

**Technical Document**

- [Tools Information](#)
- [FAQs](#)
- [Application Note](#)

**Features**

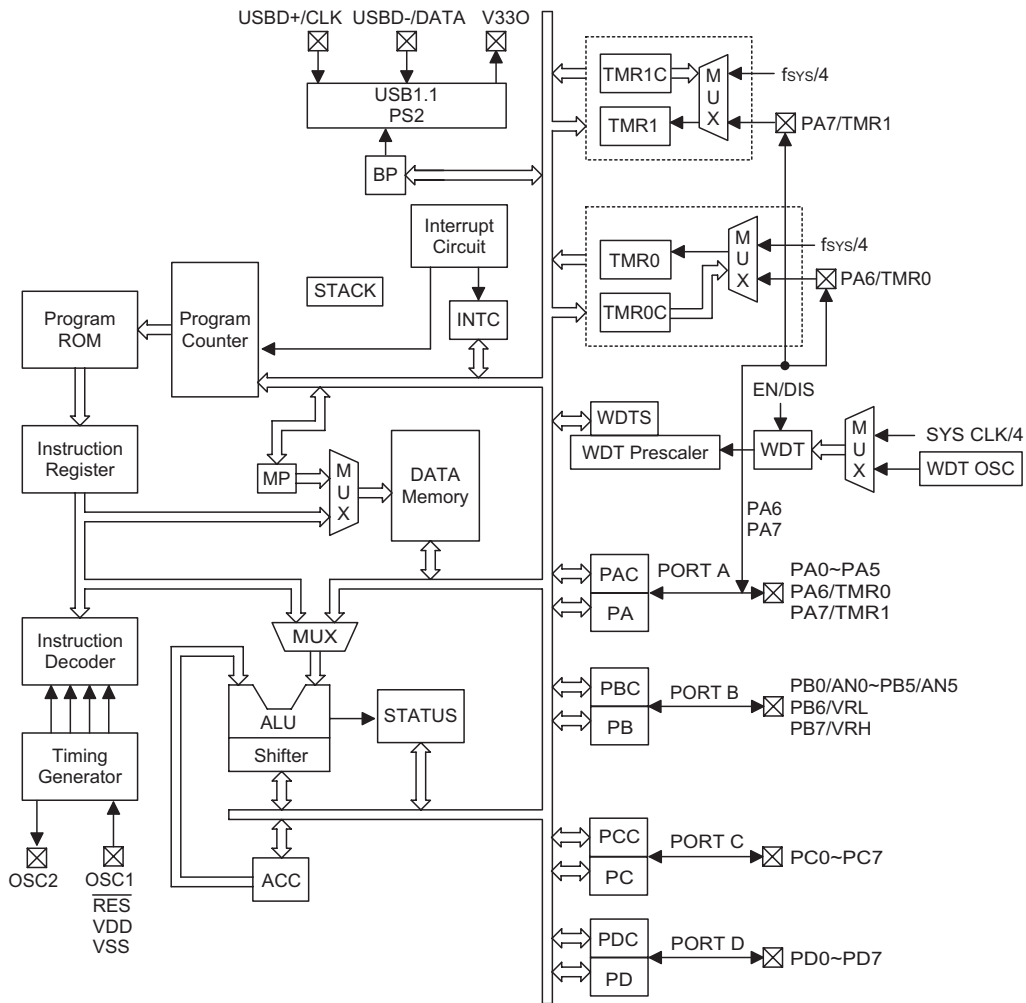
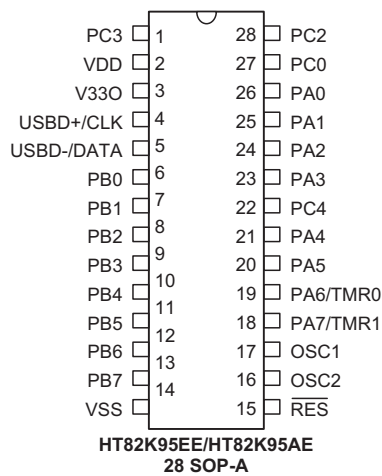
- Operating voltage:  
 $f_{SYS}=6M/12MHz: 3.3V\sim 5.5V$
- Low voltage reset function
- 32 bidirectional I/O lines (max.)
- 8-bit programmable timer/event counter with overflow interrupt
- 16-bit programmable timer/event counter and overflow interrupts
- Crystal oscillator (6MHz or 12MHz)
- Watchdog Timer
- PS2 and USB modes supported
- USB 2.0 low speed function
- 3 endpoints supported (endpoint 0 included)
- 4096×15 program memory ROM
- 160×8 data memory RAM
- 128×8 data EEPROM
- All I/O ports support wake-up options
- HALT function and wake-up feature reduce power consumption
- 8-level subroutine nesting
- Up to 0.33 $\mu s$  instruction cycle with 12MHz system clock at  $V_{DD}=5V$
- Bit manipulation instruction
- 15-bit table read instruction
- 63 powerful instructions
- All instructions in one or two machine cycles
- 28-pin SOP package

**General Description**

This device is an 8-bit high performance RISC architecture microcontroller designed for USB product applications. It is particularly suitable for use in products such as keyboards. A HALT feature is included to reduce power consumption.

The mask version HT82K95A is fully pin and functionally compatible with the OTP version HT82K95E device.

There are two dice in the HT82K95EE/HT82K95AE package: one is the HT82K95E/HT82K95A MCU, the other is a 128×8 bits EEPROM used for data memory purpose. The two dice are wire-bonded to form HT82K95EE/HT82K95AE

**Block Diagram**

**Pin Assignment**


**Pin Description**

Pin Name	I/O	ROM Code Option	Description
PA0~PA5 PA6/TMR0 PA7/TMR1	I/O	Pull-high Wake-up CMOS/NMOS/PMOS	Bidirectional 8-bit input/output port. Each bit can be configured as a wake-up input by ROM code option. The input or output mode is controlled by PAC (PA control register). Pull-high resistor options: PA0~PA7 CMOS/NMOS/PMOS options: PA0~PA7 Wake up options: PA0~PA7 PA6 and PA7 are pin-shared with TMR0 and TMR1 input, respectively.
PB0~PB7	I/O	Pull-high Wake-up	Bidirectional 8-bit input/output port. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options). Wake-up options: PB0~PB7
PD0~PD7	I/O	Pull-high Wake-up	Bidirectional I/O lines. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options). Wake-up options: PD0~PD7
VSS	—	—	Negative power supply, ground
PC0~PC7	I/O	Pull-high Wake-up	Bidirectional I/O lines. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options). Wake-up options: PC0~PC7
$\overline{\text{RES}}$	I	—	Schmitt trigger reset input. Active low
VDD	—	—	Positive power supply
V33O	O	—	3.3V regulator output
USBD+/CLK	I/O	—	USBD+ or PS2 CLK I/O line USB or PS2 function is controlled by software control register
USBD-/DATA	I/O	—	USBD- or PS2 DATA I/O line USB or PS2 function is controlled by software control register
OSC1 OSC2	I O	—	OSC1, OSC2 are connected to a 6MHz or 12MHz Crystal/resonator (determined by software instructions) for the internal system clock.

**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 .....	$0^{\circ}C$ to $70^{\circ}C$
$I_{OL}$ Total .....	150mA	$I_{OH}$ Total .....	$-100mA$
Total Power Dissipation .....	500mW		

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

**D.C. Characteristics**

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	Operating Voltage	—	f <sub>SYS</sub> =6MHz	3.3	—	5.5	V
			f <sub>SYS</sub> =12MHz	3.3	—	5.5	V
I <sub>DD1</sub>	Operating Current (6MHz Crystal)	5V	No load, f <sub>SYS</sub> =6MHz	—	6.5	12	mA
I <sub>DD2</sub>	Operating Current (12MHz Crystal)	5V	No load, f <sub>SYS</sub> =12MHz	—	7.5	16	mA
I <sub>STB1</sub>	Standby Current	5V	No load, system HALT, USB suspend*	—	—	500	μA
I <sub>STB2</sub>	Standby Current (WDT Enabled)	5V	No load, system HALT, input/output mode, set SUSPEND2 [1CH]	—	—	15	μA
V <sub>IL1</sub>	Input Low Voltage for I/O Ports	5V	—	0	—	0.8	V
V <sub>IH1</sub>	Input High Voltage for I/O Ports	5V	—	2	—	V <sub>DD</sub>	V
V <sub>IL2</sub>	Input Low Voltage ( $\overline{\text{RES}}$ )	5V	—	0	—	0.4V <sub>DD</sub>	V
V <sub>IH2</sub>	Input High Voltage ( $\overline{\text{RES}}$ )	5V	—	0.9V <sub>DD</sub>	—	V <sub>DD</sub>	V
I <sub>OL1</sub>	I/O Port Sink Current for PA1~PA7, PB, PC, PD	5V	V <sub>OL</sub> =3.4V	10	15	20	mA
I <sub>OL2</sub>	I/O Port Sink Current for PA1~PA7, PB, PC, PD	5V	V <sub>OL</sub> =0.4V	2	4	8	mA
I <sub>OL3</sub>	I/O Port Sink Current for PA0	5V	V <sub>OL</sub> =0.4V	7	10	13	mA
I <sub>OH1</sub>	I/O Port Source Current for PA1~PA7, PB, PC, PD	5V	V <sub>OH</sub> =3.4V	-2	-4	-8	mA
I <sub>OH2</sub>	I/O Port Source Current for PA0	5V	V <sub>OH</sub> =3.4V	-12	-18	-24	mA
R <sub>PH</sub>	Pull-high Resistance for PA, PB, PC, PD	5V	—	25	50	80	kΩ
V <sub>LVR</sub>	Low Voltage Reset	—	—	2	2.6	3.2	V
V <sub>V330</sub>	3.3V Regulator Output	5V	I <sub>V330</sub> =-5mA	3.0	3.3	3.6	V
C <sub>OSC1</sub>	Build_in Capacitance in OSC1	5V	—	10	15	20	pF
C <sub>OSC2</sub>	Build_in Capacitance in OSC2	5V	—	10	15	20	pF

 Note: "\*" include 15kΩ loading of USB<sub>D</sub>+, USB<sub>D</sub>- line in host terminal.

**A.C. Characteristics**

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
f <sub>SYS</sub>	System Clock (Crystal OSC)	5V	—	6	—	12	MHz
f <sub>TIMER</sub>	Timer I/P Frequency (TMR)	5V	—	0	—	12	MHz
t <sub>WDTOSC</sub>	Watchdog Oscillator	5V	—	15	31	70	μs
t <sub>WDT1</sub>	Watchdog Time-out Period (WDT OSC)	5V	Without WDT prescaler	4	8	16	ms
t <sub>WDT2</sub>	Watchdog Time-out Period (System Clock)	—	Without WDT prescaler	—	1024	—	t <sub>SYS</sub>
t <sub>RES</sub>	External Reset Low Pulse Width	—	—	1	—	—	μs
t <sub>SST</sub>	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	t <sub>SYS</sub>
			Power-up, Watchdog Time-out from normal	—	1024	—	t <sub>WDTOSC</sub>
t <sub>INT</sub>	Interrupt Pulse Width	—	—	1	—	—	μs

**EEPROM A.C. Characteristics**

Ta=25°C

Symbol	Parameter	Remark	Standard Mode*		V <sub>CC</sub> =5V±10%		Unit
			Min.	Max.	Min.	Max.	
f <sub>SK</sub>	Clock Frequency	—	—	100	—	400	kHz
t <sub>HIGH</sub>	Clock High Time	—	4000	—	600	—	ns
t <sub>LOW</sub>	Clock Low Time	—	4700	—	1200	—	ns
t <sub>r</sub>	SDA and SCL Rise Time	Note	—	1000	—	300	ns
t <sub>f</sub>	SDA and SCL Fall Time	Note	—	300	—	300	ns
t <sub>HD:STA</sub>	START Condition Hold Time	After this period the first clock pulse is generated	4000	—	600	—	ns
t <sub>SU:STA</sub>	START Condition Setup Time	Only relevant for repeated START condition	4000	—	600	—	ns
t <sub>HD:DAT</sub>	Data Input Hold Time	—	0	—	0	—	ns
t <sub>SU:DAT</sub>	Data Input Setup Time	—	200	—	100	—	ns
t <sub>SU:STO</sub>	STOP Condition Setup Time	—	4000	—	600	—	ns
t <sub>AA</sub>	Output Valid from Clock	—	—	3500	—	900	ns
t <sub>BUF</sub>	Bus Free Time	Time in which the bus must be free before a new transmission can start	4700	—	1200	—	ns
t <sub>SP</sub>	Input Filter Time Constant (SDA and SCL Pins)	Noise suppression time	—	100	—	50	ns
t <sub>WR</sub>	Write Cycle Time	—	—	5	—	5	ms

Note: These parameters are periodically sampled but not 100% tested

 \* The standard mode means V<sub>CC</sub>=2.2V to 5.5V

For relative timing, refer to timing diagrams

## Functional Description

### Execution Flow

The system clock for the microcontroller is derived from a crystal. 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 the program ROM are executed and its contents specify a full range of program memory.

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

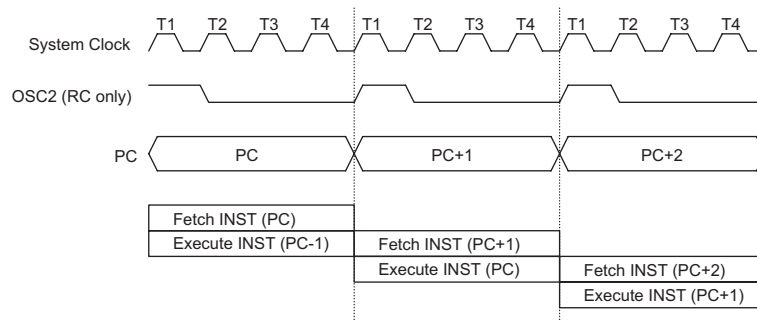
incremented by one. The program counter then points to the memory word containing the next instruction code.

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

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

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



**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
USB interrupt	0	0	0	0	0	0	0	0	0	1	0	0
Timer/Event Counter 0 overflow	0	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter 1 overflow	0	0	0	0	0	0	0	0	1	1	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

### Program Counter

Note: \*11~\*0: Program counter bits

S11~S0: Stack register bits

#11~#0: Instruction code bits

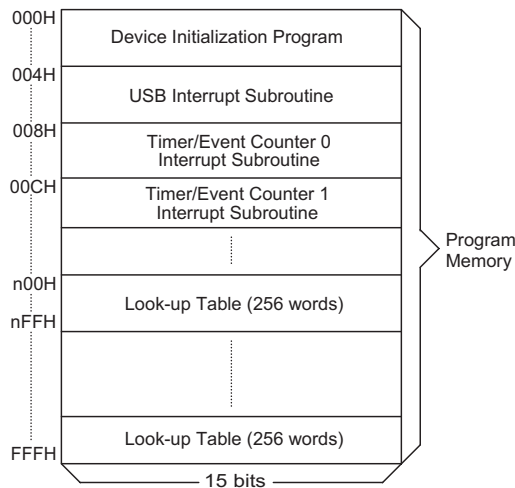
@7~@0: PCL bits

**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 4096×15 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 chip reset, the program always begins execution at location 000H.
- Location 004H  
This area is reserved for the USB interrupt service program. If the USB interrupt 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 0 interrupt service program. If a timer interrupt results from a Timer/Event Counter 0 overflow, and if the interrupt is enabled and the stack is not full, the program begins execution at location 008H.



Note: n ranges from 0 to F

**Program Memory**

- Location 00CH  
This location is reserved for the Timer/Event Counter 1 interrupt service program. If a timer interrupt results from a Timer/Event Counter 1 overflow, and 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. There are three methods to read the ROM data by two table read instructions: "TABRDC" and "TABRDL", transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH (08H). The three methods are shown as follows:
  - The instructions "TABRDC [m]" (the current page, one page=256words), where the table locations is defined by TBLP (07H) in the current page. And the ROM code option TBHP is disabled (default).
  - The instructions "TABRDC [m]", where the table locations is defined by registers TBLP (07H) and TBHP (01FH). And the ROM code option TBHP is enabled.
  - The instructions "TABRDL [m]", where the table locations is defined by Registers TBLP (07H) in the last page (0F00H~0FFFH).

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 1-bit words are read as "0". The Table Higher-order byte register (TBLH) is read only. The table pointer (TBLP, TBHP) is a read/write register (07H, 1FH), which indicates the table location. Before accessing the table, the location must be placed in the TBLP and TBHP (If the OTP option TBHP is disabled, the value in TBHP has no effect). 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 should be disabled prior to the table read instruction.

Instruction	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	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

**Table Location**

Note: \*11~\*0: Table location bits  
@7~@0: Table pointer bits

P11~P8: Current program counter bits when TBHP is disabled  
TBHP register bit3~bit0 when TBHP is enabled

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 on the requirements.

Once TBHP is enabled, the instruction "TABRDC [m]" reads the ROM data as defined by TBLP and TBHP value. Otherwise, the ROM code option TBHP is disabled, the instruction "TABRDC [m]" reads the ROM data as defined by TBLP and the current program counter bits.

### Stack Register – STACK

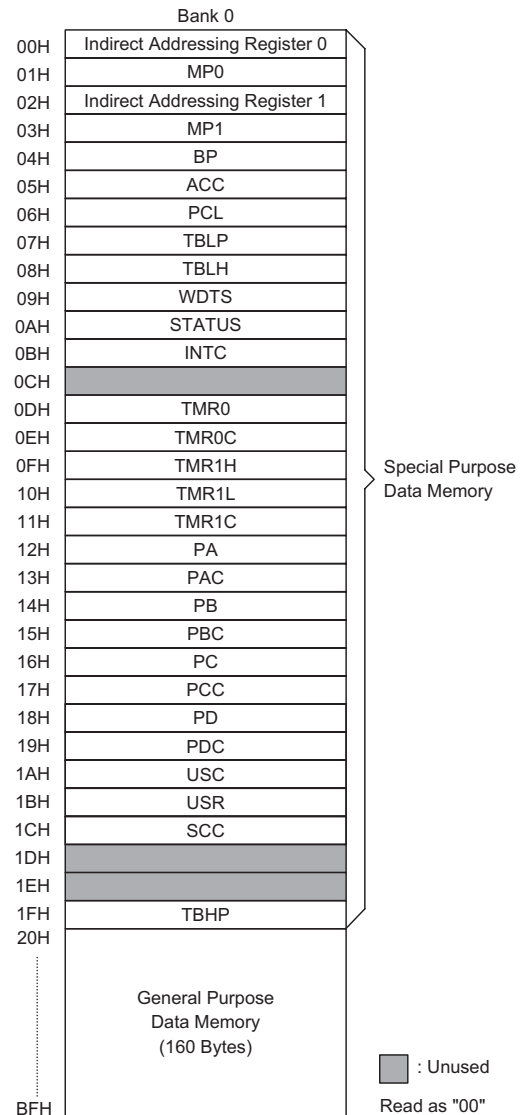
This is a special part of the memory which is used to save the contents of the program counter only. The stack is organized into 8 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the stack pointer (SP) and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, 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 signal 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 8 return addresses are stored).

### Data Memory – RAM for Bank 0

The data memory is designed with 190×8 bits. The data memory is divided into two functional groups: special function registers and general purpose data memory (160×8). Most are read/write, but some are read only.

The special function registers include the indirect addressing registers (R0;00H, R1;02H), Bank register (BP, 04H), Timer/Event Counter 0 (TMR0;0DH), Timer/Event Counter 0 control register (TMR0C;0EH), Timer/Event Counter 1 higher order byte register (TMR1H;0FH), Timer/Event Counter 1 lower order byte register (TMR1L;10H), Timer/Event Counter 1 control register (TMR1C;11H), program counter lower-order byte register (PCL;06H), memory pointer registers (MP0;01H, MP1;03H), accumulator (ACC;05H), table pointer (TBLP;07H, TBHP;1FH), table higher-order byte register (TBLH;08H), status register (STATUS;0AH), interrupt control register (INTC;0BH),



**Bank 0 RAM Mapping**

Watchdog Timer option setting register (WDTS;09H), I/O registers (PA;12H, PB;14H, PC;16H, PD;18H), I/O control registers (PAC;13H, PBC;15H, PCC;17H, PDC;19H). USB/PS2 status and control register (USC;1AH), USB endpoint interrupt status register (USR;1BH), system clock control register (SCC;1CH). The remaining space before the 20H is reserved for future expansion usage and reading these locations will get "00H". The general purpose data memory, addressed from 20H to BFH, 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 registers (MP0 or MP1).

**Data Memory – RAM for Bank 1**

The special function registers used in USB interface are located in RAM bank 1. In order to access the Bank1 register, only the Indirect addressing pointer MP1 can be used and the Bank register BP should be set to "1". The mapping of RAM bank 1 is as shown.

**Indirect Addressing Register**

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

The indirect addressing pointer (MP0) always point to Bank0 RAM addresses regardless of the value of the Bank Register (BP).

The indirect addressing pointer (MP1) can access Bank0 or Bank1 RAM data according to the value of BP which is set to "0" or "1" respectively.

The memory pointer registers (MP0 and MP1) are 8-bit registers.

40H	—
41H	PIPE_CTRL
42H	AWR
43H	STALL
44H	PIPE
45H	SIES
46H	MISC
47H	Endpt_EN
48H	FIFO0
49H	FIFO1
4AH	FIFO2
4BH	—
4CH	Undefined, reserved for future expansion
...	
...	
FFH	

**Bank 1 RAM Mapping**

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 the operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
4	PDF	PDF is cleared by system power-up or executing the "CLR WDT" instruction. PDF is set by executing the "HALT" instruction.
5	TO	TO is cleared by system power-up or executing the "CLR WDT" or "HALT" instruction. TO is set by a WDT time-out.
6~7	—	Unused bit, read as "0"

**Status (0AH) Register**
**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 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 during a system power-up.

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

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

### Interrupt

The device provides an external interrupt and internal timer/event counter interrupts. The Interrupt Control Register (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.

USB interrupts are triggered by the following USB events and the related interrupt request flag (USBF; bit 4 of the INTC) will be set.

- The corresponding USB FIFO is accessed from the PC
- The USB suspends signal from the PC
- The USB resumes signal from the PC
- The USB sends Reset signal

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 (USBF) and EMI bits will be cleared to disable other interrupts.

When the PC Host access the FIFO of the HT82K95E/HT82K95A, the corresponding request bit of the USR is set, and a USB interrupt is triggered. So user can easily decide which FIFO is accessed. When the interrupt has been served, the corresponding bit should be cleared by firmware. When the HT82K95EE/HT82K95AE receives a USB Suspend signal from the Host PC, the suspend line (bit0 of the USC) of the HT82K95EE/HT82K95AE is set and a USB interrupt is also triggered.

Also when the HT82K95EE/HT82K95AE receives a Resume signal from the Host PC, the resume line (bit3 of the USC) of HT82K95EE/HT82K95AE is set and a USB interrupt is triggered.

Whenever a USB reset signal is detected, the USB interrupt is triggered.

The internal Timer/Event Counter 0 interrupt is initialized by setting the Timer/Event Counter 0 interrupt request flag (T0F; bit 5 of INTC), caused by a timer 0 overflow. When the interrupt is enabled, 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 internal Timer/Event Counter 1 interrupt is initialized by setting the Timer/Event Counter 1 interrupt request flag (T1F; bit 6 of INTC), caused by a timer 1 overflow. When the interrupt is enabled, the stack is not full and the T1F is set, a subroutine call to location 0CH will occur. The related interrupt request flag (T1F) will be reset and the EMI bit cleared to disable further interrupts.

During the execution of an interrupt subroutine, other interrupt acknowledge signals 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.

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1= enabled; 0= disabled)
1	EUI	Controls the USB interrupt (1= enabled; 0= disabled)
2	ET0I	Controls the Timer/Event Counter 0 interrupt (1= enabled; 0= disabled)
3	ET1I	Controls the Timer/Event Counter 1 interrupt (1= enabled; 0= disabled)
4	USBF	USB interrupt request flag (1= active; 0= inactive)
5	T0F	Internal Timer/Event Counter 0 request flag (1= active; 0= inactive)
6	T1F	Internal Timer/Event Counter 1 request flag (1= active; 0= inactive)
7	—	Unused bit, read as "0"

**INTC (0BH) Register**

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.

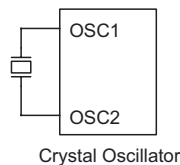
No.	Interrupt Source	Priority	Vector
a	USB interrupt	1	04H
b	Timer/Event Counter 0 overflow	2	08H
c	Timer/Event Counter 1 overflow	3	0CH

The Timer/Event Counter 0/1 interrupt request flag (T0F/T1F), USB interrupt request flag (USBF), enable Timer/Event Counter 0/1 interrupt bit (ET0I/ET1I), enable USB interrupt bit (EUI) and enable master interrupt bit (EMI) constitute an interrupt control register (INTC) which is located at 0BH in the data memory. EMI, EUI, ETI are used to control the enabling/disabling of interrupts. These bits prevent the requested interrupt from being serviced. Once the interrupt request flags (TF, USBF) 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.

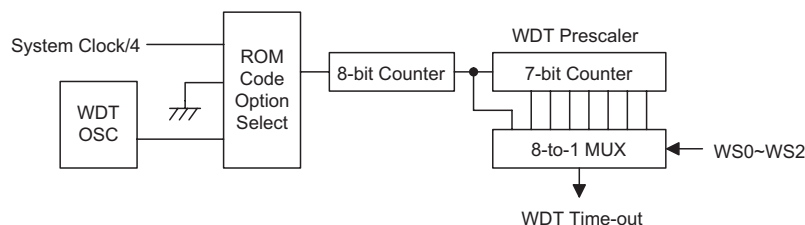
**Oscillator Configuration**

There is an oscillator circuits in the microcontroller.



**System Oscillator**

This oscillator is designed for system clocks. The HALT mode stops the system oscillator and ignores an external signal to conserve power.



**Watchdog Timer**

A crystal across OSC1 and OSC2 is needed to provide the feedback and phase shift required for the oscillator. No other external components are required. In stead 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.

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 within a period of approximately 31µs. The WDT oscillator can be disabled by ROM code option to conserve power.

**Watchdog Timer – WDT**

The WDT clock source is implemented by a dedicated RC oscillator (WDT oscillator), or instruction clock (system clock divided by 4), determines the ROM code option. This timer is designed to prevent a software malfunction or sequence from jumping to an unknown location with unpredictable results. The Watchdog Timer can be disabled by ROM code option. If the Watchdog Timer is disabled, all the executions related to the WDT result in no operation.

Once the internal WDT oscillator (RC oscillator, normally with a period of 31µs/5V) is selected, it is first divided by 256 (8-stage) to get the nominal time-out period of 8ms/5V. This time-out period may vary with temperatures, VDD and process variations. By invoking the WDT prescaler, longer time-out periods can be realized. Writing data to WS2, WS1, WS0 (bits 2, 1, 0 of the WDTS) can give different time-out periods. If WS2, WS1, and WS0 are all equal to 1, the division ratio is up to 1:128, and the maximum time-out period is 1s/5V. 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. The high nibble and bit 3 of the WDTS are reserved for user's defined flags, which can only be set to "10000" (WDTS.7~WDTS.3).

If the device operates in a noisy environment, using the on-chip 32kHz 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 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 contents of the WDT (including the WDT prescaler), three methods are employed; external reset (a low level to RES), software instruction and a "HALT" instruction. The software instruction 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 ROM code 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 wherein "CLR WDT1" and "CLR WDT2" are chosen (i.e. CLRWDT times is equal to two), these two instructions must be executed to clear the WDT, otherwise, the WDT may reset the chip as a result of time-out.

#### 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 remains 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 I/O ports 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 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 exe-

cuting 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 others remain in their original status.

The I/O ports wake-up and interrupt methods can be considered as a continuation of normal execution. Each bit in the Port A can be independently selected to wake up the device by option. PB, PC and PD can also be selected to wake up the device by option. Upon awakening from an I/O port stimulus, the program will resume execution of the next instruction. If it awakens from an interrupt, two sequence 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, the 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  $1024 t_{SYS}$  (system clock period) to resume normal operation. In other words, a dummy period will be inserted after a wake-up. If the wake-up results from an interrupt acknowledge signal, 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 completed.

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:

- $\overline{RES}$  reset during normal operation
- $\overline{RES}$  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 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".

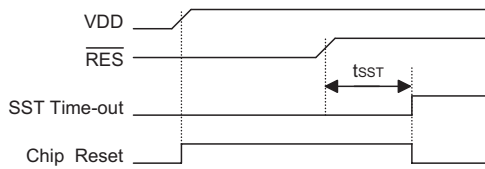
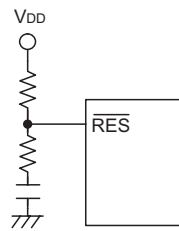
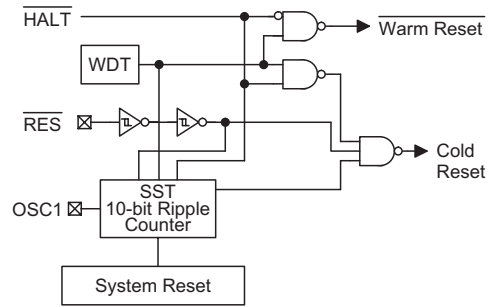
TO	PDF	RESET Conditions
0	0	$\overline{RES}$ reset during power-up
0	0	$\overline{RES}$ reset during normal operation
0	0	$\overline{RES}$ wake-up HALT
1	u	WDT time-out during normal operation
1	1	WDT wake-up HALT

Note: "u" stands for "unchanged"

To guarantee that the system oscillator is started and stabilized, the SST (System Start-up Timer) provides an

extra delay of 1024 system clock pulses when the system resets (power-up, WDT time-out or  $\overline{\text{RES}}$  reset) or when the system awakes from the HALT state.

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


**Reset Timing Chart**

**Reset Circuit**

**Reset Configuration**

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
Stack Pointer	Points to the top of the stack

The status of the registers 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)*	USB-Reset (Normal)	USB-Reset (HALT)
TMR0	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu	00-0 1000	00-0 1000
TMR1H	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR1L	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR1C	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---	00-0 1---	00-0 1---
Program Counter	000H	000H	000H	000H	000H	000H	000H
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
STATUS	--00 xxxx	--1u uuuu	--00 uuuu	--00 uuuu	--11 uuuu	--uu uuuu	--01 uuuu
INTC	-000 0000	-000 0000	--00 uuuu	-000 0000	-uuu uuuu	-000 0000	-000 0000
WDTS	1000 0111	1000 0111	1000 0111	1000 0111	uuuu uuuu	1000 0111	1000 0111
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111

Register	Reset (Power On)	WDT Time-out (Normal Operation)	RES Reset (Normal Operation)	RES Reset (HALT)	WDT Time-Out (HALT)*	USB-Reset (Normal)	USB-Reset (HALT)
PIPE_CTRL	0000 0110	0000 uuuu	0000 0110	0000 0110	0000 uuuu	0000 0110	0000 0110
AWR	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PIPE	0000 0000	xxxx xxxx	0000 0000	0000 0000	xxxx xxxx	0000 0000	0000 0000
STALL	0000 0110	0000 uuuu	0000 0110	0000 0110	0000 uuuu	0000 0110	0000 0110
SIES	0100 0000	uxux xuuu	0100 0000	0100 0000	uxux xuuu	0100 0000	0100 0000
MISC	0x00 0000	uxuu uuuu	0x00 0000	0x00 0000	uxuu uuuu	0x00 0000	0x00 0000
Endpt_EN	0000 0111	0000 uuuu	0000 0111	0000 0111	0000 uuuu	0000 0111	0000 0111
FIFO0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
USC	11xx 0000	uuxx uuuu	11xx 0000	11xx 0000	uuxx uuuu	uu00 0u00	uu00 0u00
USR	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	u1uu 0000	u1uu 0000
SCC	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0uu0 u000	0uu0 u000
TBHP	---- xxxx	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu	---- uuuu

Note: "\*" stands for "warm reset"  
 "u" stands for "unchanged"  
 "x" stands for "unknown"

#### Timer/Event Counter

Two timer/event counters (TMR0, TMR1) are implemented in the microcontroller. The Timer/Event Counter 0 contains an 8-bit programmable count-up counter and the clock may come from an external source or from  $f_{SYS}/4$ .

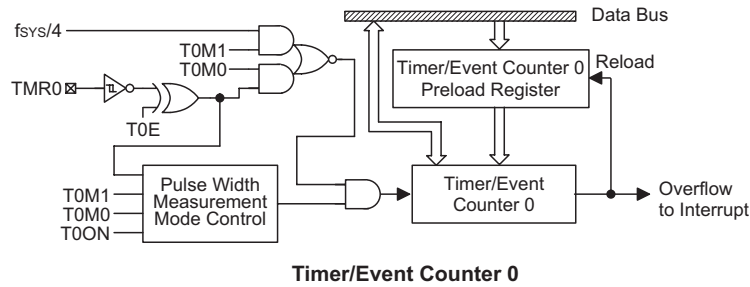
The Timer/Event Counter 1 contains an 16-bit programmable count-up counter and the clock may come from an external source or from the system clock divided by 4.

Bit No.	Label	Function
0~2, 5	—	Unused bit, read as "0"
3	T0E	To define the TMR0 active edge of Timer/Event Counter 0 (0=active on low to high; 1=active on high to low)
4	T0ON	To enable/disable timer 0 counting (0=disabled; 1=enabled)
6 7	T0M0 T0M1	To define the operating mode 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

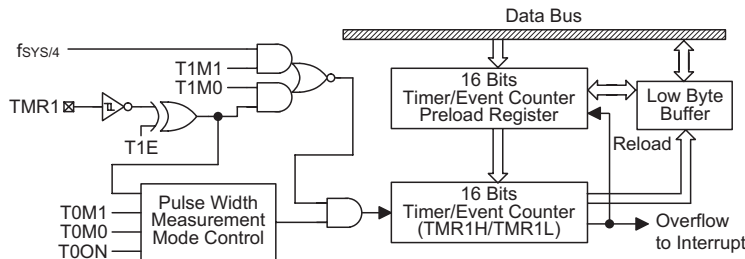
#### TMR0C (0EH) Register

Bit No.	Label	Function
0~2, 5	—	Unused bit, read as "0"
3	T1E	To define the TMR1 active edge of Timer/Event Counter 1 (0=active on low to high; 1=active on high to low)
4	T1ON	To enable/disable timer 1 counting (0=disabled; 1=enabled)
6 7	T1M0 T1M1	To define the operating mode 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

#### TMR1C (11H) Register



Timer/Event Counter 0



Timer/Event Counter 1

Using the internal clock source, there is only 1 reference time-base for Timer/Event Counter 0. The internal clock source is coming from  $f_{SYS}/4$ .

The external clock input allows the user to count external events, measure time intervals or pulse widths.

Using the internal clock source, there is only 1 reference time-base for Timer/Event Counter 1. The internal clock source is coming from  $f_{SYS}/4$ . The external clock input allows the user to count external events, measure time intervals or pulse widths.

There are 2 registers related to the Timer/Event Counter 0; TMR0 ([0DH]), TMR0C ([0EH]). Two physical registers are mapped to TMR0 location; writing TMR0 makes the starting value be placed in the Timer/Event Counter 0 preload register and reading TMR0 gets the contents of the Timer/Event Counter 0. The TMR0C is a timer/event counter control register, which defines some options.

There are 3 registers related to Timer/Event Counter 1; TMR1H (0FH), TMR1L (10H), TMR1C (11H). Writing TMR1L will only put the written data to an internal lower-order byte buffer (8 bits) and writing TMR1H will transfer the specified data and the contents of the lower-order byte buffer to TMR1H and TMR1L preload registers, respectively. The Timer/Event Counter 1 preload register is changed by each writing TMR1H operations. Reading TMR1H will latch the contents of TMR1H and TMR1L counters to the destination and the lower-order byte buffer, respectively. Reading the TMR1L will read the contents of the lower-order byte buffer. The TMR1C is the Timer/Event Counter 1 control register, which defines the operating mode, counting enable or disable and active edge.

The TOM0/TOM1, T1M0/T1M1 bits define the operating mode. The event count mode is used to count external events, which means the clock source comes from an external (TMR0/TMR1) pin. The timer mode functions as a normal timer with the clock source coming from the  $f_{SYS}/4$  (Timer0/Timer1). The pulse width measurement mode can be used to count the high or low level duration of the external signal (TMR0/TMR1). The counting is based on the  $f_{SYS}/4$  (Timer0/Timer1).

In the event count or timer mode, once the Timer/Event Counter 0/1 starts counting, it will count from the current contents in the Timer/Event Counter 0/1 to FFH or FFFFH. Once overflow occurs, the counter is reloaded from the Timer/Event Counter 0/1 preload register and generates the interrupt request flag (T0F/T1F; bit 5/6 of INTC) at the same time.

In the pulse width measurement mode with the T0ON/T1ON and TE bits equal to one, once the TMR0/TMR1 has received a transient from low to high (or high to low if the T0E/T1E bits is "0") it will start counting until the TMR0/TMR1 returns to the original level and resets the T0ON/T1ON. The measured result will remain in the Timer/Event Counter 0/1 even if the activated transient occurs again. In other words, only one cycle measurement can be done. Until setting the T0ON/T1ON, the cycle measurement will function again as long as it receives further transient pulse. Note that, in this operating mode, the Timer/Event Counter 0/1 starts counting not according to the logic level but according to the transient edges. In the case of counter overflows, the counter 0/1 is reloaded from the Timer/Event Counter 0/1 preload register and issues the interrupt request just like the other two modes. To enable the counting operation, the timer ON bit (T0ON/T1ON; bit 4 of TMR0C/TMR1C) should be set to

1. In the pulse width measurement mode, the T0ON/T1ON will be cleared automatically after the measurement cycle is completed. But in the other two modes the T0ON/T1ON can only be reset by instructions. The overflow of the Timer/Event Counter 0/1 is one of the wake-up sources. No matter what the operation mode is, writing a 0 to ET0I/ET1I can disable the corresponding interrupt services.

In the case of Timer/Event Counter 0/1 OFF condition, writing data to the Timer/Event Counter 0/1 preload register will also reload that data to the Timer/Event Counter 0/1. But if the Timer/Event Counter 0/1 is turned on, data written to it will only be kept in the Timer/Event Counter 0/1 preload register. The Timer/Event Counter 0/1 will still operate until overflow occurs (a Timer/Event Counter 0/1 reloading will occur at the same time). When the Timer/Event Counter 0/1 (reading TMR0/TMR1) 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.

**Input/Output Ports**

There are 32 bidirectional input/output lines in the microcontroller, labeled from PA to PD, which are mapped to the data memory of [12H], [14H], [16H] 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, 16H 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, PCC, PDC) to control the input/output configuration. With this control register, CMOS/NMOS/PMOS output or Schmitt trigger input with or without pull-high resistor

structures can be reconfigured dynamically under software control. To function as an input, the corresponding latch of the control register must write a "1". The input source also depends on the control register. If the control register bit is "1", the input will read the pad state. If the control register bit is "0", the contents of the latches will move to the internal bus. The latter is possible in the "read-modify-write" instruction. For output function, CMOS/NMOS/PMOS configurations can be selected (NMOS and PMOS are available for PA only). These control registers are mapped to locations 13H, 15H, 17H and 19H.

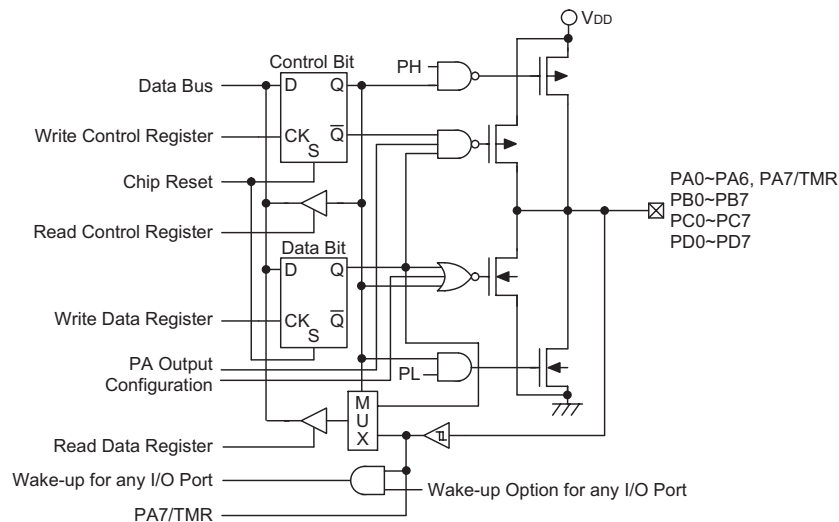
After a chip reset, these input/output lines remain at high levels or floating state (depending on the pull-high options). Each bit of these input/output latches can be set or cleared by "SET [m].i" and "CLR [m].i" (m=12H, 14H, 16H or 18H) instructions.

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 all the I/O ports have the capability of waking up the device.

There are pull-high (PA only) options available for I/O lines. Once the pull-high option of an I/O line is selected, the I/O line have pull-high resistor. Otherwise, the pull-high resistor is absent. It should be noted that a non-pull-high I/O line operating in input mode will cause a floating state.

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



**Input/Output Ports**

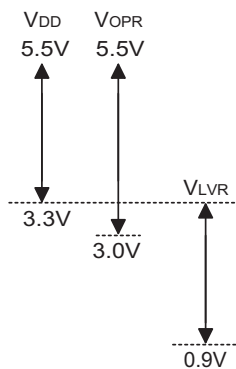
**Low Voltage Reset – LVR**

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. If the supply voltage of the device drops to within a range of 0.9V~V<sub>LVR</sub> such as might occur when changing the battery, the LVR will automatically reset the device internally.

The LVR includes the following specifications:

- For a valid LVR signal, a low voltage i.e. a voltage in the range between 0.9V~V<sub>LVR</sub> must exist for greater than 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 RES signal to perform chip reset.

The relationship between V<sub>DD</sub> and V<sub>LVR</sub> is shown below.



Note: V<sub>OPR</sub> is the voltage range for proper chip operation at 4MHz system clock.

**Data EEPROM Functional Description**

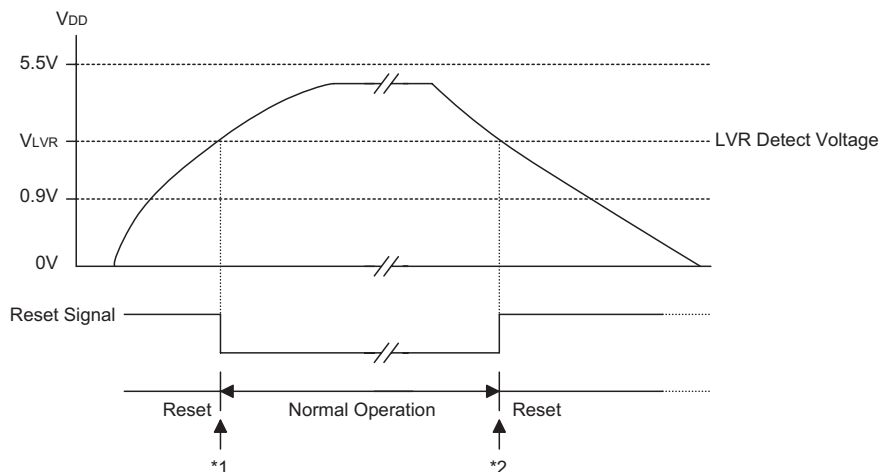
- Serial clock (SCL)  
The SCL input is used for positive edge clock data into each EEPROM device and negative edge clock data out of each device.
- Serial data (SDA)  
The SDA pin is bidirectional for serial data transfer. The pin is open-drain driven and may be wired-OR with any number of other open-drain or open collector devices.

**Memory Organization**

- 1K Serial EEPROM  
Internally organized with 128 8-bit words, the 1K requires an 8-bit data word address for random word addressing.

**Device Operations**

- Clock and data transition  
Data transfer may be initiated only when the bus is not busy. During data transfer, the data line must remain stable whenever the clock line is high. Changes in data line while the clock line is high will be interpreted as a START or STOP condition.
- Start condition  
A high-to-low transition of SDA with SCL high is a start condition which must precede any other command (refer to Start and Stop Definition Timing diagram).
- Stop condition  
A low-to-high transition of SDA with SCL high is a stop condition. After a read sequence, the stop command will place the EEPROM in a standby power mode (refer to Start and Stop Definition Timing Diagram).



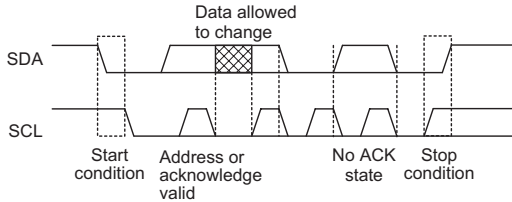
**Low Voltage Reset**

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 for over 1ms in its original state, therefore there's a 1ms delay before entering the reset mode

• Acknowledge

All addresses and data words are serially transmitted to and from the EEPROM in 8-bit words. The EEPROM sends a zero to acknowledge that it has received each word. This happens during the ninth clock cycle.



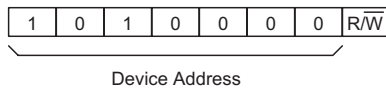
**Device Addressing**

The 1K EEPROM devices all require an 8-bit device address word following a start condition to enable the chip for a read or write operation. The device address word consist of a mandatory one, zero sequence for the first four most significant bits (refer to the diagram showing the Device Address). This is common to all the EEPROM device.

The next three bits are the fixed to be "0".

The 8th bit of device address is the read/write operation select bit. A read operation is initiated if this bit is high and a write operation is initiated if this bit is low.

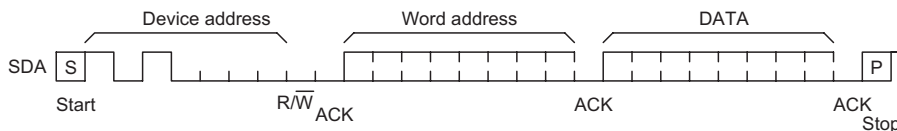
If the comparison of the device address succeed the EEPROM will output a zero at ACK bit. If not, the chip will return to a standby state.



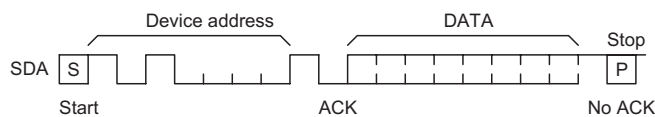
**Write Operations**

• Byte write

A write operation requires an 8-bit data word address following the device address word and acknowledgment. Upon receipt of this address, the EEPROM will again respond with a zero and then clock in the first 8-bit data word. After receiving the 8-bit data word, the EEPROM will output a zero and the addressing de-



**Byte Write Timing**

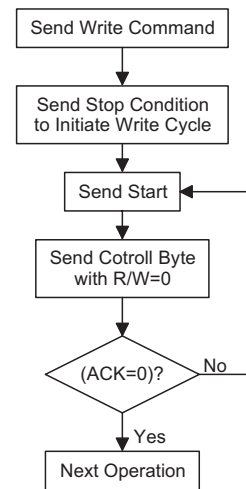


**Current Read Timing**

vice, such as a microcontroller, must terminate the write sequence with a stop condition. At this time the EEPROM enters an internally-timed write cycle to the non-volatile memory. All inputs are disabled during this write cycle and EEPROM will not respond until the write is completed (refer to Byte write timing).

• Acknowledge polling

To maximise bus throughput, one technique is to allow the master to poll for an acknowledge signal after the start condition and the control byte for a write command have been sent. If the device is still busy implementing its write cycle, then no ACK will be returned. The master can send the next read/write command when the ACK signal has finally been received.



**Acknowledge Polling Flow**

• Read operations

The data EEPROM supports three read operations, namely, current address read, random address read and sequential read. During read operation execution, the read/write select bit should be set to "1".

• Current address read

The internal data word address counter maintains the last address accessed during the last read or write operation, incremented by one. This address stays valid

between operations as long as the chip power is maintained. The address roll over during read from the last byte of the last memory page to the first byte of the first page. The address roll over during write from the last byte of the current page to the first byte of the same page. Once the device address with the read/write select bit set to one is clocked in and acknowledged by the EEPROM, the current address data word is serially clocked out. The microcontroller should respond a No ACK (High) signal and following stop condition (refer to Current read timing).

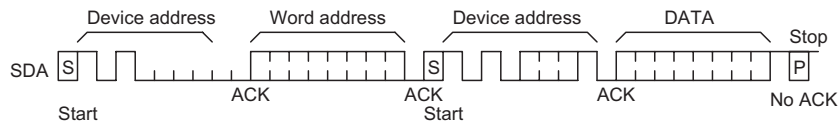
• Random read

A random read requires a dummy byte write sequence to load in the data word address which is then clocked in and acknowledged by the EEPROM. The microcontroller must then generate another start condition. The microcontroller now initiates a current address read by sending a device address with the

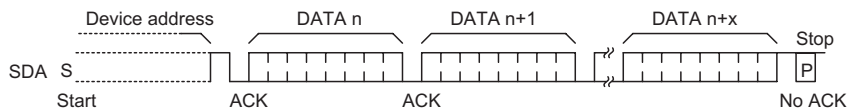
read/write select bit high. The EEPROM acknowledges the device address and serially clocks out the data word. The microcontroller should respond with a "no ACK" signal (high) followed by a stop condition. (refer to Random read timing).

• Sequential read

Sequential reads are initiated by either a current address read or a random address read. After the microcontroller receives a data word, it responds with an acknowledgment. As long as the EEPROM receives an acknowledgment, it will continue to increment the data word address and serially clock out sequential data words. When the memory address limit is reached, the data word address will roll over and the sequential read continues. The sequential read operation is terminated when the microcontroller responds with a "no ACK" signal (high) followed by a stop condition.

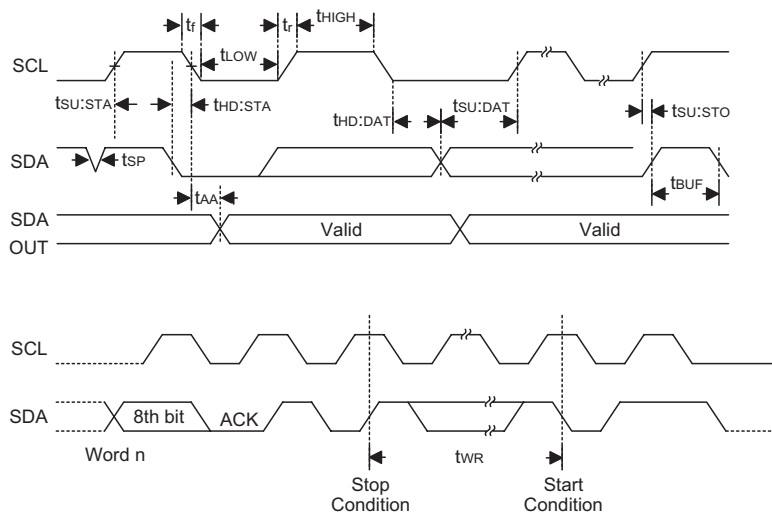


Random Read Timing



Sequential Read Timing

Data EEPROM Timing Diagrams



Note: The write cycle time  $t_{WR}$  is the time from a valid stop condition of a write sequence to the end of the valid start condition of sequential command.

### Suspend Wake-Up and Remote Wake-Up

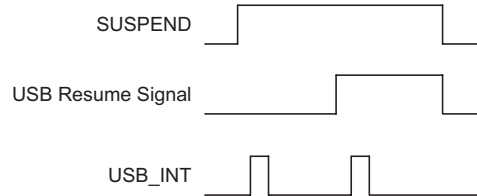
If there is no signal on the USB bus for over 3ms, the HT82K95EE/HT82K95AE will go into suspend mode. The Suspend line (bit 0 of the USC) will be set to "1" and a USB interrupt is triggered to indicate that the HT82K95EE/HT82K95AE should jump to the suspend state to meet the 500 $\mu$ A USB suspend current spec.

In order to meet the 500 $\mu$ A suspend current, the firmware should disable the USB clock by clearing the USBCKEN (bit3 of the SCC) to "0". The suspend current is 400 $\mu$ A.

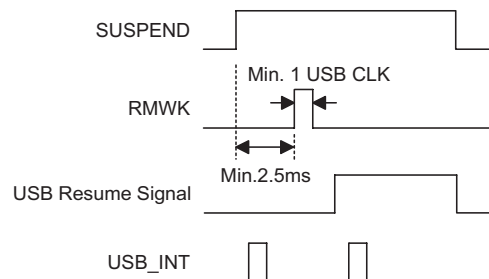
User can further decrease the suspend current to 250 $\mu$ A by setting the SUSP2 (bit4 of the SCC). If in USB mode set this bit LVR OPT must disable

When the resume signal is sent out by the host, the HT82K95EE/HT82K95AE will wake up the MCU by USB interrupt and the Resume line (bit 3 of the USC) is set. In order to make the HT82K95EE/HT82K95AE function properly, the firmware must set the USBCKEN (bit 3 of the SCC) to "1" and clear the SUSP2 (bit4 of the SCC). Since the Resume signal will be cleared before the Idle signal is sent out by the host, the Suspend line (bit 0 of the USC) will be set to "0". So when the MCU is detecting the Suspend line (bit0 of USC), the Resume line should be remembered and taken into consideration.

After finishing the resume signal, the suspend line will go inactive and a USB interrupt is triggered. The following is the timing diagram.



The device with remote wake up function can wake up the USB Host by sending a wake-up pulse through RMWK (bit 1 of the USC). Once the USB Host receives a wake-up signal from the HT82K95EE/HT82K95AE, it will send a Resume signal to the device. The timing is as follows:



### To Configure the HT82K95EE/HT82K95AE as PS2 Device

The HT82K95EE/HT82K95AE can be configured as a USB interface or PS2 interface device, by configuring the SPS2 (bit 4 of the USR) and SUSB (bit 5 of the USR). If SPS2=1, and SUSB=0, the HT82K95EE/HT82K95AE is configured as a PS2 interface, pin USBD- is configured as a PS2 Data pin and USBD+ is configured as a PS2 Clk pin. User can easily read or write to the PS2 Data or PS2 Clk pin by accessing the corresponding bit PS2DAI (bit 4 of the USC), PS2CKI (bit 5 of the USC), PS2DAO (bit 6 of the USC) and S2CKO (bit 7 of the USC) respectively.

User should make sure that in order to read the data properly, the corresponding output bit must be set to "1". For example, if it is desired to read the PS2 Data by reading PS2DAI, the PS2DAO should set to "1". Otherwise it is always read as "0".

If SPS2=0, and SUSB=1, the HT82K95EE/HT82K95AE is configured as a USB interface. Both the USBD- and USBD+ is driven by the SIE of the HT82K95E/HT82K95A. User can only write or read the USB data through the corresponding FIFO.

Both SPS2 and SUSB default is "0".

### USB Interface

There are ten registers, including PIPE\_CTRL (41H in bank 1), AWR (address + remote wake-up 42H in bank 1), STALL (43H in bank 1), PIPE (44H in bank 1), SIES (45H in bank 1), MISC (46H in bank 1), Endpt\_EN (47H in bank 1), FIFO0 (48H in bank 1), FIFO1 (49H in bank 1), and FIFO2 (4AH in bank 1) used for the USB function. AWR register contains current address and a remote wake up function control bit. The initial value of AWR is "00H". The address value extracted from the USB command is not to be loaded into this register until the SETUP stage is completed.

Bit No.	Label	R/W	Function
0	WKEN	W	Remote wake-up enable/disable
7~1	AD6~AD0	W	USB device address

### AWR (42H) Register

### STALL and PIPE, PIPE\_CTRL, Endpt\_EN Registers

PIPE register represents whether the endpoint corresponding is accessed by host or not. After ACT\_EN signal being sent out, MCU can check which endpoint had been accessed. This register is set only after the time when host access the corresponding endpoint.

STALL register shows whether the endpoint corresponding works or not. As soon as the endpoint work improperly, the bit corresponding must be set.

PIPE\_CTRL Register is used for configuring IN (Bit=1) or OUT (Bit=0) Pipe. The default is define IN pipe. Where Bit0 (DATA0) of the PIPE\_CTRL Register is used to setting the data toggle of any endpoint (except endpoint0) using data toggles to the value DATA0. Once the user want the any endpoint (except endpoint0) using data toggles to the value DATA0. the user can output a LOW pulse to this bit. The LOW pulse period must at least 10 instruction cycle.

Endpt\_EN Register is used to enable or disable the corresponding endpoint (except endpoint 0) Enable Endpoint (Bit=1) or disable Endpoint (Bit=0)

The bitmaps are list as follows:

Register Name	R/W	Register Address	Bit7~Bit3 Reserved	Bit 2	Bit 1	Bit 0	Default Value
PIPE_CTRL	R/W	01000001B	—	Pipe 2	Pipe 1	Pipe 0	00000110
STALL	R/W	01000011B	—	Pipe 2	Pipe 1	Pipe 0	00000110
PIPE	R	01000100B	—	Pipe 2	Pipe 1	Pipe 0	00000000
Endpt_EN	R/W	01000001B	—	Pipe 2	Pipe 1	Pipe 0	00000111

**PIPE\_CTRL (41H), STALL (43H), PIPE (44H) and Endpt\_EN (47H) Registers**

The SIES Register is used to indicate the present signal state which the SIE receives and also defines whether the SIE has to change the device address automatically.

Bit No.	Function	Read/Write	Register Address
7	MNI	R/W	01000001B
6~2	—	—	
1	F0_ERR	R/W	
0	Adr_set	R/W	

**SIES (45H) Register Table**

Func. Name	R/W	Description
Adr_set	R/W	This bit is used to configure the SIE to automatically change the device address with the value of the Address+Remote_WakeUp Register (42H). When this bit is set to "1" by F/W, the SIE will update the device address with the value of the Address+Remote_WakeUp Register (42H) after the PC Host has successfully read the data from the device by the IN operation. The SIE will clear the bit after updating the device address. Otherwise, when this bit is cleared to "0", the SIE will update the device address immediately after an address is written to the Address+Remote_WakeUp Register (42H) Default 0
F0_Err	R/W	This bit is used to indicate that some errors have occurred when accessing the FIFO0. This bit is set by SIE and cleared by F/W. Default 0
—	—	Unused bit, read as "0"
NMI	R/W	This bit is used to control whether the USB interrupt is output to the MCU in NAK response to the PC Host IN or OUT token. Only for Endpoint0 1: has only USB interrupt, data is transmitted to the PC host or data is received from the PC Host 0: always has USB interrupt if the USB accesses FIFO0 Default 0

**SIES Function Table**

MISC register combines a command and status to control desired endpoint FIFO action and to show the status of the desired endpoint FIFO. The MISC will be cleared by USB reset signal.

Bit No.	Label	R/W	Function
0	REQ	R/W	After setting the other status of the desired one in the MISC, endpoint FIFO can be requested by setting this bit to "1". After the job has been done, this bit has to be cleared to "0".
1	TX	R/W	This bit defines the direction of data transferring between MCU and endpoint FIFO. When the TX is set to "1", this means that the MCU wants to write data to the endpoint FIFO. After the job has been done, this bit has to be cleared to "0" before terminating request to represent the end of transferring. For reading action, this bit has to be cleared to "0" to represent that MCU wants to read data from the endpoint FIFO and has to be set to "1" after the job is done.
2	CLEAR	R/W	Clear the requested endpoint FIFO, even if the endpoint FIFO is not ready.
4 3	SELP1 SELP0	R/W	Defines which endpoint FIFO is selected, SELP1,SELP0: 00: endpoint FIFO0 01: endpoint FIFO1 10: endpoint FIFO2 11: reserved
5	SCMD	R/W	Used to show that the data in endpoint FIFO is a SETUP command. This bit has to be cleared by firmware. That is to say, even the MCU is busy, the device will not miss any SETUP commands from the host.
6	READY	R	Read only status bit, this bit is used to indicate that the desired endpoint FIFO is ready to work.
7	LEN0	R/W	Used to indicate that a 0-sized packet is sent from a host to the MCU. This bit should be cleared by firmware.

#### MISC (46H) Register

The MCU can communicate with the endpoint FIFO by setting the corresponding registers, of which address is listed in the following table. After reading the current data, next data will show after 2 $\mu$ s, used to check the endpoint FIFO status and response to MISC register, if read/write action is still going on.

Registers	R/W	Bank	Address	Bit7~Bit0
FIFO0	R/W	1	48H	Data7~Data0
FIFO1	R/W	1	49H	Data7~Data0
FIFO2	R/W	1	4AH	Data7~Data0

There are some timing constrains and usages illustrated here. By setting the MISC register, MCU can perform reading, writing and clearing actions. There are some examples shown in the following table for endpoint FIFO reading, writing and clearing.

Actions	MISC Setting Flow and Status
Read FIFO0 sequence	00H→01H→delay 2 $\mu$ s, check 41H→read* from FIFO0 register and check not ready (01H)→03H→02H
Write FIFO1 sequence	0AH→0BH→delay 2 $\mu$ s, check 4BH→write* to FIFO1 register and check not ready (0BH)→09H→08H
Check whether FIFO0 can be read or not	00H→01H→delay 2 $\mu$ s, check 41H (ready) or 01H (not ready)→00H
Check whether FIFO1 can be written or not	0AH→0BH→delay 2 $\mu$ s, check 4BH (ready) or 0BH (not ready)→0AH
Read 0-sized packet sequence form FIFO0	00H→01H→delay 2 $\mu$ s, check 81H→read once (01H)→03H→02H
Write 0-sized packet sequence to FIFO1	0AH→0BH→delay 2 $\mu$ s, check 0BH→0FH→0DH→08H

Note: \*: There are 2 $\mu$ s existing between 2 reading action or between 2 writing action

The definitions of the USB/PS2 status and control register (USC; 1AH) are as shown.

Bit No.	Label	R/W	Function
0	SUSP	R	Read only, USB suspend indication. When this bit is set to "1" (set by SIE), it indicates the USB bus enters suspend mode. The USB interrupt is also triggered on any changes of this bit.
1	RMWK	W	USB remote wake up command. It is set by MCU to force the USB host leaving the suspend mode. When this bit is set to "1", 2 $\mu$ s delay for clearing this bit to "0" is needed to insure the RMWK command is accepted by SIE.
2	URST	R/W	USB reset indication. This bit is set/cleared by USB SIE. This bit is used to detect which bus (PS2 or USB) is attached. When the URST is set to "1", this indicates that a USB reset has occurred (the attached bus is USB) and a USB interrupt will be initialized.
3	RESUME	R	USB resume indication. When the USB leaves the suspend mode, this bit is set to "1" (set by SIE). This bit will appear 20ms waiting for the MCU to detect. When the RESUME is set by the SIE, an interrupt will be generated to wake-up the MCU. In order to detect the suspend state, the MCU should set the USBCKEN and clear SUSP2 (in SCC register) to enable the SIE detecting function. The RESUME will be cleared while the SUSP is going "0". When the MCU is detecting the SUSP, the RESUME (wakes-up the MCU ) should be remembered and taken into consideration.
4	PS2DAI	R	Read only, USBD-/DATA input
5	PS2CKI	R	Read only, USBD+/CLK input
6	PS2DAO	W	Data for driving the USBD-/DATA pin when working under 3D PS2 mouse function. (Default="1")
7	PS2CKO	W	Data for driving the USBD+/CLK pin when working under 3D PS2 mouse function. (Default="1")

#### USC (1AH) Register

The USR (USB endpoint interrupt status register) register is used to indicate which endpoint is accessed and to select the serial bus (PS2 or USB). The endpoint request flags (EP0IF, EP1IF and EP2IF) are used to indicate which endpoints are accessed. If an endpoint is accessed, the related endpoint request flag will be set to "1" and the USB interrupt will occur (if the USB interrupt is enabled and the stack is not full). When the active endpoint request flag is served, the endpoint request flag has to be cleared to "0".

Bit No.	Label	R/W	Function
0	EP0IF	R/W	When this bit is set to "1" (set by the SIE), it indicates the endpoint 0 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
1	EP1IF	R/W	When this bit is set to "1" (set by the SIE), it indicates the endpoint 1 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
2	EP2IF	R/W	When this bit is set to "1" (set by the SIE), it indicates the endpoint 2 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
3, 6	—	—	Reserved
4	SPS2	R/W	The PS2 function is selected when this bit is set to "1". (Default="0")
5	SUSB	R/W	The USB function is selected when this bit is set to "1". (Default="0")
7	USB_flag	R/W	This flag is used to show the MCU is in USB mode. (Bit=1) This bit is R/W by FW and will be cleared to "0" after power-on reset. (Default="0")

#### USR (1BH) Register

There is a system clock control register implemented to select the clock used in the MCU. This register consists of the USB clock control bit (USBCKEN), second suspend mode control bit (SUSP2) and system clock selection (SYSCLK).

Bit No.	Label	R/W	Function
2~0, 7	—	—	Undefined, should be cleared to "0"
3	USBCKEN	R/W	USB clock control bit. When this bit is set to "1", it indicates that the USB clock is enabled. Otherwise, the USB clock is turned-off. (Default="0")
4	SUSP2	R/W	This bit is used to reduce power consumption in the suspend mode. In the normal mode this bit must be cleared to zero (Default=0). In the HALT mode this bit should be set high to reduce power consumption. If in USB mode set this bit LVR OPT must disable
5	PS2_flag	R/W	This flag is used to show the MCU is under PS2 mode. (Bit=1) This bit is R/W by FW and will be cleared to "0" after power-on reset. (Default="0")
6	SYSCLK	R/W	This bit is used to specify the system oscillator frequency used by the MCU. If a 6MHz crystal oscillator or resonator is used, this bit should be set to "1". If a 12MHz crystal oscillator or resonator is used, this bit should be cleared to "0" (default).

#### SCC (1CH) Register

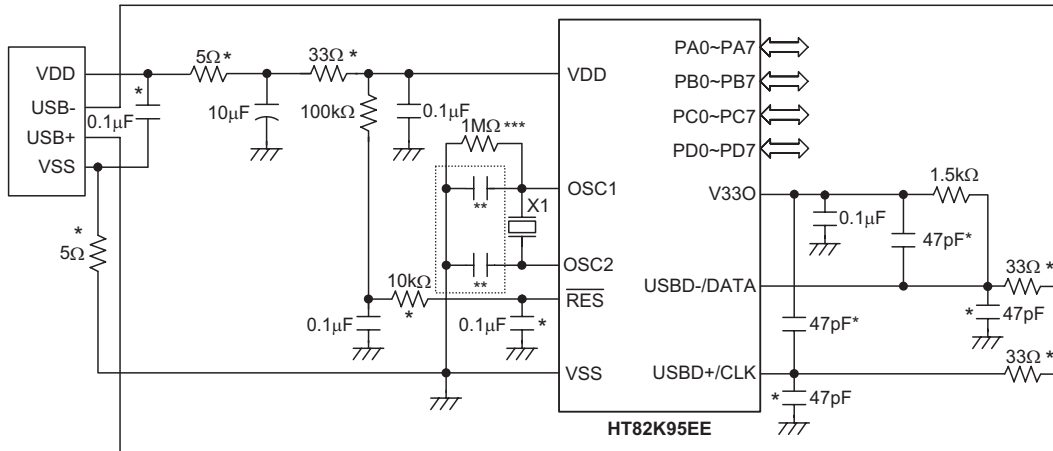
**Table High Byte Pointer for Current Table Read TBHP (Address 0X1F)**

Register	Bits	Labels	Read/Write	Option	Functions
TBHP (0X1F)	3~0	PGC3~PGC0	R	—	Store current table read bit11~bit8 data

#### Options

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

No.	Option
1	Chip lock bit (by bit)
2	PA0~PA7 pull-high resistor enabled or disabled (by bit)
3	PB0~PB7 pull-high resistor enabled or disabled (by nibble)
4	PC0~PC7 pull-high resistor enabled or disabled (by nibble)
5	PD0~PD7 pull-high resistor enabled or disabled (by nibble)
6	LVR enable or disable
7	WDT enable or disable
8	WDT clock source: $f_{SYS}/4$ or WDTOSC
9	"CLRWDT" instruction(s): 1 or 2
10	PA0~PA7 output structures: CMOS/NMOS open-drain/PMOS open-drain (by bit)
11	PA0~PA7 wake-up enabled or disabled (by bit)
12	PB0~PB7 wake-up enabled or disabled (by nibble)
13	PC0~PC7 wake-up enabled or disabled (by nibble)
14	PD0~PD7 wake-up enabled or disabled (by nibble)
15	TBHP enable or disable (default disable)

**Application Circuits**
**Crystal or Ceramic Resonator for Multiple I/O Applications for HT82K95EE**


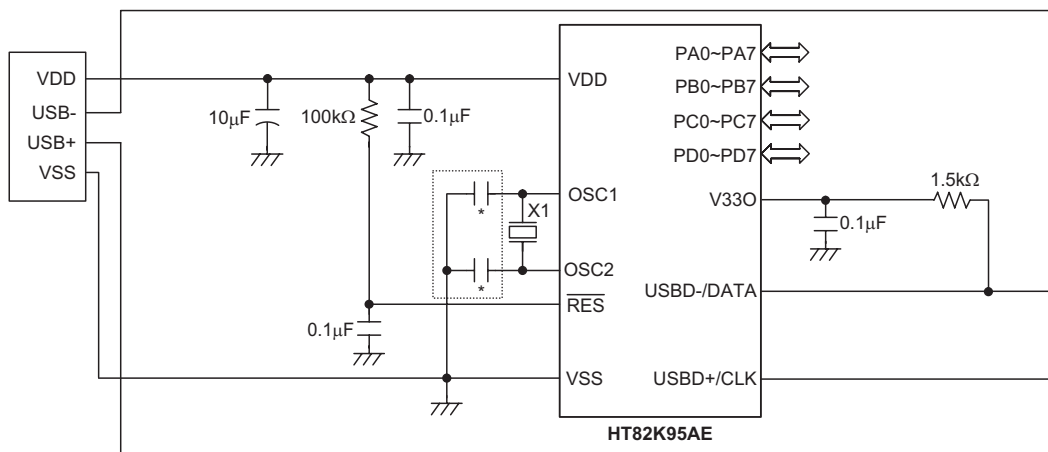
Note: The resistance and capacitance for reset circuit should be designed in such a way as to ensure that the VDD is stable and remains within a valid operating voltage range before bringing  $\overline{\text{RES}}$  to high.

X1 can use 6MHz or 12MHz, X1 as close OSC1 & OSC2 as possible.

Components with \* are used for EMC issue.

Components with \*\* are used for resonator only if necessary.

Components with \*\*\* are used for 12MHz application.

**Crystal or Ceramic Resonator for Multiple I/O Applications for HT82K95AE**


Note: X1 can use 6MHz or 12MHz, X1 as close OSC1 & OSC2 as possible.

Components with \* are used for resonator only if necessary.

## Instruction Set

### Introduction

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

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

### Instruction Timing

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

### Moving and Transferring Data

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

### Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and

subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

### Logical and Rotate Operations

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

### Branches and Control Transfer

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

### Bit Operations

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

### Table Read Operations

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

### Other Operations

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

### Instruction Set Summary

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

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0-7 number of bits

addr: Program memory address

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

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

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

**Instruction Definition**

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

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

<b>CPL [m]</b>	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	$[m] \leftarrow \overline{[m]}$
Affected flag(s)	Z
<b>CPLA [m]</b>	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow \overline{[m]}$
Affected flag(s)	Z
<b>DAA [m]</b>	Decimal-Adjust ACC for addition with result in Data Memory
Description	Convert the contents of the Accumulator value to a BCD ( Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.
Operation	$[m] \leftarrow ACC + 00H$ or $[m] \leftarrow ACC + 06H$ or $[m] \leftarrow ACC + 60H$ or $[m] \leftarrow ACC + 66H$
Affected flag(s)	C
<b>DEC [m]</b>	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	$[m] \leftarrow [m] - 1$
Affected flag(s)	Z
<b>DECA [m]</b>	Decrement Data Memory with result in ACC
Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow [m] - 1$
Affected flag(s)	Z
<b>HALT</b>	Enter power down mode
Description	This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.
Operation	TO $\leftarrow$ 0 PDF $\leftarrow$ 1
Affected flag(s)	TO, PDF

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

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

<b>RLC [m]</b>	Rotate Data Memory left through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.
Operation	$[m].(i+1) \leftarrow [m].i; (i = 0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
Affected flag(s)	C
<b>RLCA [m]</b>	Rotate Data Memory left through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC.(i+1) \leftarrow [m].i; (i = 0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
Affected flag(s)	C
<b>RR [m]</b>	Rotate Data Memory right
Description	The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.
Operation	$[m].i \leftarrow [m].(i+1); (i = 0\sim6)$ $[m].7 \leftarrow [m].0$
Affected flag(s)	None
<b>RRA [m]</b>	Rotate Data Memory right with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC.i \leftarrow [m].(i+1); (i = 0\sim6)$ $ACC.7 \leftarrow [m].0$
Affected flag(s)	None
<b>RRC [m]</b>	Rotate Data Memory right through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.
Operation	$[m].i \leftarrow [m].(i+1); (i = 0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C
<b>RRCA [m]</b>	Rotate Data Memory right through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC.i \leftarrow [m].(i+1); (i = 0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C

<b>SBC A,[m]</b>	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C
<b>SBCM A,[m]</b>	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[m] \leftarrow ACC - [m] - \bar{C}$
Affected flag(s)	OV, Z, AC, C
<b>SDZ [m]</b>	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] - 1$ Skip if $[m] = 0$
Affected flag(s)	None
<b>SDZA [m]</b>	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m] - 1$ Skip if $ACC = 0$
Affected flag(s)	None
<b>SET [m]</b>	Set Data Memory
Description	Each bit of the specified Data Memory is set to 1.
Operation	$[m] \leftarrow FFH$
Affected flag(s)	None
<b>SET [m].i</b>	Set bit of Data Memory
Description	Bit i of the specified Data Memory is set to 1.
Operation	$[m].i \leftarrow 1$
Affected flag(s)	None

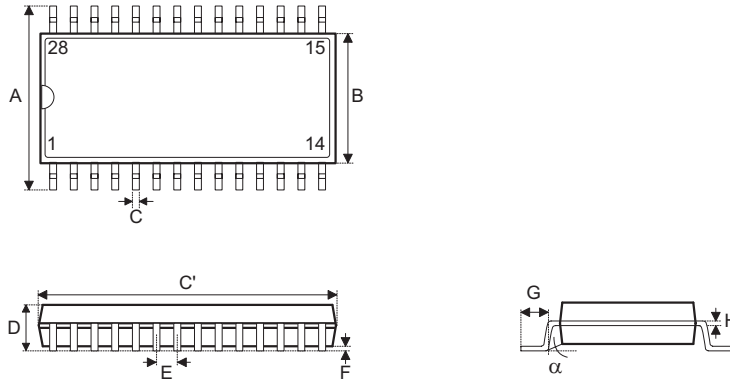
<b>SIZ [m]</b>	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] + 1$ Skip if $[m] = 0$
Affected flag(s)	None
<b>SIZA [m]</b>	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m] + 1$ Skip if $ACC = 0$
Affected flag(s)	None
<b>SNZ [m].i</b>	Skip if bit i of Data Memory is not 0
Description	If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if $[m].i \neq 0$
Affected flag(s)	None
<b>SUB A,[m]</b>	Subtract Data Memory from ACC
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C
<b>SUBM A,[m]</b>	Subtract Data Memory from ACC with result in Data Memory
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[m] \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C
<b>SUB A,x</b>	Subtract immediate data from ACC
Description	The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - x$
Affected flag(s)	OV, Z, AC, C

<b>SWAP [m]</b>	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
Affected flag(s)	None
<b>SWAPA [m]</b>	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
Affected flag(s)	None
<b>SZ [m]</b>	Skip if Data Memory is 0
Description	If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if $[m] = 0$
Affected flag(s)	None
<b>SZA [m]</b>	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m]$ Skip if $[m] = 0$
Affected flag(s)	None
<b>SZ [m].i</b>	Skip if bit i of Data Memory is 0
Description	If bit i of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	Skip if $[m].i = 0$
Affected flag(s)	None
<b>TABRDC [m]</b>	Read table (current page) to TBLH and Data Memory
Description	The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	$[m] \leftarrow$ program code (low byte) $TBLH \leftarrow$ program code (high byte)
Affected flag(s)	None
<b>TABRDL [m]</b>	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	$[m] \leftarrow$ program code (low byte) $TBLH \leftarrow$ program code (high byte)
Affected flag(s)	None

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

**Package Information**

**28-pin SOP (300mil) Outline Dimensions**

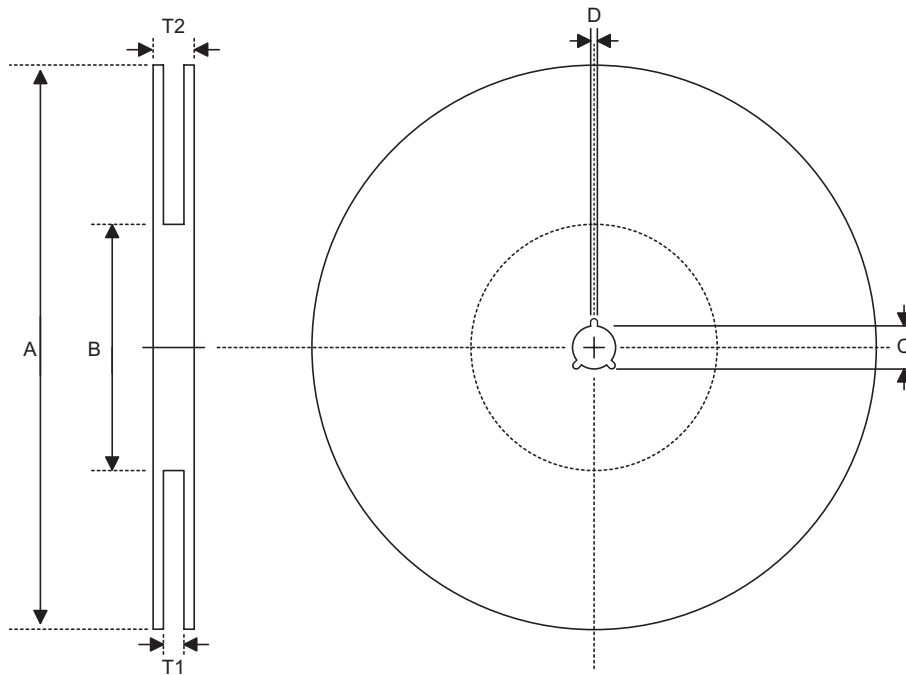


• MS-013

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
$\alpha$	0°	—	8°

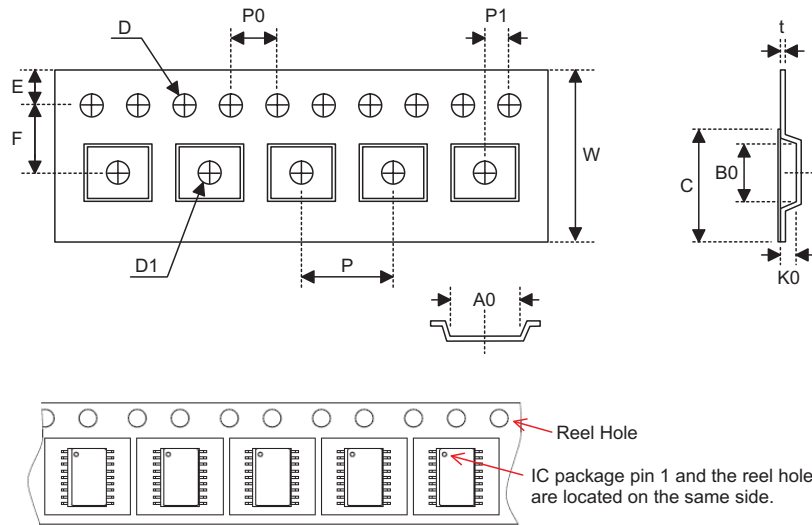
**Product Tape and Reel Specifications**

**Reel Dimensions**



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 <sup>+0.5/-0.2</sup>
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	24.8 <sup>+0.3/-0.2</sup>
T2	Reel Thickness	30.2±0.2

**Carrier Tape Dimensions**


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 <sup>+0.1/-0.0</sup>
D1	Cavity Hole Diameter	1.50 <sup>+0.25/-0.00</sup>
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	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

**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)**

G Room, 3 Floor, No.1 Building, No.2016 Yi-Shan Road, Minhang District, Shanghai, China 201103  
Tel: 86-21-5422-4590  
Fax: 86-21-5422-4705  
<http://www.holtek.com.cn>

**Holtek Semiconductor Inc. (Shenzhen Sales Office)**

5F, Unit A, Productivity Building, Gaoxin M 2nd, Middle Zone Of High-Tech Industrial Park, ShenZhen, China 518057  
Tel: 86-755-8616-9908, 86-755-8616-9308  
Fax: 86-755-8616-9722

**Holtek Semiconductor Inc. (Beijing Sales Office)**

Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031  
Tel: 86-10-6641-0030, 86-10-6641-7751, 86-10-6641-7752  
Fax: 86-10-6641-0125

**Holtek Semiconductor Inc. (Chengdu Sales Office)**

709, Building 3, Champagne Plaza, No.97 Dongda Street, Chengdu, Sichuan, China 610016  
Tel: 86-28-6653-6590  
Fax: 86-28-6653-6591

**Holtek Semiconductor (USA), Inc. (North America Sales Office)**

46729 Fremont Blvd., Fremont, CA 94538, USA  
Tel: 1-510-252-9880  
Fax: 1-510-252-9885  
<http://www.holtek.com>

Copyright © 2008 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>.