ms
t <sub>POR</sub>	Power-on Reset Low Pulse Width	—	—	1	—	—	μs

 Note: t<sub>SYS</sub>=1/f<sub>SYS</sub>

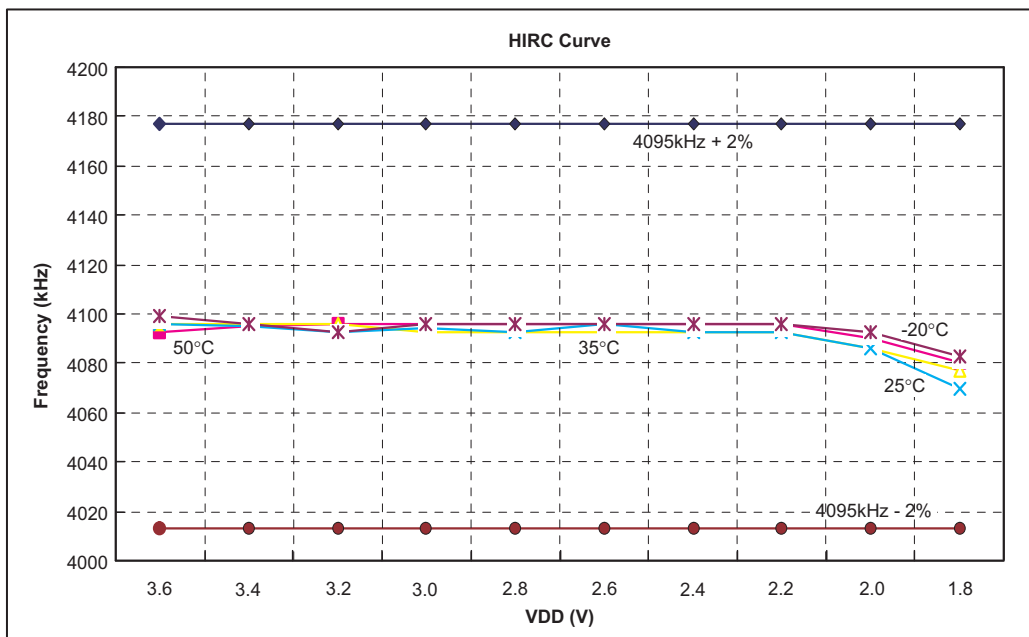
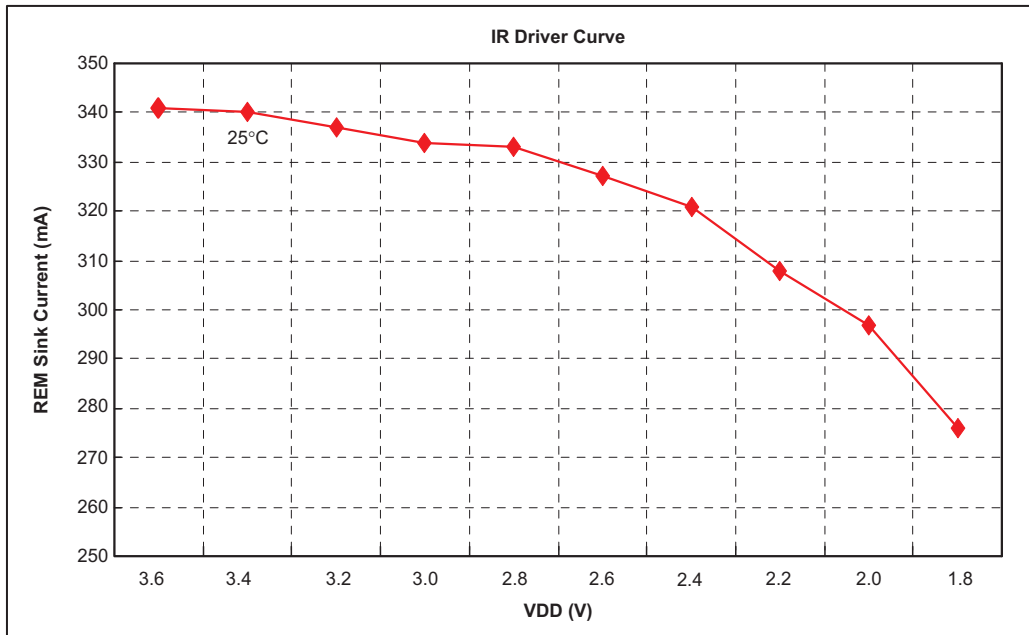
To maintain the accuracy of the internal HIRC oscillator frequency, a 0.1μF decoupling capacitor should be connected between VDD and VSS and located as close to the device as possible.

**Power-on Reset Characteristics**

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>POR</sub>	VDD Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
RR <sub>VDD</sub>	VDD raising rate to Ensure Power-on Reset	—	—	0.035	—	—	V/ms
t <sub>POR</sub>	Minimum Time for VDD Stays at V <sub>POR</sub> to Ensure Power-on Reset	—	—	1	—	—	ms



Characteristics Curves



## Functional Description

### Execution Flow

The main system clock is derived from either an external crystal oscillator which requires the connection of the external crystal or resonator or an internal RC oscillator which requires no external component for its operation. It 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 one instruction cycle while decoding and execution takes the next instruction cycle. However, the pipelining scheme causes each instruction to effectively execute within one cycle. If an instruction changes the program counter, two cycles are required to complete the instruction.

### Program Counter – PC

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

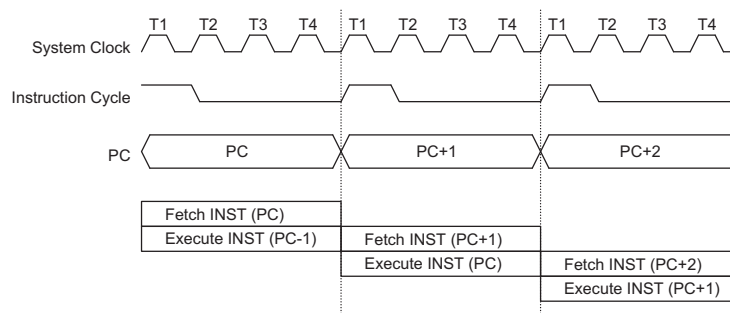
After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by one. The program counter then points to the memory word containing the next instruction code.

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

The conditional skip is activated by instruction. 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									
	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial reset	0	0	0	0	0	0	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 – ROM

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

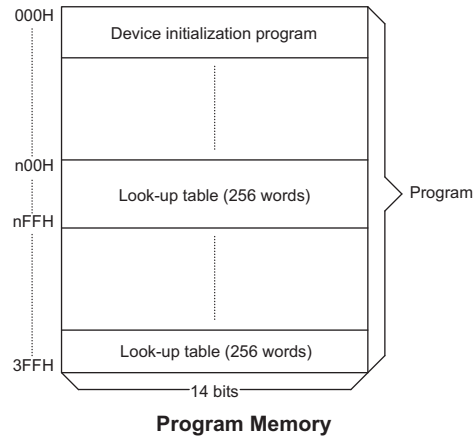
Certain locations in the program memory are reserved for special usage:

- Location 000H  
This area is reserved for the initialization program. After a device reset, the program always begins execution at location 000H.
- Table location  
Any location in the Program Memory space can be used as a look-up table. 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 register, 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 TBLH, the remaining 2 bits are read as "0". The Table Higher-order byte register (TBLH) is read only. The table pointer (TBLP) is a read/write register (07H), where P indicates the table location. Before accessing the table, the location must be placed in TBLP. The TBLH is read only and cannot be restored. All table related instructions need 2 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 used to save the contents of the program counter only. The stack is organized into one level 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 and is neither readable nor writeable. At a subroutine call the contents of the program counter are pushed onto the stack. At the end of a subroutine signaled by a return instruction, RET, 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 "CALL" is subsequently executed, stack overflow occurs and the first entry will be lost and only the most recent return address is stored.



### Data Memory – RAM

The data memory is divided into two functional groups: special function registers and general purpose data memory (32×8). Most are read/write, but some are read only.

The remaining space before the 20H is reserved for future expanded usage and reading these locations will return the result 00H. The general purpose data memory, addressed from 20H to 3FH, is used for data and control information under instruction command. All 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 instructions, respectively. They are also indirectly accessible through memory pointer register (MP;01H).

### Indirect Addressing Register

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

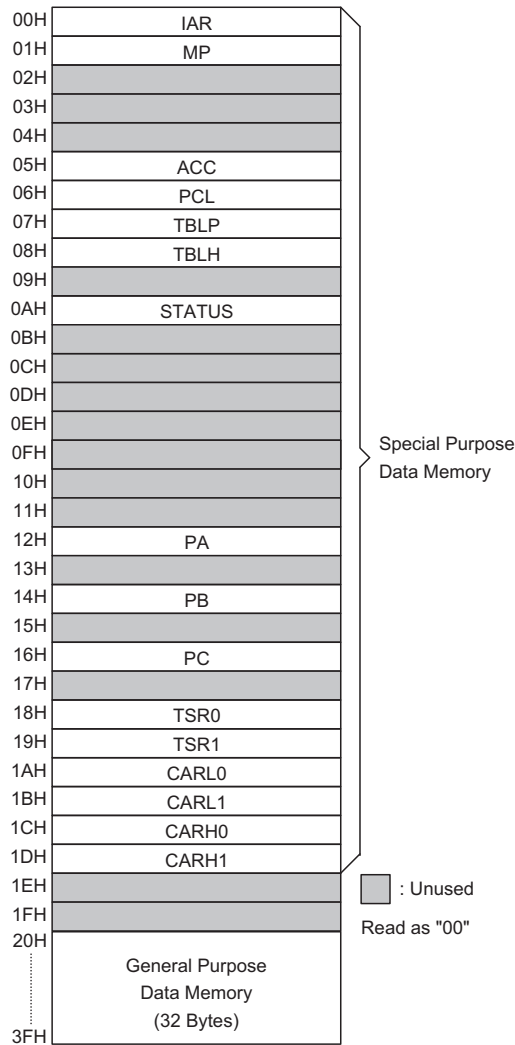
The memory pointer register MP (01H) is a 7-bit register. Bit 7 of MP is undefined and reading will return the result "1". Any writing operation to MP will only transfer the lower 7-bits of data to MP.

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

Table Location

Note: \*9~\*0: Table location bits  
P9~P8: Current program counter bits

@7~@0: Table pointer bits



**RAM Mapping**

**Accumulator**

The accumulator closely relates to ALU operations. It is also mapped to location 05H of the data memory and is capable of carrying out immediate data operations. Data movement between two data memory locations has to pass through the accumulator.

**Arithmetic and Logic Unit – ALU**

This circuit performs 8-bit arithmetic and logic operation. 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 contents of the status register.

**Status Register – STATUS**

This 8-bit status 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, the other status register bits can be altered by instructions like most other register. Any data written into the status register will not change the TO or PDF flags. In addition it should be noted that operations related to the status register may give different results from those intended. The TO and PDF flags can only be changed by the Watchdog Timer overflow, device power-up, clearing the Watchdog Timer and executing the HALT instruction.

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

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 when either a 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**

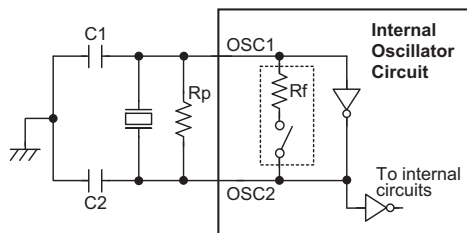


In addition, on executing a subroutine call, the status register will not be automatically pushed onto the stack. If the contents of the status are important and if the subroutine can corrupt the status register, precautions must be taken to save it properly.

### Oscillator Configuration

In this device there are two methods of generating the system clock, one external crystal oscillator and one internal RC oscillator.

- **External Crystal/Resonator Oscillator – HXT**  
The External Crystal/Ceramic System Oscillator is one of the system oscillator choices, which is selected via a configuration option. For the crystal oscillator configuration, the connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation. Two external capacitors may be required to be connected as shown. However, the feedback resistor named Rf shown in the following diagram for the crystal oscillator to oscillate properly can be selected as either an internally or externally connected type via a configuration option. When the external connection type of the feedback resistor is selected, the recommended value of the external connected feedback resistor ranges from 300kΩ to 500kΩ. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be connected as shown for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.



- Note:
1. Rp is normally not required. C1 and C2 are required.
  2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

### Crystal/Resonator Oscillator – HXT

Crystal Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
4MHz	8pF	10pF
Note: C1 and C2 values are for guidance only.		

### Crystal Recommended Capacitor Values

- **Internal RC Oscillator – HIRC**  
The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 4095kHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. As a result, at a power supply ranging from 2.2V to 3.6V and in a temperature range from -20°C to 50°C degrees, the fixed oscillation frequency of 4095kHz will have a tolerance within 2%. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins PB5 and PB6 are free for use as normal I/O pins.

**Watchdog Timer – WDT**

The WDT clock source is implemented by the instruction clock which is the system clock divided by 4 or the internal RC oscillator with the frequency of 12kHz. The clock source is processed by a frequency divider and a prescaler to provide various time out periods.

$$\text{WDT time out period} = \frac{\text{Clock Source}}{2^n}$$

Where n= 8~11 selected by a configuration option.

The WDT timer is designed to prevent a software malfunction or sequence jumping to an unknown location with unpredictable results. The Watchdog Timer can be disabled by configuration option. If the Watchdog Timer is disabled, all the executions related to the WDT result in no operation and the WDT will lose its protection purpose. In this situation the logic can only be restarted by external logic.

A WDT overflow under normal operation will initialise a "chip reset" and set the status bit "TO". To clear the contents of the WDT prescaler, two methods are adopted, software instructions or a HALT instruction. There are two types of software instructions. One type is the single instruction "CLR WDT", the other type comprises two instructions, "CLR WDT1" and "CLR WDT2". Of these two types of instructions, only one can be active depending on the configuration option – "CLR WDT times selection option". If the "CLR WDT" is selected (i.e.. CLR WDT times equal one), any execution of the CLR WDT instruction will clear the WDT. In case "CLR WDT1" and "CLR WDT2" are chosen (i.e.. CLR WDT times equal two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip due to a time-out.

**Power Down Operation – HALT**

The Power-down mode is initialised by the HALT instruction and results in the following:

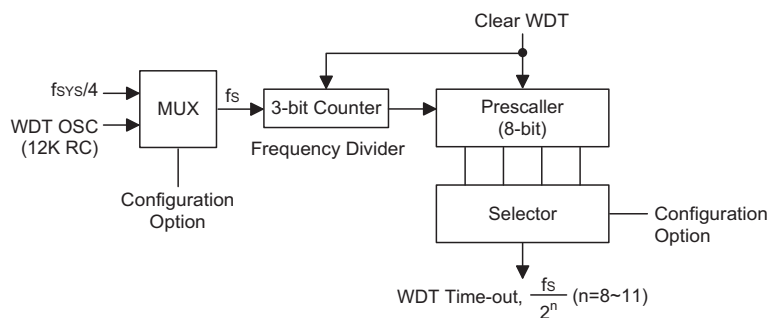
- The system oscillator turns off and the WDT stops.
- The contents of the on-chip Data Memory and registers remain unchanged.
- WDT prescaler is cleared.
- All I/O ports maintain their original status.
- The PDF flag is set and the TO flag is cleared.

The system can quit the HALT mode by means of an external falling edge signal on port B. By examining the TO and PDF flags, the reason for chip reset can be determined. The PDF flag is cleared when the system powers up or when a CLR WDT instruction is executed and is set when the HALT instruction is executed. The TO flag is set if the WDT time-out occurs during normal operation.

The port B wake-up can be considered as a continuation of normal execution. Each bit in port B can be independently selected to wake up the device by the code option. Awakening from an I/O port stimulus, the program will resume execution of the next instruction.

Once a wake-up event(s) occurs, it takes 1024 t<sub>SYS</sub> (system clock periods) to resume normal operation. In other words, a dummy cycle period will be inserted after the wake-up.

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



**Watchdog Timer**

**Reset**

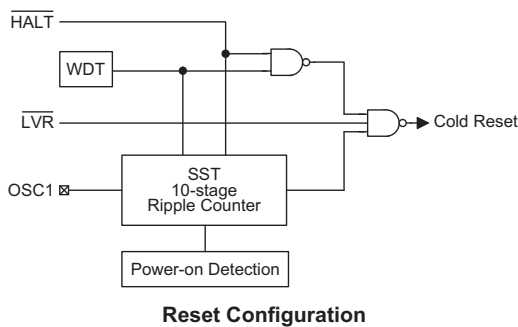
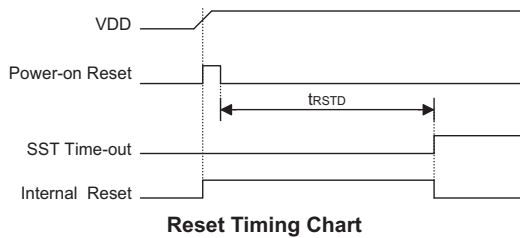
There are several ways in which a reset can occur:

- Power On reset
- $\overline{\text{RES}}$  pin reset
- Low Voltage reset
- WDT time-out reset during normal operation

Some registers remain unchanged during 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	Power-on reset
u	u	$\overline{\text{RES}}$ or LVR reset during normal operation
1	u	WDT time-out reset during normal operation
1	1	WDT time-out reset during power-down mode

Note: "u" means unchanged.



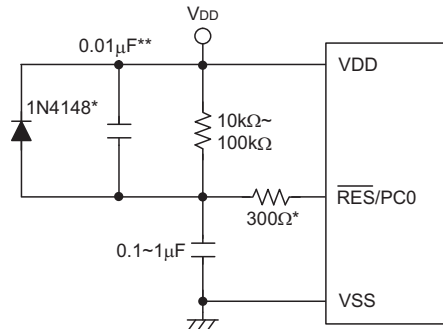
To guarantee that the system oscillator has started and stabilized, the SST (System Start-up Timer) provides an extra-delay of 1024 system clock pulses when the system powers up or when the system awakes from a HALT state.

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

The functional unit chip reset status is shown below.

Program Counter	000H
WDT Prescaler	Clear
Input/Output ports	Input mode
Stack Pointer	Points to the top of the stack
Carrier output	Low level state or floating state*

\*\*\* Determined by configuration option



Note: \*\*\* It is recommended that this component is added for added ESD protection

\*\*\*\* It is recommended that this component is added in environments where power line noise is significant.

**External  $\overline{\text{RES}}$  Circuit**

The chip reset status of the registers is summarised in the following table:

Register	Reset (Power On)	$\overline{\text{RES}}$ or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)*
Program Counter	000H	000H	000H	000H
MP	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--uu uuuu	--1u uuuu	--11 uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	---- --1	---- --1	---- --1	---- --u
TSR0	0000 0000	0000 0000	0000 0000	uuuu uuuu
TSR1	1000 0000	1000 0000	1000 0000	uuuu uuuu
CARL0	0000 0000	0000 0000	0000 0000	uuuu uuuu
CARL1	0000 0000	0000 0000	0000 0000	uuuu uuuu
CARH0	0000 0000	0000 0000	0000 0000	uuuu uuuu
CARH1	0000 0010	0000 0010	0000 0010	uuuu uuuu

Note: "u" means unchanged  
"x" means unknown  
"--" stands for unimplemented

### Input/Output Ports

There are up to 17 bidirectional input/output lines in the device, labeled PA, PB and PC which are mapped to [12H], [14H] and [16H] of the Data Memory, respectively. Each line of PA and PB can be selected as NMOS output or Schmitt trigger input with pull-high resistor by a software instruction. PC0 can be used as an input line with Schmitt trigger but without pull-high resistor or as the external  $\overline{\text{RES}}$  pin determined by the configuration option.

When the I/O ports are used for input operation, these ports are non-latched, that is, the inputs should be ready at the T2 rising edge of the instruction "MOV A, [m]" (m=12H, 14H or 16H). For I/O ports output operations, all data is latched and remains unchanged until the output latch is rewritten.

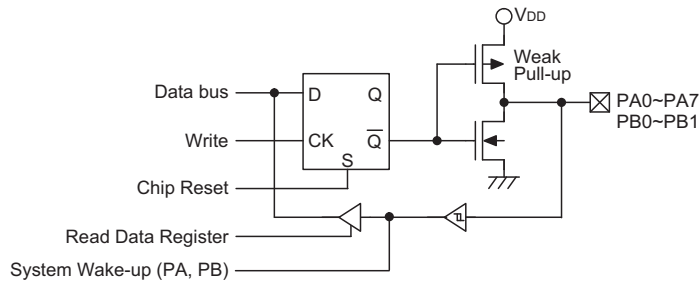
When the I/O Ports are used for input operations, it should be noted that before reading data from the pads,

a "1" should be written to the related bits to disable the NMOS device. That is, the instruction "SET [m].i" (i=0~7 for PA and PB, i=0 for PC) is executed first to disable related NMOS device, and then "MOV A, [m]" to get stable data.

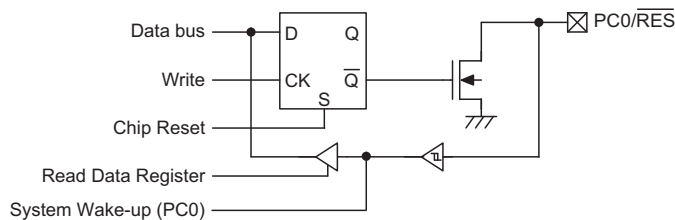
After chip reset, the I/O Ports remain at a high level input line. Each bit of the I/O ports output latches can be set or cleared by the "SET [m].i" and "CLR [m].i" (m=12H, 14H or 16H) instructions respectively.

Some instructions first input data and then follow the output operations. For example, "SET [m].i", "CLR [m]", "CPL [m]" and "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 to the accumulator.

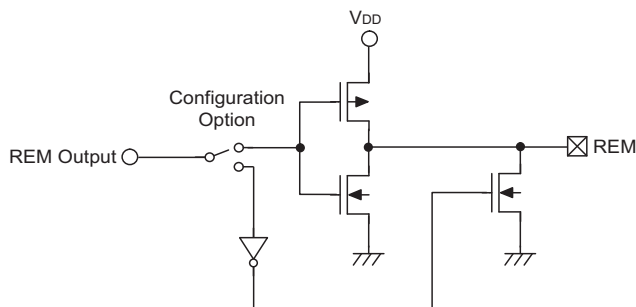
Each line of the I/O ports has a wake-up capability when the relevant pin is configured as an input line.



PA and PB Input/Output Ports



PC0 Input/Output Port



REM output pin structure

**Timer**

The timer is an internal unit for creating a remote control transmission pattern. As shown, it consists of a 9-bit down counter (t8 to t0), a flag (t9) permitting the 1-bit timer output, and a zero detector.

No.	Label	Function
0~7	t0~t7	Down counter

TSR0 (18H) Register

No.	Label	Function
0	t8	Down counter
1	t9	Timer enable, initial value is "0".
2~6	—	Unused bit, read as "0".
7	TOEF	Timer operation end flag, initial value is "1".

TSR1 (19H) Register

**Timer Operation**

The timer starts counting down when a value other than "0" is set for the down counter with a timer manipulation instruction. The timer manipulation instructions for making the timer start operation are shown below:

```

MOV A,XXH      ; XX = 00H ~ FFH
MOV TSR0,A
MOV A,XXH      ; XX ≤ 01H, t8
MOV TSR1,A
SET TSR1.1     ; The timer is started by set t9=1
    
```

Addition notes for the 9-bit timer:

- Writing to TSR0 will only put the written data to the TSR0 register (t7~t0) and writing to TSR1 (t8) will transfer the specified data and contents of TSR0 to the Down Counter. TOEF will be cleared after the data transferred from TSR1 and TSR0 to the Down Counter is completed and then wait until TSR1.1 is set by user.
- Setting TSR1.1=1, the timer will start counting. The timer will stop when its count is equal to "0" and then TOEF is set equal to "1".

- If the TSR1.1 is cleared during the timer counting, the timer will be stopped. Once the TSR1.1 is set (1→0→1), the down counter will reload data from t8~t0, and then the down counter begins counting down with the new load data.
- If TSR1.1 and TOEF are equal to 1 both, the timer can re-start, after new data is written to TSR0, TSR1 (t0~t8) in sequence.

Note: If the contents of the Down counter is 000H, set the t9 to start the timer counting, the timer will only count 1 step. The timer output time =  $64/f_{SYS}$ . →  $[(0+1) \times 64/f_{SYS} = 64/f_{SYS}]$

The down counter is decremented (-1) in the cycle of  $64/f_{SYS}$ . If the value of the down counter becomes "0", the zero detector generates the timer operation end signal to stop the timer operation. At this time, TOEF will be set to "1". The output of the timer operation end signal is continued while the down counter is "0" and the timer is stopped. The following relational expression applies between the timer's output time and the down counter's set value.

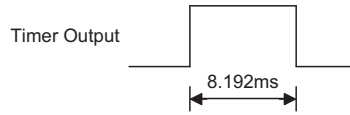
$$\text{Timer output time} = (\text{Set value} + 1) \times 64/f_{SYS}$$

An example is shown below.

```
MOV A,0FFH
MOV TSR0,A
MOV A,01H
MOV TSR1,A
SET TSR1.1
```

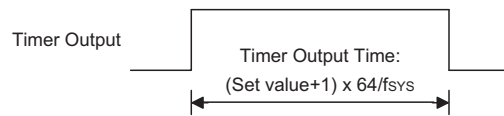
In the case above, the timer output time is as follows.

$$\begin{aligned} & (\text{Set value} + 1) \times 64/f_{SYS} \\ & = (511 + 1) \times 16\mu\text{s} \\ & = 8.192\text{ms} \end{aligned}$$

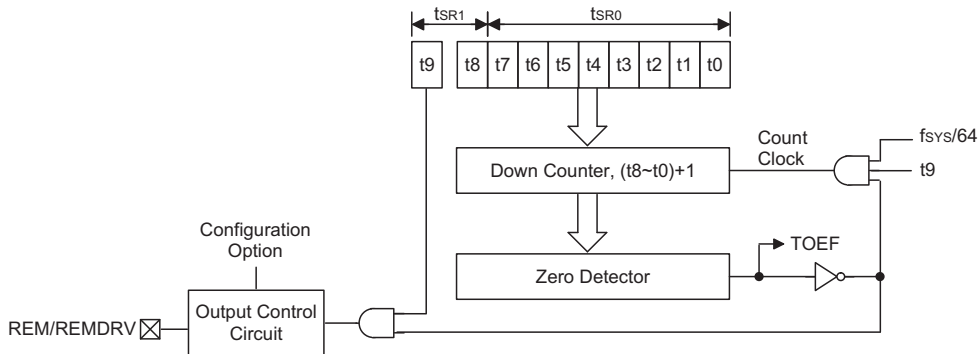


Setting the t9 bit channels the timer to the REM pin. The REM pin will be a combination of the timer and carrier signals.

Note: The carrier output results if bit 9 of the high-level period setting modulo register (CARH) is cleared ("0").



Timer Output when Carrier is not Output



Timer Configuration

**Carrier Output**

- Carrier output generator

The carrier generator consists of a 9-bit counter and two modulo registers for setting the high-level and low-level periods - CARH and CARL respectively.

Register	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CARL0	CL.7	CL.	CL.	CL.	CL.3	CL.2	CL.1	CL.0
CARL1	—	—	—	—	—	—	Fix "0"	CL.8
CARH0	CH.7	CH.6	CH.5	CH.4	CH.3	CH.2	CH.1	CH.0
CARH1	—	—	—	—	—	—	CH.9 (CARY)	CH.8

**CARL0 (1AH) Register, CARL1 (1BH), CARH0 (1CH) Register, CARH1 (1DH), Register**

Note: 1. CARH1.1 (CARY) initial value is "1".

2. CARL1.2 (CARH1.2)~CARL1.7 (CARH1.7) are unused bits, read as "0".

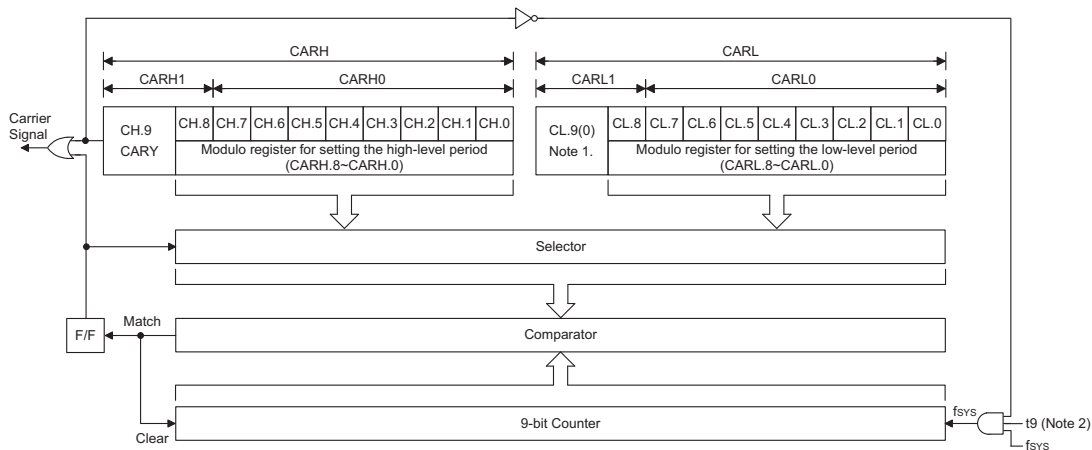
The carrier duty ratio and carrier frequency can be determined by setting the high-level and low-level widths using the respective modulo registers. Each of these widths can be set in a range of 500ns to 64µs at  $f_{SYS} = 4MHz$ .

CARH (CARH1.0, CARH0.7~CARH0.0) and CARL (CARL1.0, CARL0.7~CARL0.0) are read and written using instructions.

Example:

```

MOV A,XXH      ; XXH = 00H~FFH
MOV CARL0,A
MOV A,XXH      ; XXH ≤ 01H, CL.8 (CARL1.0)
MOV CARL1,A
MOV A,XXH      ; XXH = 00H~FFH
MOV CARH0,A
MOV A,XXH      ; XXH ≥ 02H, CH.8 (CARH1.0)
MOV CARH1,A
CLR CARH1.1    ; The carrier is started by clearing CARY(CARH1.1)="0"
    
```



**Configuration of Remote Controller Carrier Generator**

Note: 1. Bit 9 of the modulo register for setting the low-level period (CARL) is fixed to "0".

2. t9: Flag that enables timer output (timer block, see Timer Configuration)

The values of CARH and CARL can be calculated from the following expressions.

$$\text{CARL (CARL1.0, CARL0.7~CARL0.0)} = (f_{\text{SYS}} \times (1-D) \times T) - 1$$

$$\text{CARH (CARH1.0, CARH0.7~CARH0.0)} = (f_{\text{SYS}} \times D \times T) - 1$$

D: Carrier duty ratio ( $0 < D < 1$ )

$f_{\text{SYS}}$ : Input clock (MHz)

T: Carrier cycle ( $\mu\text{s}$ )

Ensure to input values in the range of 001H to 1FFH to CARL and CARH.

Example:

$$f_{\text{SYS}} = 4\text{MHz}, f_c = 38.1\text{kHz}, T = 1/f_c = 26.25\mu\text{s}, \text{duty} = 1/3$$

$$\text{CARL} = (4\text{M} \times (1-1/3) \times 26.25\mu\text{s}) - 1 = 69 = 45\text{H}$$

$$\text{CARH} = (4\text{M} \times 1/3 \times 26.25\mu\text{s}) - 1 = 34 = 22\text{H}$$

```
MOV A,045H
MOV CARL0,A
MOV A,022H
MOV CARH0,A
CLR CARH1.1 ; The carrier is started by clearing CARY(CARH1.1) = "0"
```

• Carrier output control

The remote controller carrier can be output from the REM pin by clearing to zero bit 9 (CARY) of the modulo register for setting the high-level period (CARH).

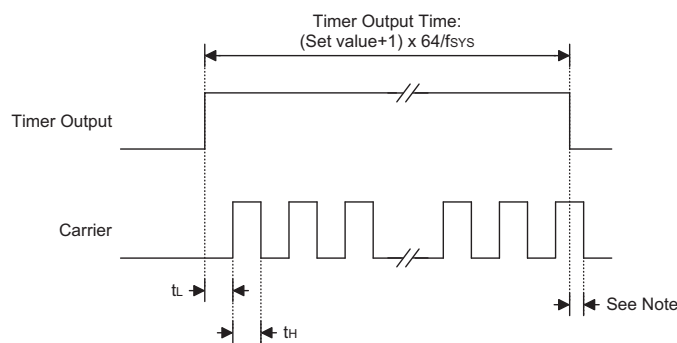
When performing a carrier output, be sure to set the timer operation after setting the CARH (CARH1.0, CARH0.7~CARH0.0) and CARL (CARL1.0, CARL0.7~CARL0.0) values.

Note that a malfunction may occur if the values of CARH (CARH1.0, CARH0.7~CARH0.0) and CARL (CARL1.0, CARL0.7~CARL0.0) are changed while the carrier is being output on the REM pin.

Executing the timer manipulation instruction starts the carrier output from the low level.

There is a dual function remote controller carrier output pin named REM. The selection of REM or REMDRV is determined by a configuration option. After a reset, the REM carrier output pin will have a low level while the REMDRV carrier output pin will be in a floating condition. The generic structures of the REM or REMDRV function are illustrated in the accompanying diagram. As the exact construction of the carrier output pin will differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the remote carrier output pins.

The output from the REM pin is in accordance with the value of bit 9 (CARY) of CARH and the timer output enable flag (t9), and the value of the timer 9-bit down counter (t0 to t8).



Timer Output when Carrier is Output

Note: When the carrier signal is active and during the time when the signal is high, if the timer output should go low, the carrier signal will first complete its high level period before going low.



CARH1.1	Timer Output Enable Flag (t9: TSR1.1)	9-bit Down Counter	REM Function (CMOS Output)	REMDRV Function (NMOS Output)
0	0	0	Low-level output	Floating output
0	0	Other than 0		
0	1	0	64/f <sub>SYS</sub> (with carrier output)	64/f <sub>SYS</sub> (with carrier output)
0	1	Other than 0	Carrier output (Note)	Carrier output
1	0	—	Low-level output	Floating output
1	1	—	High-level output	Low-level output

**REM Pin Output Control**

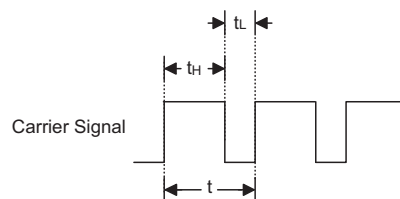
Note: Input values in the range of 001H to 1FFH to CARH (CARH1.0, CARH0.7~CARH0.0) and CARL (CARL1.0, CARL0.7~CARL0.0).

Caution: CARH (CARH1.0, CARH0.7~CARH0.0) and CARL (CARL1.0, CARL0.7~CARL0.0) must be set while the REM pin is at a low level (t9 = 0 or t0 to t8 = 0).

CARH (CARH1.0, CARH0.7~CARH0.0)	CARL (CARL1.0, CARL0.7~CARL0.0)	t <sub>H</sub> (μs)	t <sub>L</sub> (μs)	t (μs)	f <sub>c</sub> (kHz)	Duty
20H	20H	8.25	8.25	16.50	60.60	1/2
22H	45H	8.75	17.50	26.25	38.10	1/3
22H	46H	8.80	17.60	26.40	37.90	1/3
24H	49H	9.26	18.52	27.78	36.00	1/3

**Carrier Frequency Setting (f<sub>SYS</sub>=4MHz)**

CARH (CARH1.0, CARH0.7~CARH0.0)	CARL (CARL1.0, CARL0.7~CARL0.0)	t <sub>H</sub> (μs)	t <sub>L</sub> (μs)	t (μs)	f <sub>c</sub> (kHz)	Duty
25H	4BH	9.256	18.510	27.770	36	1/3
23H	47H	8.770	17.540	26.310	38	1/3
18H	30H	5.950	11.900	17.850	56	1/3

**Carrier Frequency Setting (f<sub>SYS</sub>=4095kHz)**


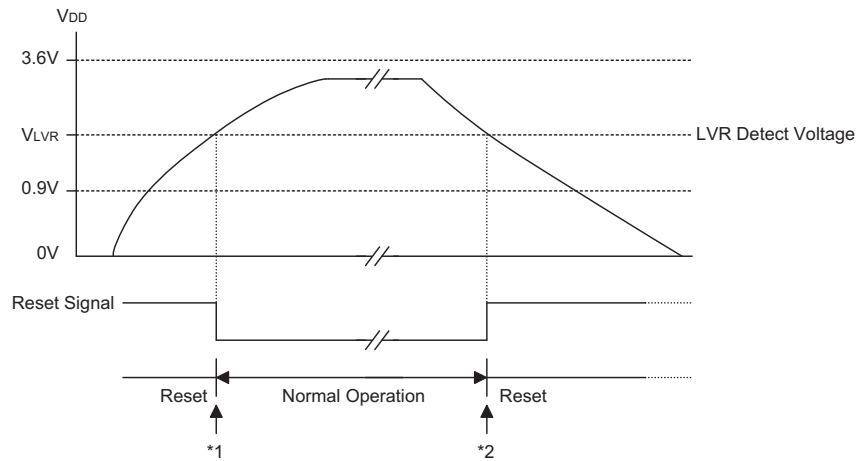
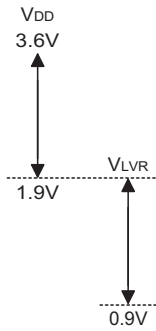
**Low Voltage Reset – LVR**

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  $0.9V \sim V_{LVR}$ , such as when changing a battery, the LVR will automatically reset the device internally.

The LVR includes the following specifications:

- The low voltage ( $0.9V \sim V_{LVR}$ ) has to remain in this state for a time in excess of 1ms. If the low voltage state does not exceed 1ms, the LVR will ignore it and will not perform a reset function.

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



**Low Voltage Reset**

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

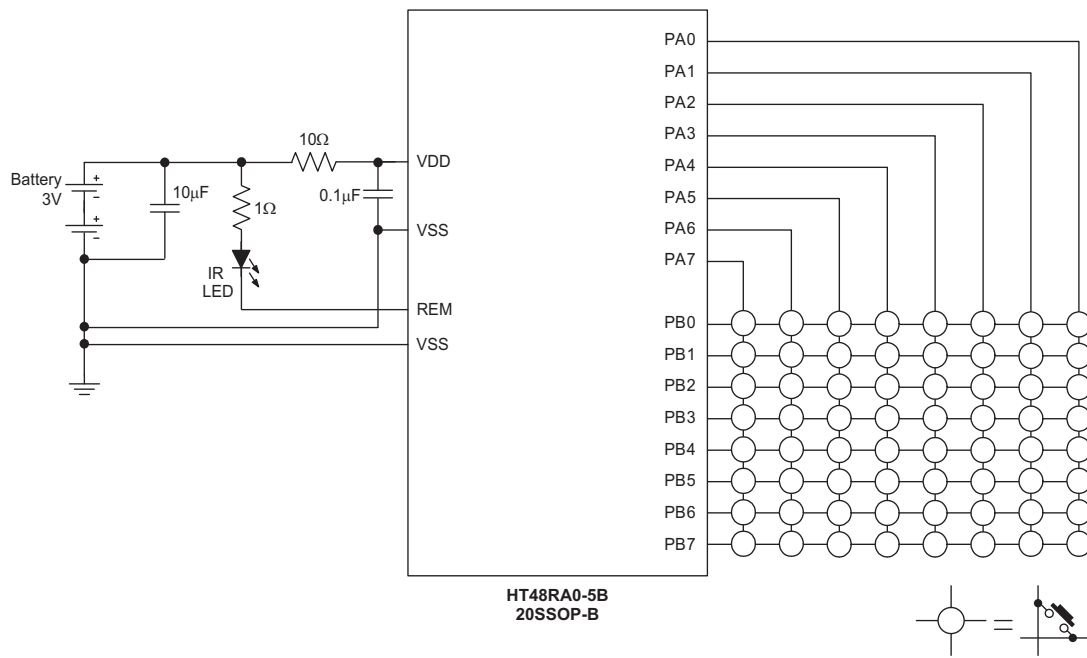
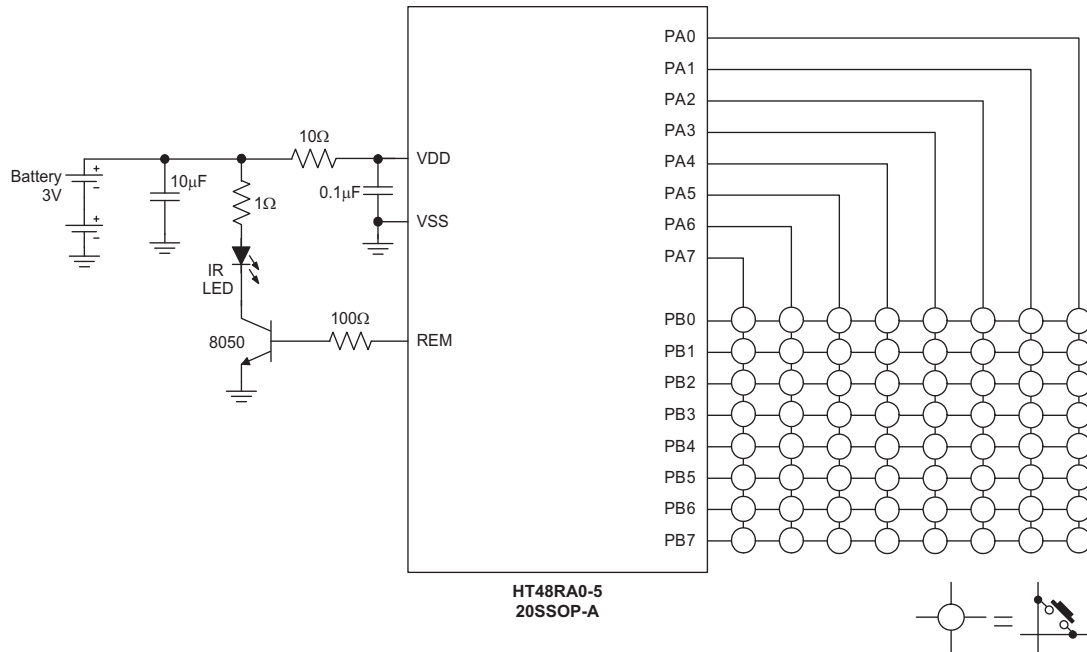
**\*\*2** Since low voltage has to be maintained in its original state and exceed 1ms, a 1ms delay enters the reset mode.

**Configuration Options**

The following table shows the range of configuration options for the device. All the configuration options must be defined to ensure proper system functioning.

No.	Code Option
<b>Oscillator Options</b>	
1	System oscillator selection - $f_{SYS}$ : XTAL oscillator without internal feedback resistor XTAL oscillator with internal feedback resistor Internal 4095kHz RC oscillator
<b>REM Pin Options</b>	
2	REM or REMDRV output function selection
<b>Watchdog Options</b>	
3	WDT clock selection - $f_S$ : Internal RC oscillator or $f_{SYS}/4$
4	WDT function: enable or disable
5	CLRWDT instruction selections: 1 or 2 instructions
6	WDT time-out period selections: $2^8/f_S$ , $2^9/f_S$ , $2^{10}/f_S$ , $2^{11}/f_S$
<b>LVR Options</b>	
7	LVR function: enable or disable
<b>Reset Pin Options</b>	
8	I/O or $\overline{RES}$ pin selection

Application Circuits



Note: The 0.1µF capacitor is required to ensure that the system clock frequency meets with the specified tolerance in the A.C. Characteristics.

## 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 0.5 $\mu$ s and branch or call instructions would be implemented within 1 $\mu$ s. 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
<b>Arithmetic</b>			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1 <sup>Note</sup>	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 <sup>Note</sup>	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	1 <sup>Note</sup>	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	1 <sup>Note</sup>	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1 <sup>Note</sup>	C
<b>Logic Operation</b>			
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	1 <sup>Note</sup>	Z
ORM A,[m]	Logical OR ACC to Data Memory	1 <sup>Note</sup>	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1 <sup>Note</sup>	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	1 <sup>Note</sup>	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
<b>Increment &amp; Decrement</b>			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1 <sup>Note</sup>	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1 <sup>Note</sup>	Z

Mnemonic	Description	Cycles	Flag Affected
<b>Rotate</b>			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	↑ <sup>Note</sup>	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	↑ <sup>Note</sup>	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	↑ <sup>Note</sup>	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	↑ <sup>Note</sup>	C
<b>Data Move</b>			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	↑ <sup>Note</sup>	None
MOV A,x	Move immediate data to ACC	1	None
<b>Bit Operation</b>			
CLR [m].i	Clear bit of Data Memory	↑ <sup>Note</sup>	None
SET [m].i	Set bit of Data Memory	↑ <sup>Note</sup>	None
<b>Branch</b>			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	↑ <sup>Note</sup>	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	↑ <sup>note</sup>	None
SZ [m].i	Skip if bit i of Data Memory is zero	↑ <sup>Note</sup>	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	↑ <sup>Note</sup>	None
SIZ [m]	Skip if increment Data Memory is zero	↑ <sup>Note</sup>	None
SDZ [m]	Skip if decrement Data Memory is zero	↑ <sup>Note</sup>	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	↑ <sup>Note</sup>	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	↑ <sup>Note</sup>	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
<b>Table Read</b>			
TABRDC [m]	Read table (current page) to TBLH and Data Memory	2 <sup>Note</sup>	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2 <sup>Note</sup>	None
<b>Miscellaneous</b>			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	↑ <sup>Note</sup>	None
SET [m]	Set Data Memory	↑ <sup>Note</sup>	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	↑ <sup>Note</sup>	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**

<b>ADC A,[m]</b>	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	$ACC \leftarrow ACC + [m] + C$
Affected flag(s)	OV, Z, AC, C
<b>ADCM A,[m]</b>	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	$[m] \leftarrow ACC + [m] + C$
Affected flag(s)	OV, Z, AC, C
<b>ADD A,[m]</b>	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	$ACC \leftarrow ACC + [m]$
Affected flag(s)	OV, Z, AC, C
<b>ADD A,x</b>	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	$ACC \leftarrow ACC + x$
Affected flag(s)	OV, Z, AC, C
<b>ADDM A,[m]</b>	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	$[m] \leftarrow ACC + [m]$
Affected flag(s)	OV, Z, AC, C
<b>AND A,[m]</b>	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	$ACC \leftarrow ACC \text{ "AND" } [m]$
Affected flag(s)	Z
<b>AND A,x</b>	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	$ACC \leftarrow ACC \text{ "AND" } x$
Affected flag(s)	Z
<b>ANDM A,[m]</b>	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	$[m] \leftarrow ACC \text{ "AND" } [m]$
Affected flag(s)	Z



<b>CALL addr</b>	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
<b>CLR [m]</b>	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None
<b>CLR [m].i</b>	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
<b>CLR WDT</b>	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
<b>CLR WDT1</b>	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. Repeatedly executing this instruction without alternately executing CLR WDT2 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
<b>CLR WDT2</b>	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. Repeatedly executing this instruction without alternately executing CLR WDT1 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

<b>CPL [m]</b>	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
<b>CPLA [m]</b>	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
<b>DAA [m]</b>	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
<b>DEC [m]</b>	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	$[m] \leftarrow [m] - 1$
Affected flag(s)	Z
<b>DECA [m]</b>	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
<b>HALT</b>	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

<b>INC [m]</b>	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	$[m] \leftarrow [m] + 1$
Affected flag(s)	Z
<b>INCA [m]</b>	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	$ACC \leftarrow [m] + 1$
Affected flag(s)	Z
<b>JMP addr</b>	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	$Program\ Counter \leftarrow addr$
Affected flag(s)	None
<b>MOV A,[m]</b>	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	$ACC \leftarrow [m]$
Affected flag(s)	None
<b>MOV A,x</b>	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	$ACC \leftarrow x$
Affected flag(s)	None
<b>MOV [m],A</b>	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	$[m] \leftarrow ACC$
Affected flag(s)	None
<b>NOP</b>	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None
<b>OR A,[m]</b>	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	$ACC \leftarrow ACC \text{ "OR" } [m]$
Affected flag(s)	Z

<b>OR A,x</b>	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
<b>ORM A,[m]</b>	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
<b>RET</b>	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
<b>RET A,x</b>	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
<b>RETI</b>	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
<b>RL [m]</b>	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
<b>RLA [m]</b>	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

<b>RLC [m]</b>	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	$[m].(i+1) \leftarrow [m].i; (i = 0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
Affected flag(s)	C
<b>RLCA [m]</b>	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	$ACC.(i+1) \leftarrow [m].i; (i = 0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
Affected flag(s)	C
<b>RR [m]</b>	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	$[m].i \leftarrow [m].(i+1); (i = 0\sim6)$ $[m].7 \leftarrow [m].0$
Affected flag(s)	None
<b>RRA [m]</b>	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	$ACC.i \leftarrow [m].(i+1); (i = 0\sim6)$ $ACC.7 \leftarrow [m].0$
Affected flag(s)	None
<b>RRC [m]</b>	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	$[m].i \leftarrow [m].(i+1); (i = 0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C
<b>RRCA [m]</b>	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	$ACC.i \leftarrow [m].(i+1); (i = 0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C

<b>SBC A,[m]</b>	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	$ACC \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C
<b>SBCM A,[m]</b>	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	$[m] \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C
<b>SDZ [m]</b>	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
<b>SDZA [m]</b>	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
<b>SET [m]</b>	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	$[m] \leftarrow FFH$
Affected flag(s)	None
<b>SET [m].i</b>	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

<b>SIZ [m]</b>	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	$[m] \leftarrow [m] + 1$ Skip if $[m] = 0$
Affected flag(s)	None
<b>SIZA [m]</b>	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	$ACC \leftarrow [m] + 1$ Skip if $ACC = 0$
Affected flag(s)	None
<b>SNZ [m].i</b>	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 $[m].i \neq 0$
Affected flag(s)	None
<b>SUB A,[m]</b>	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	$ACC \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C
<b>SUBM A,[m]</b>	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	$[m] \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C
<b>SUB A,x</b>	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	$ACC \leftarrow ACC - x$
Affected flag(s)	OV, Z, AC, C

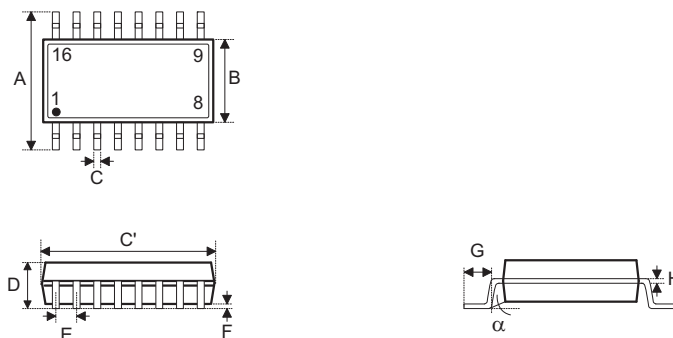
<b>SWAP [m]</b>	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
Affected flag(s)	None
<b>SWAPA [m]</b>	Swap nibbles of Data Memory with result in ACC
Description	The 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	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
Affected flag(s)	None
<b>SZ [m]</b>	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
<b>SZA [m]</b>	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	$ACC \leftarrow [m]$ Skip if $[m] = 0$
Affected flag(s)	None
<b>SZ [m].i</b>	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
<b>TABRDC [m]</b>	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	$[m] \leftarrow$ program code (low byte) $TBLH \leftarrow$ program code (high byte)
Affected flag(s)	None
<b>TABRDL [m]</b>	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	$[m] \leftarrow$ program code (low byte) $TBLH \leftarrow$ program code (high byte)
Affected flag(s)	None



<b>XOR A,[m]</b>	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
<b>XORM A,[m]</b>	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
<b>XOR A,x</b>	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**

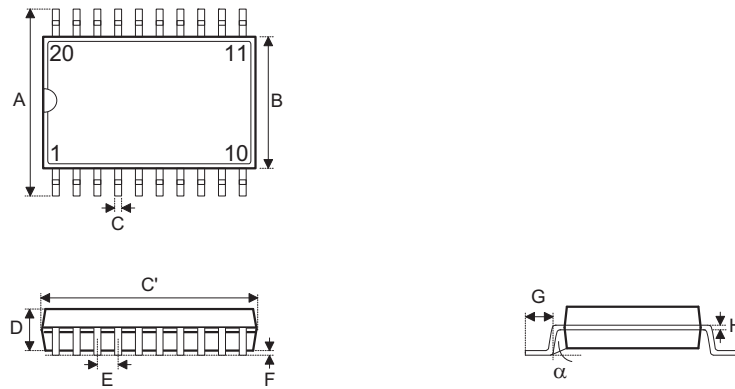
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website (<http://www.holtek.com.tw/english/literature/package.pdf>) for the latest version of the package information.

**16-pin NSOP (150mil) Outline Dimensions**


• MS-012

Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.157
C	0.012	—	0.020
C'	0.386	—	0.394
D	—	—	0.069
E	—	0.050	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.007	—	0.010
$\alpha$	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.79	—	6.20
B	3.81	—	3.99
C	0.30	—	0.51
C'	9.80	—	10.01
D	—	—	1.75
E	—	1.27	—
F	0.10	—	0.25
G	0.41	—	1.27
H	0.18	—	0.25
$\alpha$	0°	—	8°

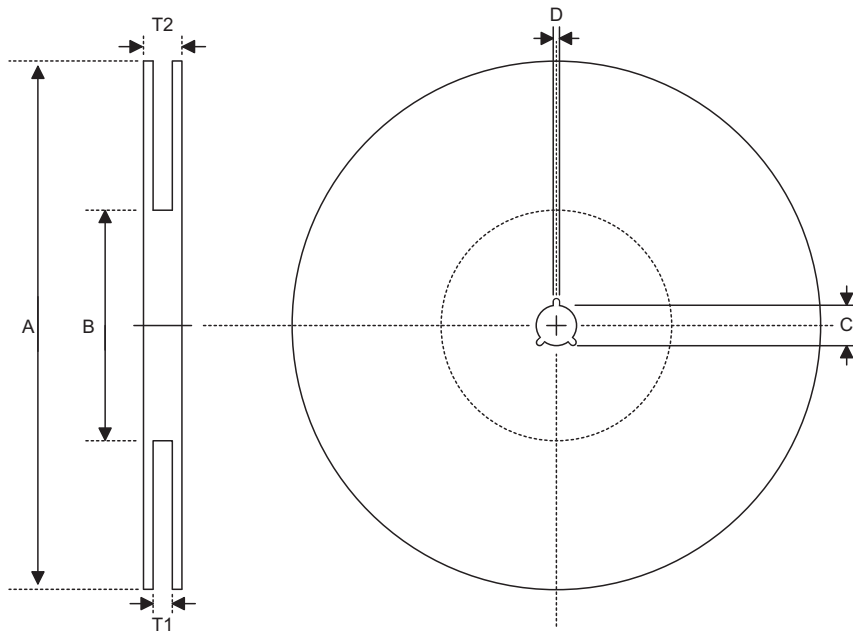
**20-pin SSOP (150mil) Outline Dimensions**


Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.158
C	0.008	—	0.012
C'	0.335	—	0.347
D	0.049	—	0.065
E	—	0.025	—
F	0.004	—	0.010
G	0.015	—	0.050
H	0.007	—	0.010
$\alpha$	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.79	—	6.20
B	3.81	—	4.01
C	0.20	—	0.30
C'	8.51	—	8.81
D	1.24	—	1.65
E	—	0.64	—
F	0.10	—	0.25
G	0.38	—	1.27
H	0.18	—	0.25
$\alpha$	0°	—	8°

**Product Tape and Reel Specifications**

**Reel Dimensions**

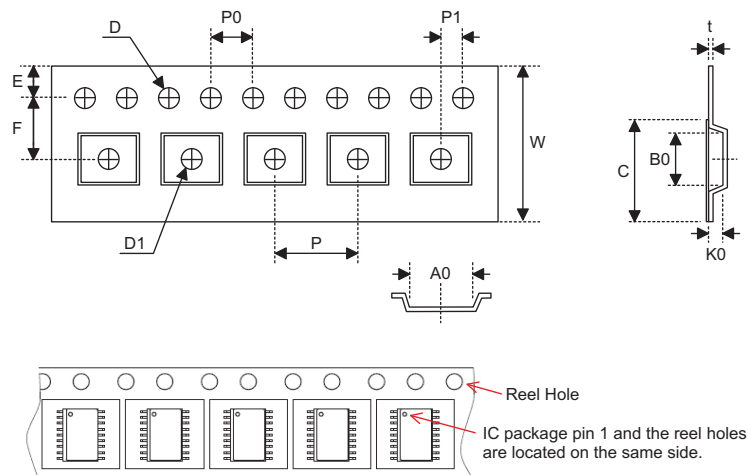


SOP 16N (150mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	13.0 <sup>+0.5/-0.2</sup>
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	16.8 <sup>+0.3/-0.2</sup>
T2	Reel Thickness	22.2±0.2

SSOP 20S (150mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	13.0 <sup>+0.5/-0.2</sup>
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	16.8 <sup>+0.3/-0.2</sup>
T2	Reel Thickness	22.2±0.2

**Carrier Tape Dimensions**

**SOP 16N (150mil)**

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0±0.3
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	7.5±0.1
D	Perforation Diameter	1.55 <sup>+0.10/-0.00</sup>
D1	Cavity Hole Diameter	1.50 <sup>+0.25/-0.00</sup>
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	10.3±0.1
K0	Cavity Depth	2.1±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

**SSOP 20S (150mil)**

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0 <sup>+0.3/-0.1</sup>
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	7.5±0.1
D	Perforation Diameter	1.5 <sup>+0.1/-0.0</sup>
D1	Cavity Hole Diameter	1.50 <sup>+0.25/-0.00</sup>
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	9.0±0.1
K0	Cavity Depth	2.3±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

**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. (Shenzhen Sales Office)**

5F, Unit A, Productivity Building, No.5 Gaoxin M 2nd Road, Nanshan District, Shenzhen, China 518057  
Tel: 86-755-8616-9908, 86-755-8616-9308  
Fax: 86-755-8616-9722

**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>

Copyright © 2012 by HOLTEK SEMICONDUCTOR INC.

The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Holtek assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Holtek makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Holtek's products are not authorized for use as critical components in life support devices or systems. Holtek reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at <http://www.holtek.com.tw>.