

Technical Document

- [Tools Information](#)
- [FAQs](#)
- [Application Note](#)
 - [HA0003E Communicating between the HT48 & HT46 Series MCUs and the HT93LC46 EEPROM](#)
 - [HA0004E HT48 & HT46 MCU UART Software Implementation Method](#)
 - [HA0005E Controlling the I2C bus with the HT48 & HT46 MCU Series](#)
 - [HA0007E Using the MCU Look Up Table Instructions](#)
 - [HA0049E Read and Write Control of the HT1380](#)

Features

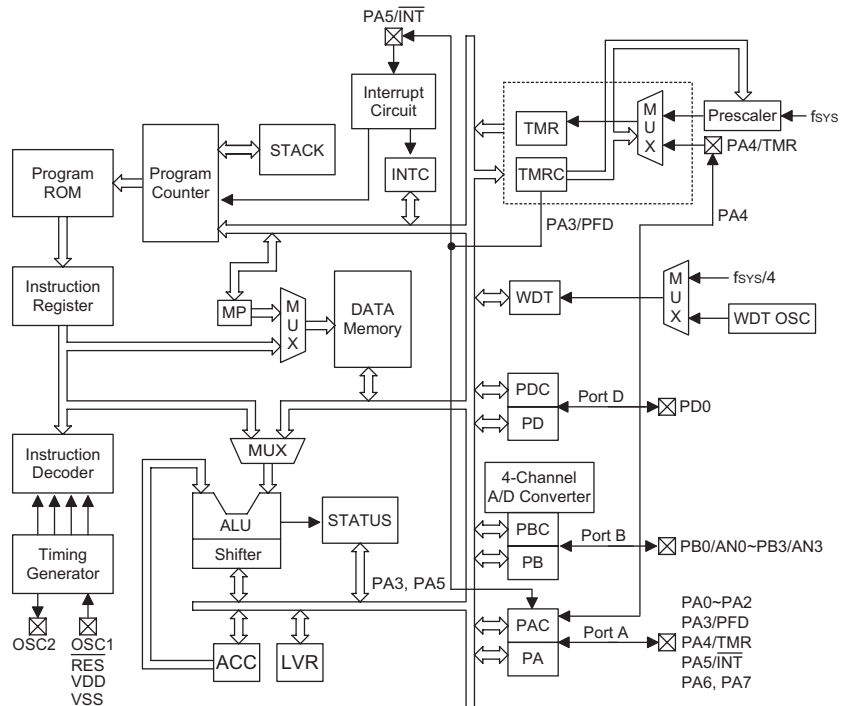
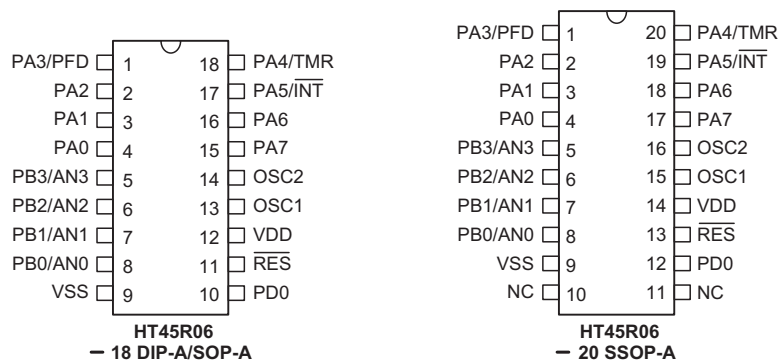
- Operating voltage: 2.2V~5.5V
($I_{DD} < 200\mu A$, when $f_{SYS} = 455kHz$, $V_{DD} = +5V$)
- Operating frequency: 400kHz~2MHz
455kHz: $V_{DD} = 2.2V \sim 5.5V$
1MHz: $V_{DD} = 2.4V \sim 5.5V$
2MHz: $V_{DD} = 3.3V \sim 5.5V$
- 13 bidirectional I/O lines (PA, PB0~PB3, PD0)
- One interrupt input shared with an I/O line
- 8-bit programmable timer/event counter with overflow interrupt and 7-stage prescaler
- On-chip crystal and RC oscillator
- Watchdog Timer
- 1024×14 program memory
- 64×8 data memory RAM
- Supports PFD for sound generation
- HALT function and wake-up feature reduce power consumption
- Up to 2 μs instruction cycle with 2MHz system clock at $V_{DD} = 5V$
- 4-level subroutine nesting
- 4 channels 8-bit resolution A/D converter
- PA with wake-up function
- Bit manipulation instruction
- 14-bit table read instruction
- 63 powerful instructions
- All instructions in 1 or 2 machine cycles
- Low voltage reset function
- Fast start-up: < 5ms
($f_{SYS} = 455kHz$, the \overline{RES} is connected to VDD)
- 18-pin DIP/SOP, 20-pin SSOP packages

General Description

The HT45R06 is an 8-bit high performance, RISC architecture microcontroller devices specifically designed for cost-effective multiple I/O control product applications.

The advantages of low power consumption, I/O flexibility, timer functions, oscillator options, watchdog timer,

buzzer driver, multi-channel A/D converter, Pulse Width Modulation function, HALT and wake-up functions, enhance the versatility of these devices to suit a wide range of A/D application possibilities such as security systems, smoke detectors and smart tags.

Block Diagram

Pin Assignment


Pin Description

Pin Name	I/O	Option	Description
PA0~PA2 PA3/PFD PA4/TMR PA5/INT PA6~PA7	I/O	Pull-high Wake-up PFD	I/O lines with pull-high resistors (bit option). I/O modes of each line are controlled by related control register bit (PAC). Each line of PA can be optioned as a wake-up input (bit option). I/O configurations: Schmitt trigger/CMOS. The PA3, PA4 and PA5 are pin-shared with PFD, TMR and INT, respectively.
PB0/AN0~ PB3/AN3	I/O	Pull-high	I/O lines with pull-high resistors (bit option). I/O modes of each line are controlled by related control register bit (PBC). I/O configurations: Schmitt trigger/CMOS. Each PB line is pin shared with an A/D converter input.
PD0	I/O	Pull-high	I/O lines with pull-high resistors (bit option). I/O modes of each line are controlled by related control register bit (PDC). I/O configurations: Schmitt trigger/CMOS.
OSC1 OSC2	I O	Crystal or RC	A resistor connected across OSC1 and VSS or a crystal across OSC1 and OSC2 will generate the system clock.
RES	—	—	Schmitt trigger reset input. Active low.
VDD	—	—	Positive power supply
VSS	—	—	Negative power supply, ground.

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

Absolute Maximum Ratings

Supply Voltage	$V_{SS}-0.3V$ to $V_{SS}+6.0V$	Storage Temperature	$-50^{\circ}C$ to $125^{\circ}C$
Input Voltage	$V_{SS}-0.3V$ to $V_{DD}+0.3V$	Operating Temperature	$-40^{\circ}C$ to $85^{\circ}C$

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
V _{DD}	Operating Voltage	—	f _{SYS} =455kHz	2.2	—	5.5	V
			f _{SYS} =1MHz	2.4	—	5.5	V
			f _{SYS} =2MHz	3.3	—	5.5	V
I _{DD}	Operating Current (Crystal OSC, RC OSC)	3V	No load, f _{SYS} =2MHz, ADC disable	—	0.5	1.3	mA
		5V		—	1.5	3	mA
		3V	No load, f _{SYS} =455kHz	—	100	150	μA
		5V		—	200	300	μA
I _{STB1}	Standby Current (WDT Enabled)	3V	No load, system HALT	—	—	5	μA
		5V		—	—	10	μA
I _{STB2}	Standby Current (WDT Disabled)	3V	No load, system HALT	—	—	1	μA
		5V		—	—	2	μA
V _{IL1}	Input Low Voltage for I/O Ports, TMR and INT	—	—	0	—	0.3V _{DD}	V
V _{IH1}	Input High Voltage for I/O Ports, TMR and INT	—	—	0.7V _{DD}	—	V _{DD}	V
V _{IL2}	Input Low Voltage ($\overline{\text{RES}}$)	—	—	0	—	0.4V _{DD}	V
V _{IH2}	Input High Voltage ($\overline{\text{RES}}$)	—	—	0.9V _{DD}	—	V _{DD}	V
V _{LVR}	Low Voltage Reset	—	LVR enabled	2.7	3.0	3.3	V
I _{OL}	I/O Port Sink Current	3V	V _{OL} =0.1V _{DD}	4	8	—	mA
		5V		10	20	—	mA
I _{OH}	I/O Port Source Current	3V	V _{OH} =0.9V _{DD}	-2	-4	—	mA
		5V		-5	-10	—	mA
R _{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ
V _{AD}	A/D Input Voltage	—	—	0	—	V _{DD}	V
E _{AD}	A/D Conversion Error	—	—	—	±0.5	±1	LSB
I _{ADC}	Additional Power Consumption, if A/D Converter is Used	3V	—	—	0.5	1	mA
		5V	—	—	1.5	3	mA

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V _{DD}	Conditions				
f _{SYS1}	System Clock (Crystal OSC)	—	2.2V~5.5V	—	455	—	kHz
			2.4V~5.5V	—	1000	—	kHz
			3.3V~5.5V	—	2000	—	kHz
f _{SYS2}	System Clock (RC OSC)	—	2.2V~5.5V	—	455	—	kHz
			2.4V~5.5V	—	1000	—	kHz
			3.3V~5.5V	—	2000	—	kHz
f _{TIMER}	Timer I/P Frequency (TMR)	—	2.2V~5.5V	0	—	455	kHz
			2.4V~5.5V	0	—	1000	kHz
			3.3V~5.5V	0	—	2000	kHz
t _{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V		32	65	130	μs
t _{WDT1}	Watchdog Time-out Period (RC)	3V	—	1.4	2.8~5.6	11	S
		5V		1.1	2.3~4.7	9.4	S
t _{WDT2}	Watchdog Time-out Period (System Clock)	—	—	2 ¹⁷	—	2 ¹⁸	t _{SYS}
t _{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t _{CSST}	System Start-up Timer Period	—	Reset conditions occur, when f _{SYS} = 455kHz and RES connect VDD	—	2.5	5	ms
t _{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t _{AD}	A/D Clock Period	—	—	1	—	—	μs
t _{ADC}	A/D Conversion Time	—	—	—	76	—	t _{AD}
t _{ADCS}	A/D Sampling Time	—	—	—	32	—	t _{AD}

Functional Description

Execution Flow

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

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

Program Counter – PC

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

After accessing a program memory word to fetch an instruction code, the contents of the program counter are

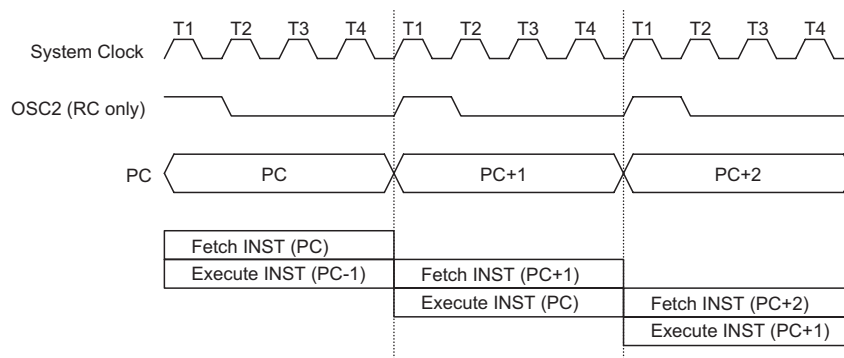
incremented by one. The program counter then points to the memory word containing the next instruction code.

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

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

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

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



Execution Flow

Mode	Program Counter									
	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial Reset	0	0	0	0	0	0	0	0	0	0
External Interrupt	0	0	0	0	0	0	0	1	0	0
Timer/Event Counter Overflow	0	0	0	0	0	0	1	0	0	0
A/D Converter Interrupt	0	0	0	0	0	0	1	1	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

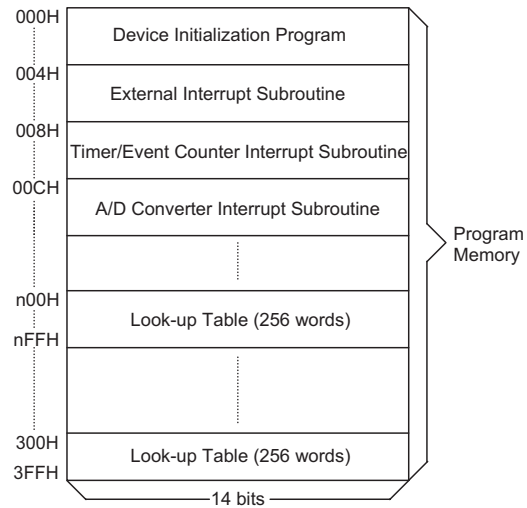
Program Memory – ROM

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

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

- Location 000H
This area is reserved for program initialization. After a chip reset, the program always begins execution at location 000H.
- Location 004H
This area is reserved for the external interrupt service program. If the $\overline{\text{INT}}$ input pin is activated, the interrupt is enabled and the stack is not full, the program begins execution at location 004H.
- Location 008H
This area is reserved for the timer/event counter interrupt service program. If a timer interrupt results from a timer/event counter overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 008H.
- Location 00CH
This area is reserved for the A/D converter interrupt service program. If the interrupt is activated (when the A/D conversion is completed), the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.
- Table location
Any location in the program memory can be used as look-up tables. The instructions "TABRDC [m]" (the current page, 1 page=256 words) and "TABRDL [m]" (the last page) transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH (08H). Only the destination of the lower-order byte in the table is well-defined, the

other bits of the table word are transferred to the lower portion of TBLH, and the remaining 2 bits are read as "0". The Table Higher-order byte register (TBLH) is read only. The table pointer (TBLP) is a read/write register (07H), which indicates the table location. Before accessing the table, the location must be placed in TBLP. The TBLH is read only and cannot be restored. If the main routine and the ISR (Interrupt Service Routine) both employ the table read instruction, the contents of the TBLH in the main routine are likely to be changed by the table read instruction used in the ISR. Errors can occur. In other words, using the table read instruction in the main routine and the ISR simultaneously should be avoided. However, if the table read instruction has to be applied in both the main routine and the ISR, the interrupt is supposed to be disabled prior to the table read instruction. It will not be enabled until the TBLH has been backed up. All table related instructions require two cycles to complete the operation. These areas may function as normal program memory depending upon the requirements.



Program Memory

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

Table Location

Note: *9~*0: Table location bits

P9, P8: Current program counter bits

@7~@0: Table pointer bits

Stack Register – STACK

This is a special part of the memory which is used to save the contents of the program counter only. The stack is organized into 4 levels and is neither part of the data nor part of the program space, and is neither readable nor writable. The activated level is indexed by the stack pointer (SP) and is neither readable nor writeable. At a subroutine call or interrupt acknowledgment, the contents of the program counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction (RET or RETI), the program counter is restored to its previous value from the stack. After a chip reset, the SP will point to the top of the stack.

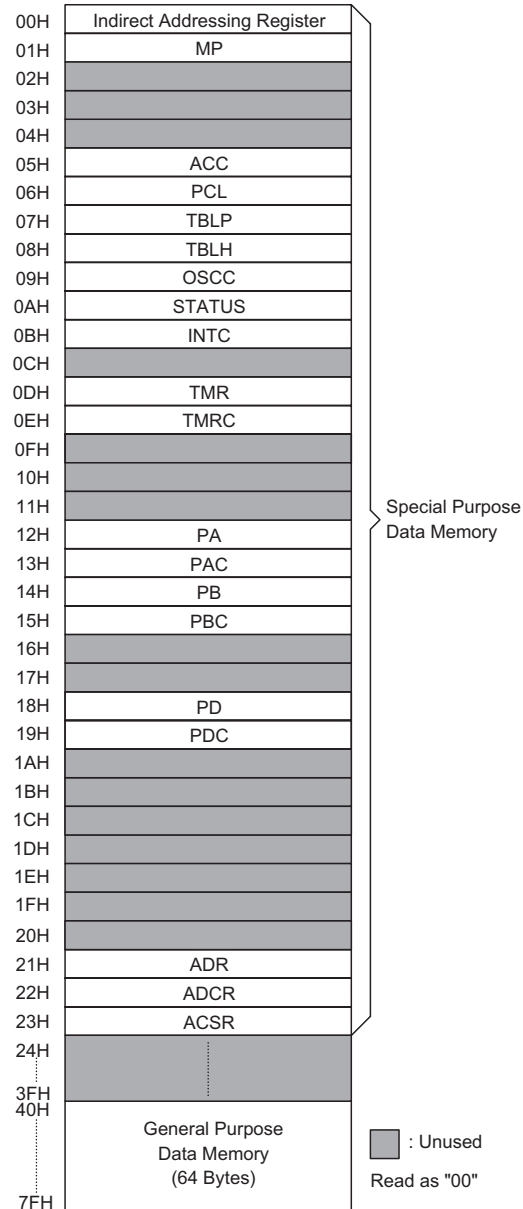
If the stack is full and a non-masked interrupt takes place, the interrupt request flag will be recorded but the acknowledge will be inhibited. When the stack pointer is decremented (by RET or RETI), the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. In a similar case, if the stack is full and a "CALL" is subsequently executed, stack overflow occurs and the first entry will be lost (only the most recent 4 return addresses are stored).

Data Memory – RAM

The data memory is designed with 85×8 bits. The data memory is divided into 2 functional groups: special function registers and general purpose data memory (64×8). Most of them are read/write, but some are read only.

The special function registers include the indirect addressing register (00H), program counter lower-order byte register (PCL;06H), memory pointer register (MP;01H), accumulator (ACC;05H), table pointer (TBLP;07H), table higher-order byte register (TBLH;08H), system quick start-up (OSCC;09H) status register (STATUS;0AH), interrupt control register (INTC;0BH), a Timer/Event Counter register (TMR;0DH), a Timer/Event Counter control register (TMRC;0EH), I/O port data registers (PA;12H, PB;14H, PD;18H), I/O port control registers (PAC;13H, PBC;15H, PDC;19H), A/D result register (ADR;21H), A/D control register (ADCR;22H) and A/D clock setting register (ACSR;23H). The remaining space before the 40H is reserved for future expansion and reading these locations will return the result "00H". The general purpose data memory, addressed from 40H to 7FH, is used for data and control information under instruction commands.

All of the data memory areas can handle arithmetic, logic, increment, decrement and rotate operations directly. Except for some dedicated bits, each bit in the data memory can be set and reset by "SET [m].i" and "CLR [m].i". They are also indirectly accessible through memory pointer register (MP;01H).



RAM Mapping

Indirect Addressing Register

Location 00H is an indirect addressing register that is not physically implemented. Any read/write operation of [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. 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 is closely related to ALU operations. It is also mapped to location 05H of the data memory and can carry out immediate data operations. The data movement between two data memory locations must pass through the accumulator.

Arithmetic and Logic Unit – ALU

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

- Arithmetic operations (ADD, ADC, SUB, SBC, DAA)
- Logic operations (AND, OR, XOR, CPL)
- Rotation (RL, RR, RLC, RRC)
- Increment and Decrement (INC, DEC)
- Branch decision (SZ, SNZ, SIZ, SDZ)

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

Status Register – STATUS

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

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

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

In addition, on entering the interrupt sequence or executing 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.

Interrupt

The microcontroller provides external interrupts, an internal timer/event counter overflow interrupt, and an A/D converter end-of-conversion interrupt. The interrupt control registers (INTC;0BH) contains the interrupt control bits to set the enable/disable and the interrupt request flags.

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

All these kinds of interrupts have a wake-up capability. As an interrupt is serviced, a control transfer occurs by pushing the program counter onto the stack, followed by a branch to a subroutine at specified location in the program memory. Only the program counter is pushed onto the stack. If the contents of the register or status register (STATUS) are altered by the interrupt service program which corrupts the desired control sequence, the contents should be saved in advance.

Bit No.	Label	Function
0	C	C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation, otherwise C is cleared. C is also affected by a rotate through carry instruction.
1	AC	AC is set if an operation results in a carry out of the low nibbles in addition or no borrow from the high nibble into the low nibble in subtraction, otherwise AC is cleared.
2	Z	Z is set if the result of an arithmetic or logic operation is zero, otherwise Z is cleared.
3	OV	OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
4	PDF	PDF is cleared by 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

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1=enable; 0=disable)
1	E EI	Controls the external interrupt (1=enable; 0=disable)
2	ETI	Controls the timer/event counter interrupt (1=enable; 0=disable)
3	EADI	Controls the A/D converter interrupt (1=enable; 0=disable)
4	EIF	External interrupt request flag (1=active; 0=inactive)
5	TF	Internal timer/event counter request flag (1=active; 0=inactive)
6	ADF	A/D converter request flag (1=active; 0=inactive)
7	—	Unused bit, read as "0"

INTC (0BH) Register

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

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

The A/D converter end-of-conversion interrupt is initialized by setting the A/D end-of-conversion interrupt request flag (bit 6 of the INTC), caused by an end of A/D conversion. When the interrupt is enabled, the stack is not full and the end of A/D conversion interrupt request flag is set, a subroutine call to location 00CH will occur. The related interrupt request flag will be reset and the EMI bit cleared to disable further interrupts.

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

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

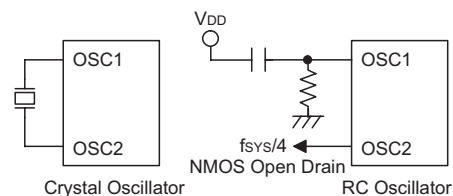
Interrupt Source	Priority	Vector
External Interrupt	1	04H
Timer/Event Counter Overflow	2	08H
A/D Converter Interrupt	3	0CH

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

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

Oscillator Configuration

There are two oscillator circuits in the microcontroller.



System Oscillator

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

If an RC oscillator is used, an external resistor between OSC1 and VSS is required and the resistance must range from 30kΩ to 750kΩ. The system clock, divided by 4, is available on OSC2, which can be used to synchronize external logic. The RC oscillator provides the most cost effective solution. However, the frequency of oscillation may vary with VDD, temperatures and the chip itself due to process variations. It is, therefore, not suitable for timing sensitive operations where an accurate oscillator frequency is desired.

If the Crystal oscillator is used, a crystal across OSC1 and OSC2 is needed to provide the feedback and phase shift required for the oscillator, and no other external components are required. Instead of a crystal, a resonator can also be connected between OSC1 and OSC2 to get a frequency reference, but two external capacitors in OSC1 and OSC2 are required (if the oscillating frequency is less than 1MHz).

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

Oscillator Control Register – OSCC

The QOSC is a control bit for system quick start-up oscillation. The QOSC default setting is enabled. After system start-up is finished, the QOSC must be cleared by user to reduce power consumption.

Bit No.	Label	Function
0	QOSC	System quick start-up oscillation 0=Disable 1=Enable (Default setting)
1~7	—	Unused bit, read as "0"

OSCC (09H) Register

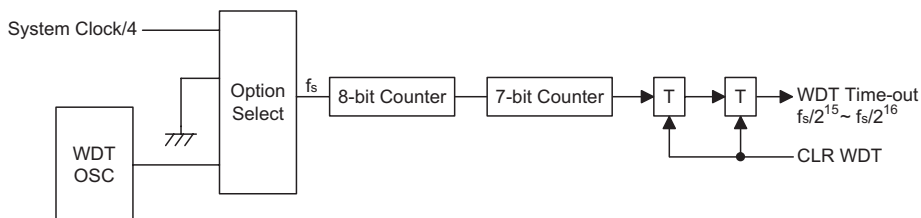
Watchdog Timer – WDT

The WDT clock source is implemented by a dedicated RC oscillator (WDT oscillator) or instruction clock (system clock divided by 4) determined by options. 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 option. If the watchdog timer is disabled, all executions related to the WDT result in no operation.

Once the internal WDT oscillator (RC oscillator with a period of 65μs at 5V normally) is selected, it is first divided by 2¹⁶ to get the nominal time-out period of approximately 5.1s at 5V. This time-out period may vary with temperature, VDD and process variations. By invoking the WDT, prescaler, longer time-out periods can be realized. If the WDT oscillator is disabled, the WDT clock may still come from the instruction clock and operates in the same manner except that in the HALT state the WDT may stop counting and lose its protecting purpose. In this situation the logic can only be restarted by external logic.

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

The WDT overflow under normal operation will initialize a "chip reset" and set the status bit "TO". But in the HALT mode, the overflow will initialize a "warm reset", and only the Program Counter and SP are reset to zero. To clear the WDT contents (including the WDT prescaler), three methods are adopted; external reset (a low level to \overline{RES}), software instruction and a "HALT" instruction. The software instructions include "CLR WDT" and the other set – "CLR WDT1" and "CLR WDT2". Of these two types of instruction, only one can be active depending on the option – "CLR WDT times selection option". If the "CLR WDT" is selected (i.e. CLRWDT times equal one), any execution of the "CLR WDT" instruction will clear the WDT. In the case that "CLR WDT1" and "CLR WDT2" are chosen (i.e. CLRWDT times equal two), these two instructions must be executed to clear the WDT, otherwise, the WDT may reset the chip as a result of 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 will be turned off but the WDT oscillator keeps running (if the WDT oscillator is selected).
- The contents of the on-chip RAM and registers remain unchanged.
- WDT and WDT prescaler will be cleared and re-counted again (if the WDT clock is from the WDT oscillator).
- All of the I/O ports maintain their original status.
- The PDF flag is set and the TO flag is cleared.

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

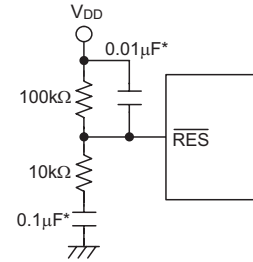
The port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each bit in port A can be independently selected to wake up the device by the options. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If it awakens from an interrupt, two sequences may occur. If the related interrupt is disabled or the interrupt is enabled but the stack is full, the program will resume execution at the next instruction. If the interrupt is enabled and the stack is not full, a regular interrupt response takes place. If an interrupt request flag is set to "1" before entering the HALT mode, the wake-up function of the related interrupt will be disabled. Once a wake-up event occurs, it takes 2.5ms (system clock period) to resume normal operation. In other words, a dummy period will be inserted after wake-up. If the wake-up results from an interrupt acknowledge, the actual interrupt subroutine execution will be delayed by one or more cycles. If the wake-up results in the next instruction execution, this will be executed immediately after the dummy period is finished.

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

Reset

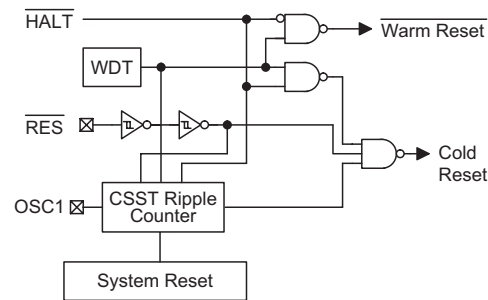
There are three ways in which a reset can occur:

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

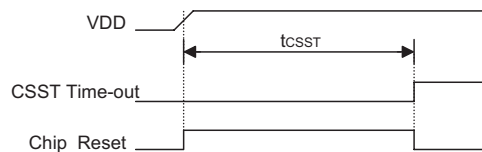


Reset Circuit

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



Reset Configuration



Reset Timing Chart

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

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

Note: "u" means "unchanged"

To guarantee that the system oscillator is started and stabilized, the CSST (System Start-up Timer) provides an extra delay of 2.5ms when the system resets (power-up, WDT time-out or $\overline{\text{RES}}$ reset) or awakes from the HALT state.

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

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

The functional unit chip reset status are shown below.

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

The registers state are summarized in the following table.

Register	Reset (Power On)	WDT time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (HALT)*
MP	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
Program Counter	000H	000H	000H	000H	000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	--xx xxxx	--uu uuuu	--uu uuuu	--uu uuuu	--uu uuuu
OSCC	---- --1	---- --1	---- --1	---- --1	---- --1
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRC	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
PBC	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
PD	---- --1	---- --1	---- --1	---- --1	---- --u
PDC	---- --1	---- --1	---- --1	---- --1	---- --u
ADR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	1--- --00	1--- --00	1--- --00	1--- --00	u--- --uu

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

Timer/Event Counter

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

Using external clock input allows the user to count external events, measure time internals or pulse widths, or generate an accurate time base. Using the internal clock allows the user to generate an accurate time base.

The timer/event counter can generate a PFD signal by using external or internal clock and PFD frequency is determine by the equation $f_{INT}/[2 \times (256-N)]$.

There are two registers related to the timer/event counter; TMR ([0DH]), TMRC ([0EH]). Two physical registers are mapped to TMR location. Writing TMR makes the starting value be placed in the timer/event counter preload register and reading TMR retrieves the contents of the timer/event counter. The TMRC is a timer/event counter control register, which defines some options.

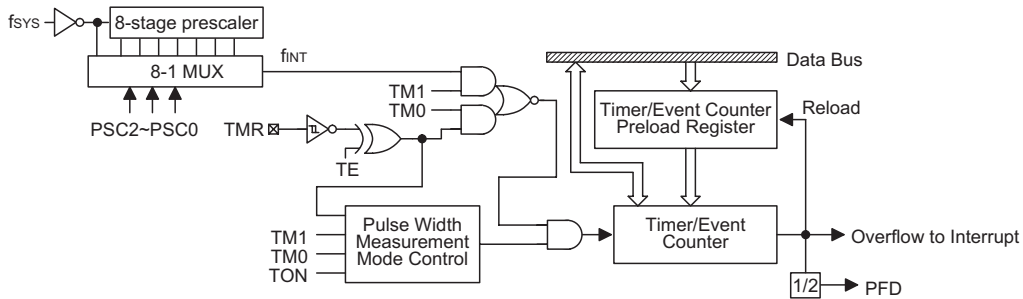
The TM0, TM1 bits define the operating mode. The event count mode is used to count external events, which means the clock source comes from an external (TMR) pin. The timer mode functions as a normal timer with the clock source coming from the f_{INT} clock. The pulse width measurement mode can be used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

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

In the pulse width measurement mode with the TON and TE bits equal to one, once the TMR has received a transient from low to high (or high to low if the TE bits is "0") it will start counting until the TMR returns to the original level and resets the TON. The measured result will remain in the timer/event counter even if the activated transient occurs again. In other words, only one cycle measurement can be done. Until setting the TON, the cycle measurement will function again as long as it receives further transient pulse. Note that in this operating mode, the timer/event counter starts counting not according to the logic level but according to the transient edges. In the case of counter overflows, the counter is reloaded from the timer/event counter preload register and issues an interrupt request just like the other two modes. To enable the counting operation, the timer ON bit (TON; bit 4 of the TMRC) should be set to 1. In the pulse width measurement mode, the TON will be cleared automatically after the measurement cycle is completed. But in the other two modes the TON can only be reset by instructions. The timer/event counter overflow is one of the wake-up sources. No matter what the operation mode is, writing a 0 to ET1 can disable the interrupt service.

Bit No.	Labels	Function
0 1 2	PSC0 PSC1 PSC2	Defines the prescaler stages, PSC2, PSC1, PSC0= 000: $f_{INT}=f_{SYS}/1$ 001: $f_{INT}=f_{SYS}/2$ 010: $f_{INT}=f_{SYS}/4$ 011: $f_{INT}=f_{SYS}/8$ 100: $f_{INT}=f_{SYS}/16$ 101: $f_{INT}=f_{SYS}/32$ 110: $f_{INT}=f_{SYS}/64$ 111: $f_{INT}=f_{SYS}/128$
3	TE	Defines the TMR active edge of the timer/event counter: In Event Counter Mode (TM1, TM0)=(0, 1): 1: count on falling edge; 0: count on rising edge In Pulse Width measurement mode (TM1, TM0)=(1, 1): 1: start counting on the rising edge, stop on the falling edge; 0: start counting on the falling edge, stop on the rising edge
4	TON	Enables or disables the timer counting (0=disable; 1=enable)
5	—	Unused bit, read as "0"
6 7	TM0 TM1	Defines the operating mode, TM1, TM0= 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMRC (0EH) Register



Timer/Event Counter

In the case of timer/event counter OFF condition, writing data to the timer/event counter preload register will also reload that data to the timer/event counter. But if the timer/event counter is turned on, data written to it will only be kept in the timer/event counter preload register. The timer/event counter will still operate until overflow occurs. When the timer/event counter (reading TMR) is read, the clock will be blocked to avoid errors. As clock blocking may result in a counting error, this must be taken into consideration by the programmer.

The bits 0~2 of the TMRC can be used to define the pre-scaling stages of the internal clock sources of the timer/event counter. The definitions are as shown. The overflow signal of the timer/event counter can be used to generate PFD signals for buzzer driving.

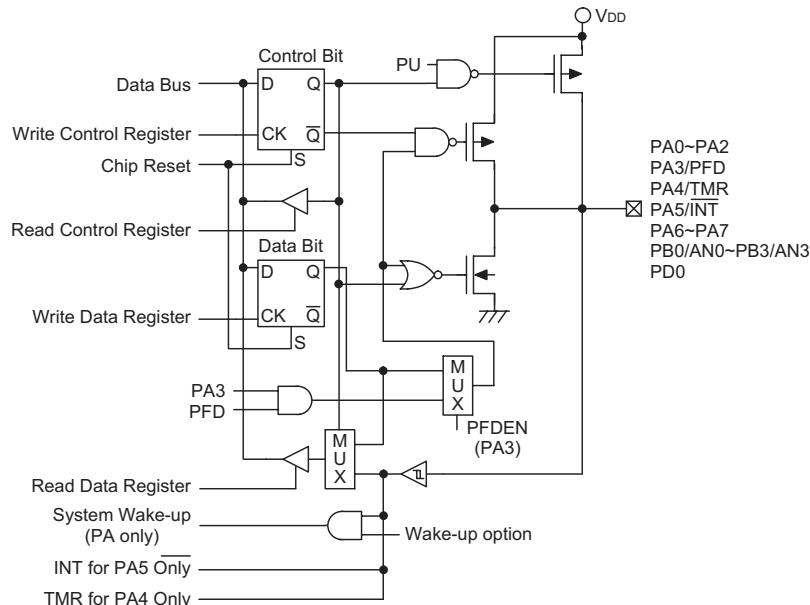
Input/Output Ports

There are 13 bidirectional input/output lines in the microcontroller, labeled from PA, PB and PD, which are mapped to the data memory of [12H], [14H] and [18H] respectively. All of these I/O ports can be used for input

and output operations. For input operation, these ports are non-latching, that is, the inputs must be ready at the T2 rising edge of instruction "MOV A,[m]" (m=12H, 14H or 18H). For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Each I/O line has its own control register (PAC, PBC, PDC) to control the input/output configuration. With this control register, CMOS output or Schmitt trigger input with or without pull-high resistor structures can be re-configured dynamically under software control. To function as an input, the corresponding latch of the control register must write "1". The input source also depends on the control register. If the control register bit is "1", the input will read the pad state. If the control register bit is "0", the contents of the latches will move to the internal bus. The latter is possible in the "read-modify-write" instruction.

For output function, CMOS is the only configuration. These control registers are mapped to locations 13H, 15H and 19H.



Input/Output Ports

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

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

Each line of port A has the capability of waking-up the device. The highest 7-bit of port D and 4 bits of port B are not physically implemented, on reading them a "0" is returned whereas writing then results in no-operation. See Application note.

There is a pull-high option available for all I/O lines. Once the pull-high option is selected, all I/O lines have pull-high resistors. Otherwise, the pull-high resistors are absent. It should be noted that a non-pull-high I/O line operating in input mode will cause a floating state.

The PA3 is pin-shared with the PFD. If the PFD option is selected, the output signal in output mode of PA3 will be the PFD signal generated by the timer/event counter overflow signal. Those in the input mode always maintain their original functions. Once the PFD option is selected, the PFD output signal is controlled by PA3 data register only. Writing "1" to PA3 data register will enable the PFD output function and writing "0" will force the PA3 to remain at "0". The I/O functions of PA3 are shown below.

I/O Mode	I/P (Normal)	O/P (Normal)	I/P (PFD)	O/P (PFD)
PA3	Logical Input	Logical Output	Logical Input	PFD (Timer on)

Note: The PFD frequency is the timer/event counter overflow frequency divided by 2.

The PA4 and PA5 are pin-shared with TMR and $\overline{\text{INT}}$ pins respectively.

The PB can also be used as A/D converter inputs. The A/D function will be described later.

A/D Converter

The 4 channels and 8-bit resolution A/D converter are implemented in this microcontroller. The reference voltage is VDD. The A/D converter contains three special registers, namely, ADR (21H), ADCR (22H) and ACSR (23H). The ADR is A/D result register and is read-only. After the A/D conversion is completed, the ADR should be read to retrieve the conversion result data. The ADCR is an A/D converter control register, which defines the A/D channel number, analog channel select, start A/D conversion control bit and end of A/D conversion flag. If users want to start an A/D conversion, they should define the PB configuration, select the converted analog channel, and give START bit a raising edge and falling edge (0→1→0). At the end of A/D conversion, the EOCB bit is cleared and an A/D converter interrupt occurs (if the A/D converter interrupt is enabled). The ACSR is A/D clock setting register, which is used to select the A/D clock source.

The A/D converter control register is used to control the A/D converter. The bit2~bit0 of the ADCR are used to select an analog input channel. There are a total of eight channels to select. The bit5~bit3 of the ADCR are used to set PB configurations. PB can be an analog input or as digital I/O line determined by these 3 bits. Once a PB line is selected as an analog input, the I/O functions and pull-high resistor of this I/O line are disabled and the A/D converter circuit is powered on. The EOCB bit (bit6 of the ADCR) is end of A/D conversion flag. Check this bit to know when A/D conversion is completed. The START bit of the ADCR is used to begin the conversion of the A/D converter. Giving START bit a rising edge and falling edge means that the A/D conversion has started. In order to ensure that A/D conversion is completed, the START should remain at "0" until the EOCB is cleared to "0" (end of A/D conversion).

The bit 7 of the ACSR is used for testing purposes only. It cannot be used by the users. The bit1 and bit0 of the ACSR are used to select the A/D clock sources.

Bit No.	Label	Function
0 1	ADCS0 ADCS1	Selects the A/D converter clock source 00=system clock/2 01=system clock/8 10=system clock/32 11=undefined
2~6	—	Unused bit, read as "0"
7	TEST	For test mode used only

ACSR (23H) Register

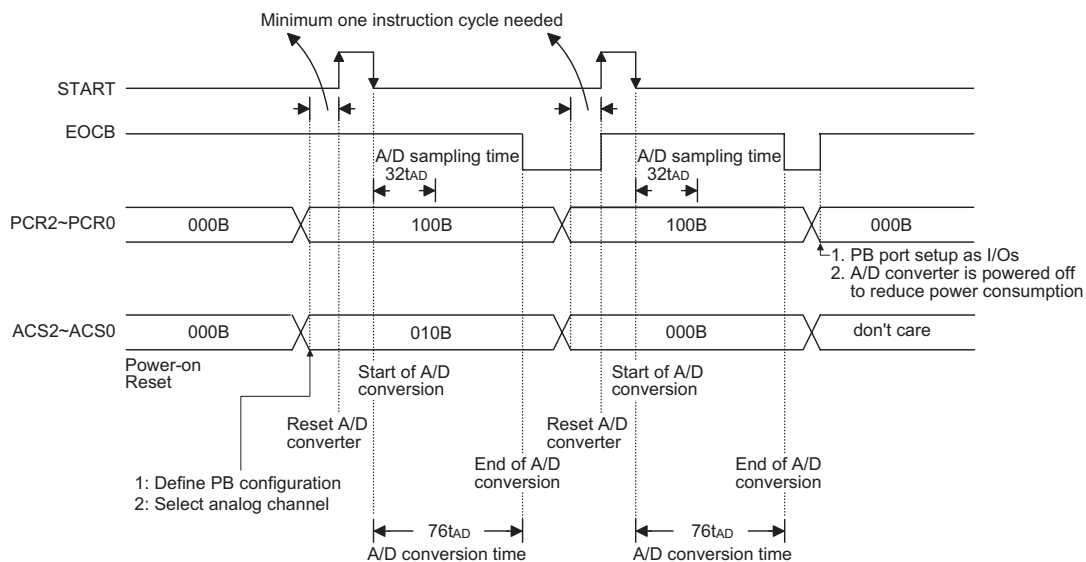
Bit No.	Label	Function
0	ACS0	ACS2, ACS1, ACS0: Select A/D channel 0, 0, 0: AN0 0, 0, 1: AN1 0, 1, 0: AN2 0, 1, 1: AN3 1, X, X: undefined, cannot be used
1	ACS1	
2	ACS2	
3	PCR0	PCR2, PCR1, PCR0: PB3~PB0 configurations 0, 0, 0: PB3 PB2 PB1 PB0 (The ADC circuit is power off to reduce power consumption.) 0, 0, 1: PB3 PB2 PB1 AN0 0, 1, 0: PB3 PB2 AN1 AN0 0, 1, 1: PB3 AN2 AN1 AN0 1, x, x: AN3 AN2 AN1 AN0
4	PCR1	
5	PCR2	
6	EOCB	End of A/D conversion flag. (0: end of A/D conversion)
7	START	Start the A/D conversion 0→1→0= Start 0→1= Reset A/D converter and set EOCB to "1"

ADCR (22H) Register

When the A/D conversion is completed, the A/D interrupt request flag is set. The EOCB bit is set to "1" when the START bit is set from "0" to "1".

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
D7	D6	D5	D4	D3	D2	D1	D0

Note: D0~D7 is A/D conversion result data bit LSB~MSB.

ADR (21H) Register


Note: A/D clock must be $f_{sys}/2$, $f_{sys}/8$ or $f_{sys}/32$

A/D Conversion Timing

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the EOCB bit in the ADCR register is used to detect when the conversion cycle is complete, whereas in the second example, the A/D interrupt is used to determine when the conversion is complete.

Example: using EOCB Polling Method to detect end of conversion

```

clr INTC0.3           ; disable A/D interrupt in interrupt control register
mov a,00100000B
mov ADCR,a           ; setup ADCR register to configure Port PB0~PB3 as A/D inputs and select
                    ; AN0 to be connected to the A/D converter

mov a,00000001B
mov ACSR,a           ; setup the ACSR register to select fsys/8 as the A/D clock

```

Start_conversion:

```

clr ADCR.7
set ADCR.7           ; reset A/D
clr ADCR.7           ; start A/D

```

Polling_EOC:

```

sz ADCR.6           ; poll the ADCR register EOCB bit to detect end of A/D conversion
jmp polling_EOC     ; continue polling
mov a,ADR            ; read conversion result from the ADR register
mov adr_buffer,a    ; save result to user defined register
:
:
jmp start_conversion ; start next A/D conversion

```

Example: using the Interrupt method to detect end of conversion

```

set INTC0.0         ; interrupt global enable
set INTC0.3         ; enable A/D interrupt in interrupt control register
mov a,00100000B
mov ADCR,a           ; setup ADCR register to configure Port PB0~PB3 as A/D inputs and select
                    ; AN0 to be connected to the A/D converter

mov a,00000001B
mov ACSR,a           ; setup the ACSR register to select fsys/8 as the A/D clock

```

start_conversion:

```

clr ADCR.7
set ADCR.7           ; reset A/D
clr ADCR.7           ; start A/D
:
:

```

; interrupt service routine

EOC_service routine:

```

mov a_buffer,a      ; save ACC to user defined register
mov a,ADR            ; read conversion result from the ADR register
mov adr_buffer,a    ; save result to user defined register

clr ADCR.7
set ADCR.7           ; reset A/D
clr ADCR.7           ; start A/D

mov a,a_buffer      ; restore ACC from temporary storage
reti

```

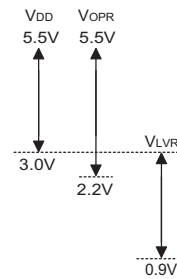
Low Voltage Reset – LVR

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.

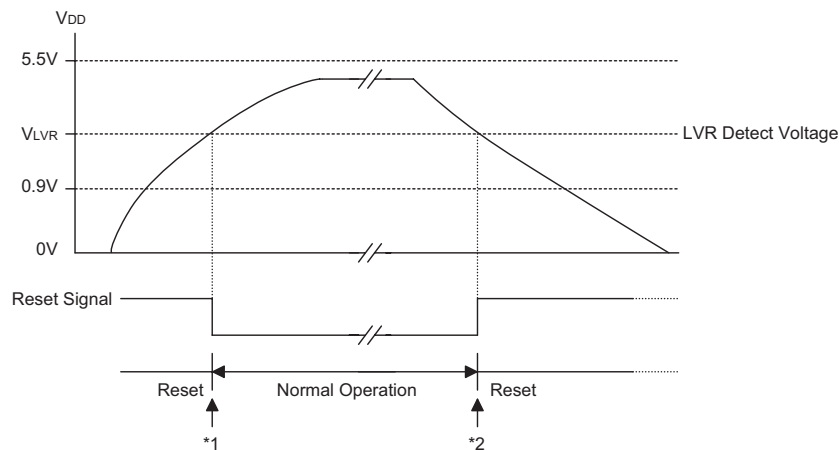
The LVR includes the following specifications:

- The low voltage (0.9V~V_{LVR}) state has to be maintained for more than 1ms, and the other circuits remain in their original state. 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 RES signal to perform a chip reset.

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



Note: V_{OPR} is the voltage range for proper chip operation at 2MHz system clock.



Low Voltage Reset

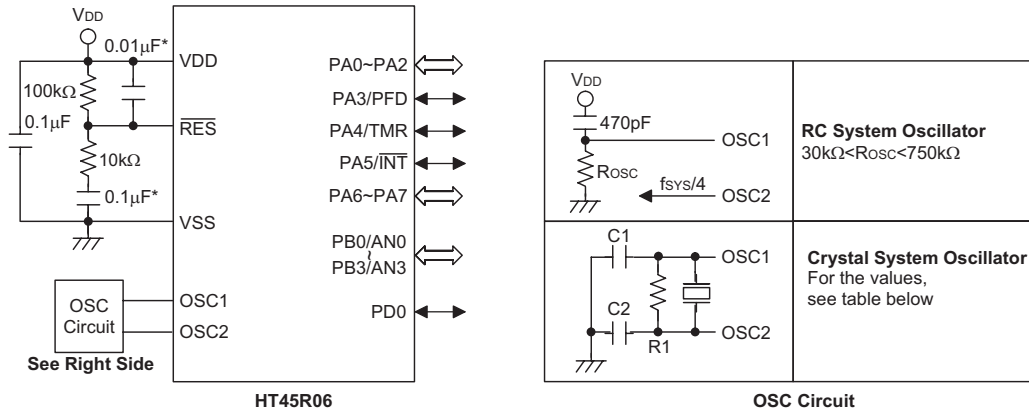
Note: *1: To make sure that the system oscillator has stabilized, the CSST provides an extra delay of 2.5ms before entering the normal operation.

*2: Since low voltage has to be maintained for more than 1ms, otherwise the system remain in their original state. Therefore a 1ms delay has to elapse before entering the reset mode.

Options

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

No.	Options
1	WDT clock source: WDTOSC or f _{sys} /4
2	WDT function: enable or disable
3	LVR function: enable or disable
4	CLRWDT instruction (s): One or two clear WDT instruction (s)
5	System oscillator: RC or crystal
6	Pull-high resistors (PA, PB and PD): none or pull-high
7	PFD function: enable or disable
8	PA0~PA7 wake-up: enable or disable

Application Circuits


The following table shows the C1, C2 and R1 values corresponding to the different crystal values. (For reference only)

Crystal or Resonator	C1, C2	R1
2MHz Crystal & Resonator	25pF	10kΩ
1MHz Crystal	35pF	27kΩ
480kHz Resonator	300pF	9.1kΩ
455kHz Resonator	300pF	10kΩ
429kHz Resonator	300pF	10kΩ

The function of the resistor R1 is to ensure that the oscillator will switch off should low voltage conditions occur. Such a low voltage, as mentioned here, is one which is less than the lowest value of the MCU operating voltage. Note however that if the LVR is enabled then R1 can be removed.

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

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

Instruction Set Summary

Mnemonic	Description	Instruction Cycle	Flag Affected
Arithmetic			
ADD A,[m]	Add data memory to ACC	1	Z,C,AC,OV
ADDM A,[m]	Add ACC to data memory	1 ⁽¹⁾	Z,C,AC,OV
ADD A,x	Add immediate data to ACC	1	Z,C,AC,OV
ADC A,[m]	Add data memory to ACC with carry	1	Z,C,AC,OV
ADCM A,[m]	Add ACC to data memory with carry	1 ⁽¹⁾	Z,C,AC,OV
SUB A,x	Subtract immediate data from ACC	1	Z,C,AC,OV
SUB A,[m]	Subtract data memory from ACC	1	Z,C,AC,OV
SUBM A,[m]	Subtract data memory from ACC with result in data memory	1 ⁽¹⁾	Z,C,AC,OV
SBC A,[m]	Subtract data memory from ACC with carry	1	Z,C,AC,OV
SBCM A,[m]	Subtract data memory from ACC with carry and result in data memory	1 ⁽¹⁾	Z,C,AC,OV
DAA [m]	Decimal adjust ACC for addition with result in data memory	1 ⁽¹⁾	C
Logic Operation			
AND A,[m]	AND data memory to ACC	1	Z
OR A,[m]	OR data memory to ACC	1	Z
XOR A,[m]	Exclusive-OR data memory to ACC	1	Z
ANDM A,[m]	AND ACC to data memory	1 ⁽¹⁾	Z
ORM A,[m]	OR ACC to data memory	1 ⁽¹⁾	Z
XORM A,[m]	Exclusive-OR ACC to data memory	1 ⁽¹⁾	Z
AND A,x	AND immediate data to ACC	1	Z
OR A,x	OR immediate data to ACC	1	Z
XOR A,x	Exclusive-OR immediate data to ACC	1	Z
CPL [m]	Complement data memory	1 ⁽¹⁾	Z
CPLA [m]	Complement data memory with result in ACC	1	Z
Increment & Decrement			
INCA [m]	Increment data memory with result in ACC	1	Z
INC [m]	Increment data memory	1 ⁽¹⁾	Z
DECA [m]	Decrement data memory with result in ACC	1	Z
DEC [m]	Decrement data memory	1 ⁽¹⁾	Z
Rotate			
RRA [m]	Rotate data memory right with result in ACC	1	None
RR [m]	Rotate data memory right	1 ⁽¹⁾	None
RRCA [m]	Rotate data memory right through carry with result in ACC	1	C
RRC [m]	Rotate data memory right through carry	1 ⁽¹⁾	C
RLA [m]	Rotate data memory left with result in ACC	1	None
RL [m]	Rotate data memory left	1 ⁽¹⁾	None
RLCA [m]	Rotate data memory left through carry with result in ACC	1	C
RLC [m]	Rotate data memory left through carry	1 ⁽¹⁾	C
Data Move			
MOV A,[m]	Move data memory to ACC	1	None
MOV [m],A	Move ACC to data memory	1 ⁽¹⁾	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of data memory	1 ⁽¹⁾	None
SET [m].i	Set bit of data memory	1 ⁽¹⁾	None

Mnemonic	Description	Instruction Cycle	Flag Affected
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if data memory is zero	1 ⁽²⁾	None
SZA [m]	Skip if data memory is zero with data movement to ACC	1 ⁽²⁾	None
SZ [m].i	Skip if bit i of data memory is zero	1 ⁽²⁾	None
SNZ [m].i	Skip if bit i of data memory is not zero	1 ⁽²⁾	None
SIZ [m]	Skip if increment data memory is zero	1 ⁽³⁾	None
SDZ [m]	Skip if decrement data memory is zero	1 ⁽³⁾	None
SIZA [m]	Skip if increment data memory is zero with result in ACC	1 ⁽²⁾	None
SDZA [m]	Skip if decrement data memory is zero with result in ACC	1 ⁽²⁾	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read			
TABRDC [m]	Read ROM code (current page) to data memory and TBLH	2 ⁽¹⁾	None
TABRDL [m]	Read ROM code (last page) to data memory and TBLH (This instruction is not valid for HT48R05A-1/HT48C05)	2 ⁽¹⁾	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear data memory	1 ⁽¹⁾	None
SET [m]	Set data memory	1 ⁽¹⁾	None
CLR WDT	Clear Watchdog Timer	1	TO,PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO ⁽⁴⁾ ,PDF ⁽⁴⁾
CLR WDT2	Pre-clear Watchdog Timer	1	TO ⁽⁴⁾ ,PDF ⁽⁴⁾
SWAP [m]	Swap nibbles of data memory	1 ⁽¹⁾	None
SWAPA [m]	Swap nibbles of data memory with result in ACC	1	None
HALT	Enter power down mode	1	TO,PDF

Note: x: Immediate data

m: Data memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

√: Flag is affected

–: Flag is not affected

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

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

⁽³⁾: ⁽¹⁾ and ⁽²⁾

⁽⁴⁾: The flags may be affected by the execution status. If the Watchdog Timer is cleared by executing the "CLR WDT1" or "CLR WDT2" instruction, the TO and PDF are cleared. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition
ADC A,[m]

Add data memory and carry to the accumulator

Description

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

Operation

 $ACC \leftarrow ACC+[m]+C$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADCM A,[m]

Add the accumulator and carry to data memory

Description

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

Operation

 $[m] \leftarrow ACC+[m]+C$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADD A,[m]

Add data memory to the accumulator

Description

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

Operation

 $ACC \leftarrow ACC+[m]$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADD A,x

Add immediate data to the accumulator

Description

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

Operation

 $ACC \leftarrow ACC+x$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADDM A,[m]

Add the accumulator to the data memory

Description

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

Operation

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

AND A,[m] Logical AND accumulator with data memory
 Description Data in the accumulator and the specified data memory perform a bitwise logical_AND operation. The result is stored in the accumulator.

Operation $ACC \leftarrow ACC \text{ "AND" } [m]$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

AND A,x Logical AND immediate data to the accumulator
 Description Data in the accumulator and the specified data perform a bitwise logical_AND operation. The result is stored in the accumulator.

Operation $ACC \leftarrow ACC \text{ "AND" } x$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

ANDM A,[m] Logical AND data memory with the accumulator
 Description Data in the specified data memory and the accumulator perform a bitwise logical_AND operation. The result is stored in the data memory.

Operation $[m] \leftarrow ACC \text{ "AND" } [m]$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

Operation $Stack \leftarrow Program\ Counter + 1$
 $Program\ Counter \leftarrow addr$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m] Clear data memory
 Description The contents of the specified data memory are cleared to 0.

Operation $[m] \leftarrow 00H$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m].i Clear bit of data memory
 Description The bit i of the specified data memory is cleared to 0.
 Operation $[m].i \leftarrow 0$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR WDT Clear Watchdog Timer
 Description The WDT is cleared (clears the WDT). The power down bit (PDF) and time-out bit (TO) are cleared.
 Operation $WDT \leftarrow 00H$
 $PDF \text{ and } TO \leftarrow 0$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
0	0	—	—	—	—

CLR WDT1 Preclear Watchdog Timer
 Description Together with CLR WDT2, clears the WDT. PDF and TO are also cleared. Only execution of this instruction without the other preclear instruction just sets the indicated flag which implies this instruction has been executed and the TO and PDF flags remain unchanged.
 Operation $WDT \leftarrow 00H^*$
 $PDF \text{ and } TO \leftarrow 0^*$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CLR WDT2 Preclear Watchdog Timer
 Description Together with CLR WDT1, clears the WDT. PDF and TO are also cleared. Only execution of this instruction without the other preclear instruction, sets the indicated flag which implies this instruction has been executed and the TO and PDF flags remain unchanged.
 Operation $WDT \leftarrow 00H^*$
 $PDF \text{ and } TO \leftarrow 0^*$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CPL [m] Complement data memory
 Description Each bit of the specified data memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice-versa.
 Operation $[m] \leftarrow \overline{[m]}$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

DEC [m] Decrement data memory
 Description Data in the specified data memory is decremented by 1.

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

DECA [m] Decrement data memory and place result in the accumulator
 Description Data in the specified data memory is decremented by 1, leaving the result in the accumulator. The contents of the data memory remain unchanged.

Operation $ACC \leftarrow [m] - 1$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

HALT	Enter power down mode												
Description	This instruction stops program execution and turns off the system clock. The contents of the RAM and registers are retained. The WDT and prescaler are cleared. The power down bit (PDF) is set and the WDT time-out bit (TO) is cleared.												
Operation	Program Counter \leftarrow Program Counter+1 PDF \leftarrow 1 TO \leftarrow 0												
Affected flag(s)	<table border="1"> <thead> <tr> <th>TO</th> <th>PDF</th> <th>OV</th> <th>Z</th> <th>AC</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>1</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> </tr> </tbody> </table>	TO	PDF	OV	Z	AC	C	0	1	—	—	—	—
TO	PDF	OV	Z	AC	C								
0	1	—	—	—	—								
INC [m]	Increment data memory												
Description	Data in the specified data memory is incremented by 1												
Operation	[m] \leftarrow [m]+1												
Affected flag(s)	<table border="1"> <thead> <tr> <th>TO</th> <th>PDF</th> <th>OV</th> <th>Z</th> <th>AC</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>—</td> <td>—</td> <td>—</td> <td>√</td> <td>—</td> <td>—</td> </tr> </tbody> </table>	TO	PDF	OV	Z	AC	C	—	—	—	√	—	—
TO	PDF	OV	Z	AC	C								
—	—	—	√	—	—								
INCA [m]	Increment data memory and place result in the accumulator												
Description	Data in the specified data memory is incremented by 1, leaving the result in the accumulator. The contents of the data memory remain unchanged.												
Operation	ACC \leftarrow [m]+1												
Affected flag(s)	<table border="1"> <thead> <tr> <th>TO</th> <th>PDF</th> <th>OV</th> <th>Z</th> <th>AC</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>—</td> <td>—</td> <td>—</td> <td>√</td> <td>—</td> <td>—</td> </tr> </tbody> </table>	TO	PDF	OV	Z	AC	C	—	—	—	√	—	—
TO	PDF	OV	Z	AC	C								
—	—	—	√	—	—								
JMP addr	Directly jump												
Description	The program counter are replaced with the directly-specified address unconditionally, and control is passed to this destination.												
Operation	Program Counter \leftarrow addr												
Affected flag(s)	<table border="1"> <thead> <tr> <th>TO</th> <th>PDF</th> <th>OV</th> <th>Z</th> <th>AC</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> </tr> </tbody> </table>	TO	PDF	OV	Z	AC	C	—	—	—	—	—	—
TO	PDF	OV	Z	AC	C								
—	—	—	—	—	—								
MOV A,[m]	Move data memory to the accumulator												
Description	The contents of the specified data memory are copied to the accumulator.												
Operation	ACC \leftarrow [m]												
Affected flag(s)	<table border="1"> <thead> <tr> <th>TO</th> <th>PDF</th> <th>OV</th> <th>Z</th> <th>AC</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> </tr> </tbody> </table>	TO	PDF	OV	Z	AC	C	—	—	—	—	—	—
TO	PDF	OV	Z	AC	C								
—	—	—	—	—	—								

MOV A,x Move immediate data to the accumulator
 Description The 8-bit data specified by the code is loaded into the accumulator.
 Operation $ACC \leftarrow x$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

MOV [m],A Move the accumulator to data memory
 Description The contents of the accumulator are copied to the specified data memory (one of the data memories).
 Operation $[m] \leftarrow ACC$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

NOP No operation
 Description No operation is performed. Execution continues with the next instruction.
 Operation Program Counter \leftarrow Program Counter+1
 Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

OR A,[m] Logical OR accumulator with data memory
 Description Data in the accumulator and the specified data memory (one of the data memories) perform a bitwise logical_OR operation. The result is stored in the accumulator.
 Operation $ACC \leftarrow ACC \text{ "OR" } [m]$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

OR A,x Logical OR immediate data to the accumulator
 Description Data in the accumulator and the specified data perform a bitwise logical_OR operation. The result is stored in the accumulator.
 Operation $ACC \leftarrow ACC \text{ "OR" } x$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

ORM A,[m] Logical OR data memory with the accumulator
 Description Data in the data memory (one of the data memories) and the accumulator perform a bitwise logical_OR operation. The result is stored in the data memory.
 Operation $[m] \leftarrow ACC \text{ "OR" } [m]$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

RET

Return from subroutine

Description

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

Operation

 Program Counter \leftarrow Stack

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RET A,x

Return and place immediate data in the accumulator

Description

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

Operation

 Program Counter \leftarrow Stack

 ACC \leftarrow x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RETI

Return from interrupt

Description

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

Operation

 Program Counter \leftarrow Stack

 EMI \leftarrow 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RL [m]

Rotate data memory left

Description

The contents of the specified data memory are rotated 1 bit left with bit 7 rotated into bit 0.

Operation

 $[m].(i+1) \leftarrow [m].i$; $[m].i$:bit i of the data memory ($i=0\sim 6$)

 $[m].0 \leftarrow [m].7$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RLA [m]

Rotate data memory left and place result in the accumulator

Description

Data in the specified data memory is rotated 1 bit left with bit 7 rotated into bit 0, leaving the rotated result in the accumulator. The contents of the data memory remain unchanged.

Operation

 ACC. $(i+1) \leftarrow [m].i$; $[m].i$:bit i of the data memory ($i=0\sim 6$)

 ACC.0 $\leftarrow [m].7$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RLC [m] Rotate data memory left through carry
 Description The contents of the specified data memory and the carry flag are rotated 1 bit left. Bit 7 replaces the carry bit; the original carry flag is rotated into the bit 0 position.

Operation $[m].(i+1) \leftarrow [m].i$; $[m].i$:bit i of the data memory (i=0~6)
 $[m].0 \leftarrow C$
 $C \leftarrow [m].7$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

Description Data in the specified data memory and the carry flag are rotated 1 bit left. Bit 7 replaces the carry bit and the original carry flag is rotated into bit 0 position. The rotated result is stored in the accumulator but the contents of the data memory remain unchanged.

Operation $ACC.(i+1) \leftarrow [m].i$; $[m].i$:bit i of the data memory (i=0~6)
 $ACC.0 \leftarrow C$
 $C \leftarrow [m].7$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

RR [m] Rotate data memory right

Description The contents of the specified data memory are rotated 1 bit right with bit 0 rotated to bit 7.

Operation $[m].i \leftarrow [m].(i+1)$; $[m].i$:bit i of the data memory (i=0~6)
 $[m].7 \leftarrow [m].0$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

Description Data in the specified data memory is rotated 1 bit right with bit 0 rotated into bit 7, leaving the rotated result in the accumulator. The contents of the data memory remain unchanged.

Operation $ACC.(i) \leftarrow [m].(i+1)$; $[m].i$:bit i of the data memory (i=0~6)
 $ACC.7 \leftarrow [m].0$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RRC [m] Rotate data memory right through carry

Description The contents of the specified data memory and the carry flag are together rotated 1 bit right. Bit 0 replaces the carry bit; the original carry flag is rotated into the bit 7 position.

Operation $[m].i \leftarrow [m].(i+1)$; $[m].i$:bit i of the data memory (i=0~6)
 $[m].7 \leftarrow C$
 $C \leftarrow [m].0$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

RRCA [m]	Rotate right through carry and place result in the accumulator												
Description	Data of the specified data memory and the carry flag are rotated 1 bit right. Bit 0 replaces the carry bit and the original carry flag is rotated into the bit 7 position. The rotated result is stored in the accumulator. The contents of the data memory remain unchanged.												
Operation	$ACC.i \leftarrow [m].(i+1)$; $[m].i$:bit i of the data memory (i=0~6) $ACC.7 \leftarrow C$ $C \leftarrow [m].0$												
Affected flag(s)	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>TO</th> <th>PDF</th> <th>OV</th> <th>Z</th> <th>AC</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>√</td> </tr> </tbody> </table>	TO	PDF	OV	Z	AC	C	—	—	—	—	—	√
TO	PDF	OV	Z	AC	C								
—	—	—	—	—	√								
SBC A,[m]	Subtract data memory and carry from the accumulator												
Description	The contents of the specified data memory and the complement of the carry flag are subtracted from the accumulator, leaving the result in the accumulator.												
Operation	$ACC \leftarrow ACC + \overline{[m]} + C$												
Affected flag(s)	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>TO</th> <th>PDF</th> <th>OV</th> <th>Z</th> <th>AC</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>—</td> <td>—</td> <td>√</td> <td>√</td> <td>√</td> <td>√</td> </tr> </tbody> </table>	TO	PDF	OV	Z	AC	C	—	—	√	√	√	√
TO	PDF	OV	Z	AC	C								
—	—	√	√	√	√								
SBCM A,[m]	Subtract data memory and carry from the accumulator												
Description	The contents of the specified data memory and the complement of the carry flag are subtracted from the accumulator, leaving the result in the data memory.												
Operation	$[m] \leftarrow ACC + \overline{[m]} + C$												
Affected flag(s)	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>TO</th> <th>PDF</th> <th>OV</th> <th>Z</th> <th>AC</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>—</td> <td>—</td> <td>√</td> <td>√</td> <td>√</td> <td>√</td> </tr> </tbody> </table>	TO	PDF	OV	Z	AC	C	—	—	√	√	√	√
TO	PDF	OV	Z	AC	C								
—	—	√	√	√	√								
SDZ [m]	Skip if decrement data memory is 0												
Description	The contents of the specified data memory are decremented by 1. If the result is 0, the next instruction is skipped. If the result is 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).												
Operation	Skip if $([m]-1)=0$, $[m] \leftarrow ([m]-1)$												
Affected flag(s)	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>TO</th> <th>PDF</th> <th>OV</th> <th>Z</th> <th>AC</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> </tr> </tbody> </table>	TO	PDF	OV	Z	AC	C	—	—	—	—	—	—
TO	PDF	OV	Z	AC	C								
—	—	—	—	—	—								
SDZA [m]	Decrement data memory and place result in ACC, skip if 0												
Description	The contents of the specified data memory are decremented by 1. If the result is 0, the next instruction is skipped. The result is stored in the accumulator but the data memory remains unchanged. If the result is 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).												
Operation	Skip if $([m]-1)=0$, $ACC \leftarrow ([m]-1)$												
Affected flag(s)	<table border="1" style="width: 100%; text-align: center;"> <thead> <tr> <th>TO</th> <th>PDF</th> <th>OV</th> <th>Z</th> <th>AC</th> <th>C</th> </tr> </thead> <tbody> <tr> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> <td>—</td> </tr> </tbody> </table>	TO	PDF	OV	Z	AC	C	—	—	—	—	—	—
TO	PDF	OV	Z	AC	C								
—	—	—	—	—	—								

SET [m] Set data memory
 Description Each bit of the specified data memory is set to 1.
 Operation $[m] \leftarrow FFH$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m]. i Set bit of data memory
 Description Bit i of the specified data memory is set to 1.
 Operation $[m].i \leftarrow 1$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SIZ [m] Skip if increment data memory is 0
 Description The contents of the specified data memory are incremented by 1. If the result is 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).
 Operation Skip if $([m]+1)=0$, $[m] \leftarrow ([m]+1)$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SIZA [m] Increment data memory and place result in ACC, skip if 0
 Description The contents of the specified data memory are incremented by 1. If the result is 0, the next instruction is skipped and the result is stored in the accumulator. The data memory remains unchanged. If the result is 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).
 Operation Skip if $([m]+1)=0$, $ACC \leftarrow ([m]+1)$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SNZ [m].i Skip if bit i of the data memory is not 0
 Description If bit i of the specified data memory is not 0, the next instruction is skipped. If bit i of the data memory is not 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).
 Operation Skip if $[m].i \neq 0$
 Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SUB A,[m] Subtract data memory from the accumulator
 Description The specified data memory is subtracted from the contents of the accumulator, leaving the result in the accumulator.

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SUBM A,[m] Subtract data memory from the accumulator
 Description The specified data memory is subtracted from the contents of the accumulator, leaving the result in the data memory.

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

Operation $ACC \leftarrow ACC + \overline{x} + 1$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SWAP [m] Swap nibbles within the data memory
 Description The low-order and high-order nibbles of the specified data memory (1 of the data memories) are interchanged.

Operation $[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SWAPA [m] Swap data memory and place result in the accumulator
 Description The low-order and high-order nibbles of the specified data memory are interchanged, writing the result to the accumulator. The contents of the data memory remain unchanged.

Operation $ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$

$ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SZ [m] Skip if data memory is 0

Description If the contents of the specified data memory are 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if [m]=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

Description The contents of the specified data memory are copied to the accumulator. If the contents is 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if [m]=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

Description If bit i of the specified data memory is 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation Skip if [m].i=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

Description The low byte of ROM code (last page) addressed by the table pointer (TBLP) is moved to the data memory and the high byte transferred to TBLH directly.
Note that this instruction is not valid for HT48R05A-1/HT48C05

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

XOR A,[m]

Logical XOR accumulator with data memory

Description

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

Operation

 $ACC \leftarrow ACC \text{ "XOR" } [m]$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

XORM A,[m]

Logical XOR data memory with the accumulator

Description

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

Operation

 $[m] \leftarrow ACC \text{ "XOR" } [m]$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

XOR A,x

Logical XOR immediate data to the accumulator

Description

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

Operation

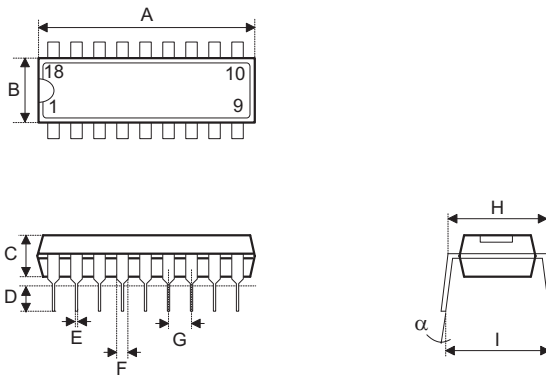
 $ACC \leftarrow ACC \text{ "XOR" } x$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

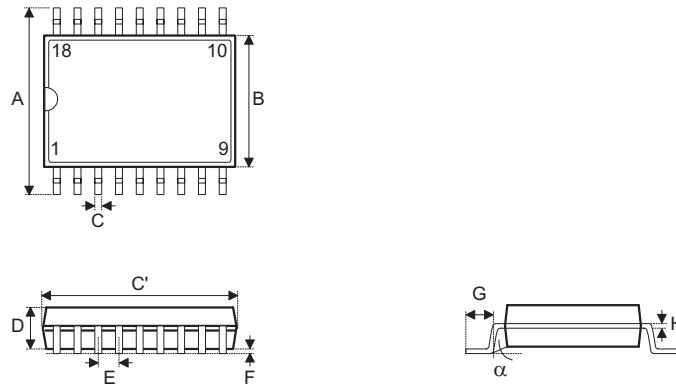
Package Information

18-pin DIP (300mil) Outline Dimensions



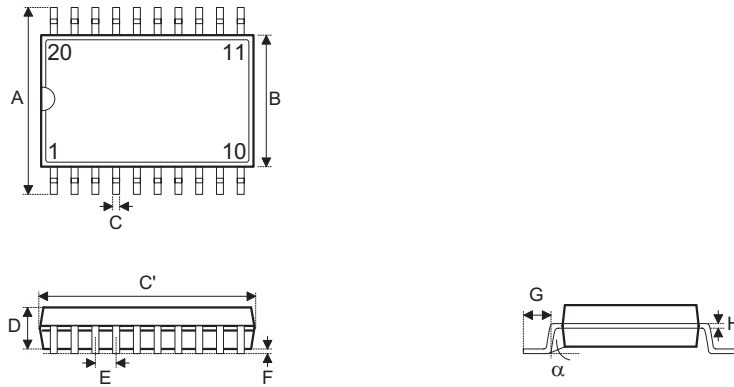
Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	895	—	915
B	240	—	260
C	125	—	135
D	125	—	145
E	16	—	20
F	50	—	70
G	—	100	—
H	295	—	315
I	335	—	375
α	0°	—	15°

18-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	394	—	419
B	290	—	300
C	14	—	20
C'	447	—	460
D	92	—	104
E	—	50	—
F	4	—	—
G	32	—	38
H	4	—	12
α	0°	—	10°

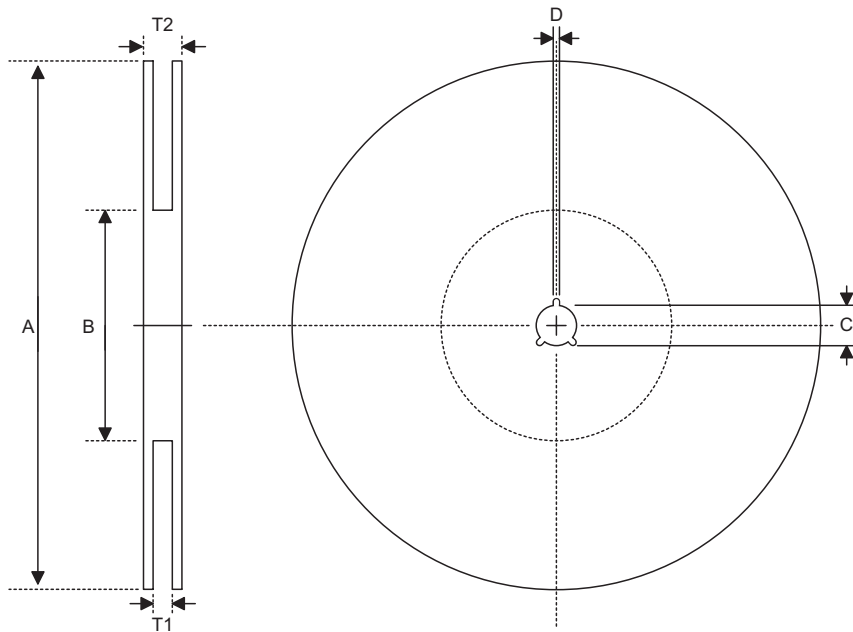
20-pin SSOP (150mil) Outline Dimensions



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 Reel and Tape Specifications

Reel Dimensions

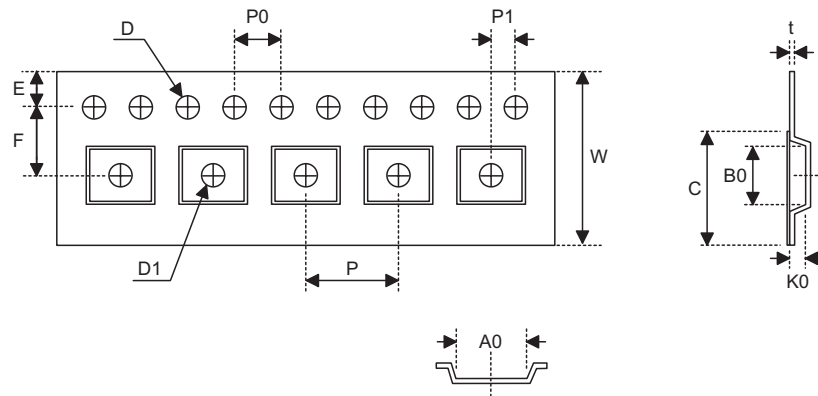


SOP 18W

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	24.8+0.3 -0.2
T2	Reel Thickness	30.2±0.2

SSOP 20S (150mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1
B	Reel Inner Diameter	62±1.5
C	Spindle Hole Diameter	13+0.5 -0.2
D	Key Slit Width	2±0.5
T1	Space Between Flange	16.8+0.3 -0.2
T2	Reel Thickness	22.2±0.2

Carrier Tape Dimensions

SOP 18W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0+0.3 -0.1
P	Cavity Pitch	16.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5±0.1
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.9±0.1
B0	Cavity Width	12.0±0.1
K0	Cavity Depth	2.8±0.1
t	Carrier Tape Thickness	0.3±0.05
C	Cover Tape Width	21.3

SSOP 20S (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16+0.3 -0.1
P	Cavity Pitch	8±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±0.1
P1	Cavity to Perforation (Length Direction)	2±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	9±0.1
K0	Cavity Depth	2.3±0.1
t	Carrier Tape Thickness	0.3±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 Inc. (Shanghai Sales Office)

7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China 200233
Tel: 021-6485-5560
Fax: 021-6485-0313
<http://www.holtek.com.cn>

Holtek Semiconductor Inc. (Shenzhen Sales Office)

43F, SEG Plaza, Shen Nan Zhong Road, Shenzhen, China 518031
Tel: 0755-8346-5589
Fax: 0755-8346-5590
ISDN: 0755-8346-5591

Holtek Semiconductor Inc. (Beijing Sales Office)

Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031
Tel: 010-6641-0030, 6641-7751, 6641-7752
Fax: 010-6641-0125

Holmate Semiconductor, Inc. (North America Sales Office)

46712 Fremont Blvd., Fremont, CA 94538
Tel: 510-252-9880
Fax: 510-252-9885
<http://www.holmate.com>

Copyright © 2005 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>.