HT82K96E

USB Multimedia Keyboard Encoder 8-Bit OTP MCU

Technical Document

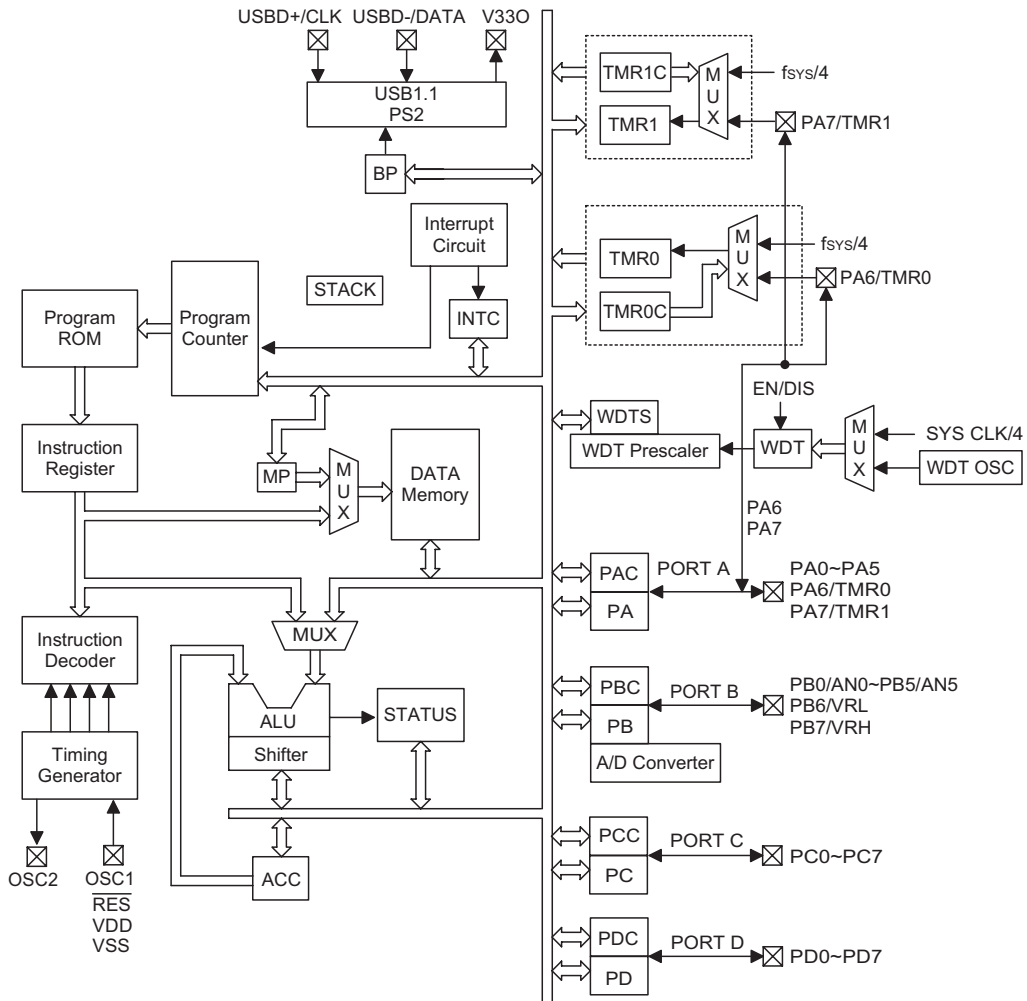- Tools Information
- FAQs
- Application Note

Features

- Operating voltage:
  f_{SYS}=6M/12MHz: 4.4V~5.5V
- Low voltage reset function
- 32 bidirectional I/O lines (max.)
- 8-bit programmable timer/event counter with over-flow interrupt
- 16-bit programmable timer/event counter and over-flow interrupts
- Crystal oscillator (6MHz or 12MHz)
- Watchdog Timer
- 6 channels 8-bit A/D converter
- PS2 and USB modes supported
- USB 2.0 low speed function

- 4 endpoints supported (endpoint 0 included)
- 4096×15 program memory ROM
- 160×8 data memory RAM
- HALT function and wake-up feature reduce power consumption
- 8-level subroutine nesting
- Up to 0.33μs 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
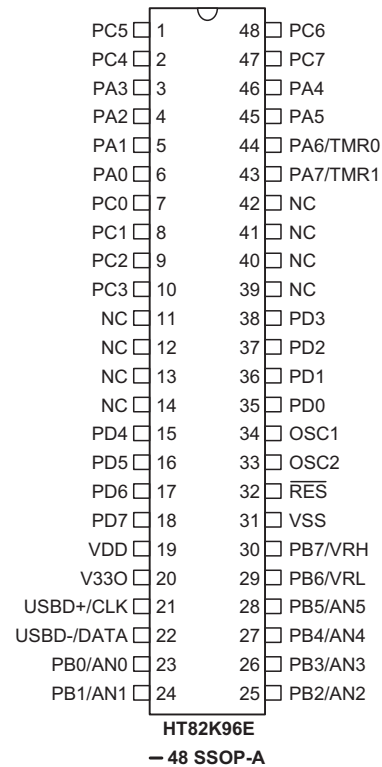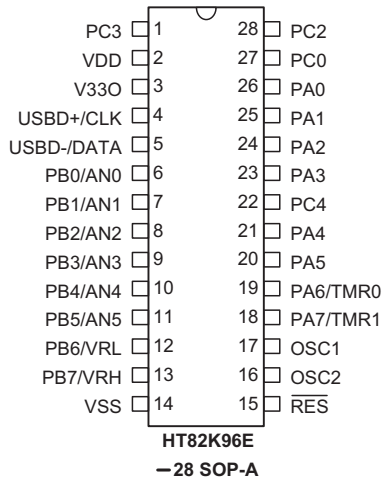- 28-pin SOP, 48-pin SSOP package

General Description

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

Block Diagram

Pin Assignment

```
              ┌───────┐                          ┌───────┐
        PC5 ┤ 1    48 ┤ PC6
        PC4 ┤ 2    47 ┤ PC7
        PA3 ┤ 3    46 ┤ PA4
        PA2 ┤ 4    45 ┤ PA5
        PA1 ┤ 5    44 ┤ PA6/TMR0
        PA0 ┤ 6    43 ┤ PA7/TMR1
        PC0 ┤ 7    42 ┤ NC
        PC1 ┤ 8    41 ┤ NC
        PC2 ┤ 9    40 ┤ NC
        PC3 ┤ 10   39 ┤ NC
         NC ┤ 11   38 ┤ PD3
         NC ┤ 12   37 ┤ PD2
         NC ┤ 13   36 ┤ PD1
         NC ┤ 14   35 ┤ PD0
        PD4 ┤ 15   34 ┤ OSC1
        PD5 ┤ 16   33 ┤ OSC2
        PD6 ┤ 17   32 ┤ RES
        PD7 ┤ 18   31 ┤ VSS
        VDD ┤ 19   30 ┤ PB7/VRH
       V33O ┤ 20   29 ┤ PB6/VRL
   USBD+/CLK ┤ 21  28 ┤ PB5/AN5
  USBD-/DATA ┤ 22  27 ┤ PB4/AN4
     PB0/AN0 ┤ 23  26 ┤ PB3/AN3
     PB1/AN1 ┤ 24  25 ┤ PB2/AN2
              └───────┘
```

PC3 ┤ 1 28 ┤ PC2
VDD ┤ 2 27 ┤ PC0
V33O ┤ 3 26 ┤ PA0
USBD+/CLK ┤ 4 25 ┤ PA1
USBD-/DATA ┤ 5 24 ┤ PA2
PB0/AN0 ┤ 6 23 ┤ PA3
PB1/AN1 ┤ 7 22 ┤ PC4
PB2/AN2 ┤ 8 21 ┤ PA4
PB3/AN3 ┤ 9 20 ┤ PA5
PB4/AN4 ┤ 10 19 ┤ PA6/TMR0
PB5/AN5 ┤ 11 18 ┤ PA7/TMR1
PB6/VRL ┤ 12 17 ┤ OSC1
PB7/VRH ┤ 13 16 ┤ OSC2
VSS ┤ 14 15 ┤ RES

HT82K96E
－28 SOP-A

HT82K96E
－48 SSOP-A

Pin Description

Pin Name	I/O	ROM Code Option	Description
PA0~PA5 PA6/TMR0 PA7/TMR1	I/O	Pull-low 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 Pull-low resistor options: PA0~PA5 CMOS/NMOS/PMOS options: PA0~PA7 Wake up options: PA0~PA7 PA6 and PA7 are pin-shared with TMR0 and TMR1 input, respectively. PA0~PA5 can be used as USB mouse X1, X2, Y1, Y2, Z1, Z2 input for mouse hardware wake-up function PA6, PA7 can be used as USB mouse button input for mouse hardware wake-up function
PB0/AN0 PB1/AN1 PB2/AN2 PB3/AN3 PB4/AN4 PB5/AN5 PB6/VRL PB7/VRH	I/O	Pull-high Analog input	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). The PB can be used as analog input of the analog to digital converter (determined by options). PB6, PB7 can be used as USB mouse button input for mouse Hardware wake-up function
PD0~PD7	I/O	Pull-high	Bidirectional I/O lines. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by 1-bit pull-high option). PD4 can be used as USB mouse button input for mouse hardware wake-up function

Pin Name	I/O	ROM Code Option	Description
VSS	—	—	Negative power supply, ground
PC0~PC7	I/O	Pull-high	Bidirectional I/O lines. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options). PC0 can be used as USB mouse IRPT control pin for mouse hardware wake-up function
$\overline{RES}$	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 an 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

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

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

Operating Temperature ....0°C to 70°C

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=6MHz	4.4	—	5.5	V
			f_{SYS}=12MHz	4.4	—	5.5	V
I_{DD1}	Operating Current (6MHz Crystal)	5V	No load, f_{SYS}=6MHz	—	6.5	12	mA
I_{DD2}	Operating Current (12MHz Crystal)	5V	No load, f_{SYS}=12MHz	—	7.5	16	mA
I_{STB1}	Standby Current (WDT Enabled)	5V	No load, system HALT, USB suspend	—	—	250	μA
I_{STB2}	Standby Current (WDT Disabled)	5V	No load, system HALT, USB suspend	—	—	230	μA
V_{IL1}	Input Low Voltage for I/O Ports	5V	—	0	—	0.8	V
V_{IH1}	Input High Voltage for I/O Ports	5V	—	2	—	5	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	5V	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	5V	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL1}	I/O Port Sink Current for PB, PC1~PC7, PD	5V	V_{OL}=3.4V	12	17	—	mA
I_{OL2}	I/O Port Sink Current for PB, PC1~PC7, PD	5V	V_{OL}=0.4V	2	4	—	mA
I_{OL3}	I/O Port Sink Current for PA	5V	V_{OL}=0.4V	5	10	—	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_DD	Conditions				
I_{OL4}	I/O Port Sink Current for PC0	5V	V_{OL}=0.4V	10	25	—	mA
I_{OH1}	I/O Port Source Current for PC0	5V	V_{OH}=3.4V	−8	−16	—	mA
I_{OH2}	I/O Port Source Current for PA, PB, PC1~PC7, PD	5V	V_{OH}=3.4V	−2	−5	—	mA
R_{PH}	Pull-high Resistance for PA, PB, PC, PD	5V	—	25	50	80	kΩ
R_{PL}	Pull-low Resistance for PA1~PA5	5V	—	15	30	45	kΩ
V_{LVR}	Low Voltage Reset	—	—	3	3.4	4.0	V
V_{V33O}	3.3V Regulator Output	5V	I_{V33O}=−5mA	3.0	3.3	3.6	V
$E_{A/D}$	A/D Conversion Error	5V	Total error	—	1	2	LSB

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_DD	Conditions				
f_{SYS}	System Clock (Crystal OSC)	5V	—	6	—	12	MHz
f_{TIMER}	Timer I/P Frequency (TMR0/TMR1)	5V	—	0	—	12	MHz
t_{WDTOSC}	Watchdog Oscillator	5V	—	15	31	70	μs
t_{WDT1}	Watchdog Time-out Period (WDT OSC)	5V	Without WDT prescaler	4	8	16	ms
t_{WDT2}	Watchdog Time-out Period (System Clock)	—	Without WDT prescaler	—	1024	—	t_{SYS}
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	t_{SYS}
			Power-up, Watchdog Time-out from normal	—	1024	—	t_{WDTOSC}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{ADC}	A/D Conversion Time	—	—	—	64	—	$t_{A/D}$

Note: $t_{A/D}=\dfrac{1}{f_{A/D}}$, $f_{A/D}$=A/D clock source frequencies (6MHz, 3MHz, 1.5MHz, 0.75MHz)

Functional Description

Execution Flow

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

Instruction fetching and execution are pipelined in such a way that a fetch takes an instruction cycle while decoding and execution takes the next instruction cycle. However, the pipelining scheme causes each instruction to effectively execute 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.
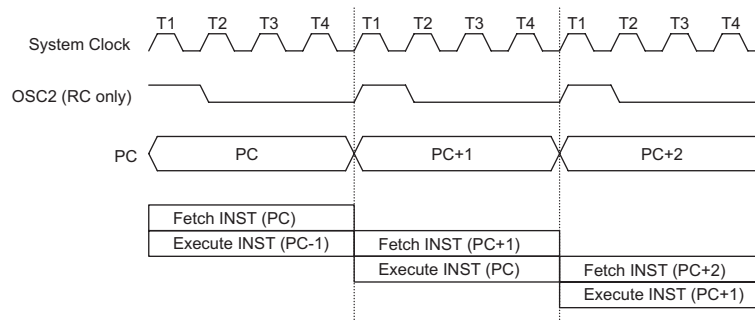
Execution Flow

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 .
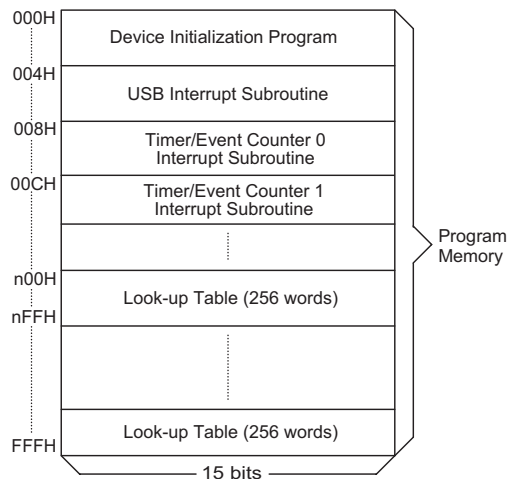
Note: n ranges from 0 to F

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. The instructions ″TABRDC [m]″ (the current page, one page=256 words) and ″TABRDL [m]″ (the last page) transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH (08H). Only the destination of the lower-order byte in the table is well-defined, the other bits of the table word are transferred to the lower portion of TBLH, and the remaining 1-bit words are read as ″0″. The Table Higher-order byte register (TBLH) is read only. The table pointer (TBLP) is a read/write register (07H), which indicates the table location. Before accessing the table, the location must be placed in the TBLP. The TBLH is read only and cannot be restored. If the main routine and the ISR (Interrupt Service Routine) both employ the table read instruction, the contents of the TBLH in the main routine are likely to be changed by the table read instruction used in the ISR. Errors can occur. In other words, using the table read instruction in the main routine and the ISR simultaneously should be avoided. However, if the table read instruction has to be applied in both the main routine and the ISR, the interrupt is supposed to be disabled prior to the table read instruction. It will not be enabled until the TBLH has been backed up. All table related instructions require two cycles to complete the operation. These areas may function as normal program memory depending upon the requirements.

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.

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   P11~P8: Current program counter bits
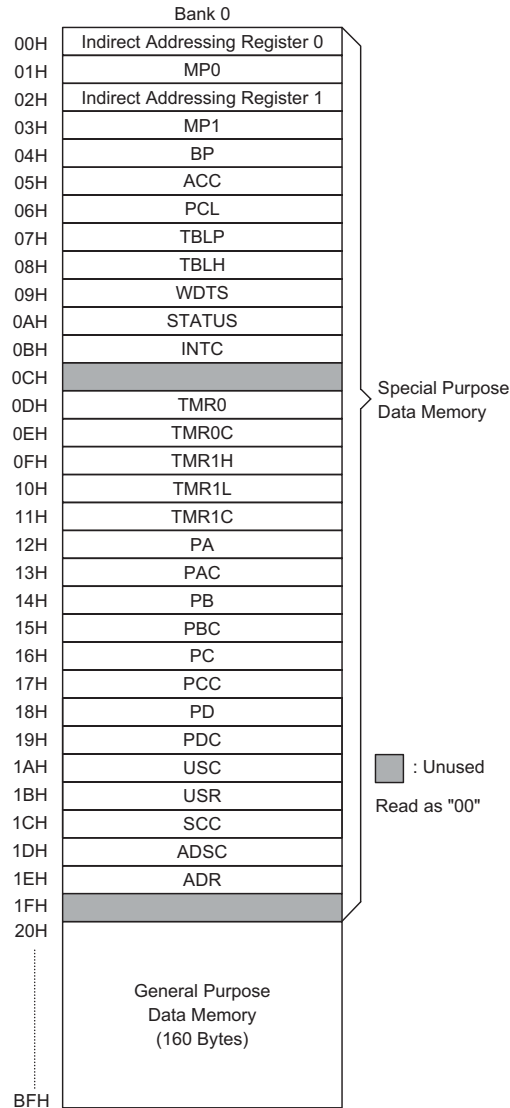
@7~@0: Table pointer bits

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), table higher-order byte register (TBLH;08H), status register (STATUS;0AH), interrupt control register (INTC;0BH), 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). A/D converter status and control register (ADSC;1DH) and A/D converter result register (ADR;1EH). The remaining space before the 20H is reserved for future expanded 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.

	Bank 0
00H	Indirect Addressing Register 0
01H	MP0
02H	Indirect Addressing Register 1
03H	MP1
04H	BP
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	WDTS
0AH	STATUS
0BH	INTC
0CH	
0DH	TMR0
0EH	TMR0C
0FH	TMR1H
10H	TMR1L
11H	TMR1C
12H	PA
13H	PAC
14H	PB
15H	PBC
16H	PC
17H	PCC
18H	PD
19H	PDC
1AH	USC
1BH	USR
1CH	SCC
1DH	ADSC
1EH	ADR
1FH	
20H	General Purpose Data Memory (160 Bytes)
BFH	

Special Purpose Data Memory

: Unused
Read as "00"

Bank 0 RAM Mapping

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 Bank1 register, only the Indirect addressing pointer MP1 can be used and the Bank register BP should set to 1. The mapping of RAM bank 1 is as shown.

40H	—
41H	—
42H	AWR
43H	STALL
44H	PIPE
45H	SIES
46H	MISC
47H	—
48H	FIFO0
49H	FIFO1
4AH	FIFO2
4BH	FIFO3
4CH ⋮ FFH	Undefined, reserved for future expansion

RAM Bank 1

Note: Register 45H is defined for version C or later version

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 no matter the value of Bank Register (BP).

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

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

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.

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

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 INTC) will be set.

- The access of the corresponding USB FIFO from PC
- The USB suspend signal from PC
- The USB resume signal from PC
- USB 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 PC Host access the FIFO of the HT82K96E, the corresponding request bit of USR is set, and a USB interrupt is triggered. So user can easy to decide which FIFO is accessed. When the interrupt has been served, the corresponding bit should be cleared by firmware. When HT82K96E receive a USB Suspend signal from Host PC, the suspend line (bit0 of USC) of the HT82K96E is set and a USB interrupt is also triggered.

Also when HT82K96E receive a Resume signal from Host PC, the resume line (bit3 of USC) of HT82K96E is set and a USB interrupt is triggered.

Whatever there are 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 (; 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/even counter 1 interrupt is initialized by setting the Timer/Event Counter 1 interrupt request flag (;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.

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

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.

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, ET0I and ET1I are used to control the enabling/disabling of interrupts. These bits prevent the requested interrupt from being serviced. Once the interrupt request flags (T0F, T1F, 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.
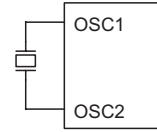

Crystal Oscillator

System Oscillator

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

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.
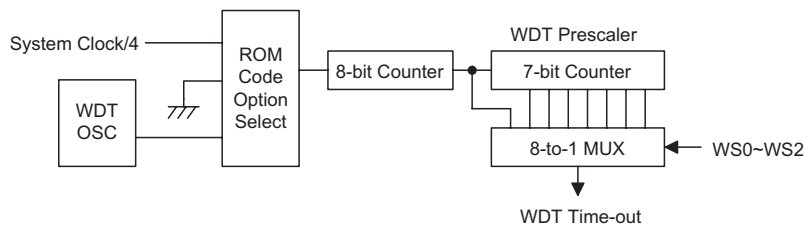
Watchdog Timer

Once the internal WDT oscillator (RC oscillator with a period of 31μs/5V normally) 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 (bit 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 ″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 WDT (including the WDT prescaler), three methods are adopted; external reset (a low level to $\overline{RES}$), 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 that ″CLR WDT″ and ″CLR WDT″ are chosen (i.e. CLRWDT times equal two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip as a result of time-out.

The time-out periods defined in WDTS can used as ″wake-up period″ in the Mouse Hardware wake-up function. Please reference to Mouse Hardware Wake-up function description.

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 recounted again (if the WDT clock is from the WDT oscillator).

• All of the I/O ports maintain their original status.

• The PDF flag is set and the TO flag is cleared.

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

The port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each bit in port A can be independently selected to wake up the device by mask option. 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 finished.

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

Reset

There are three ways in which a reset can occur:

• $\overline{RES}$ 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 re-set″ 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
u	u	$\overline{RES}$ reset during normal operation
0	1	$\overline{RES}$ 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 reset (power-up, WDT time-out or $\overline{RES}$ reset) or the system awakes from the HALT state.

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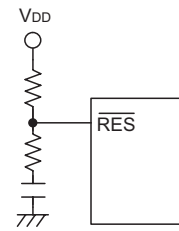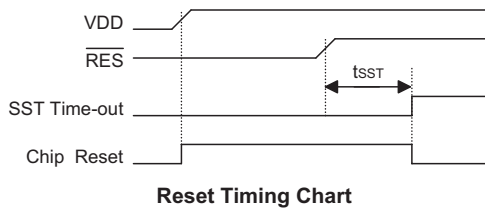
Reset Circuit
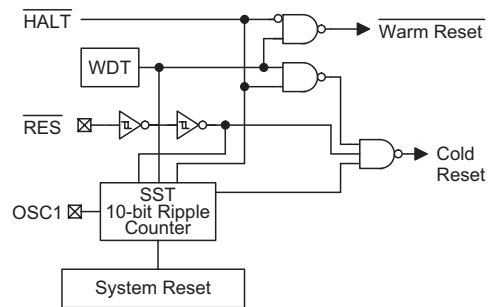
Reset Configuration

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

Reset Timing Chart

The states of the registers is summarized in the table.

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)
TMR0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	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	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR1L	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	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	--uu uuuu	--01 uuuu	--11 uuuu	--uu uuuu	--01 uuuu
INTC	-000 0000	-000 0000	-000 0000	-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
AWR	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PIPE	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
STALL	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MISC	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
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
FIFO3	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	0100 0000	uuuu uuuu	0100 0000	0100 0000	uuuu uuuu	01uu 0000	01uu 0000
SCC	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0u00 u000	0u00 u000
ADSC	1000 0000	uuuu uuuu	1000 0000	1000 0000	uuuu uuuu	1000 0000	1000 0000
ADR	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx

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 comes 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	TE	To define the TMR0 active edge of Timer/Event Counter 0 (0=active on low to high; 1=active on high to low)
4	TON	To enable/disable timer 0 counting (0=disabled; 1=enabled)
6 7	TM0 TM1	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	TE	To define the TMR1 active edge of Timer/Event Counter 1 (0=active on low to high; 1=active on high to low)
4	TON	To enable/disable timer 1 counting (0=disabled; 1=enabled)
6 7	TM0 TM1	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
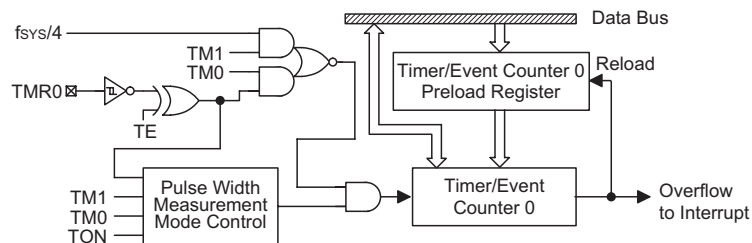
Timer/Event Counter 0
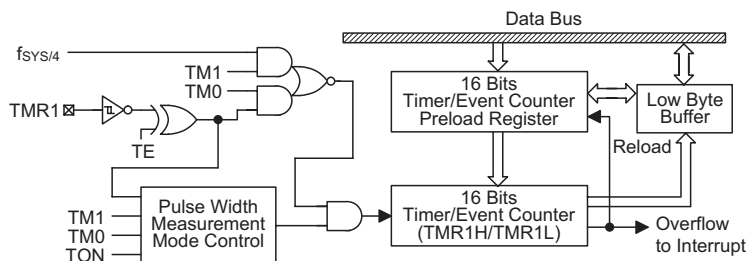
Timer/Event Counter 1

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 TM0, TM1 bits define the operating mode. The event count mode is used to count external events, which means the clock source comes from an external (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 TON and TE bits equal to one, once the TMR0/TMR1 has received a transient from low to high (or high to low if the TE bits is ″0″) it will start counting until the TMR0/TMR1 returns to the original level and resets the TON. 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 TON, the cycle measurement will function again as long as it receives further transient pulse. Note that, in this operating mode, the Timer/Event Counter 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 (TON; bit 4 of TMR0C/TMR1C) should be set to 1. In the pulse width measurement mode, the TON will be cleared automatically after the measurement cycle is completed. But in the other two modes the TON can only be reset by instructions. The 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 results 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/low resistor structures can be reconfigured dynamically (i.e. on-the-fly) under software control. To function as an input, the corresponding latch of the control register must write ″1″. The 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/low 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 port A has the capability of waking-up the device.

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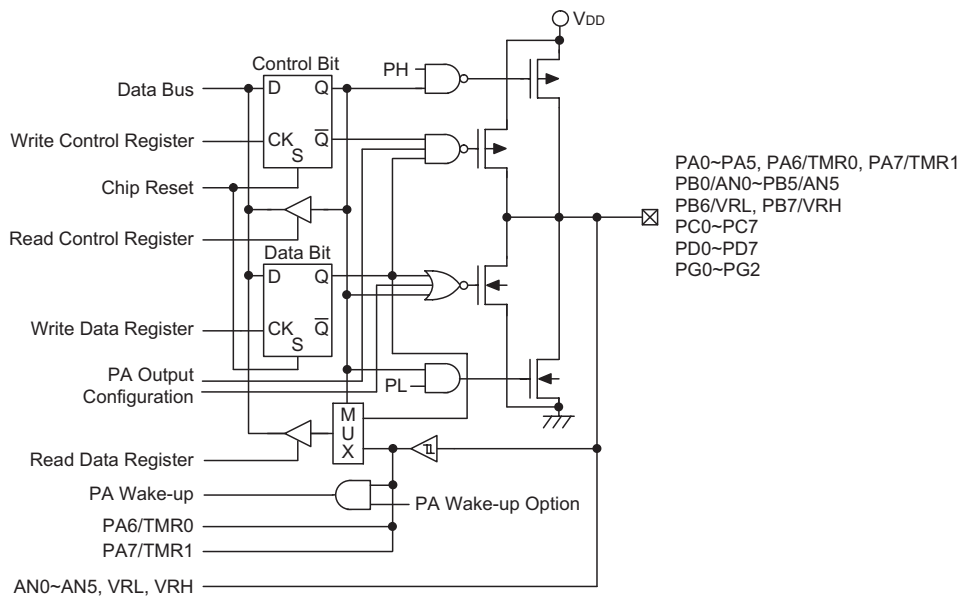

Input/Output Ports

Low Voltage Reset − LVR

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

The LVR includes the following specifications:

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

- The LVR uses the ″OR″ function with the external RES signal to perform chip reset.

The relationship between V_{DD} and V_{LVR} is shown below.
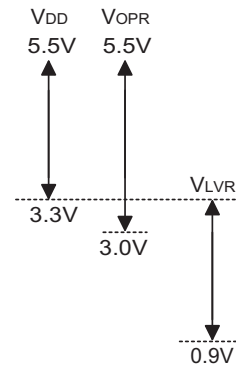
Note: V_{OPR} is the voltage range for proper chip operation at 4MHz system clock.
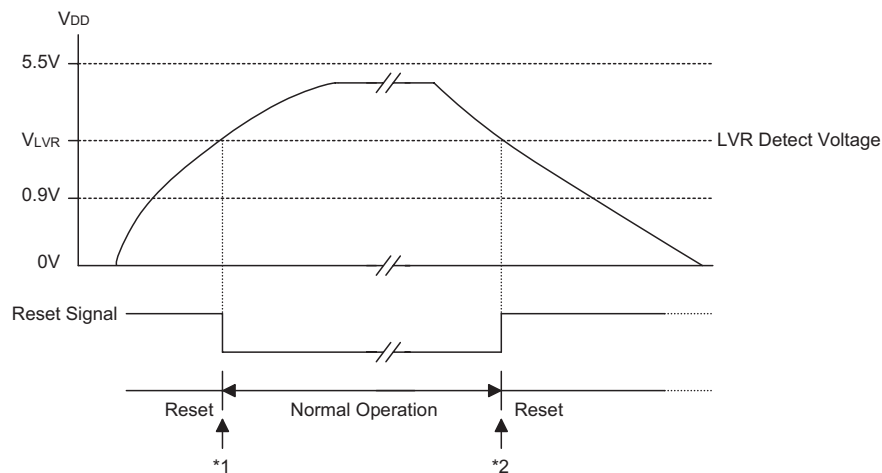
Low Voltage Reset

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

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

Mouse Hardware Wake-Up Function

When the HT82K96E is used for USB mouse application, in order to decrease the power consumption of the HT82K96E in suspend mode. The HT82K96E has built-in Mouse Hardware wake-up function. Once the HT82K96E jump to suspend mode, and the HWKUPSB (bit7 of SCC) is set to 1. The HT82K96E will automatically switch the IRPT control pin (PC0) and detect movement of the X1, X2, Y1, Y2, Z1, Z2, corresponding to (PA0~PA5) and the state of the five button corresponding to PA6, PA7, PB6, PB7, and PD4. Once there are mouse movement or state change. The HT82K96E will wake-up the MCU by I/O method, otherwise the MCU is in suspend mode.

How long the HT82K96E to turn on the IRPT, and the low pulse period of the PC0 is defined by bit0~3 of the WDTS (wake-up period) and the bit0~bit2 of the SCC (LED_on period) respectively. The following diagram show the IRPT control pin timing.
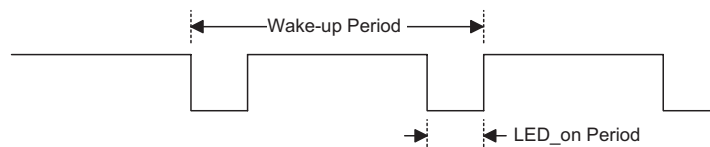
Suspend Wake-Up Remote Wake-Up

If there is no signal on USB bus is over 3ms, the HT82K96E will go into suspend mode . The Suspend line (bit 0 of USC) will be set to 1 and a USB interrupt is triggered to indicate the HT82K96E should jump to suspend state to meet the 500μA USB suspend current spec.

In order to meet the 500μA suspend current, the firmware should disable the USB clock by clear the USBCKEN (bit3 of the SCC) to ″0″. The suspend current is about 400μA.

Also the user can further decrease the suspend current to 250μA by set the SUSP2 (bit4 of the SCC). But if the SUSP2 is set, the user make sure cannot enable the LVR OPT option, otherwise the HT82K96E will be reset.

When the resume signal is sent out by the host, the HT82K96E will wake up the MCU by USB interrupt and the Resume line (bit 3 of USC) is set. In order to make HT82K96E work properly, the firmware must set the USBCKEN (bit 3 of 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 and the Suspend line (bit 0 of USC) is going 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
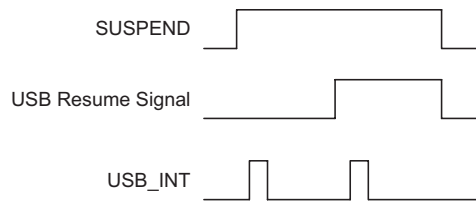
The device with remote wake up function can wake-up the USB Host by sending a wake-up pulse through RMWK (bit 1 of USC). Once the USB Host receive the wake-up signal from HT82K96E. it will send a Resume signal to device. The timing as follow:
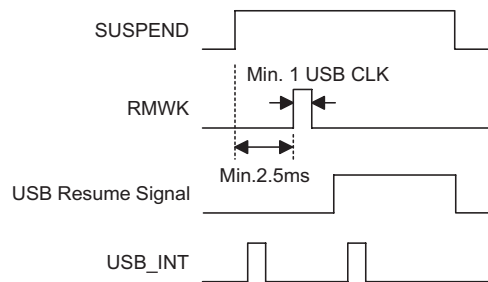
To Configure the HT82K96E as PS2 Device

The HT82K96E can be define as USB interface or PS2 interface by configuring the SPS2 (bit 4 of USR) and SUSB (bit 5 of USR). If SPS2=1, and SUSB=0, the HT82K96E is defined as PS2 interface, pin USBD- is now defined as PS2 Data pin and USBD+ is now defined as PS2 Clk pin. The user can easy to read or write the PS2 Data or PS2 Clk pin by accessing the corresponding bit PS2DAI (bit 4 of USC), PS2CKI (bit 5 of USC), PS2DAO (bit 6 of USC) and S2CKO (bit 7 of USC) respectively.

The user should make sure that in order to read the data properly, the corresponding output bit must set to 1. For example, if it want to read PS2 Data by reading PS2DAI, the PS2DAO should set to 1. Otherwise it always read 0.

If SPS2=0, and SUSB=1, the HT82K96E is defined as USB interface. Both the USBD- and USBD+ is driving by SIE of the HT82K96E. The user only write or read the USB data through the corresponding FIFO.

Both SPS2 and SUSB is default ″0″.

To Configure the ADC Block

The HT82K96E has built-in a 8-bit A/D converter with 6 channels (PB0~PB5). In order to make the A/D converter more flexibility, there are two mode: External Reference voltage and Internal Reference voltage. It can easy to configure by setting the ADREF (bit 6 of USR). For External Reference voltage, the reference voltage of the A/D converter comes from external PB6/VRL and PB7/VRH pins. Otherwise, the reference voltage is coming from the VDD and VSS of MCU.

PB0~PB5 is the 6-channels input of the A/D converter, it can easy to define which channel is converting by configuring ACS2~ACS0 (bit 2~0 of ADSC). Also there are four converter clock source to be selected by setting ADCS1 (bit 4 of ADSC), ADCS0 (bit 3 of ADSC).

Once the ADON (bit 6 of ADSC) is set and send the start pulse through START (bit 5 of ADSC). The A/D converter will be in operation. There are EOCB (bit 7 of ADSC) to indicate whether the A/D converter is busy or not. The EOCB is clear when the conversion is completed. The user can read the converter data by reading the register ADR. In order to meet 500uA suspend current spec. . The user should disable the A/D by clearing ADON before jump to suspend mode.

The following is A/D converter timing diagrams
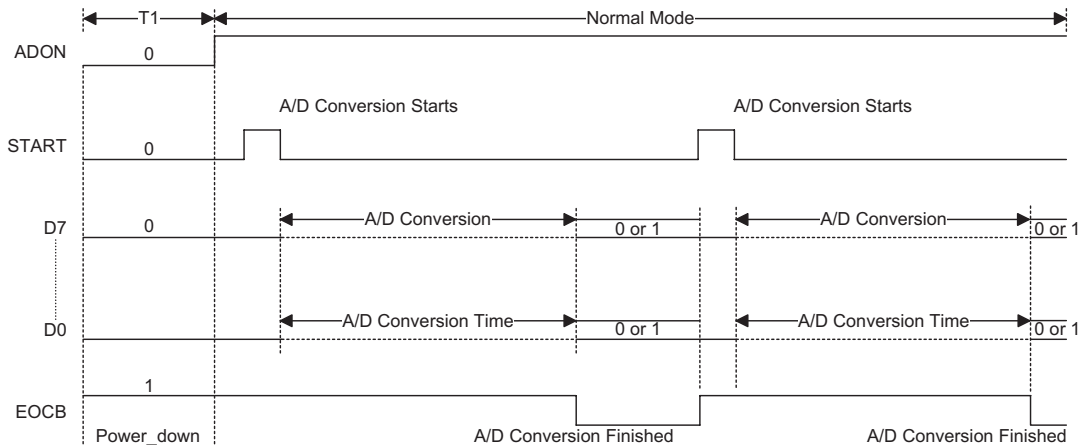
USB Interface and A/D Converter

There are 7 registers, including AWR (address + remote wake up; 42H in bank 1), STALL (43H in bank 1), PIPE (44H in bank 1), MISC (46H in bank 1), FIFO0 (48H in bank 1), FIFO1 (49H in bank 1), FIFO2 (4AH in bank 1) and FIFO3 (4BH 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 has not to be loaded into this register until the SETUP stage being finished.

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

PIPE register represents whether the corresponding endpoint is accessed by host or not. This register is set only after the time when host accesses the corresponding endpoint. Only the last accessed endpoint is shown in this register. STALL register shows whether the corresponding endpoint works properly or not. As soon as the endpoint works improperly, the related bit in the STALL has to be set to ″1″. The STALL will be cleared by USB reset signal.

Bit No.	Label	R/W	Function
0	STL0	W	Stall the endpoint 0
1	STL1	W	Stall the endpoint 1
2	STL2	W	Stall the endpoint 2
3	STL3	W	Stall the endpoint 3
7~4	—	W	Unused bit, read as ″0″

STALL (43H) Register

Bit No.	Label	R/W	Function
0	EP0RW	R	Endpoint 0 accessed
1	EP1RW	R	Endpoint 1 accessed
2	EP2RW	R	Endpoint 2 accessed
3	EP3RW	R	Endpoint 3 accessed
7~4	—	R	Unused bit, read as ″0″

PIPE (44H) Register

SIES. Register (for version C or later version) 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.

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Func.				Reserved bit			F0_ERR	Adr_set
R/W							R/W	R/W
Reg_Adr				01000101B				

Note: Bit7 must be ″0″

Func. Name	R/W	Description
	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

SIES (45H) Register Table

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

Bit No.	Label	R/W	Function
0	REQ	R/W	After setting other status of desired one in the MISC, endpoint FIFO can be requested by setting this bit to ″1″. After 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 MCU wants to write data to endpoint FIFO. After the job has been done, this bit has to be cleared to ″0″ before terminating request to represent end of transferring. For reading action, this bit has to be cleared to ″0″ to represent that MCU wants to read data from endpoint FIFO and has to be set to ″1″ after the job done.
2	CLEAR	R/W	Clear the requested endpoint FIFO, even the endpoint FIFO is not ready.
4 3	SELP1 SELP0	R/W	To define which endpoint FIFO is selected, SELP1,SELP0: 00: endpoint FIFO0 01: endpoint FIFO1 10: endpoint FIFO2 11: endpoint FIFO3
5	SCMD	R/W	It is used to show that the data in endpoint FIFO is SETUP command. This bit has to be cleared by firmware. That is to say, even the MCU is busing, the device will not miss any SETUP commands from 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	It is used to indicate that a 0-sized packet is sent from host to MCU. This bit should be cleared by firmware.

MISC (46H) Register

MCU can communicate with endpoint FIFO by setting the corresponding registers, of which address is listed in the following table. After reading current data, next data will show on after 2μs. using to check 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
FIFO3	R/W	1	4BH	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 2μs, check 41H→read* from FIFO0 register and check not ready (01H)→03H→02H
Write FIFO1 sequence	0AH→0BH→delay 2μs, check 4BH→write* to FIFO1 register and check not ready (0BH)→09H→08H
Check whether FIFO0 can be read or not	00H→01H→delay 2μs, check 41H (ready) or 01H (not ready)→00H
Check whether FIFO1 can be written or not	0AH→0BH→delay 2μs, check 4BH (ready) or 0BH (not ready)→0AH
Read 0-sized packet sequence form FIFO0	00H→01H→delay 2μs, check 81H→read once (01H)→03H→02H
Write 0-sized packet sequence to FIFO1	0AH→0BH→delay 2μs, check 0BH→0FH→0DH→08H

Note: *: There are 2μs 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 changing 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″, 2μs 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 a USB reset is occurred (The attached bus is USB) and a USB interrupt will be initialized.
3	RESUME	R	USB resume indication. When the USB leaves suspend mode, this bit is set to ″1″ (set by SIE). This bit will appear 20ms waiting for MCU to detect. When the RESUME is set by SIE, an interrupt will be generated to wake-up the MCU. In order to detecting the suspend state, MCU should set 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 MCU is detecting the SUSP, the RESUME (causes MCU to wake-up) 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 USBD-/DATA pin when work under 3D PS2 mouse function. (Default=″1″)
7	PS2CKO	W	Data for driving USBD+/CLK pin when work 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 serial bus (PS2 or USB) and A/D converter operation modes. The endpoint request flags (EP0IF, EP1IF, EP2IF and EP3IF) 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 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 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 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 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	EP3IF	R/W	When this bit is set to ″1″ (set by SIE), it indicates the endpoint 3 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
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″)
6	ADREF	R/W	The reference voltage of A/D converter is coming from the VDD and VSS of MCU when this bit is set ″1″. Otherwise, the reference voltage of A/D converter comes from external PB6/VRL and PB7/VRH pins. (Default=″1″)
7	FIFO-cntl	W	For ICE only, 0 for FIFO read (Default=″0″); 1 for FIFO write

USR (1BH) Register

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

Bit No.	Label	R/W	Function
2~0	Led_on Period	R/W	To define low pulse period of IRPT (PC0) for mouse hardware function. The time base is 31.25μs (1/32kHz). Default value is 000. 000: 2×base 001: 3×base 010: 5×base 011: 9×base 100: 17×base 101: 33×base 110: 65×base 111: 127×base
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 for decreasing power consumption in suspend mode. In normal mode clean this bit=0 (Default=″0″) In HALT mode set this bit=1 for decreasing power consumption.
5	—	R/W	Undefined, should be cleared to ″0″
6	SYSCLK	R/W	This bit is used to specify the system oscillator frequency used by 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).
7	HWKUPSB	R/W	Hardware HALT mode wake-up detect circuit active under power down mode. Low active. ″0″: WDT timer overflow will wake-up MCU system ″1″: WDT timer overflow will start hardware wake-up detect circuit but not wake-up MCU system.

SCC (1CH) Register

The A/D converter implemented in the MCU is a 6-channel 8-bit A/D converter. The reference voltage (high reference voltage and low reference voltage) can be selected as coming from external pins (PB6/VRL and PB7/VRH) or internal power supplies of MCU (VDD and VSS). The VRL and VRH are used to set the minimal and maximal boundaries of the full-scale range of the A/D converter. If an analog inputs, VRL or VRH is not used for A/D conversion, it also can be used as a general purpose I/O line. The ADSC (A/D converter status and control register) register is used to set the configurations and A/D clock sources of A/D converter and control the operation of A/D converter.

Bit No.	Label	Function
2~0	ACS2~ACS0	These 3 bits are use to select one of eight A/D converter channels for the conversion. The A/D converter input channels AN0~AN5 are pin-shared with PB0~PB5. PB6/VRL and PB7/VRH are used for the A/D converter reference inputs. ACS2,ACS1,ACS0 : 000/001/010/011/100/101/110/111: AN0/AN1/AN2/AN3/AN4/AN5/VRL/VRH
4 3	ADCS1 ADCS0	A/D converter clock source selection. ADCS1,ADCS0: 00: 6MHz 01: 3MHz 10: 1.5MHz 11: 0.75MHz
5	START	Start the A/D conversion. (0→1→0: start, 0→1: reset A/D converter and A/D data register)
6	ADON	This bit is used to control the enable/disable of A/D converter circuit. If this bit is set to "1" the A/D converter enters operating mode. Otherwise, the A/D converter will be turned-off
7	EOCB	End of A/D conversion indication. (0: end of A/D conversion)

ADSC (1DH) Register

The A/D converter data register is used to store the result of A/D conversion.

| Bit No.	Label	Function
7~0	D7~D0	Result of A/D conversion

ADR (1EH) Register

Mask Options

The following table shows all kinds of mask option in the microcontroller. All of the mask 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	PA0~PA5 pull down resistor enabled or disabled (by bit)
4	PB0~PB7 pull-high resistor enabled or disabled (by nibble)
5	PC0~PC7 pull-high resistor enabled or disabled (by nibble)
6	PD0~PD7 pull-high resistor enabled or disabled (by nibble)
7	LVR enable or disable
8	WDT enable or disable
9	WDT clock source: f_{SYS}/4 or WDTOSC
10	"CLRWDT" instruction(s): 1 or 2
11	PA0~PA7 output structures: CMOS/NMOS open-drain/PMOS open-drain (by bit)
12	PA0~PA7 wake-up enabled or disabled (by bit)
13	A/D converter enabled or disabled

Application Circuits

Crystal or Ceramic Resonator for Multiple I/O Applications
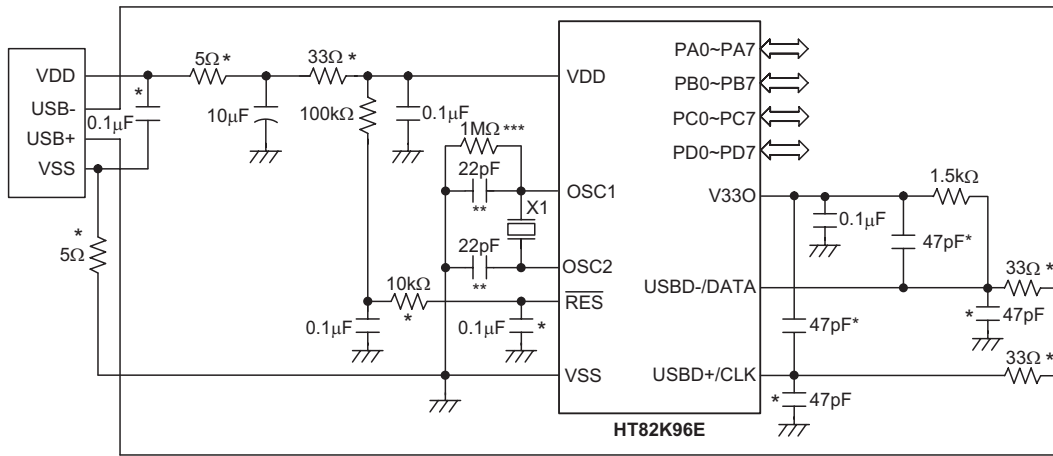
Note: The resistance and capacitance for reset circuit should be designed in such a way as to ensure that the VDD is stable and remains within a valid operating voltage range before bringing RES to high.

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.

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

Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

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

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

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.

    2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

    3. For the ″CLR WDT1″ and ″CLR WDT2″ instructions the TO and PDF flags may be affected by the execution status. The TO and PDF flags are cleared after both ″CLR WDT1″ and ″CLR WDT2″ instructions are consecutively executed. Otherwise the TO and PDF flags remain unchanged.

Instruction Definition

ADC A,[m]   Add Data Memory to ACC with Carry

Description   The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.

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

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

ADCM A,[m]   Add ACC to Data Memory with Carry

Description   The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.

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

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

ADD A,[m]   Add Data Memory to ACC

Description   The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.

Operation $ACC \leftarrow ACC + [m]$

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

ADD A,x   Add immediate data to ACC

Description   The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.

Operation $ACC \leftarrow ACC + x$

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

ADDM A,[m]   Add ACC to Data Memory

Description   The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.

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

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

AND A,[m]   Logical AND Data Memory to ACC

Description   Data in the Accumulator and the specified Data Memory perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation $ACC \leftarrow ACC\ ''AND''\ [m]$

Affected flag(s)   Z

AND A,x   Logical AND immediate data to ACC

Description   Data in the Accumulator and the specified immediate data perform a bitwise logical AND operation. The result is stored in the Accumulator.

Operation $ACC \leftarrow ACC\ ''AND''\ x$

Affected flag(s)   Z

ANDM A,[m]   Logical AND ACC to Data Memory

Description   Data in the specified Data Memory and the Accumulator perform a bitwise logical AND operation. The result is stored in the Data Memory.

Operation $[m] \leftarrow ACC\ ''AND''\ [m]$

Affected flag(s)   Z

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

CPL [m]   Complement Data Memory

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

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

Affected flag(s)   Z

CPLA [m]   Complement Data Memory with result in ACC

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

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

Affected flag(s)   Z

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

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

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

Affected flag(s)   C

DEC [m]   Decrement Data Memory

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

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

Affected flag(s)   Z

DECA [m]   Decrement Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] - 1$

Affected flag(s)   Z

HALT   Enter power down mode

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

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

Affected flag(s)   TO, PDF

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z

INCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] + 1
Affected flag(s)	Z

JMP addr	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter ← addr
Affected flag(s)	None

MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None

MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC ← x
Affected flag(s)	None

MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None

NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None

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

OR A,x   Logical OR immediate data to ACC

Description   Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator.

Operation   ACC ← ACC ″OR″ x

Affected flag(s)   Z

ORM A,[m]   Logical OR ACC to Data Memory

Description   Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory.

Operation   [m] ← ACC ″OR″ [m]

Affected flag(s)   Z

RET   Return from subroutine

Description   The Program Counter is restored from the stack. Program execution continues at the restored address.

Operation   Program Counter ← Stack

Affected flag(s)   None

RET A,x   Return from subroutine and load immediate data to ACC

Description   The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.

Operation   Program Counter ← Stack
                         ACC ← x

Affected flag(s)   None

RETI   Return from interrupt

Description   The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.

Operation   Program Counter ← Stack
                         EMI ← 1

Affected flag(s)   None

RL [m]   Rotate Data Memory left

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

Operation $[m].(i+1) \leftarrow [m].i; (i = 0 \sim 6)$
                         $[m].0 \leftarrow [m].7$

Affected flag(s)   None

RLA [m]   Rotate Data Memory left with result in ACC

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

Operation $ACC.(i+1) \leftarrow [m].i; (i = 0 \sim 6)$
                         $ACC.0 \leftarrow [m].7$

Affected flag(s)   None

RLC [m]	Rotate Data Memory left through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i = 0~6) [m].0 ← C C ← [m].7
Affected flag(s)	C

RLCA [m]	Rotate Data Memory left through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.(i+1) ← [m].i; (i = 0~6) ACC.0 ← C C ← [m].7
Affected flag(s)	C

RR [m]	Rotate Data Memory right
Description	The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i = 0~6) [m].7 ← [m].0
Affected flag(s)	None

RRA [m]	Rotate Data Memory right with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.i ← [m].(i+1); (i = 0~6) ACC.7 ← [m].0
Affected flag(s)	None

RRC [m]	Rotate Data Memory right through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i = 0~6) [m].7 ← C C ← [m].0
Affected flag(s)	C

RRCA [m]	Rotate Data Memory right through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.i ← [m].(i+1); (i = 0~6) ACC.7 ← C C ← [m].0
Affected flag(s)	C

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

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

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

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

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

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

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

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

SET [m] Set Data Memory

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

Operation $[m] \leftarrow FFH$

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation $[m].i \leftarrow 1$

Affected flag(s) None

SIZ [m]	Skip if increment Data Memory is 0
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	[m] ← [m] + 1 Skip if [m] = 0
Affected flag(s)	None

SIZA [m]	Skip if increment Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] + 1 Skip if ACC = 0
Affected flag(s)	None

SNZ [m].i	Skip if bit i of Data Memory is not 0
Description	If bit i of the specified Data Memory is not 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is 0 the program proceeds with the following instruction.
Operation	Skip if [m].i ≠ 0
Affected flag(s)	None

SUB A,[m]	Subtract Data Memory from ACC
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − [m]
Affected flag(s)	OV, Z, AC, C

SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	[m] ← ACC − [m]
Affected flag(s)	OV, Z, AC, C

SUB A,x	Subtract immediate data from ACC
Description	The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	ACC ← ACC − x
Affected flag(s)	OV, Z, AC, C

SWAP [m]   Swap nibbles of Data Memory

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

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

Affected flag(s)   None

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

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

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

Affected flag(s)   None

SZ [m]   Skip if Data Memory is 0

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

Operation   Skip if $[m] = 0$

Affected flag(s)   None

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

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

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

Affected flag(s)   None

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

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

Operation   Skip if $[m].i = 0$

Affected flag(s)   None

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

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

Operation $[m] \leftarrow$ program code (low byte)
$TBLH \leftarrow$ program code (high byte)

Affected flag(s)   None

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

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

Operation $[m] \leftarrow$ program code (low byte)
$TBLH \leftarrow$ program code (high byte)

Affected flag(s)   None

XOR A,[m]　　　　　Logical XOR Data Memory to ACC

Description　　　　　Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation　　　　　$ACC \leftarrow ACC \; ''XOR'' \; [m]$

Affected flag(s)　　　Z

XORM A,[m]　　　　Logical XOR ACC to Data Memory

Description　　　　　Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.

Operation　　　　　$[m] \leftarrow ACC \; ''XOR'' \; [m]$

Affected flag(s)　　　Z

XOR A,x　　　　　Logical XOR immediate data to ACC

Description　　　　　Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.

Operation　　　　　$ACC \leftarrow ACC \; ''XOR'' \; x$

Affected flag(s)　　　Z

Package Information

28-pin SOP (300mil) Outline Dimensions
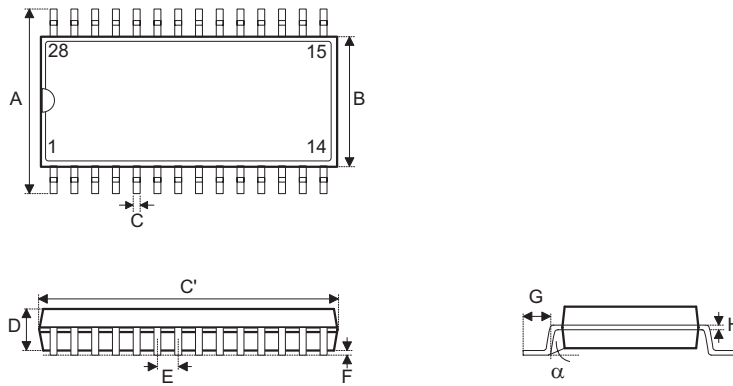
Symbol	Dimensions in mil		
	Min.	**Nom.**	**Max.**
A	394	—	419
B	290	—	300
C	14	—	20
C′	697	—	713
D	92	—	104
E	—	50	—
F	4	—	—
G	32	—	38
H	4	—	12
α	0°	—	10°

48-pin SSOP (300mil) Outline Dimensions
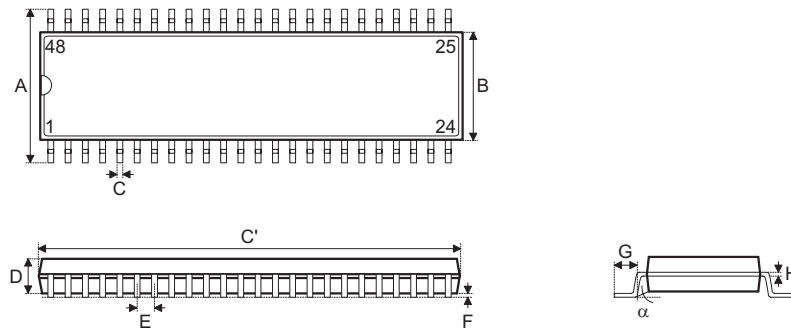
Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	395	—	420
B	291	—	299
C	8	—	12
C′	613	—	637
D	85	—	99
E	—	25	—
F	4	—	10
G	25	—	35
H	4	—	12
α	0°	—	8°

Product Tape and Reel Specifications

Reel Dimensions
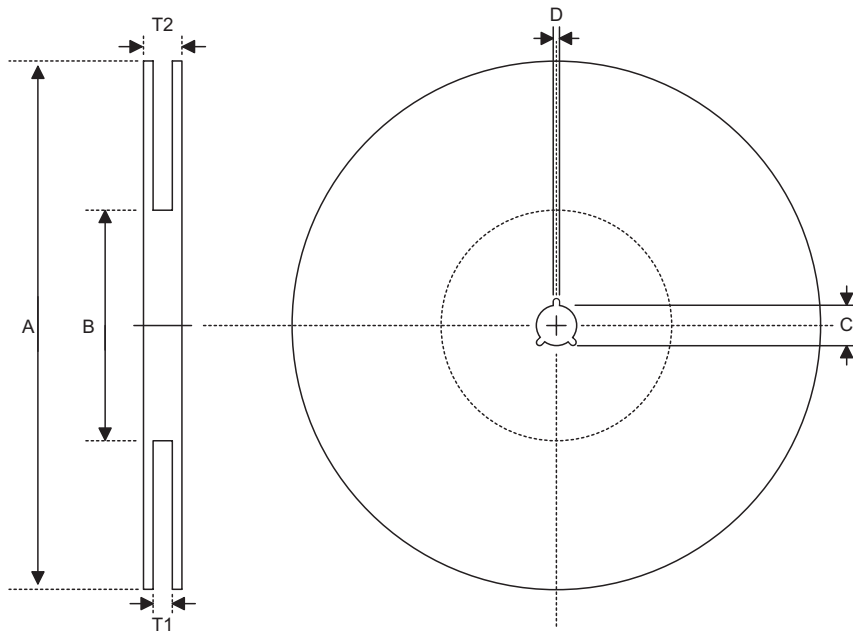
SOP 28W (300mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1
B	Reel Inner Diameter	62±1.5
C	Spindle Hole Diameter	13+0.5 −0.2
D	Key Slit Width	2±0.5
T1	Space Between Flange	24.8+0.3 −0.2
T2	Reel Thickness	30.2±0.2

SSOP 48W

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330±1
B	Reel Inner Diameter	100±0.1
C	Spindle Hole Diameter	13+0.5 −0.2
D	Key Slit Width	2±0.5
T1	Space Between Flange	32.2+0.3 −0.2
T2	Reel Thickness	38.2±0.2

Carrier Tape Dimensions
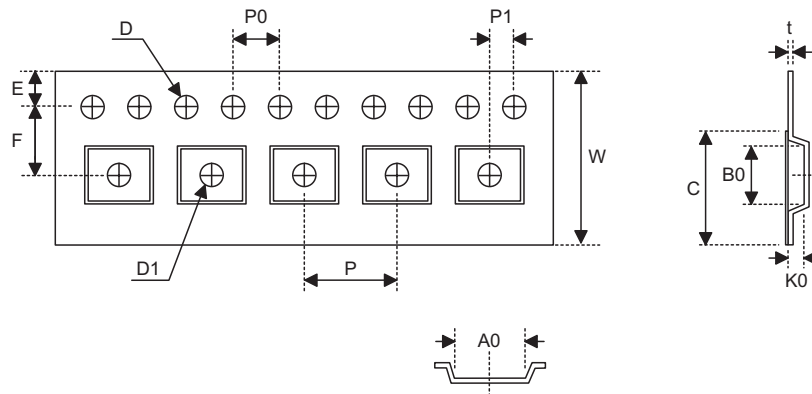
SOP 28W (300mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24±0.3
P	Cavity Pitch	12±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5+0.1
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4±0.1
P1	Cavity to Perforation (Length Direction)	2±0.1
A0	Cavity Length	10.85±0.1
B0	Cavity Width	18.34±0.1
K0	Cavity Depth	2.97±0.1
t	Carrier Tape Thickness	0.35±0.01
C	Cover Tape Width	21.3
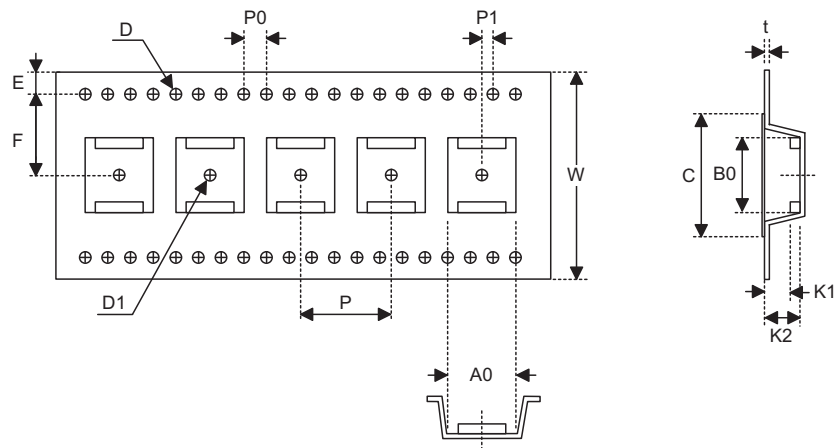
SSOP 48W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	32±0.3
P	Cavity Pitch	16±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	14.2±0.1
D	Perforation Diameter	2 Min.
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4±0.1
P1	Cavity to Perforation (Length Direction)	2±0.1
A0	Cavity Length	12±0.1
B0	Cavity Width	16.2±0.1
K1	Cavity Depth	2.4±0.1
K2	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.35±0.05
C	Cover Tape Width	25.5

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

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

Holtek Semiconductor Inc. (Shanghai Sales Office)
7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China 200233
Tel: 86-21-6485-5560
Fax: 86-21-6485-0313
http://www.holtek.com.cn

Holtek Semiconductor Inc. (Shenzhen Sales Office)
5/F, Unit A, Productivity Building, Cross of Science M 3rd Road and Gaoxin M 2nd Road, Science Park, Nanshan District,
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
Tel: 1-510-252-9880
Fax: 1-510-252-9885
http://www.holtek.com