HT48RA0-2/HT48CA0-2
# Remote Type 8-Bit MCU

Technical Document

Features

- Operating voltage: 2.0V~3.6V
- Ten bidirectional I/O lines
- 4 Schmitt trigger input lines
- One carrier output (1/2 or 1/3 duty)
- On-chip crystal and RC oscillator
- Watchdog Timer
- $1K \times 14$ program memory
- 32×8 data RAM
- HALT function and wake-up feature reduce power consumption

- 62 powerful instructions
- Up to $1\mu s$ instruction cycle with 4MHz system clock
- All instructions in 1 or 2 machine cycles
- 14-bit table read instructions
- One-level subroutine nesting
- Bit manipulation instructions
- Low voltage reset function
- 20-pin SSOP package

General Description

The HT48RA0-2/HT48CA0-2 are 8-bit high performance, RISC architecture microcontroller devices specifically designed for multiple I/O control product applications. The mask version HT48CA0-2 is fully pin and functionally compatible with the OTP version HT48RA0-2 device.

The advantages of low power consumption, I/O flexibility, timer functions, oscillator options, 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
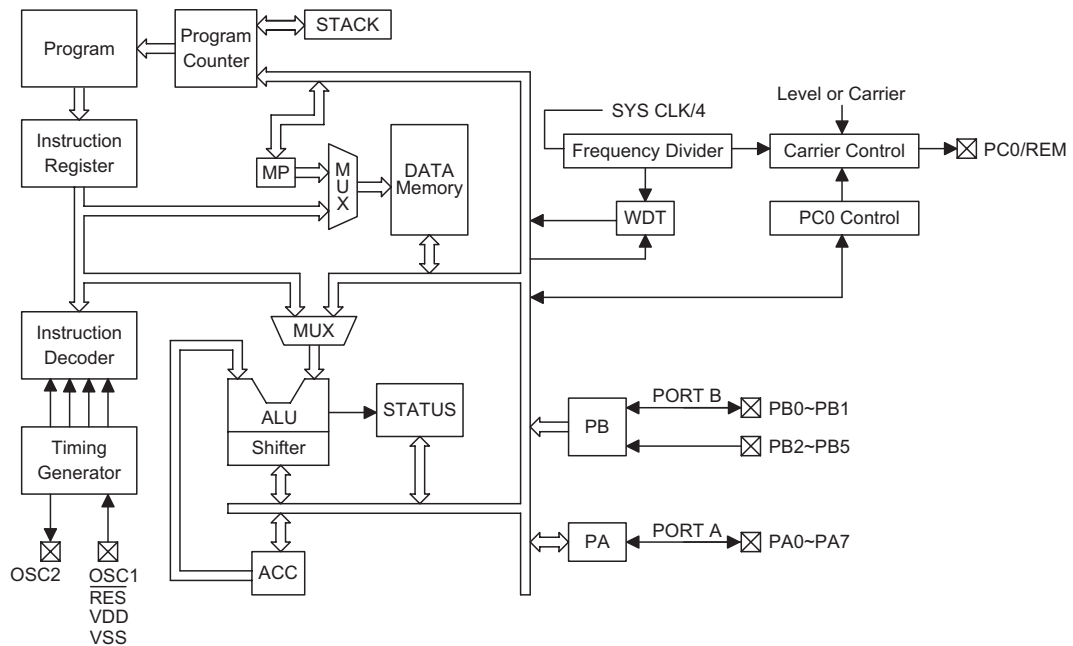
Pin Assignment
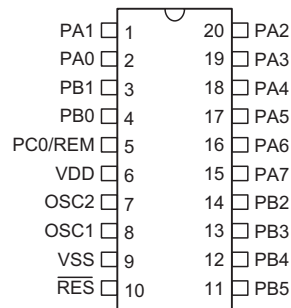
HT48RA0-2/HT48CA0-2
20 SSOP-A

Pin Description

Pin Name	I/O	Code Option	Description
PA0~PA7	I/O	—	Bidirectional 8-bit input/output port with pull-high resistors. Each bit can be determined as NMOS output or Schmitt trigger input by software instructions.
PB0, PB1	I/O	Wake-up or None	2-bit bidirectional input/output lines with pull-high resistors. Each bit can be determined as NMOS output or Schmitt trigger input by software instructions. Each bit can also be configured as wake-up input by code option.
PB2~PB5	I	Wake-up or None	4-bit Schmitt trigger input lines with pull-high resistors. Each bit can be configured as a wake-up input by code option.
PC0/REM	O	Level or Carrier	Level or carrier output pin PC0 can be set as CMOS output pin or carrier output pin by code option.
VDD	—	—	Positive power supply
VSS	—	—	Negative power supply, ground
OSC2 OSC1	O I	Crystal or RC	OSC1, OSC2 are connected to an RC network or a crystal (determined by code option) for the internal system clock. In the case of RC operation, OSC2 is the output terminal for 1/4 system clock (NMOS open drain output).
$\overline{\text{RES}}$	I	—	Schmitt trigger reset input. Active low.

Absolute Maximum Ratings

Supply Voltage ....V_{SS}−0.3V to V_{SS}+4.0V

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

I_{OL} Total 150mA

Total Power Dissipation 500mW

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

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

I_{OH} Total...−100mA

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	—	—	2.0	—	3.6	V
I_{DD}	Operating Current	3V	No load, f_{SYS}=4MHz	—	0.7	1.5	mA
I_{STB}	Standby Current	3V	No load, system HALT	—	—	1	μA
V_{IL1}	Input Low Voltage for I/O Ports	3V	—	0	—	$0.2V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports	3V	—	$0.8V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{\text{RES}}$)	3V	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{\text{RES}}$)	3V	—	$0.9V_{DD}$	—	V_{DD}	V
V_{LVR}	Low Voltage Reset Voltage	—	Temperature=25°C*	1.8	1.9	2.0	V
I_{OL}	I/O Ports Sink Current	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
I_{OH}	PC0/REM Output Source Current	3V	V_{OH}=0.9V_{DD}	−2	−4	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ

Note: ″*″ Guaranteed at 25°C only

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
t_{SYS}	System Clock	3V	—	400	—	4000	kHz
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Power-up, reset or wake-up from HALT	—	1024	—	t_{SYS}
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1	2	ms

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

Functional Description

Execution Flow

The HT48RA0-2/HT48CA0-2 system clock can be derived from a crystal/ceramic resonator oscillator. 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 ROM 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.
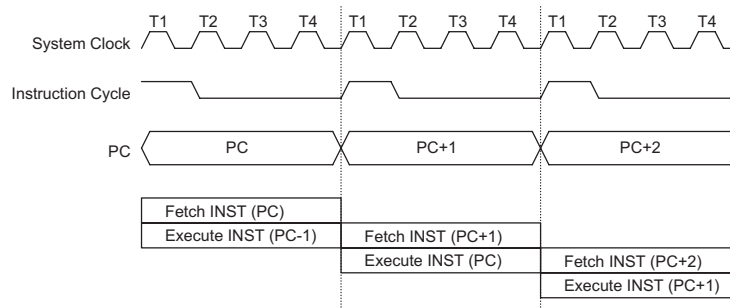
Execution Flow

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   S9~S0: Stack register bits
        #9~#0: Instruction code bits   @7~@0: PCL bits

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.

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 chip reset, the program always begins execution at location 000H.

- Table location
  Any location in the EPROM space can be used as look-up tables. The instructions TABRDC [m] (the current page, one page=256 words) and TABRDL [m] (the last page) transfer the contents of the lower-order byte to the specified data memory, and the

higher-order byte to TBLH (08H). Only the destination of the lower-order byte in the table is well-defined, the other bits of the table word are transferred to the lower portion of 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 (PC) 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 (SP) 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 (only the most recent return address is stored).

Data Memory − RAM

The data memory is designed with 42×8 bits. The data memory is divided into two functional groups: special function registers and general purpose data memory (32×8). Most of them are read/write, but some are read only.

The unused space before 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).

Note: n ranges from 0 to 3

Program Memory

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   @7~@0: Table pointer bits
      P9~P8: Current program counter bits

00H	Indirect Addressing Register
01H	MP
02H	
03H	
04H	
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	
0AH	STATUS
0BH	
0CH	
0DH	
0EH	
0FH	
10H	
11H	
12H	PA
13H	
14H	PB
15H	
16H	PC
17H	
1FH	
20H	General Purpose Data Memory (32 Bytes)
3FH	

Special Purpose Data Memory

▨ : Unused
Read as "00"

RAM Mapping

Indirect Addressing Register

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

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

Accumulator

The accumulator closely relates to ALU operations. It is also mapped to location 05H of the data memory and is 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, bits in the status register 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

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

may give different results from those intended. The TO and PDF flags can only be changed by the Watchdog Timer overflow, chip 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.

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

Oscillator Configuration

There are two oscillator circuits implemented in the microcontroller.
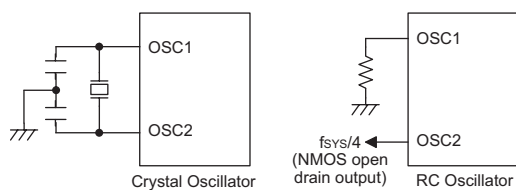
System Oscillator

Both are designed for system clocks; the RC oscillator and the Crystal oscillator, which are determined by code options. No matter what oscillator type is selected, the signal provides the system clock. The HALT mode stops the system oscillator and ignores the external signal to conserve power.

If an RC oscillator is used, an external resistor between OSC1 and VSS in needed and the resistance must range from 51kΩ to 1MΩ. The system clock, divided by 4, is available on OSC2, which can be used to synchronize external logic. The RC oscillator provides the most cost effective solution. However, the frequency of the oscillation may vary with V_{DD}, temperature and the chip itself due to process variations. It is, therefore, not suitable for timing sensitive operations where 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 for the oscillator. No other external components are needed. Instead of a crystal, the resonator can also be connected between OSC1 and OSC2 to get a frequency reference, but two external capacitors in OSC1 and OSC2 are required.

Watchdog Timer – WDT

The clock source of the WDT is implemented by instruction clock (system clock divided by 4). The clock source is processed by a frequency divider and a prescaller to yield various time out periods.

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

Where n= 8~11 selected by code option.

This 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 code 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 an external logic.

A WDT overflow under normal operation will initialize ″chip reset″ and set the status bit ″TO″. To clear the contents of the WDT prescaler, three methods are adopted; external reset (a low level to $\overline{\text{RES}}$), 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 code 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.
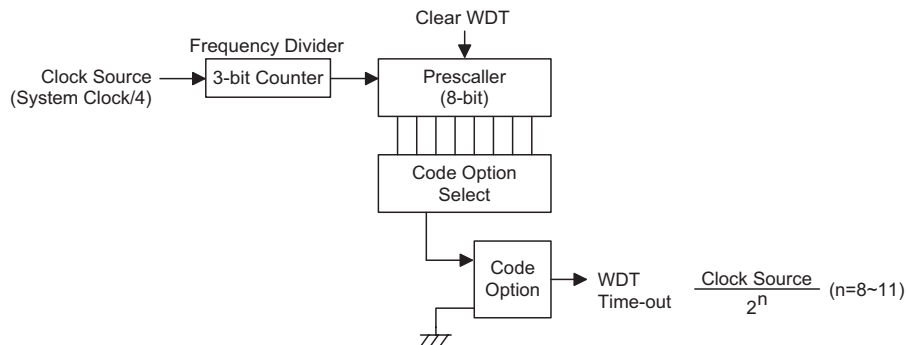
Watchdog Timer

Power Down Operation – HALT

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

- The system oscillator turns off and the WDT stops.
- The contents of the on-chip RAM and registers remain unchanged.
- WDT prescaler are 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 reset or an external falling edge signal on port B. An external reset causes a device initialization. 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 execute the CLR WDT instruction and is set when the HALT instruction is executed. The TO flag is set if the WDT time-out occurs, and causes a wake-up that only resets the program counter and SP, the others keep their original status.

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_{SYS} (system clock period) 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.

Reset

There are three ways in which a reset can occur:

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

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. But when the reset comes from the $\overline{RES}$ pin, the SST delay is disabled. 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

Reset Timing Chart
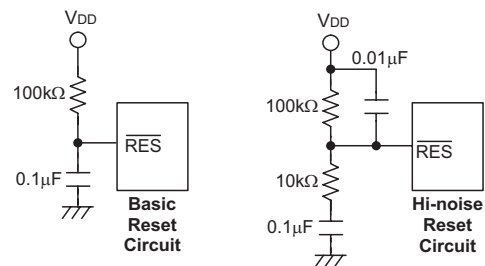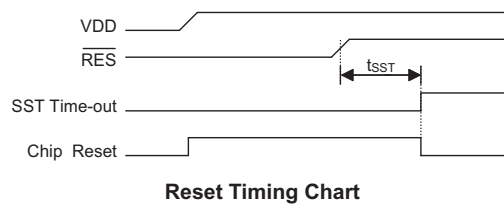
Reset Circuit

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.

Reset Configuration
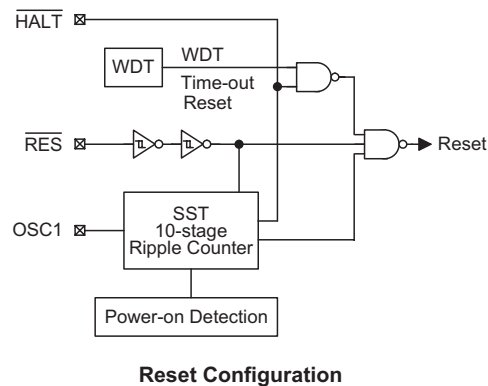
The chip reset status of the registers is summarized in the following table:

Register	Reset (Power On)	WDT Time-out (Normal Operation)	RES Reset (Normal Operation)	RES Reset (HALT)	WDT Time-out (HALT)*
Program Counter	000H	000H	000H	000H	000H
MP	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	0011 1111	0011 1111	0011 1111	0011 1111	uuuu uuuu
PC	---- ---1	---- ---1	---- ---1	---- ---1	---- ---u

Note: "u" means unchanged

"x" means unknown

Carrier

The HT48RA0-2/HT48CA0-2 provides a carrier output which shares the pin with PC0. It can be selected to be a carrier output (REM) or level output pin (PC0) by code option. If the carrier output option is selected, setting PC0="0" to enable carrier output and setting PC0="1" to disable it at low level output.

The clock source of the carrier is implemented by instruction clock (system clock divided by 4) and processed by a frequency divider to yield various carry frequency.

$$\text{Carry Frequency} = \frac{\text{Clock Source}}{m \times 2^n}$$

where m=2 or 3 and n=0~3, both are selected by code option. If m=2, the duty cycle of the carrier output is 1/2 duty. If m=3, the duty cycle of the carrier output can be 1/2 duty or 1/3 duty also determined by code option (with the exception of n=0).

Detailed selection of the carrier duty is shown below:

$m \times 2^n$	Duty Cycle
2, 4, 8, 16	1/2
3	1/3
6, 12, 24	1/2 or 1/3
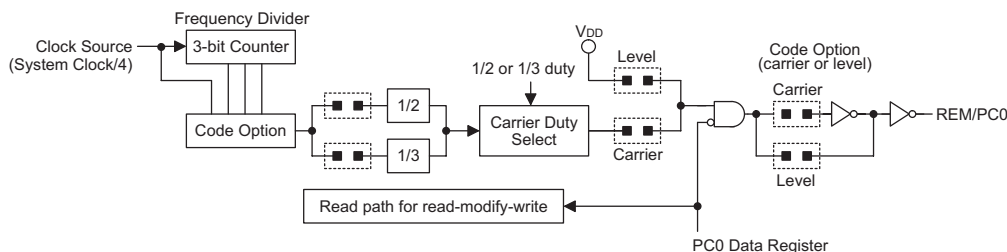
The following table shows examples of carrier frequency selection.

| f_{SYS}	$f_{CARRIER}$	Duty	$m \times 2^n$
455kHz	37.92kHz	1/3 only	3
	56.9kHz	1/2 only	2

Input/Output Ports

There are an 8-bit bidirectional input/output port, a 4-bit input with 2-bit I/O port and one-bit output port in the HT48RA0-2/HT48CA0-2, labeled PA, PB and PC which are mapped to [12H], [14H], [16H] of the RAM, respectively. Each bit of PA can be selected as NMOS output or Schmitt trigger with pull-high resistor by software instruction. PB0~PB1 have the same structure with PA, while PB2~PB5 can only be used for input operation (Schmitt trigger with pull-high resistors). PC is only one-bit output port shares the pin with carrier output. If the level option is selected, the PC is CMOS output.

Both PA and PB for the 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 or 14H). For PA, PB0~PB1 and PC output operation, all data are latched and remain unchanged until the output latch is rewritten.

Carrier/Level Output
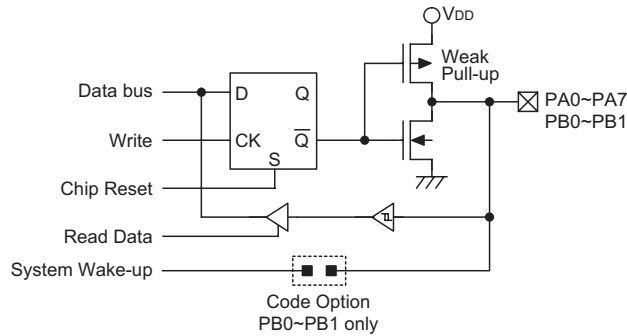
PA, PB Input/Output Lines
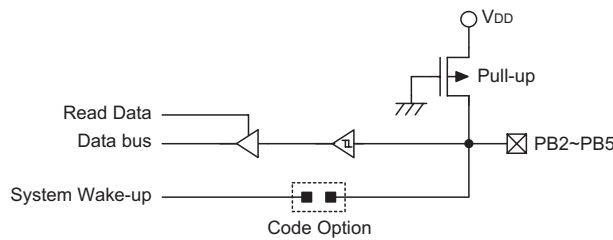
PB Input Lines

When the PA and PB0~PB1 is used for input operation, it should be noted that before reading data from 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, i=0~1 for PB) is executed first to disable related NMOS device, and then ″MOV A, [m]″ to get stable data.

After chip reset, PA and PB remain at a high level input line while PC remain at high level output, if the level option is selected.

Each bit of PA, PB0~PB1 and PC 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]″, ″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 PB has a wake-up capability to the device by code option. The highest seven bits of PC are not physically implemented, on reading them a ″0″ is returned and writing results in a no-operation.

Note: The bit 6 and Bit 7 the PB register (14H) are unused in the HT48RA0-2/HT48CA0-2, any read from that will return the value ″0″. User Should be very careful in transferring the program from the HT48RA0A or HT48RA0-1/HT48CA0-1 device to the HT48RA0-2/HT48CA0-2 device.

Low Voltage Reset − LVR

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

The LVR includes the following specifications:

- The low voltage ($0.9V~V_{LVR}$) has to remain in their original state to exceed 1ms. If the low voltage state does not exceed 1ms, the LVR will ignore it and do not perform a reset function.

- The LVR uses the ″OR″ function with the external $\overline{RES}$ signal to perform chip reset.

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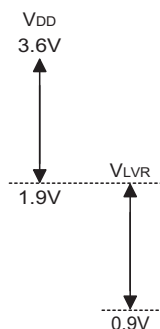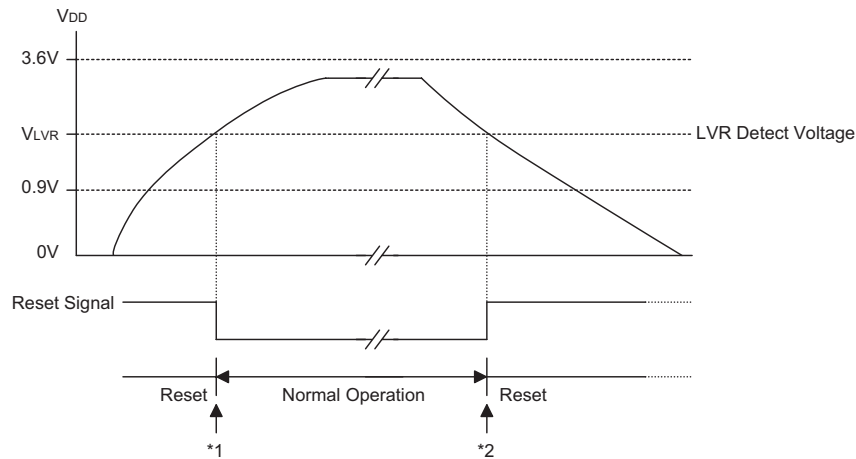
Low Voltage Reset

Note: "*1" To make sure that the system oscillator has stabilized, the SST provides an extra delay of 1024 system clock pulses before entering the normal operation.
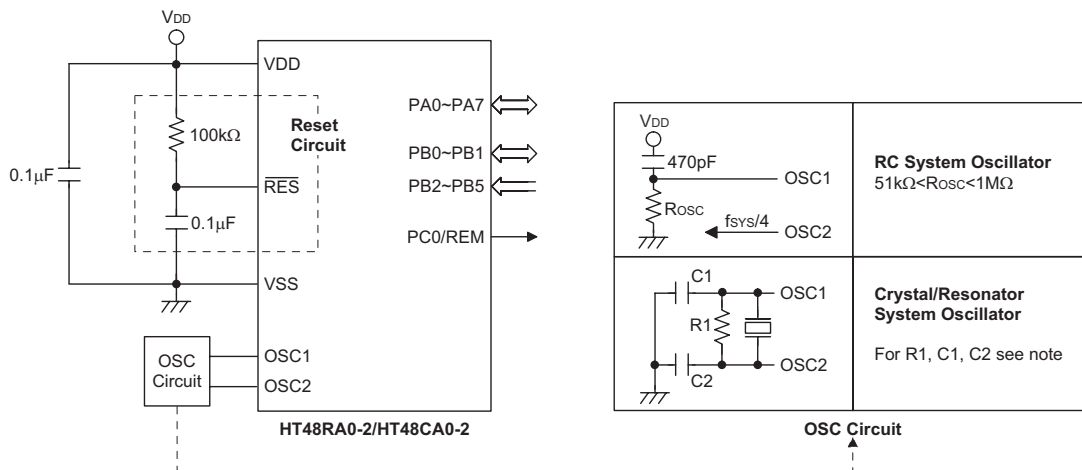
"*2" Since low voltage has to be maintained in its original state and exceed 1ms, therefore 1ms delay enters the reset mode.

Code Option

The following table shows eight kinds of code option in the HT48RA0-2/HT48CA0-2. All the code options must be defined to ensure proper system functioning.

No.	Code Option
1	WDT time-out period selection Time-out period= $\dfrac{2^n}{\text{Clock Source}}$, where n=8~11.
2	WDT enable/disable selection. This option is to decide whether the WDT timer is enabled or disabled.
3	CLR WDT times selection. This option defines how to clear the WDT by instruction. "One time" means that the CLR WDT instruction can clear the WDT. "Two times" means only if both of the CLR WDT1 and CLR WDT2 instructions have been executed, the WDT can be cleared.
4	Wake-up selection. This option defines the wake-up activity function. External input pins (PB only) all have the capability to wake-up the chip from a HALT.
5	Carrier/level output selection. This option defines the activity of PC0 to be carrier output or level output.
6	Carry frequency selection. Carry frequency= $\dfrac{\text{Clock Source}}{(2 \text{ or } 3) \times 2^n}$, where n=0~3.
7	Carrier duty selection. There are two types of selection: 1/2 duty or 1/3 duty. If carrier frequency=Clock Source/(2, 4, 8 or 16), the duty cycle will be 1/2 duty. If carrier frequency=Clock Source/3, the duty cycle will be 1/3 duty. If carrier frequency=Clock Source/(6, 12 or 24), the duty cycle can be 1/2 duty or 1/3 duty.
8	System oscillator selection. RC or crystal oscillator.
9	LVR function: enable or disable

Application Circuits

HT48RA0-2/HT48CA0-2

OSC Circuit

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 V_{DD} 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.

Example

HT48RA0-2/HT48CA0-2
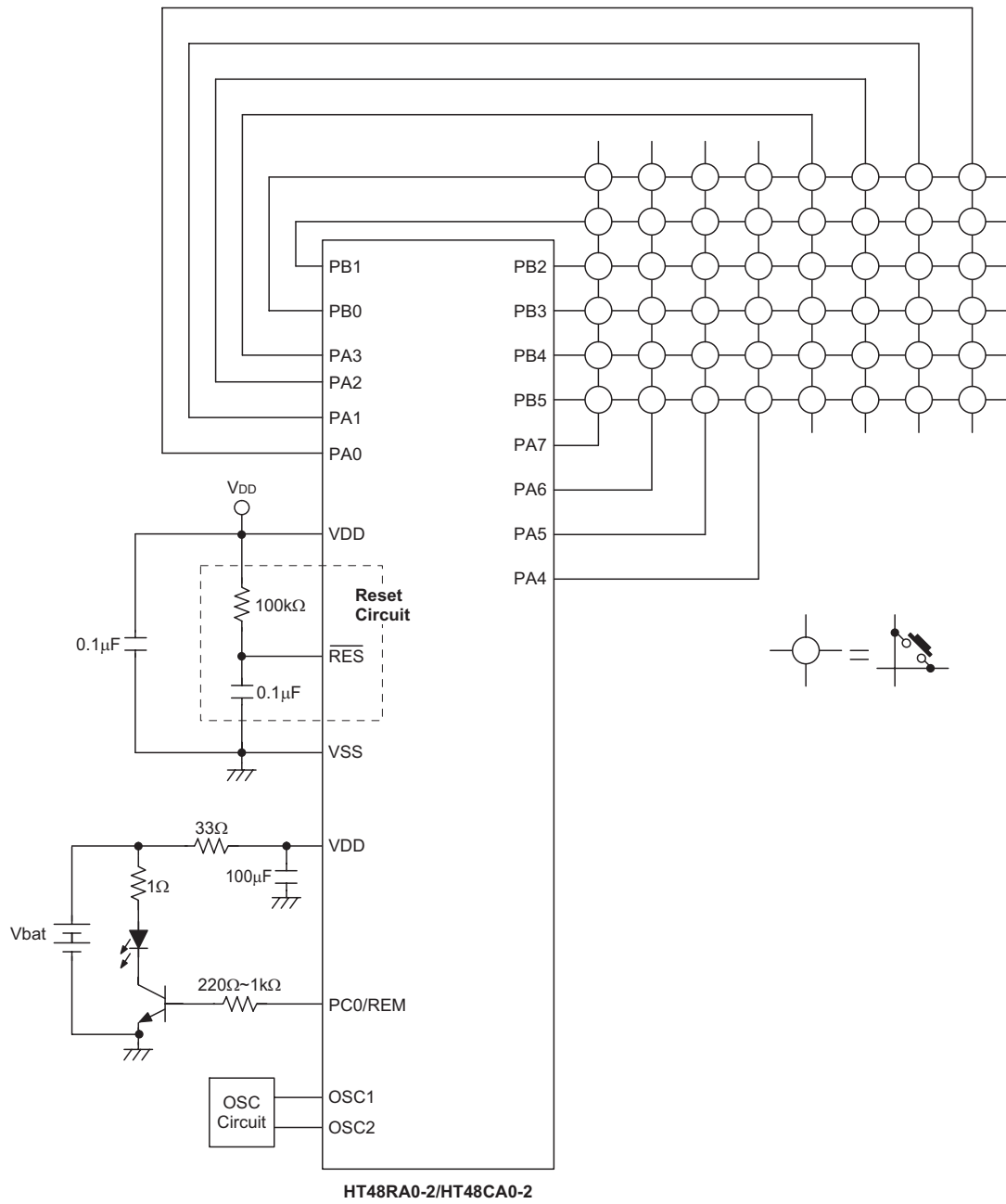
Instruction Set

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	1^{Note}	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^{Note}	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^{Note}	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^{Note}	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1^{Note}	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	1^{Note}	Z
ORM A,[m]	Logical OR ACC to Data Memory	1^{Note}	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1^{Note}	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^{Note}	Z
CPLA [m]	Complement Data Memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment Data Memory with result in ACC	1	Z
INC [m]	Increment Data Memory	1^{Note}	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1^{Note}	Z

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

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 $ACC \leftarrow [m] + 1$

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 $\leftarrow$ 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 $ACC \leftarrow [m]$

Affected flag(s)   None

MOV A,x   Move immediate data to ACC

Description   The immediate data specified is loaded into the Accumulator.

Operation $ACC \leftarrow x$

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 $[m] \leftarrow ACC$

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 $ACC \leftarrow ACC\ ″OR″\ [m]$

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 $[m].(i+1) \leftarrow [m].i; (i = 0\sim6)$
$[m].0 \leftarrow C$
$C \leftarrow [m].7$

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 $ACC.(i+1) \leftarrow [m].i; (i = 0\sim6)$
$ACC.0 \leftarrow C$
$C \leftarrow [m].7$

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 $[m].i \leftarrow [m].(i+1); (i = 0\sim6)$
$[m].7 \leftarrow [m].0$

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 $ACC.i \leftarrow [m].(i+1); (i = 0\sim6)$
$ACC.7 \leftarrow [m].0$

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 $[m].i \leftarrow [m].(i+1); (i = 0\sim6)$
$[m].7 \leftarrow C$
$C \leftarrow [m].0$

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 $ACC.i \leftarrow [m].(i+1); (i = 0\sim6)$
$ACC.7 \leftarrow C$
$C \leftarrow [m].0$

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 sub-tracted 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 sub-tracted from the Accumulator. The result is stored in the Data Memory. Note that if the re-sult 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 in-struction 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 $\leftrightarrow$ [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 $\leftarrow$ [m].7 ~ [m].4 ACC.7 ~ ACC.4 $\leftarrow$ [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 $\leftarrow$ [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] $\leftarrow$ program code (low byte) TBLH $\leftarrow$ 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] $\leftarrow$ program code (low byte) TBLH $\leftarrow$ 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 op-
                            eration. 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 op-
                            eration. 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
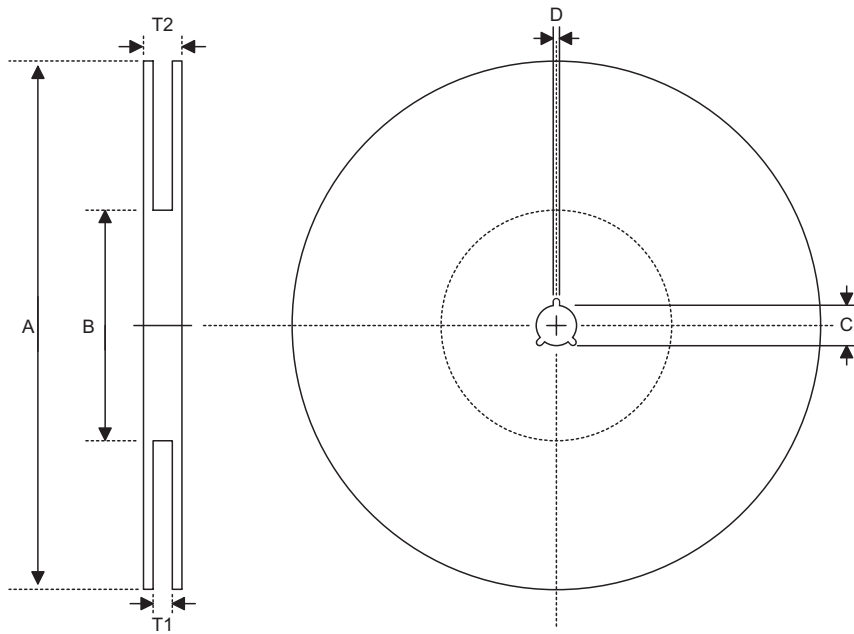
Affected flag(s)   Z

Package Information

20-pin SSOP (150mil) Outline Dimensions
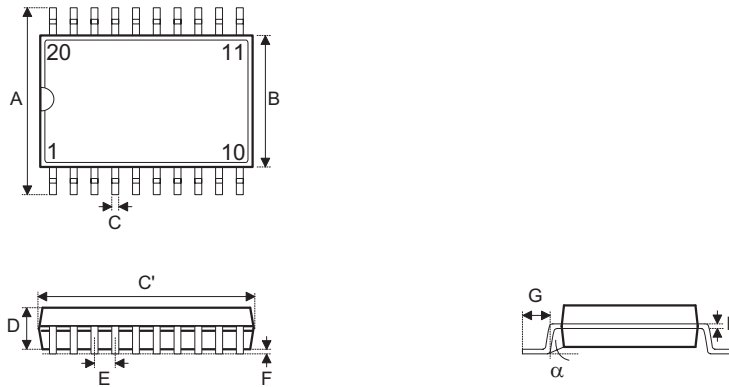
Symbol	Dimensions in mil		
	Min.	**Nom.**	**Max.**
A	228	—	244
B	150	—	158
C	8	—	12
C′	335	—	347
D	49	—	65
E	—	25	—
F	4	—	10
G	15	—	50
H	7	—	10
α	0°	—	8°

Product Tape and Reel Specifications

Reel Dimensions

SSOP 20S (150mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1.0
B	Reel Inner Diameter	62±1.5
C	Spindle Hole Diameter	13.0+0.5 −0.2
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	16.8+0.3 −0.2
T2	Reel Thickness	22.2±0.2

Carrier Tape Dimensions
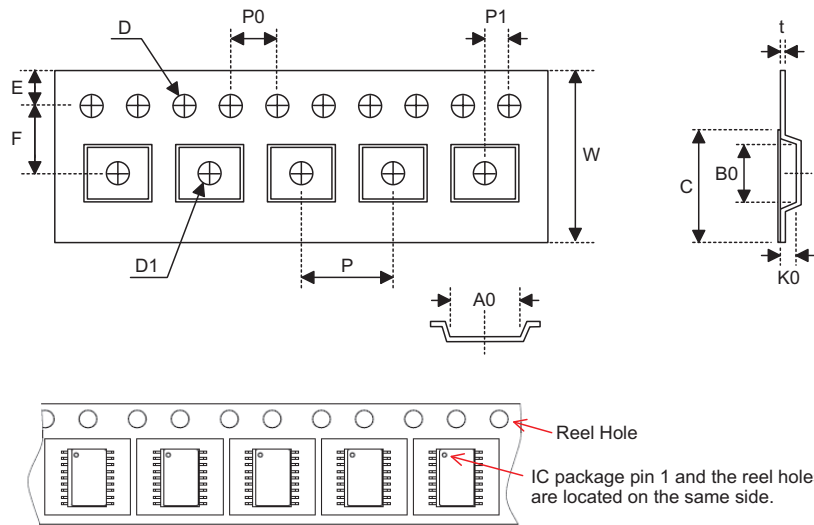
SSOP 20S (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0+0.3 −0.1
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	1.5+0.1
D1	Cavity Hole Diameter	1.5+0.25
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

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 (China) Inc. (Dongguan Sales Office)
Building No. 10, Xinzhu Court, (No. 1 Headquarters), 4 Cuizhu Road, Songshan Lake, Dongguan, China 523808
Tel: 86-769-2626-1300
Fax: 86-769-2626-1311

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