Multimedia Keyboard Encoder 8-Bit MCU

Technical Document

- Tools Information
- FAQs

Features

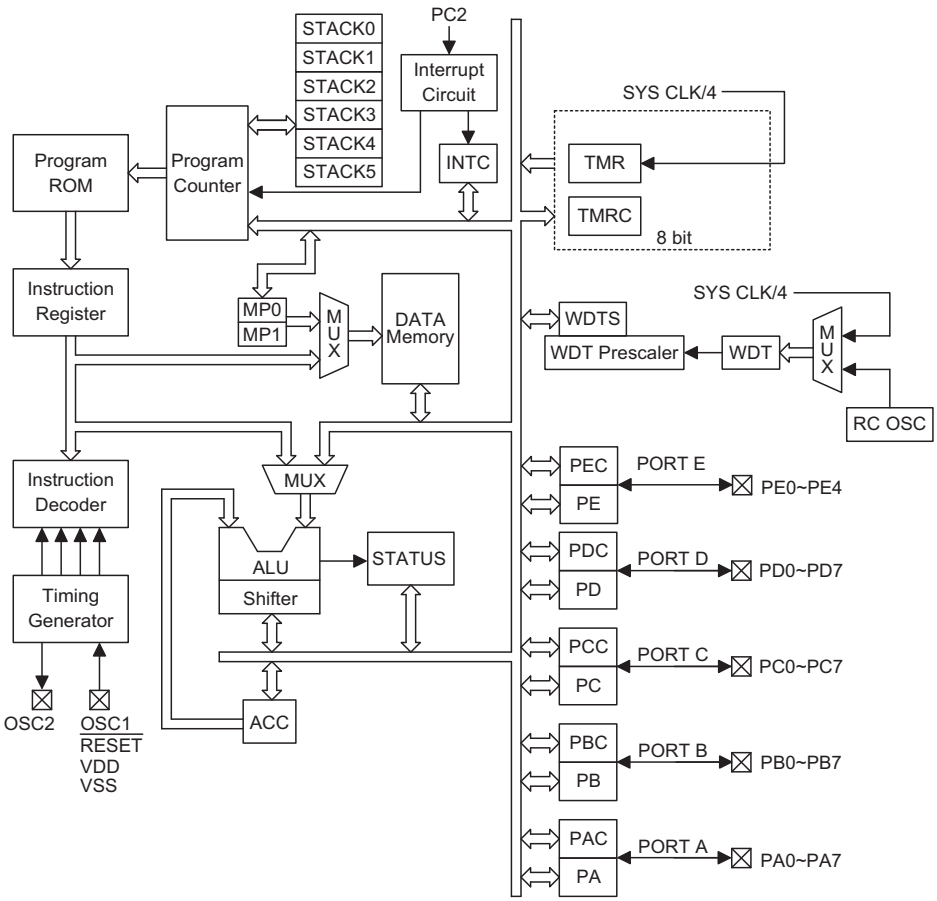
- Operating voltage: 1.8V~5.5V
- 34 bidirectional I/O line and 3 CMOS output
- One 8-bit programmable timer counter with overflow interrupts
- Crystal or RC oscillator
- Watchdog Timer
- 3K×16 program EPROM
- 160×8 data RAM
- One external interrupt pin (shared with PC2)
- 2.0V LVR by option (default disable)
- HALT function and wake-up feature reduce power consumption
- Six-level subroutine nesting
- Bit manipulation instructions
- 16-bit table read instructions
- 63 powerful instructions
- All instructions in 1 or 2 machine cycles
- 20/28-pin SOP, 32-pin QFN and 48-pin SSOP packages

General Description

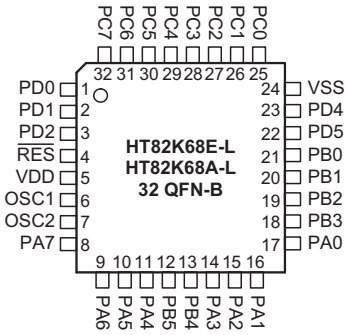
This device is an 8-bit high performance peripheral interface IC, designed for multiple I/O products and multimedia applications. It supports interface to a low speed PC with multimedia keyboard or wireless keyboard in Windows 95, Windows 98 or Windows 2000 environment. A HALT feature is included to reduce power consumption.

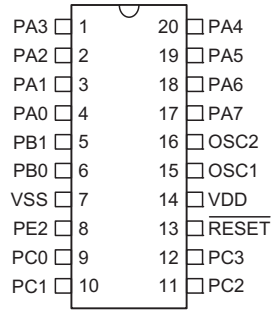
The mask version HT82K68A-L is fully pin and functionally compatible with the OTP version HT82K68E-L device.

Block Diagram

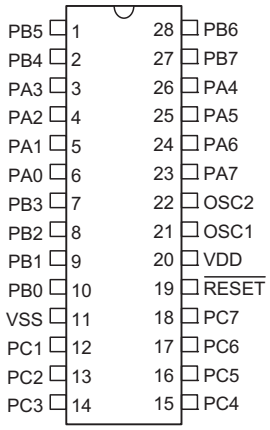


Pin Assignment

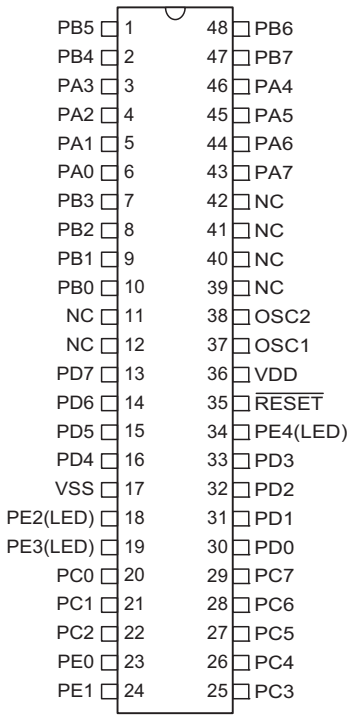




HT82K68E-L/HT82K68A-L
—20 SOP-A



HT82K68E-L/HT82K68A-L
—28 SOP-A



HT82K68E-L/HT82K68A-L
— 48 SSOP-A

Pin Description

Pin Name	I/O	Mask Option	Description
PA0~PA7	I/O	Wake-up Pull-high or None	Bidirectional 8-bit input/output port. Each bit can be configured as a wake-up input by mask option. Software* instructions determine the CMOS output or Schmitt Trigger input with or without 12K pull-high resistor.
PB0~PB7	I/O	Pull-high or None	Bidirectional 8-bit input/output port. Software* instructions determine the output or Schmitt Trigger input with or without pull-high resistor.
PC0	I/O	Wake-up Pull-high or None	This pin is an I/O port. NMOS open drain output with pull-high resistor and can be used as DATA or CLOCK line of PS2. This pin can be configured as a wake-up input by mask option.
PC1	I/O	Wake-up Pull-high or None	This pin is an I/O port. NMOS open drain output with pull-high resistor and can be used as DATA or CLOCK line of PS2. This pin can be configured as a wake-up input by mask option.
PC2~PC3	I/O	Wake-up Pull-high or None	Bidirectional 2-bit input/output port. Each bit can be configured as a wake-up input by mask option. Software* instructions determine the CMOS output or Schmitt Trigger input with or without pull-high resistor. PC2 also as external interrupt input pin. PE0 determine whether rising edge or falling edge of PC2 to trigger the INT circuit.
PC4~PC7	I/O	Pull-high or None	Bidirectional 4-bit input/output port. Software* instructions determine the CMOS output or Schmitt Trigger input with or without pull-high resistor.
PD0~PD7	I/O	Pull-high or None	Bidirectional 8-bit input/output port. Software* instructions determine the CMOS output or Schmitt Trigger input with or without pull-high resistor.

Pin Name	I/O	Mask Option	Description
PE0~PE1	I/O	Pull-high or None	Bidirectional input/output port. Software* instruction determine the CMOS output or Schmitt Trigger input with or without pull-high resistor. If PE0 output 1, rising edge of PC2 trigger INT circuit. PE0 output 0, falling edge of PC2 trigger INT circuit.
PE2	O		This pin is a CMOS output structure. The pad can function as LED (SCR) drivers for the keyboard. I_{OL}=18mA at V_{OL}=3.4V
PE3	O		This pin is a CMOS output structure. The pad can function as LED (NUM) drivers for the keyboard. I_{OL}=18mA at V_{OL}=3.4V
PE4	O		This pin is a CMOS output structure. The pad can function as LED (CAP) drivers for the keyboard. I_{OL}=18mA at V_{OL}=3.4V
VDD	—	—	Positive power supply
VSS	—	—	Negative power supply, ground
$\overline{\text{RESET}}$	I	—	Chip reset input. Active low. Built-in power-on reset circuit to reset the entire chip. Chip can also be externally reset via $\overline{\text{RESET}}$ pin
OSC1 OSC2	I O	Crystal or RC	OSC1, OSC2 are connected to an RC network or a crystal for the internal system clock. In the case of RC operation, OSC2 is the output terminal for the 1/4 system clock; A 110kΩ resistor is connected to OSC1 to generate a 2 MHZ frequency.

Note: *: Software means the HT−IDE (Holtek Integrated Development Environment) can be configured by mask option.

Absolute Maximum Ratings

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

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

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

Operating Temperature .....−25°C to 70°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	—	—	1.8	—	5.5	V
I_{DD1}	Operating Current (Crystal OSC)	3V	No load, f_{SYS}= 6MHz	—	0.7	1.5	mA
		5V		—	2	5	mA
I_{DD2}	Operating Current (RC OSC)	3V	No load, f_{SYS}= 6MHz	—	0.5	1.5	mA
		5V		—	2	5	mA
I_{STB1}	Standby Current (WDT enabled)	3V	No load, system HALT	—	—	8	μA
		5V		—	—	15	μA
I_{STB2}	Standby Current (WDT Disabled)	3V	No load, system HALT	—	—	3	μA
		5V		—	—	6	μA
V_{IL1}	Input Low Voltage for I/O Ports (Schmitt)	3V	—	0	—	$0.3V_{DD}$	V
		5V	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports (Schmitt)	3V	—	$0.7V_{DD}$	—	V_{DD}	V
		5V	—	$0.7V_{DD}$	—	V_{DD}	V

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{IL2}	Input Low Voltage ($\overline{RESET}$)	3V	—	0	—	0.7	V
		5V	—	0	—	1.3	V
V_{IH2}	Input High Voltage ($\overline{RESET}$)	3V	—	$0.9V_{DD}$	—	V_{DD}	V
		5V	—	$0.9V_{DD}$	—	V_{DD}	V
V_{LVR}	Low Voltage Reset	—	—	—	2.0	—	V
I_{OL}	I/O Port Sink Current of PA, PB, PC, PD, PE0~1	5V	V_{OL}= $0.1V_{DD}$	2	4	—	mA
I_{OH}	I/O Port Source Current of PA, PB, PC, PD, PE0~4	5V	V_{OH}= $0.9V_{DD}$	–2.5	–4	—	mA
I_{LED}	LED Sink Current (SCR, NUM, CAP)	5V	V_{OL}=3.4V	10	17	25	mA
t_{POR}	Power-on Reset Time	5V	R=100kΩ, C=0.1μF	50	100	150	ms
R_{PH}	Internal Pull-high Resistance of PA, PB, PC, PD, PE Port	3V	—	30	60	90	kΩ
		5V	—	15	30	45	kΩ
R_{PH1}	Internal Pull-high Resistance of DATA, CLK	3V	—	4	9	15	kΩ
		5V	—	2	4.7	8	kΩ
Δf/f	Frequency Variation	5V	Crystal	—	—	±1	%
Δf/f1	Frequency Variation	5V	RC	—	—	±20	%

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS1}	System Clock (Crystal OSC)	1.8V	—	450	—	4000	kHz
		5V	—	450	—	8000	kHz
f_{SYS2}	System Clock (RC OSC)	3V	—	450	—	6000	kHz
		5V	—	450	—	8000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	35	78	130	μs
t_{WDT1}	Watchdog Time-out Period (RC)	3V	Without WDT prescaler	12	23	45	ms
		5V		9	19	35	ms
t_{WDT2}	Watchdog Time-out Period (System Clock)	—	Without WDT prescaler	—	1024	—	t_{SYS}
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Power-up or wake-up from HALT	—	1024	—	t_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs

Note: t_{SYS}= $1/f_{SYS1}$ or $1/f_{SYS2}$

Functional Description

Execution Flow

The device system clock 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 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 12-bit program counter (PC) controls the sequence in which the instructions stored in the program ROM are executed and its contents specify a maximum of 4096 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, internal interrupt, external interrupt or return from subroutine, the PC manipulates the program transfer by loading the address corresponding to each instruction.

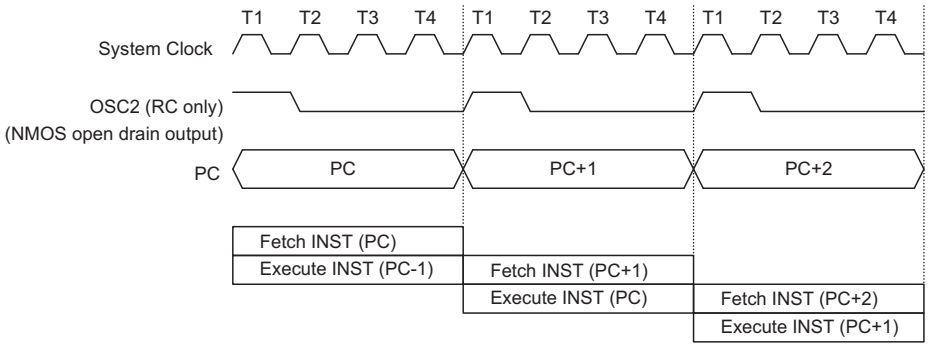
The conditional skip is activated by instruction. Once the condition is met, the next instruction, fetched during the current instruction execution, is discarded and a dummy cycle replaces it to get the proper instruction. Otherwise proceed with the next instruction.

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

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

Program Memory – ROM

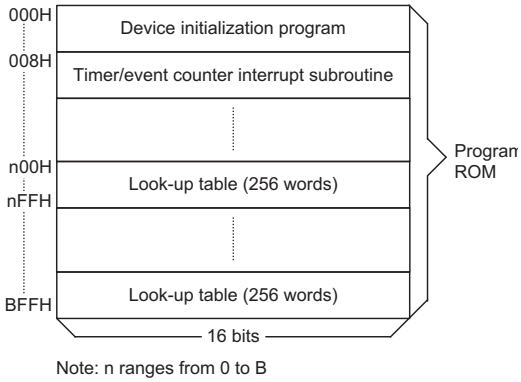
The program memory is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized with 3072×16 bits, addressed by the program counter and table pointer.



Execution Flow

Mode	Program Counter											
	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial reset	0	0	0	0	0	0	0	0	0	0	0	0
External interrupt	0	0	0	0	0	0	0	0	0	1	0	0
Timer counter overflow	0	0	0	0	0	0	0	0	1	0	0	0
Skip	Program Counter+2											
Loading PCL	*11	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, call branch	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from subroutine	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Note: *11~*0: Program counter bits
#11~#0: Instruction code bits
S11~S0: Stack register bits
@7~@0: PCL bits



Note: n ranges from 0 to B

Program Memory

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

- Location 000
  This area is reserved for the initialization program. After chip reset, the program always begins execution at location 000H.
- Location 004H
  Location 004H is reserved for external interrupt service program. If the PC2 (external input pin) is activated, the interrupt is enabled, and the stack is not full, the program begins execution at location 004H. The pin PE0 determine whether the rising or falling edge of the PC2 to activate external interrupt service program.
- Location 008H
  This area is reserved for the timer counter interrupt service program. If timer interrupt results from a timer counter overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 008H.
- Table location
  Any location in the ROM 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 1 bit is read as 0. The Table Higher-order byte register (TBLH) is read only. 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. 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. 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 which is used to save the contents of the program counter (PC) only. The stack is organized into six levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the stack pointer (SP) and is neither readable nor writeable. At a subroutine call or interrupt acknowledgement, 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.

Data Memory – RAM

The data memory is designed with 184 × 8 bits. It is divided into two functional groups: special function registers and general purpose data memory (160×8). Most of them are read/write, but some are read only.

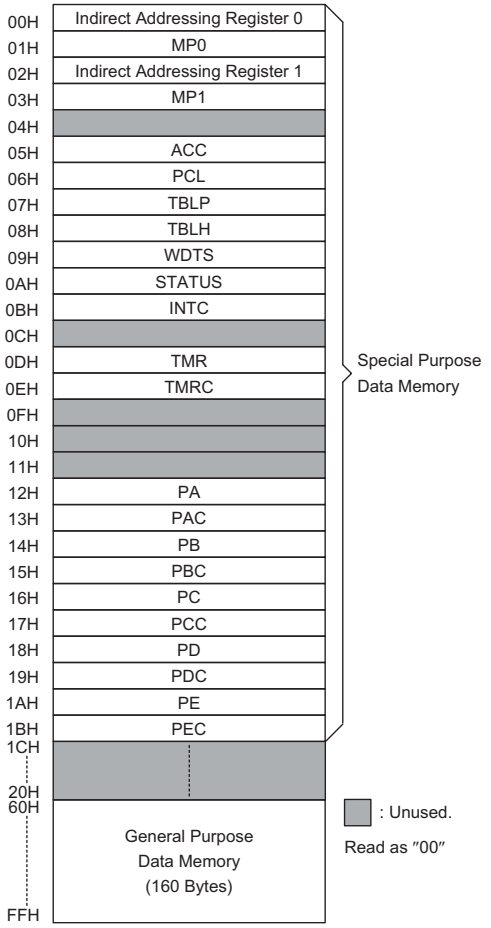
The unused space before 60H is reserved for future expanded usage and reading these locations will get the result 00H. The general purpose data memory, addressed from 60H to FFH, is used for data and control information under instruction command. All data memory areas can handle arithmetic, logic, increment, decrement and rotate operations directly. Except for some dedicated bits, each bit in the data memory can be set

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

Note: *11~*0: Table location bits P11~P8: Current program counter bits

@7~@0: Table location bits

and reset by the SET [m].i and CLR [m].i instructions, respectively. They are also indirectly accessible through Memory pointer registers (MP0;01H, MP1;03H).



RAM Mapping

Indirect Addressing Register

Location 00H and 02H are indirect addressing registers that are not physically implemented. Any read/write operation of [00H] and [02H] can access the data memory pointed to by MP0 (01H) and MP1 (03H) respectively. Reading location 00H or 02H indirectly will return the result 00H. Writing indirectly results in no operation.

The function of data movement between two indirect addressing registers is not supported. The memory pointer registers, MP0 and MP1, are 8-bit registers which can be used to access the data memory by combining corresponding indirect addressing registers.

Accumulator

The accumulator is closely related to the ALU operations. It is also mapped to location 05H of the data memory and is capable of carrying 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 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 status register.

Status Register – Status

The 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 watch dog time-out flag (TO). The status register not only records the status information but also 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 flags. It should be noted that operations related to the status register may give different results from those intended. The TO and PDF flags can only be changed by system power up, Watchdog Timer overflow, executing the HALT instruction and clearing the Watchdog Timer.

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

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

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 if 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 logical 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 when either a system power-up or executing the CLR WDT instruction. PDF is set by executing a 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

Interrupt

The device provides an internal timer counter interrupt and an external interrupt shared with PC2. The interrupt control register (INTC;0BH) contains the interrupt control bits to set not only the enable/disable status but also the interrupt request flags.

Once an interrupt subroutine is serviced, all 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 interrupt have the 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 the specified location in the program memory. Only the program counter is pushed onto the stack. If the contents of the register and Status register (STATUS) are altered by the interrupt service program which corrupt the desired control sequence, the contents should be saved in advance.

The internal timer counter interrupt is initialized by setting the timer counter interrupt request flag (T0F; bit 5 of INTC), which is normally caused by a timer counter overflow. When the interrupt is enabled, and the stack is not full and the T0F bit is set, a subroutine call to location 08H will occur. The related interrupt request flag (T0F) will be reset and the EMI bit cleared to disable further interrupts.

The external interrupt is shared with PC2. The external interrupt is activated, the related interrupt request flag (EIF; bit4 of INTC) is then 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 also be cleared to disable other interrupts.

The external interrupt (PC2) can be triggered by a high to low transition, or a low to high transition of the PC2, which is dependent on the output level of the PE0. When PE0 is output high, the external interrupt is triggered by a low to high transition of the PC2. When PE0 is output low, the external interrupt is triggered by a high to low transition of PC2.

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1= enabled; 0= disabled)
1	EEI	Control the external interrupt
2	ET0I	Controls the timer counter interrupt (1= enabled; 0= disabled)
3	—	Unused bit, read as ″0″
4	EIF	External interrupt flag
5	T0F	Internal timer counter request flag (1= active; 0= inactive)
6, 7	—	Unused bit, read as ″0″

INTC (0BH) Register

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

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

Interrupt Source	Vector
External interrupt 1	04H
Timer counter overflow	08H

Once the interrupt request flags (T0F) are set, they will remain in the INTC register until the interrupts are serviced or cleared by a software instruction.

It is suggested that a program does not use the "CALL subroutine" within the interrupt subroutine. Because 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, once the "CALL subroutine" operates in the interrupt subroutine it will damage the original control sequence.

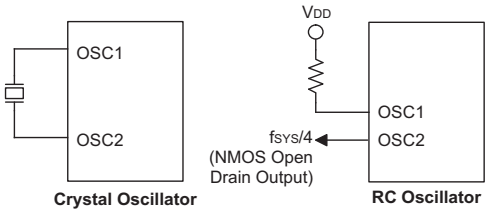
Oscillator Configuration

There are two oscillator circuits in the microcontroller. Both are designed for system clocks; the RC oscillator and the Crystal oscillator, which are determined by mask options. No matter what oscillator type is selected, the signal provides the system clock. The HALT mode stops the system oscillator and resists the external signal to conserve power.

If an RC oscillator is used, an external resistor between OSC1 and VDD is needed and the resistance must range from 20kΩ to 510kΩ. 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 VDD, 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 needed for 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.

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 for a period of approximately 78μs. The WDT oscillator can be disabled by mask option to conserve power.

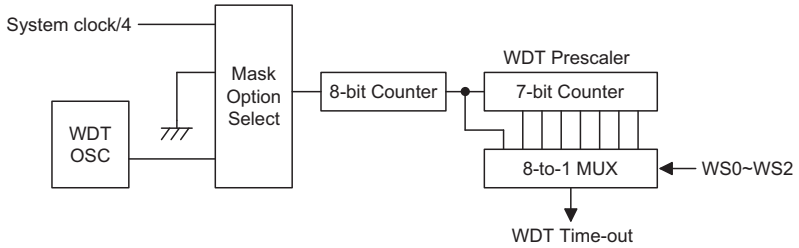


System Oscillator

Watchdog Timer – WDT

The WDT clock source is implemented by a dedicated RC oscillator (WDT oscillator) or instruction clock (system clock divided by 4), decided by mask 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 mask option. If the Watchdog Timer is disabled, all the executions related to the WDT results in no operation.

Once the internal WDT oscillator (RC oscillator normally with a period of 78μs) is selected, it is first divided by 256 (8-stages) to get the nominal time-out period of approximately 20ms. This time-out period may vary with temperature, VDD and process variations. By invoking the WDT prescaler, longer time-out periods can be realized. Writing data to WS2, WS1, WS0 (bit 2,1,0 of the WDTS) can give different time-out periods. If WS2, WS1, WS0 are all equal to 1, the division ratio is up to 1:128, and the maximum time-out period is 2.6 seconds.



Watchdog Timer

If the WDT oscillator is disabled, the WDT clock may still come from the instruction clock and operate in the same manner except that in the HALT state the WDT may stop counting and lose its protecting purpose. In this situation the WDT logic can be restarted by external logic. The high nibble and bit 3 of the WDTS are reserved for user defined flags, which can be used to indicate some specified status.

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.

WS2	WS1	WS0	Division Ratio
0	0	0	1:1
0	0	1	1:2
0	1	0	1:4
0	1	1	1:8
1	0	0	1:16
1	0	1	1:32
1	1	0	1:64
1	1	1	1:128

WDTS (09H) Register

The WDT overflow under normal operation will initialize "chip reset" and set the status bit TO. An overflow in the HALT mode, initializes a "warm reset" only when the program counter and stack pointer are reset to zero. To clear the contents of the WDT (including the WDT prescaler), three methods are adopted; external reset (a low level to $\overline{\text{RESET}}$), software instruction(s), or a HALT instruction. There are two types of software instructions; CLR WDT and CLR WDT1/CLR WDT2. Of these two types of instruction, only one can be active depending on the mask option – "CLR WDT times selection option". If the "CLR WDT" is selected (ie. 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 (ie. CLRWDT times equal two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip because of the time-out.

Power Down Operation – HALT

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

- The system oscillator will turn 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 recount again (if the WDT clock has come from the WDT oscillator).
- All 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, interrupt, and external falling edge signal on port A and port C [0:3] or a WDT overflow. An external reset causes a device initialization and the WDT overflow performs a "warm reset". Examining the TO and PDF flags, the reason for chip reset can be determined. The PDF flag is cleared when system power-up or executing 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 stack pointer, the others keep their original status.

On the other hand, awakening from an external interrupt (PC2), two sequences may happen. If the interrupt is disabled or the interrupt is enabled but the stack is full, the program will resume execution at the next instruction. But if the interrupt is enabled and the stack is not full, the regular interrupt response takes place.

The port A or port C [0:3] wake-up can be considered as a continuation of normal execution. Each bit in port A can be independently selected to wake up the device by mask option. Awakening from an I/O port stimulus, the program will resume execution of the next instruction.

Once a wake-up event occurs, and the system clock comes from a crystal, it takes 1024 t_{SYS} (system clock period) to resume normal operation. In other words, the device will insert a dummy period after the wake-up. If the system clock comes from an RC oscillator, it continues operating immediately. If the wake-up results in next instruction execution, this will execute immediately after the dummy period is completed.

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:

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

The WDT time-out during HALT is different from other chip reset conditions, since it can perform a warm reset that just resets the program counter and stack pointer, leaving the other circuits to remain 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	$\overline{\text{RESET}}$ reset during power-up
u	u	$\overline{\text{RESET}}$ reset during normal operation
0	0	$\overline{\text{RESET}}$ wake-up HALT
1	u	WDT time-out during normal operation
1	1	WDT wake-up HALT

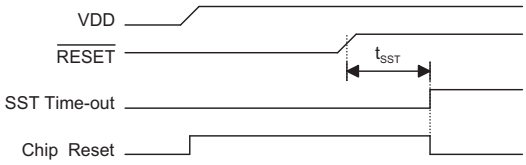
Note: "u" means unchanged

To guarantee that the system oscillator 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 it awakes from the HALT state.

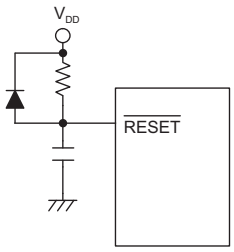
When a system power-up occurs, the SST delay is added during the reset period. But when the reset comes from the $\overline{\text{RESET}}$ pin, the SST delay is disabled. Any wake-up from HALT will enable the SST delay.

The functional unit chip reset status is shown below.

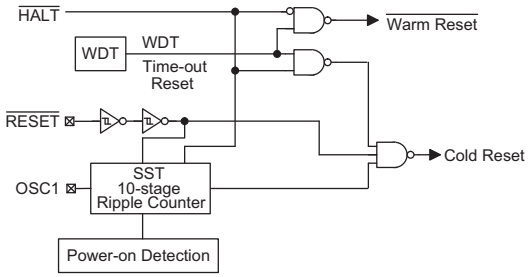
Program Counter	000H
Prescaler	Clear
WDT	Clear. After master reset, WDT begins counting
Timer counter	Off
Input/output ports	Input mode
Stack Pointer	Points to the top of the stack



Reset Timing Chart



Reset Circuit



Reset Configuration

Timer Counter

A timer counter (TMR) is implemented in the microcontroller. The timer counter contains an 8-bit programmable count-up counter and the clock may come from the system clock divided by 4.

Using the internal instruction clock, there is only one reference time-base.

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

Bit No.	Label	Function
0~3	—	Unused bit, read as "0"
4	TON	To enable/disable timer counting (0= disabled; 1= enabled)
5	—	Unused bit, read as "0"
6 7	TM0 TM1	10= Timer mode (internal clock)

TMRC (0EH) Register

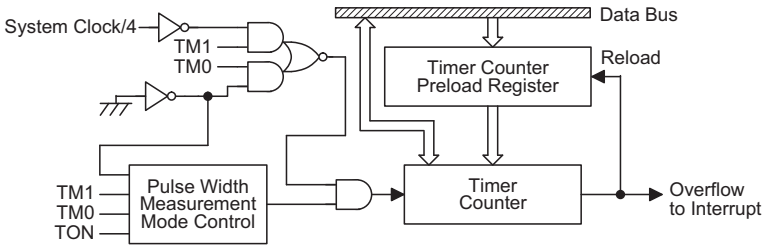
The state of the registers is summarized in the following table:

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{\text{RESET}}$ Reset (Normal Operation)	$\overline{\text{RESET}}$ Reset (HALT)	WDT Time-out (HALT)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu 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	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--00 uuuu	--11 uuuu
INTC	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMRC	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PE	---1 1111	---1 1111	---1 1111	---1 1111	---u uuuu
PEC	---1 1111	---1 1111	---1 1111	---1 1111	---u uuuu

Note: "*" stands for warm reset

"u" stands for unchanged

"x" stands for unknown



Timer Counter

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

To enable the counting operation, the timer ON bit (TON; bit 4 of TMRC) should be set to 1. In the case of timer counter OFF condition, writing data to the timer counter preload register will also reload that data to the timer counter. But if the timer counter is turned on, data written to it will only be kept in the timer counter preload register. The timer counter will still operate until overflow occurs. When the timer counter (reading TMR) is read, the clock will be blocked to avoid errors. As clock blocking may results in a counting error, this must be taken into consideration by the programmer.

Input/Output Ports

There are 34 bidirectional input/output lines in the microcontroller, labeled from PA to PE, which are mapped to the data memory of [12H], [14H], [16H], [18H] and [1AH] respectively. All 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, 16H, 18H or 1AH). For output operation, all data is latched and remains unchanged until the output latch is rewritten.

Each I/O line has its own control register (PAC, PBC, PCC, PDC, PEC) to control the input/output configuration. With this control register, CMOS output or Schmitt trigger input with or without pull-high resistor (mask option) structures can be reconfigured dynamically (i.e., on-the-fly) under software control. To function as an input, the corresponding latch of the control register must write "1". The pull-high resistance will exhibit automatically if the pull-high option is selected. The input source(s) also depend(s) on the control register. If the control register bit is "1", 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 "read-modify-write" instruction. For output function, CMOS is the only configuration. These control registers are mapped to locations 13H, 15H, 17H, 19H and 1BH.

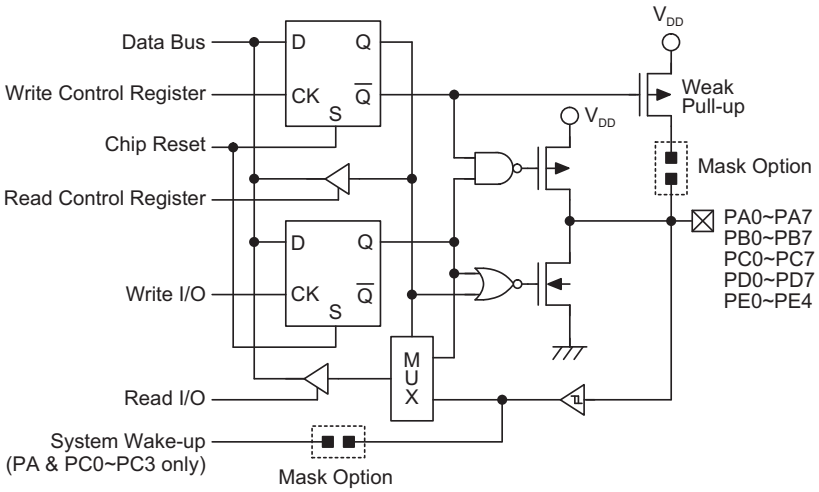
After a chip reset, these input/output lines stay at high levels or floating (mask option). Each bit of these input/output latches can be set or cleared by the SET [m].i or CLR [m].i (m=12H, 14H, 16H, 18H or 1AH) instruction.

Some instructions first input data and then follow the output operations. For example, the SET [m].i, CLR [m].i, CPL [m] and CPLA [m] instructions 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 and port C [0:3] has the capability to wake-up the device.

PC2 is shared with the external interrupt pin, PE2~PE4 is defined as CMOS output pins only. PE0 can determine whether the high to low transition, or the low to high transition of PC2 to activate the external subroutine, when PE0 output high, the low to high transition of PC2 to trigger the external subroutine, when PE0 output low, the high to low transition of PC2 to trigger the external subroutine.

PE2~PE4 is configured as CMOS output only and is used to drive the LED. PC0, PC1 is configured as NMOS open drain output with 4.6kΩ pull-high resistor such that it can easy to use as DATA or CLOCK line of PS2 keyboard application.



Input/Output Ports

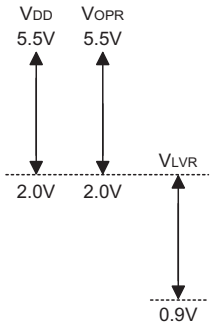
Low Voltage Reset – LVR

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.

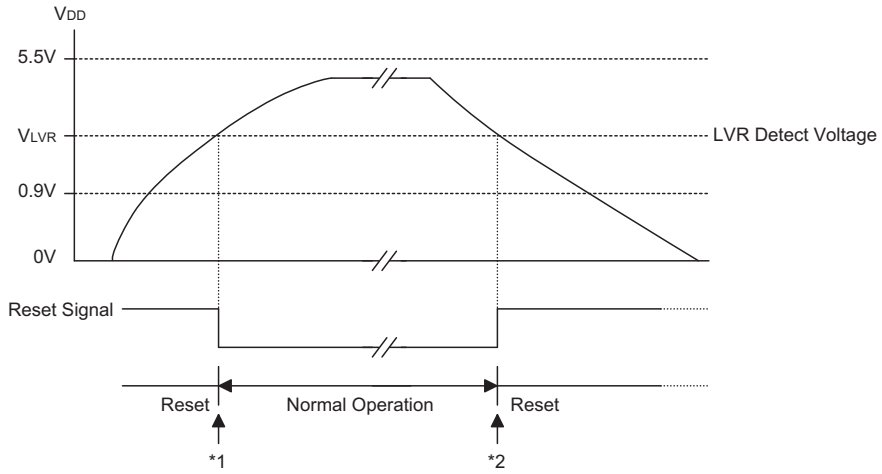
The LVR includes the following specifications:

- 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.

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



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



Low Voltage Reset

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

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

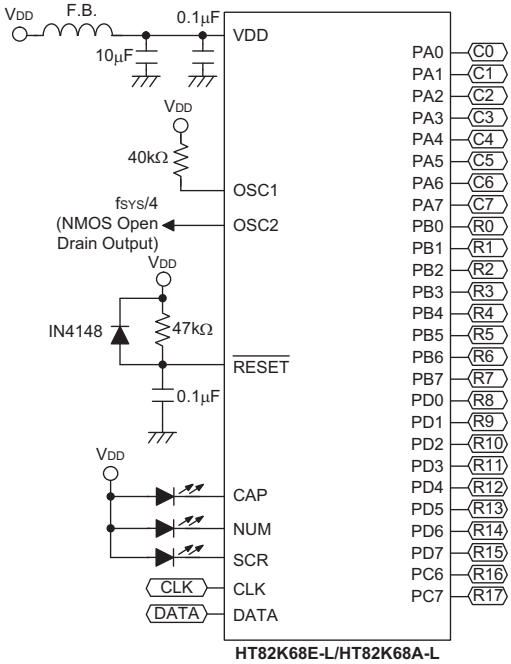
ROM Code Option

The following shows six kinds of ROM code option in the device. All the ROM code options must be defined to ensure proper system function.

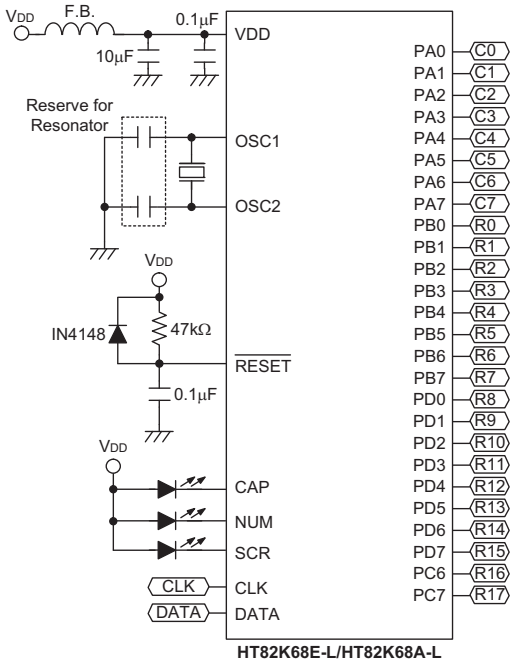
No.	ROM Code Option
1	OSC type selection. This option is to decide if an RC or Crystal oscillator is chosen as system clock. If the Crystal oscillator is selected, the XST (Crystal Start-up Timer) default is activated, otherwise the XST is disabled.
2	WDT source selection. There are three types of selection: on-chip RC oscillator, instruction clock or disable the WDT.
3	CLRWDT times selection. This option defines the way 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, only then will the WDT be cleared.
4	Wake-up selection. This option defines the wake-up function activity. External I/O pins (PA and PC [0:3] only) all have the capability to wake-up the chip from a HALT.
5	Pull-high selection. This option is to decide whether the pull-high resistance is visible or not in the input mode of the I/O ports. Each bit of an I/O port can be independently selected.
6	LVR enable/disable. User can configure whether enable or disable the circuit by configuration option.
7	The Input type only Schmitt Trigger input type can used for HT82K68E-L. The Input type Schmitt Trigger input or inverter input type can used for HT82K68A-L.

Application Circuits

RC Oscillator for Multiple I/O Applications



Crystal Oscillator or Ceramic Resonator for Multiple I/O Applications



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

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

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

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

Instruction Definition

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

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

CPL [m] Complement Data Memory

Description Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.

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

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

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

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

HALT Enter power down mode

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

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

Affected flag(s) TO, PDF

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

OR A,x	Logical OR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ x
Affected flag(s)	Z
ORM A,[m]	Logical OR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″OR″ [m]
Affected flag(s)	Z
RET	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter ← Stack
Affected flag(s)	None
RET A,x	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter ← Stack ACC ← x
Affected flag(s)	None
RETI	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None
RL [m]	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i = 0~6) [m].0 ← [m].7
Affected flag(s)	None
RLA [m]	Rotate Data Memory left with result in ACC
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i = 0~6) ACC.0 ← [m].7
Affected flag(s)	None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

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

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

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

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

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

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

SET [m] Set Data Memory

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

Operation $[m] \leftarrow FFH$

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation $[m].i \leftarrow 1$

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation ACC ← ACC – [m]

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

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

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

Operation [m] ← ACC – [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC – x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7 ~ [m].4

Affected flag(s) None

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

Description The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.

Operation ACC.3 ~ ACC.0 ← [m].7 ~ [m].4
ACC.7 ~ ACC.4 ← [m].3 ~ [m].0

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m] = 0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m] = 0

Affected flag(s) None

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

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

Operation Skip if [m].i = 0

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

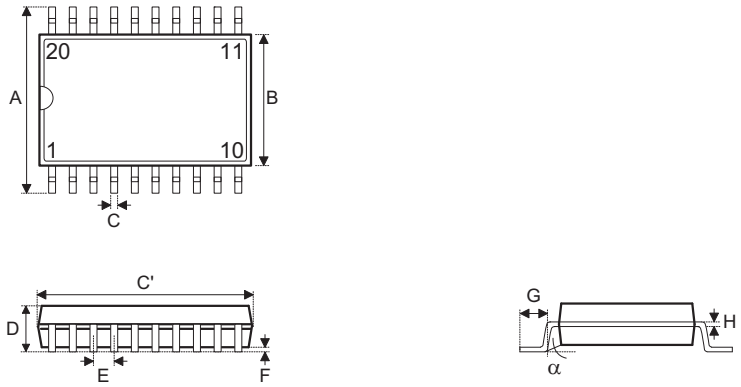
XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

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

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

Package Information

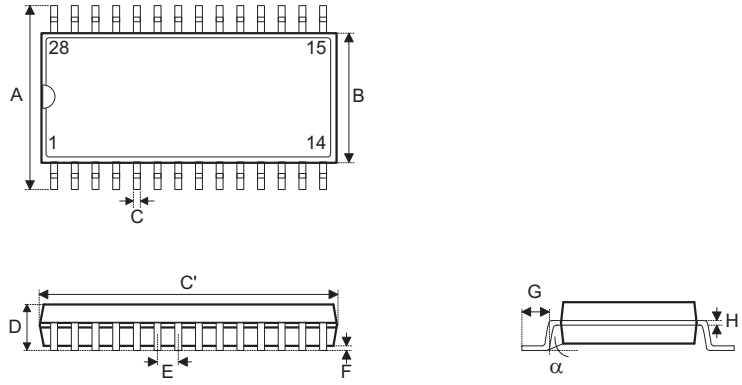
20-pin SOP (300mil) Outline Dimensions



- MS-013

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	393	—	419
B	256	—	300
C	12	—	20
C′	496	—	512
D	—	—	104
E	—	50	—
F	4	—	12
G	16	—	50
H	8	—	13
α	0°	—	8°

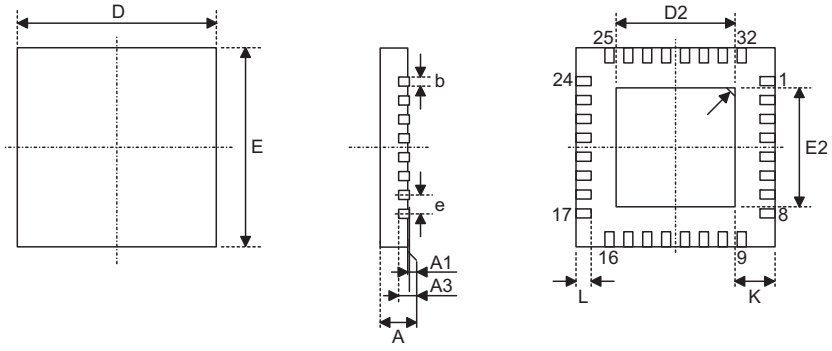
28-pin SOP (300mil) Outline Dimensions



• MS-013

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	393	—	419
B	256	—	300
C	12	—	20
C′	697	—	713
D	—	—	104
E	—	50	—
F	4	—	12
G	16	—	50
H	8	—	13
α	0°	—	8°

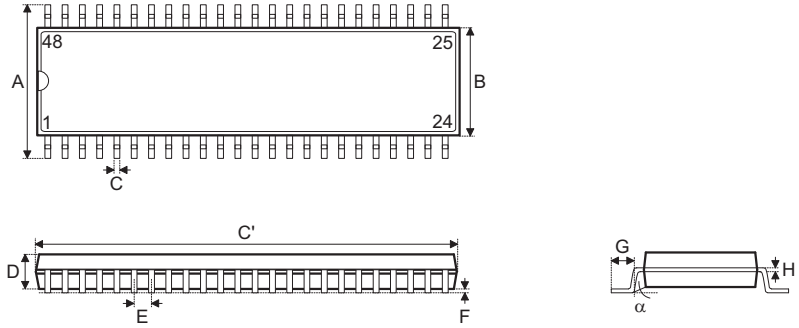
SAW Type 32-pin (5mm×5mm) QFN Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.028	—	0.031
A1	0.000	—	0.002
A3	—	0.008	—
b	0.007	—	0.012
D	—	0.197	—
E	—	0.197	—
e	—	0.020	—
D2	0.049	—	0.128
E2	0.049	—	0.128
L	0.012	—	0.020
K	—	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.70	—	0.80
A1	0.00	—	0.05
A3	—	0.20	—
b	0.18	—	0.30
D	—	5.00	—
E	—	5.00	—
e	—	0.50	—
D2	1.25	—	3.25
E2	1.25	—	3.25
L	0.30	—	0.50
K	—	—	—

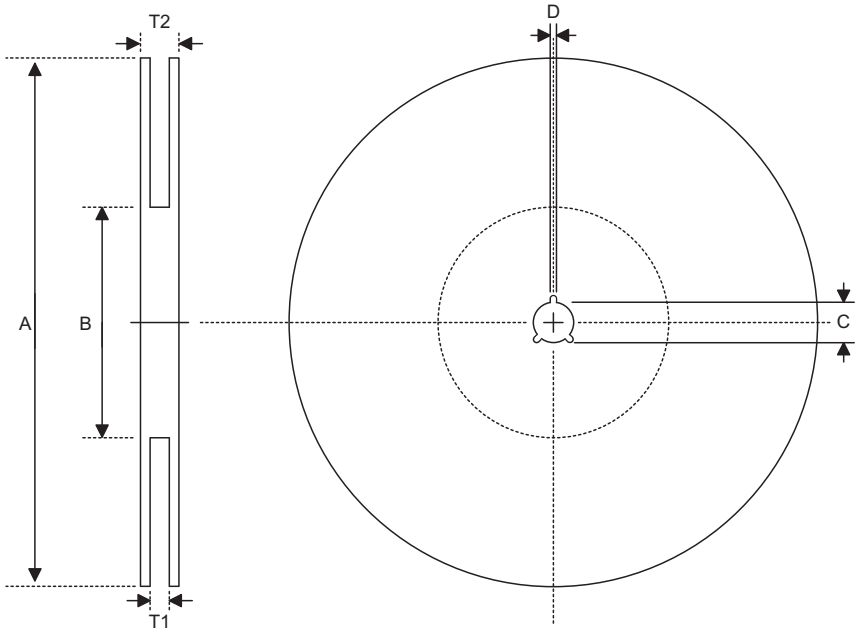
48-pin SSOP (300mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	395	—	420
B	291	—	299
C	8	—	12
C′	613	—	637
D	85	—	99
E	—	25	—
F	4	—	10
G	25	—	35
H	4	—	12
α	0°	—	8°

Product Tape and Reel Specifications

Reel Dimensions



SOP 20W

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	13.0 $^{+0.5/-0.2}$
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

SOP 28W (300mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	13.0 $^{+0.5/-0.2}$
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

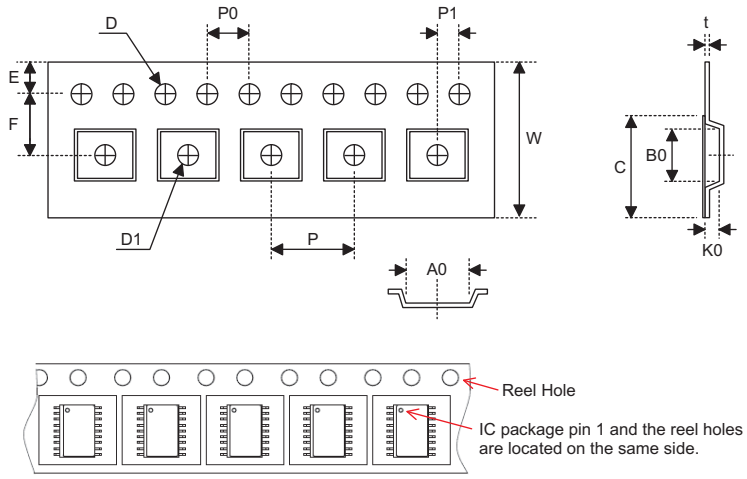
SAW Type QFN32-pin (5mm×5mm)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±0.1
C	Spindle Hole Diameter	13.0 $^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	12.5 $^{+0.3/-0.2}$
T2	Reel Thickness	—

SSOP 48W

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±0.1
C	Spindle Hole Diameter	13.0 $^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	32.2 $^{+0.3/-0.2}$
T2	Reel Thickness	38.2±0.2

Carrier Tape Dimensions

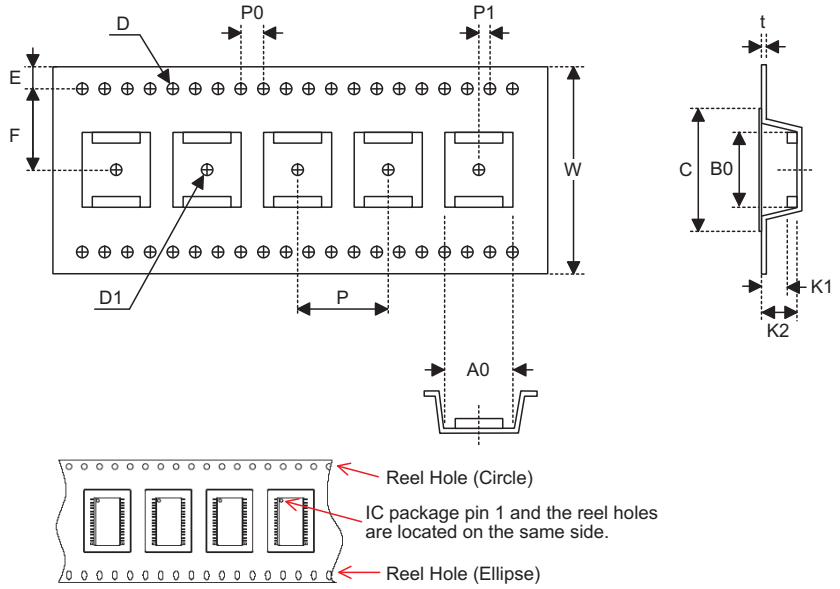


SOP 20W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0 +0.3/-0.1
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5 +0.1/-0.0
D1	Cavity Hole Diameter	1.50 +0.25/-0.00
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.8±0.1
B0	Cavity Width	13.3±0.1
K0	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	21.3±0.1

SOP 28W (300mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0±0.3
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5 +0.1/-0.0
D1	Cavity Hole Diameter	1.50 +0.25/-0.00
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.85±0.10
B0	Cavity Width	18.34±0.10
K0	Cavity Depth	2.97±0.10
t	Carrier Tape Thickness	0.35±0.01
C	Cover Tape Width	21.3±0.1



SSOP 48W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	32.0±0.3
P	Cavity Pitch	16.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	14.2±0.1
D	Perforation Diameter	2 Min.
D1	Cavity Hole Diameter	1.50 $^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	12.0±0.1
B0	Cavity Width	16.2±0.1
K1	Cavity Depth	2.4±0.1
K2	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.35±0.05
C	Cover Tape Width	25.5±0.1

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

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

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

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