8-Bit USB Multimedia Keyboard Encoder OTP MCU

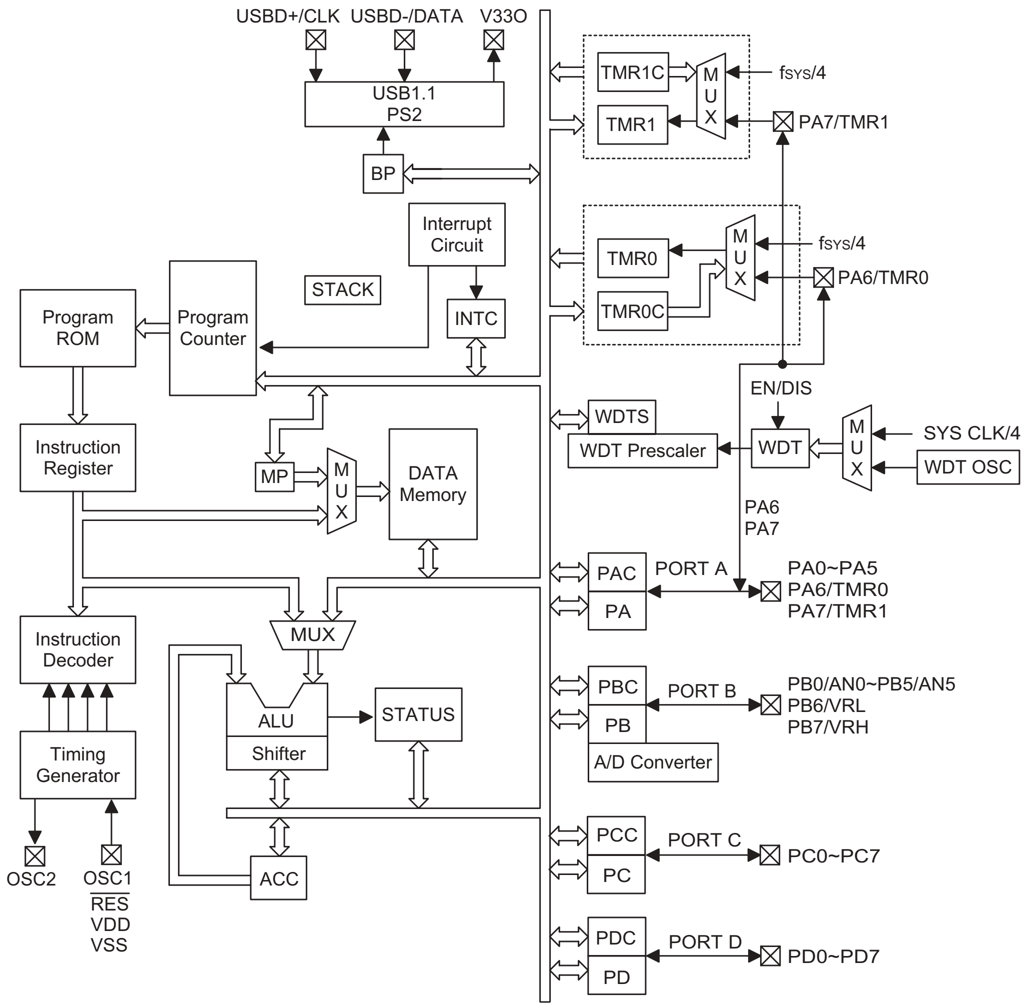
Features

- Operating voltage:
  f_{SYS}=6M/12MHz: 4.0V~5.5V
- Low voltage reset function
- 32 bidirectional I/O lines (max.)
- 8-bit programmable timer/event counter with overflow interrupt
- 16-bit programmable timer/event counter and overflow interrupts
- Crystal oscillator (6MHz or 12MHz)
- Watchdog Timer
- 6 channels 8-bit A/D converter
- PS2 and USB modes supported
- USB1.1 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
- 20-pin DIP/SOP, 28-pin SOP, 40-pin DIP, 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

1	PA3	20	PA4
2	PA2	19	PA5
3	PA1	18	PA6/TMR0
4	PA0	17	PA7/TMR1
5	PC0	16	OSC1
6	PD4	15	OSC2
7	VDD	14	$\overline{\text{RES}}$
8	V33O	13	VSS
9	USBD+/CLK	12	PB7/VRH
10	USBD-/DATA	11	PB6/VRL

HT82K96E
−20 DIP-A/SOP-A

1	PC3	28	PC2
2	VDD	27	PC0
3	V33O	26	PA0
4	USBD+/CLK	25	PA1
5	USBD-/CLK	24	PA2
6	PB0	23	PA3
7	PB1	22	PC4
8	PB2	21	PA4
9	PB3	20	PA5
10	PB4	19	PA6
11	PB5	18	PA7
12	PB6	17	OSC1
13	PB7	16	OSC2
14	VSS	15	$\overline{\text{RES}}$

HT82K96E
−28 SOP-A

1	PC5	40	PC6
2	PC4	39	PC7
3	PA3	38	PA4
4	PA2	37	PA5
5	PA1	36	PA6/TMR0
6	PA0	35	PA7/TMR1
7	PC0	34	PD3
8	PC1	33	PD2
9	PC2	32	PD1
10	PC3	31	PD0
11	PD4	30	OSC1
12	PD5	29	OSC2
13	PD6	28	$\overline{\text{RES}}$
14	PD7	27	VSS
15	VCC	26	PB7/VRH
16	V33O	25	PB6/VRL
17	USBD+/CLK	24	PB5/AN5
18	USBD-/DATA	23	PB4/AN4
19	PB0/AN0	22	PB3/AN3
20	PB1/AN1	21	PB2/AN2

HT82K96E
−40 DIP-A

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

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
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
$\overline{\text{RES}}$	I	—	Schmitt trigger reset input. Active low
VDD	—	—	Positive power supply
V33O	O	—	3.3V regulator output
USBD+/CLK	I/O	—	USBD+ or PS2 CLK I/O line USB OR PS2 function is controlled by software control register
USBD-/DATA	I/O	—	USBD- or PS2 DATA I/O line USB or PS2 function is controlled by software control register
OSC1 OSC2	I O	—	OSC1, OSC2 are connected to 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}+5.5V

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

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

Operating Temperature..........................–40°C to 85°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_{DD1}	Operating Voltage	—	f_{SYS}=6MHz	4.0	—	5.5	V
V_{DD2}	Operating Voltage	—	f_{SYS}=12MHz	4.0	—	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	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports	—	—	$0.7V_{DD}$	—	V_{DD}	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
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_{SST1}	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	t_{SYS}
t_{SST2}	System Start-up Timer Period		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: $t_{A/D}=\frac{1}{f_{A/D}}$, $f_{A/D}$=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.

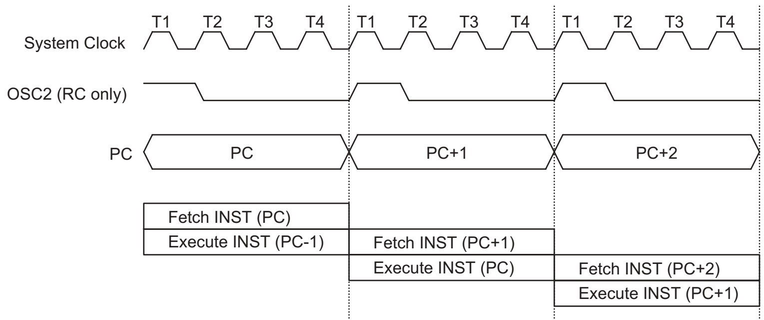
After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by one. The program counter then points to the memory word containing the next instruction code.

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	PC+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

#11~#0: Instruction code bits

S11~S0: Stack register bits

@7~@0: PCL bits

Program memory – ROM

The program memory is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized into 4096×15 bits, addressed by the program counter and table pointer.

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

- Location 000H

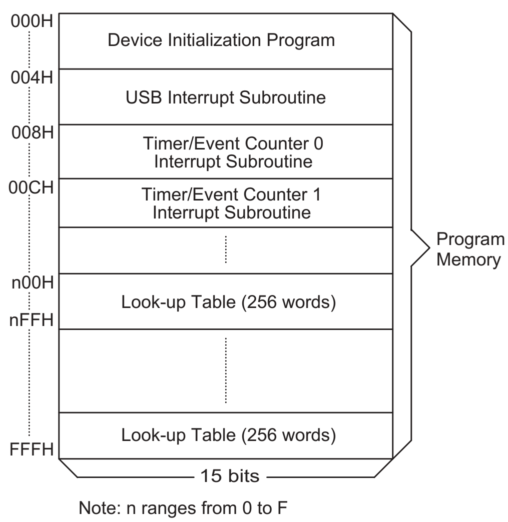
  This area is reserved for program initialization. After chip reset, the program always begins execution at location 000H.

- 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 .



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 (PC) 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

@7~@0: Table pointer bits

P11~P8: Current program counter 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 (;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	Special Purpose DATA MEMORY
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	(Unused)	
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	(Unused)	
20H ... BFH	General Purpose DATA MEMORY (160 Bytes)	

: 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.

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

RAM bank 1

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 (PD), and watchdog time-out flag (TO). It also records the status information and controls the operation sequence.

With the exception of the TO and PD 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 PD flag. In addition operations related to the status register may give different results from those intended.

Labels	Bits	Function
C	0	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.
AC	1	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.
Z	2	Z is set if the result of an arithmetic or logic operation is zero; otherwise Z is cleared.
OV	3	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.
PD	4	PD is cleared by system power-up or executing the ″CLR WDT″ instruction. PD is set by executing the ″HALT″ instruction.
TO	5	TO is cleared by system power-up or executing the ″CLR WDT″ or ″HALT″ instruction. TO is set by a WDT time-out.
—	6	Unused bit, read as ″0″
—	7	Unused bit, read as ″0″

Status 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 PD 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.

Register	Bit No.	Label	Function
INTC (0BH)	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 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.

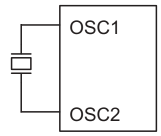
No.	Interrupt Source	Priority	Vector
a	USB interrupt	1	04H
b	Timer/Event Counter 0 overflow	2	08H
c	Timer/Event Counter 1 overflow	3	0CH

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

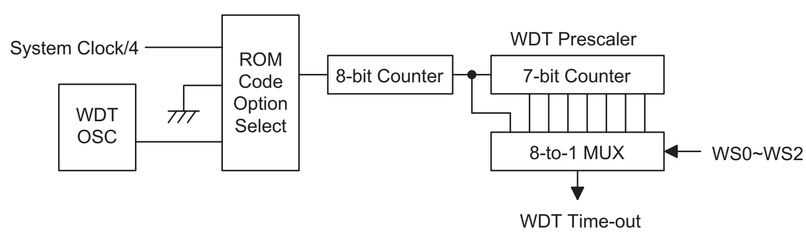
This oscillator is designed for system clocks. The HALT mode stops the system oscillator and ignores an external signal to conserve power.

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 72μs. 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 72μs@5V normally) is selected, it is first divided by 256 (8-stage) to get the nominal time-out period of 18.6ms@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 2.4s@5V seconds. 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 RC oscillator (WDT OSC) or 32kHz crystal oscillator (RTC 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 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 PC 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 $\overline{\text{RES}}$), 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 PD 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 PD flags are examined, the reason for chip reset can be determined. The PD 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 PC 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. The RTC oscillator still runs in the HALT mode (if the RTC oscillator is enabled).

Reset

There are three ways in which a reset can occur:

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

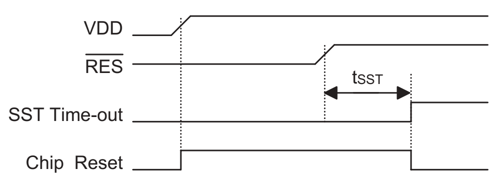
The WDT time-out during HALT is different from other chip reset conditions, since it can perform a "warm reset" that resets only the PC and SP, leaving the other circuits in their original state. Some registers remain unchanged during other reset conditions. Most registers are reset to the "initial condition"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

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

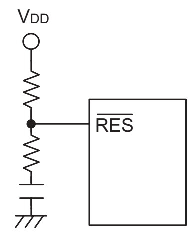
Note: "u" stands for "unchanged"

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 $\overline{\text{RES}}$ reset) or the system awakes from the HALT state.

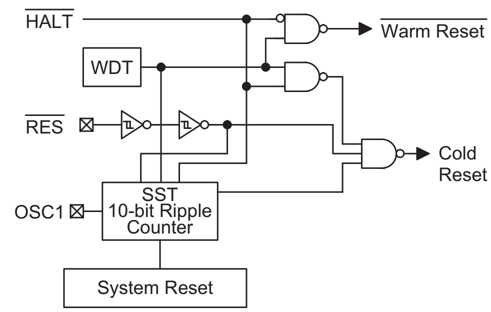
When a system reset occurs, the SST delay is added during the reset period. Any wake-up from HALT will enable the SST delay.



Reset timing chart



Reset circuit



Reset configuration

The functional unit chip reset status are shown below.

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

The states of the registers is summarized in the table.

Register	Reset (Power On)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset (Normal Operation)	$\overline{RES}$ 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	uuuu uuuu	uuuu uuuu
FIFO1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
FIFO2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
FIFO3	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
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	0000 u000	0000 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.

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.

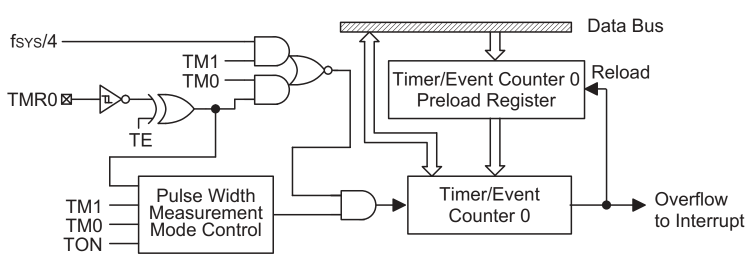
Using the internal clock source, there is only 1 reference time-base for Timer/Event Counter 1. The internal clock source is coming from $f_{SYS}/4$. The external clock input allows the user to count external events, measure time intervals or pulse widths.

Label (TMR0C)	Bits	Function
—	0~2	Unused bit, read as ″0″
TE	3	To define the TMR0 active edge of Timer/Event Counter 0 (0=active on low to high; 1=active on high to low)
TON	4	To enable/disable timer 0 counting (0=disabled; 1=enabled)
—	5	Unused bit, read as ″0″
TM0 TM1	6 7	To define the operating mode 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

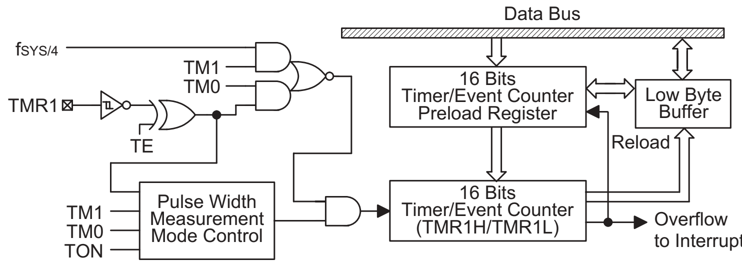
TMR0C register

Label (TMR1C)	Bits	Function
—	0~2	Unused bit, read as ″0″
TE	3	To define the TMR1 active edge of Timer/Event Counter 1 (0=active on low to high; 1=active on high to low)
TON	4	To enable/disable timer 1 counting (0=disabled; 1=enabled)
—	5	Unused bit, read as ″0″
TM0 TM1	6 7	To define the operating mode 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMR1C register



Timer/Event Counter 0



Timer/Event Counter 1

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.

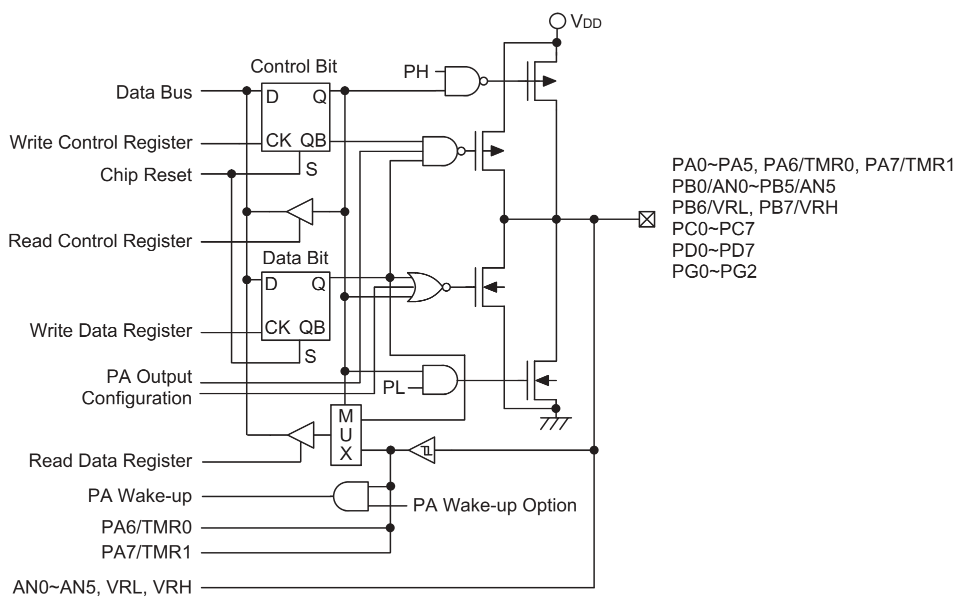
After a chip reset, these input/output lines remain at high levels or floating state (depending on the pull-high/low options). Each bit of these input/output latches can be set or cleared by ″SET [m].i″ and ″CLR [m].i″ (m=12H, 14H, 16H or 18H) instructions.

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

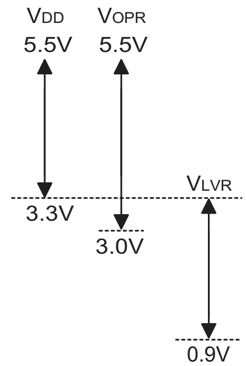
Low voltage reset – LVR

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~V_{LVR} such as changing a battery, the LVR will automatically reset the device internally.

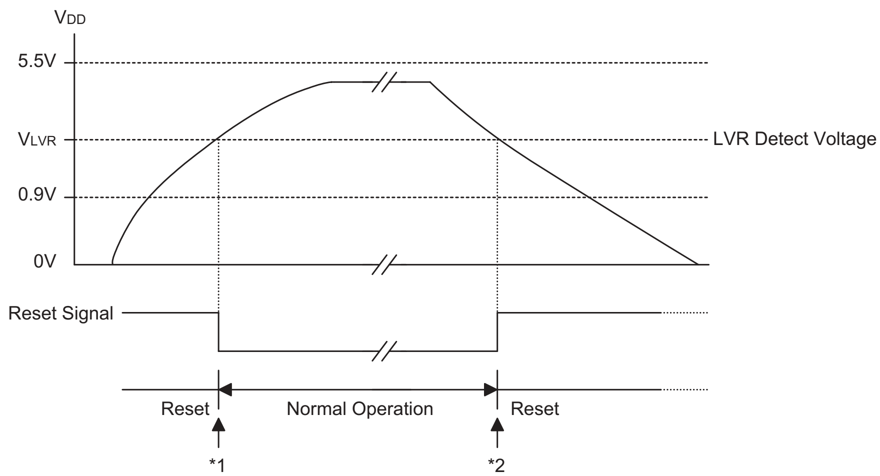
The LVR includes the following specifications:

- The low voltage (0.9V~V_{LVR}) has to remain in their original state to exceed 1ms. If the low voltage state does not exceed 1ms, the LVR will ignore it and do not perform a reset function.
- The LVR uses the "OR" function with the external $\overline{RES}$ signal to perform chip reset.

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



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



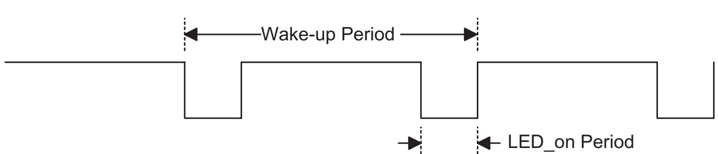
Low voltage reset

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

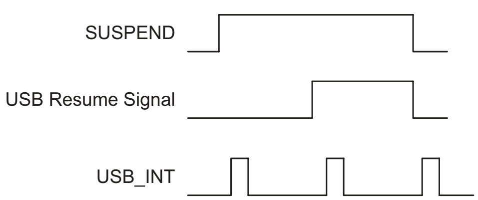
If there is no signal on USB bus is over 3ms, the HT82K96E will go into suspend mode . The Suspend line (bit 0 of USC) will be set to 1 and a USB interrupt is triggered to indicate the HT82K96E should jump to suspend state to meet the 500μA USB suspend current spec.

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.

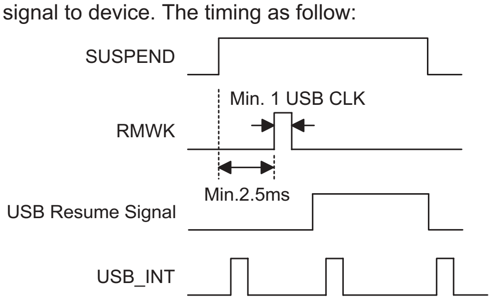
Also the user can further decrease the suspend current to 250μA by set the SUSP2 (bit4 of the SCC). But if the SUSP2 is set, the user make sure cannot enable the LVR OPT option, otherwise the HT82K96E will be reset.

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 token 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″.

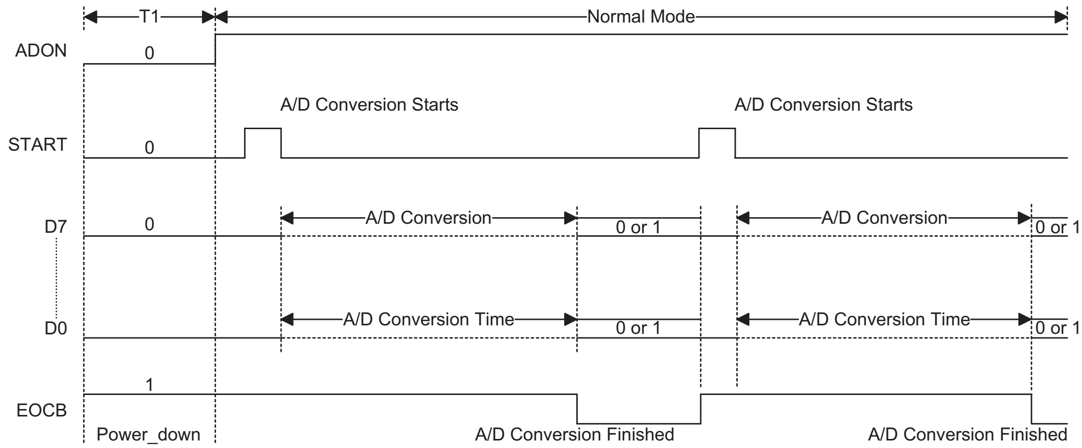
To configure the ADC block

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.

AWR	Bits	R/W	Function
WKEN	0	W	Remote wake-up enable/disable
AD6~AD0	7~1	W	USB device address

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.

PIPE	Bits	R/W	Function
EP0RW	0	R	Endpoint 0 accessed
EP1RW	1	R	Endpoint 1 accessed
EP2RW	2	R	Endpoint 2 accessed
EP3RW	3	R	Endpoint 3 accessed
—	7~4	R	Unused bit, read as ″0″

STALL	Bits	R/W	Function
STL0	0	W	Stall the endpoint 0
STL1	1	W	Stall the endpoint 1
STL2	2	W	Stall the endpoint 2
STL3	3	W	Stall the endpoint 3
—	7~4	W	Unused bit, read as ″0″

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.

MISC	Bits	R/W	Function
REQ	0	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″
TX	1	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.
CLEAR	2	R/W	Clear the requested endpoint FIFO, even the endpoint FIFO is not ready.
SELP1 SELP0	4 3	R/W	To define which endpoint FIFO is selected, SELP1,SELP0: 00: endpoint FIFO0 01: endpoint FIFO1 10: endpoint FIFO2 11: endpoint FIFO3
SCMD	5	R/W	Read only status bit, 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.
READY	6	R	Read only status bit, this bit is used to indicate that the desired endpoint FIFO is ready to work.
LEN0	7	R	Read only status bit, it is sued to indicate that a 0-sized packet is sent from host to MCU. This bit should be cleared by firmware.

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 MSIC 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.

USC	Bits	R/W	Function
SUSP	0	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.
RMWK	1	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.
URST	2	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 an USB reset is occurred (The attached bus is USB) and an USB interrupt will be initialized.
RESUME	3	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 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 token into consideration.
PS2DAI	4	R	Read only, USBD-/DATA input
PS2CKI	5	R	Read only, USBD+/CLK input
PS2DAO	6	W	Data for driving USBD-/DATA pin when work under 3D PS2 mouse function. (Default=″1″)
PS2CKO	7	W	Data for driving USBD+/CLK pin when work under 3D PS2 mouse function. (Default=″1″)

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″.

USR	Bits	R/W	Function
EP0IF	0	R/W	When this bit is set to ″1″ (set by SIE), it indicates the endpoint 0 is accessed and an USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
EP1IF	1	R/W	When this bit is set to ″1″ (set by SIE), it indicates the endpoint 1 is accessed and an USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
EP2IF	2	R/W	When this bit is set to ″1″ (set by SIE), it indicates the endpoint 2 is accessed and an USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
EP3IF	3	R/W	When this bit is set to ″1″ (set by SIE), it indicates the endpoint 3 is accessed and an USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
SPS2	4	R/W	The PS2 function is selected when this bit is set to ″1″. (Default=″0″)
SUSB	5	R/W	The USB function is selected when this bit is set to ″1″. (Default=″0″)
ADREF	6	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″)
FIFO-cntl	7	W	For ICE only, 0 for FIFO read (Default=″0″); 1 for FIFO write

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).

SCC	Bits	R/W	Function
Led_on Period	2~0	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
USBCKEN	3	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″)
SUSP2	4	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″
SYSCLK	6	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).
HWKUPSB	7	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.

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.

ADSC	Bits	Function
ACS2~ACS0	2~0	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
ADCS1 ADCS0	4 3	A/D converter clock source selection. ADCS1,ADCS0: 00: 6MHz 01: 3MHz 10: 1.5MHz 11: 0.75MHz
START	5	Start the A/D conversion. (0→1→0: start, 0→1: reset A/D converter and A/D data register)
ADON	6	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
EOCB	7	End of A/D conversion indication. (0: end of A/D conversion)

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

ADR	Bits	Function
D7~D0	7~0	Result of A/D conversion

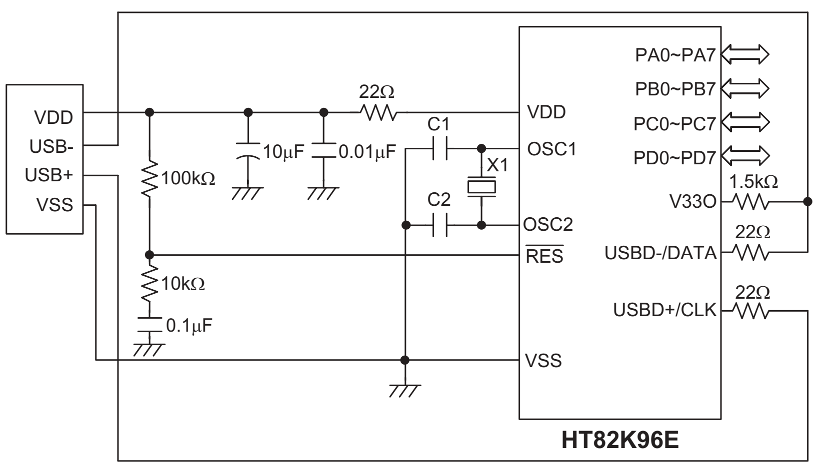
Mask options

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/disabled (by bit)
3	PA0~PA5 pull down resistor enabled/disabled (by bit)
4	PB0~PB7 pull-high resistor enabled/disabled (by nibble)
5	PC0~PC7 pull-high resistor enabled/disabled (by nibble)
6	PD0~PD7 pull-high resistor enabled/disabled (by nibble)
7	LVR enable/disable
8	WDT enable/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/disabled (by bit)
13	A/D converter enabled/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 $\overline{RES}$ to high.

0.01μF capacitor, 22Ω resistor connected to HT82K96E-VDD as close as possible.

X1 can use 6MHz or 12MHz

Instruction Set Summary

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

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

Note: x: Immediate data

m: Data memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

√: Flag is affected

–: Flag is not affected

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

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

(3): (1) and (2)

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

Instruction Definition

ADC A,[m] Add data memory and carry to the accumulator

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

Operation: ACC ← ACC+[m]+C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

ADCM A,[m] Add the accumulator and carry to data memory

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

Operation: [m] ← ACC+[m]+C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

ADD A,[m] Add data memory to the accumulator

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

Operation: ACC ← ACC+[m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

ADD A,x Add immediate data to the accumulator

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

Operation: ACC ← ACC+x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

ADDM A,[m] Add the accumulator to the data memory

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

Operation: [m] ← ACC+[m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

AND A,[m] Logical AND accumulator with data memory

Description: Data in the accumulator and the specified data memory perform a bitwise logical_AND operation. The result is stored in the accumulator.

Operation: ACC ← ACC ″AND″ [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

AND A,x Logical AND immediate data to the accumulator

Description: Data in the accumulator and the specified data perform a bitwise logical_AND operation. The result is stored in the accumulator.

Operation: ACC ← ACC ″AND″ x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

ANDM A,[m] Logical AND data memory with the accumulator

Description: Data in the specified data memory and the accumulator perform a bitwise logical_AND operation. The result is stored in the data memory.

Operation: [m] ← ACC ″AND″ [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

CALL addr Subroutine call

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

Operation: Stack ← PC+1
PC ← addr

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

CLR [m] Clear data memory

Description: The contents of the specified data memory are cleared to 0.

Operation: [m] ← 00H

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

CLR [m].i Clear bit of data memory

Description The bit i of the specified data memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

CLR WDT Clear Watchdog Timer

Description The WDT and the WDT Prescaler are cleared (re-counting from 0). The power down bit (PD) and time-out bit (TO) are cleared.

Operation WDT and WDT Prescaler ← 00H
PD and TO ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	0	0	—	—	—	—

CLR WDT1 Preclear Watchdog Timer

Description The TO, PD flags, WDT and the WDT Prescaler has cleared (re-counting from 0), if the other preclear WDT instruction has been executed. Only execution of this instruction without the other preclear instruction just sets the indicated flag which implies this instruction has been executed and the TO and PD flags remain unchanged.

Operation WDT and WDT Prescaler ← 00H*
PD and TO ← 0*

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	0*	0*	—	—	—	—

CLR WDT2 Preclear Watchdog Timer

Description The TO, PD flags, WDT and the WDT Prescaler are cleared (re-counting from 0), if the other preclear WDT instruction has been executed. Only execution of this instruction without the other preclear instruction, sets the indicated flag which implies this instruction has been executed and the TO and PD flags remain unchanged.

Operation WDT and WDT Prescaler ← 00H*
PD and TO ← 0*

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	0*	0*	—	—	—	—

CPL [m] Complement data memory

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

CPLA [m] Complement data memory and place result in the accumulator

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

Operation: ACC ← $\overline{[m]}$

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

DAA [m] Decimal-Adjust accumulator for addition

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	√

DEC [m] Decrement data memory

Description: Data in the specified data memory is decremented by 1.

Operation: [m] ← [m]–1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

DECA [m] Decrement data memory and place result in the accumulator

Description: Data in the specified data memory is decremented by 1, leaving the result in the accumulator. The contents of the data memory remain unchanged.

Operation: ACC ← [m]–1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

HALT Enter power down mode

Description This instruction stops program execution and turns off the system clock. The contents of the RAM and registers are retained. The WDT and prescaler are cleared. The power down bit (PD) is set and the WDT time-out bit (TO) is cleared.

Operation PC ← PC+1
PD ← 1
TO ← 0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	0	1	—	—	—	—

INC [m] Increment data memory

Description Data in the specified data memory is incremented by 1

Operation [m] ← [m]+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

INCA [m] Increment data memory and place result in the accumulator

Description Data in the specified data memory is incremented by 1, leaving the result in the accumulator. The contents of the data memory remain unchanged.

Operation ACC ← [m]+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

JMP addr Directly jump

Description The program counter are replaced with the directly-specified address unconditionally, and control is passed to this destination.

Operation PC ←addr

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

MOV A,[m] Move data memory to the accumulator

Description The contents of the specified data memory are copied to the accumulator.

Operation ACC ← [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

MOV A,x Move immediate data to the accumulator

Description The 8-bit data specified by the code is loaded into the accumulator.

Operation ACC ← x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

MOV [m],A Move the accumulator to data memory

Description The contents of the accumulator are copied to the specified data memory (one of the data memories).

Operation [m] ←ACC

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

NOP No operation

Description No operation is performed. Execution continues with the next instruction.

Operation PC ← PC+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

OR A,[m] Logical OR accumulator with data memory

Description Data in the accumulator and the specified data memory (one of the data memories) perform a bitwise logical_OR operation. The result is stored in the accumulator.

Operation ACC ← ACC ″OR″ [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

OR A,x Logical OR immediate data to the accumulator

Description Data in the accumulator and the specified data perform a bitwise logical_OR operation. The result is stored in the accumulator.

Operation ACC ← ACC ″OR″ x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

ORM A,[m] Logical OR data memory with the accumulator

Description Data in the data memory (one of the data memories) and the accumulator perform a bitwise logical_OR operation. The result is stored in the data memory.

Operation [m] ←ACC ″OR″ [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

RET Return from subroutine

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

Operation PC ← Stack

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

RET A,x Return and place immediate data in the accumulator

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

Operation PC ← Stack
ACC ← x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

RETI Return from interrupt

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

Operation PC ← Stack
EMI ← 1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

RL [m] Rotate data memory left

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

RLA [m] Rotate data memory left and place result in the accumulator

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

RLC [m] Rotate data memory left through carry

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	√

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

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	√

RR [m] Rotate data memory right

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

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

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

RRC [m] Rotate data memory right through carry

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	√

RRCA [m] Rotate right through carry and place result in the accumulator

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	√

SBC A,[m] Subtract data memory and carry from the accumulator

Description: The contents of the specified data memory and the complement of the carry flag are subtracted from the accumulator, leaving the result in the accumulator.

Operation: ACC ← ACC+$\overline{[m]}$+C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

SBCM A,[m] Subtract data memory and carry from the accumulator

Description: The contents of the specified data memory and the complement of the carry flag are subtracted from the accumulator, leaving the result in the data memory.

Operation: [m] ← ACC+$\overline{[m]}$+C

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

SDZ [m] Skip if decrement data memory is 0

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

Operation: Skip if ([m]–1)=0, [m] ← ([m]–1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SDZA [m] Decrement data memory and place result in ACC, skip if 0

Description: The contents of the specified data memory are decremented by 1. If the result is 0, the next instruction is skipped. The result is stored in the accumulator but the data memory remains unchanged. If the result is 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation: Skip if ([m]–1)=0, ACC ← ([m]–1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SET [m] Set data memory

Description: Each bit of the specified data memory is set to 1.

Operation: [m] ← FFH

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SET [m]. i Set bit of data memory

Description: Bit i of the specified data memory is set to 1.

Operation: [m].i ← 1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SIZ [m] Skip if increment data memory is 0

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

Operation: Skip if ([m]+1)=0, [m] ← ([m]+1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SIZA [m] Increment data memory and place result in ACC, skip if 0

Description: The contents of the specified data memory are incremented by 1. If the result is 0, the next instruction is skipped and the result is stored in the accumulator. The data memory remains unchanged. If the result is 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation: Skip if ([m]+1)=0, ACC ← ([m]+1)

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SNZ [m].i Skip if bit i of the data memory is not 0

Description: If bit i of the specified data memory is not 0, the next instruction is skipped. If bit i of the data memory is not 0, the following instruction, fetched during the current instruction execution, is discarded and a dummy cycle is replaced to get the proper instruction (2 cycles). Otherwise proceed with the next instruction (1 cycle).

Operation: Skip if [m].i≠0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SUB A,[m] Subtract data memory from the accumulator

Description: The specified data memory is subtracted from the contents of the accumulator, leaving the result in the accumulator.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

SUBM A,[m] Subtract data memory from the accumulator

Description: The specified data memory is subtracted from the contents of the accumulator, leaving the result in the data memory.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

SUB A,x Subtract immediate data from the accumulator

Description: The immediate data specified by the code is subtracted from the contents of the accumulator, leaving the result in the accumulator.

Operation: ACC ← ACC+$\overline{x}$+1

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	√	√	√	√

SWAP [m] Swap nibbles within the data memory

Description: The low-order and high-order nibbles of the specified data memory (1 of the data memories) are interchanged.

Operation: [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SWAPA [m] Swap data memory and place result in the accumulator

Description: The low-order and high-order nibbles of the specified data memory are interchanged, writing the result to the accumulator. The contents of the data memory remain unchanged.

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

SZ [m] Skip if data memory is 0

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

Operation: Skip if [m]=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

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

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

Operation: Skip if [m]=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

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

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

Operation: Skip if [m].i=0

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

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

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

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

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

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

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	—	—	—

XOR A,[m] Logical XOR accumulator with data memory

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

Operation: ACC ← ACC ″XOR″ [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

XORM A,[m] Logical XOR data memory with the accumulator

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

Operation: [m] ← ACC ″XOR″ [m]

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

XOR A,x Logical XOR immediate data to the accumulator

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

Operation: ACC ← ACC ″XOR″ x

Affected flag(s)

TC2	TC1	TO	PD	OV	Z	AC	C
—	—	—	—	—	√	—	—

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

Holtek Semiconductor Inc. (Sales Office)
11F, No.576, Sec.7 Chung Hsiao E. Rd., Taipei, Taiwan
Tel: 886-2-2782-9635
Fax: 886-2-2782-9636
Fax: 886-2-2782-7128 (International sales hotline)

Holtek Semiconductor (Hong Kong) Ltd.
RM.711, Tower 2, Cheung Sha Wan Plaza, 833 Cheung Sha Wan Rd., Kowloon, Hong Kong
Tel: 852-2-745-8288
Fax: 852-2-742-8657

Holtek Semiconductor (Shanghai) Inc.
7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China
Tel: 021-6485-5560
Fax: 021-6485-0313
http://www.holtek.com.cn

Holmate Technology Corp.
48531 Warm Springs Boulevard, Suite 413, Fremont, CA 94539
Tel: 510-252-9880
Fax: 510-252-9885
http://www.holmate.com