HT82M99EE/HT82M99AE
USB Mouse Encoder 8-Bit MCU

Technical Document

- Tools Information
- FAQs
- Application Note

Features

- Flexible total solution for applications that combine PS/2 and low-speed USB interface, such as mice, joysticks, and many others
- USB Specification Compliance
  - Conforms to USB specification V1.1
  - Conforms to USB HID specification V1.1
- Supports 1 Low-speed USB control endpoint and 1 interrupt endpoint
- Each endpoint has 8×8 bytes FIFO
- Integrated USB transceiver
- 3.3V regulator output
- External 6MHz or 12MHz ceramic resonator or crystal
- 8-bit RISC microcontroller, with 2K×14 program memory (000H~7FFH)
- 96 bytes RAM (20H~7FH)
- 128×8 data EEPROM
- 6MHz/12MHz internal CPU clock
- 4-level stacks
- Two 7-bit indirect addressing registers
- One 16-bit programmable timer counter with overflow interrupt (shared with PA7, vector 0CH)
- One USB interrupt input (vector 04H)
- HALT function and wake-up feature reduce power consumption
- PA0~PA7, PB4 and PB7 support wake-up function
- Internal Power-On reset (POR)
- Watchdog Timer (WDT)
- 12 I/O ports
- 20-pin DIP/SOP package

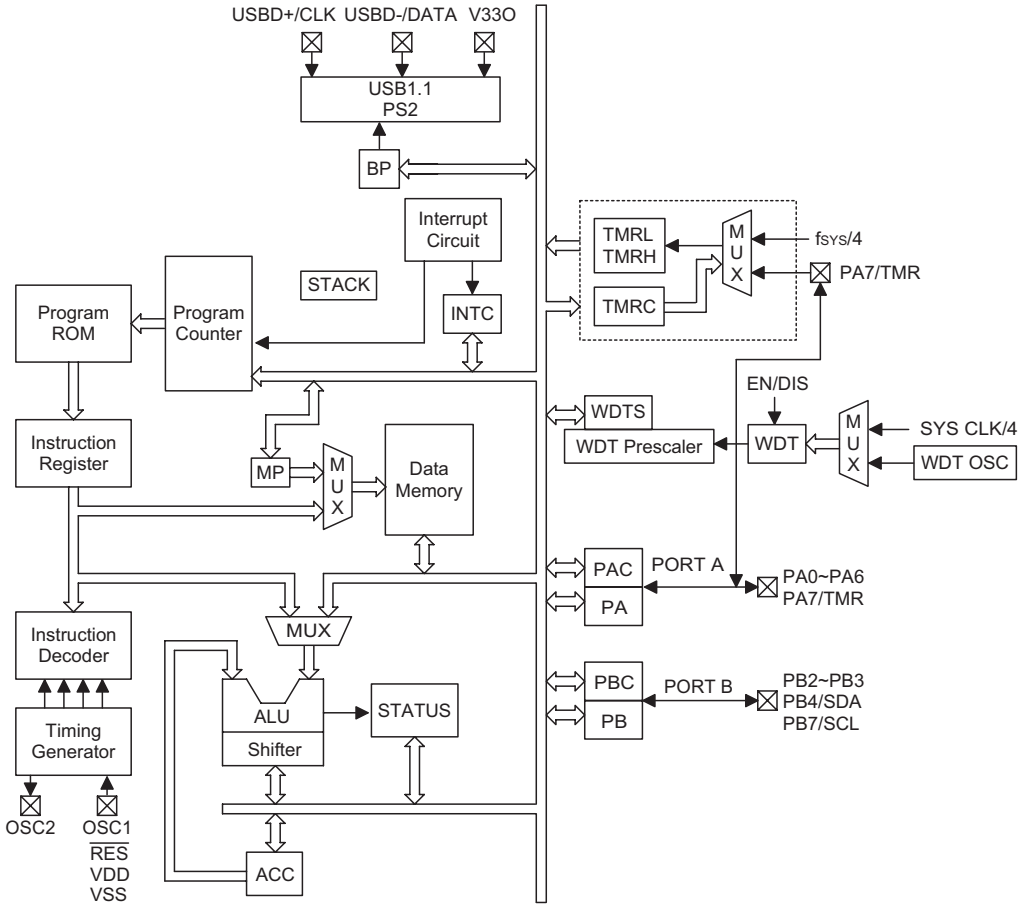
General Description

The USB MCU HT82M99EE/HT82M99AE are suitable for USB mouse and USB joystick devices. It consists of a Holtek high performance 8-bit MCU core for control unit, built-in USB SIE, 2K×14 program memory and 96 bytes data RAM.

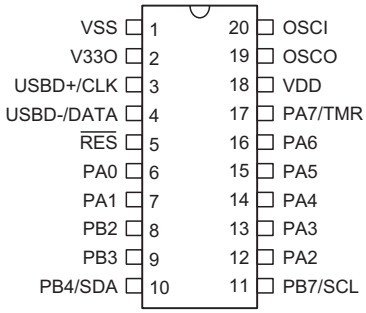
The mask version HT82M99AE is fully pin and function-ally compatible with the OTP version HT82M99EE device.

There are two dice in the HT82M99EE/HT82M99AE package: one is the HT82M99EE/HT82M99AE MCU, the other is a 128×8 bits EEPROM used for data memory purpose. The two dice are wire-bonded to form HT82M99EE/HT82M99AE.

Block Diagram



Pin Assignment



HT82M99EE/HT82M99AE
– 20 DIP-A/SOP-A

Pin Description

Pin Name	I/O	ROM Code Option	Description
PA0~PA6 PA7/TMR	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~PA3 CMOS/NMOS/PMOS options: PA0~PA7 Falling edge wake-up options: PA0~PA1, PA4~PA7 Rising and falling edge wake-up options: PA2~PA3 PA7 is wire-bonded with TMR
PB2~PB3	I/O	Pull-high Pull-low	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). Pull-low resistor for options: PB2, PB3
PB4/SDA, PB7/SCL	I/O	Pull-high Wake-up	Bidirectional 8-bit input/output port. Software instructions determine the CMOS output or Schmitt trigger input with pull-high resistor (determined by pull-high options). The PB can be used as analog input of the analog to digital converter (determined by options). Falling edge wake-up options: PB4, PB7 PB4 is wire-bonded with SDA pad of the Data EEPROM PB7 is wire-bonded with SCL pad of the Data EEPROM
VSS	—	—	Negative power supply, ground
$\overline{RES}$	I	—	Schmitt trigger reset input. Active low.
VDD	—	—	Positive power supply
V33O	O	—	3.3V regulator output
USBD+/CLK	I/O	—	USBD+ or PS2 CLK I/O line USB or PS2 function is controlled by software control register
USBD-/DATA	I/O	—	USBD- or PS2 DATA I/O line USB or PS2 function is controlled by software control register
OSCI OSCO	I O	—	OSCI, OSCO are connected to a 6MHz or 12MHz crystal/resonator (determined by software instructions) for the internal system clock.

Absolute Maximum Ratings

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

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

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

Operating Temperature ..... 0°C to 70°C

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	—	4	—	5.5	V
I_{DD}	Operating Current (6MHz Crystal)	5V	No load, f_{SYS}=6MHz	—	7	9	mA
I_{STB}	Standby Current	5V	No load, system HALT	—	300	500	μA
V_{IL1}	Input Low Voltage for I/O Ports	5V	—	0	—	0.8	V
V_{IH1}	Input High Voltage for I/O Ports	5V	—	2	—	5	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	5V	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	5V	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	Output Sink Current for Other Ports PA0~PA7, PB2~PB4, PB7	5V	V_{OL}=0.4V	2	4	—	mA
I_{OH}	Output Port Source Current	5V	V_{OL}=3.4V	–2.5	–4	—	mA
R_{PD}	Pull-down Resistance for PA0~PA3, PB2 and PB3	5V	—	10	30	50	kΩ
R_{PH1}	Pull-high Resistance for CLK and DATA	—	—	2	4.7	6	kΩ
R_{PH2}	Pull-high Resistance for PA0~PA7, PB2~PB4, PB7	—	—	30	50	70	kΩ
V_{LVR}	Low Voltage Reset	5V	—	2.4	2.7	3	V

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_{RCSYS}	RC Clock with 8-bit Prescaler Register	5V	—	0	32	—	kHz
t_{WDT}	Watchdog Time-out Period (System Clock)	—	Without WDT prescaler	1024	—	—	t_{RCSYS}
t_{RF}	USBD+, USBD- Rising & falling Time	—	—	75	—	300	ns
t_{SST}	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	t_{SYS}
t_{OSC}	Crystal Setup	—	—	—	5	10	ms

Note: Power-on period=t_{WDT}+t_{SST}+t_{OSC}

WDT Time-out in normal mode=1/f_{RCSYS}×256×WDTS+t_{WDT}

WDT Time-out in HALT mode=1/f_{RCSYS}×256×WDTS+t_{SST}+t_{OSC}

EEPROM A.C. Characteristics Ta=25°C

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

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

* The standard mode means V_{CC}=2.2V to 5.5V

For relative timing, refer to timing diagrams

Functional Description

Execution Flow

The system clock for the microcontroller is derived from either 6MHz or 12MHz crystal oscillator, which used a frequency that is determined by the SCLKSEL bit of the SCC Register. The default system frequency is 12MHz. 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 be effectively executed in a cycle. If an instruction changes the program counter, two cycles are required to complete the instruction.

Program Counter – PC

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

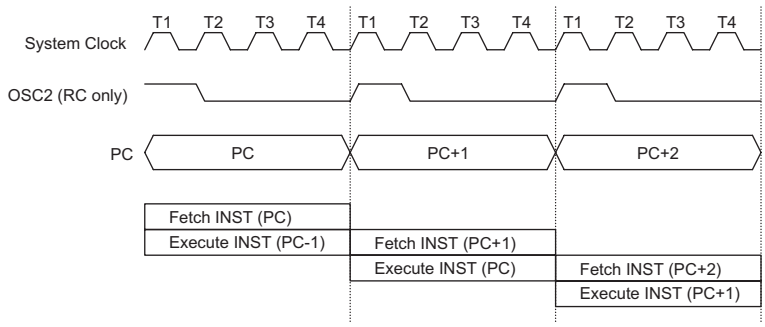
After accessing a program memory word to fetch an instruction code, the contents of the program counter are incremented by one. The program counter then points to the memory word containing the next instruction code.

When executing a jump instruction, conditional skip execution, loading to the PCL register, performing a 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 with the next instruction.

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

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



Execution Flow

Mode	Program Counter										
	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial Reset	0	0	0	0	0	0	0	0	0	0	0
USB Interrupt	0	0	0	0	0	0	0	0	1	0	0
Timer/Event Counter Overflow	0	0	0	0	0	0	0	1	1	0	0
Skip	Program Counter+2										
Loading PCL	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

Note: *10~*0: Program counter bits
#10~#0: Instruction code bits
S10~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 2048×14 bits, addressed by the program counter and table pointer.

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

- Location 000H

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

- Location 004H

  This area is reserved for the 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 00CH

  This location is reserved for the Timer/Event Counter interrupt service program. If a timer interrupt results from a Timer/Event Counter overflow, and the interrupt is enabled and the stack is not full, the program begins execution at location 00CH.

- Table location

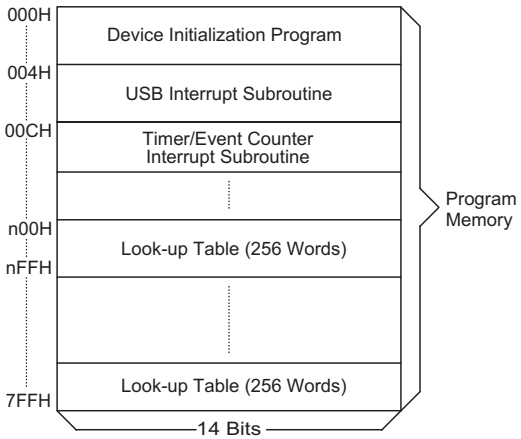
  Any location in the program memory can be used as look-up tables. There are three method to read the ROM data by two table read instructions: "TABRDC" and "TABRDL", transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH (08H).

  The three methods are shown as follows:

  - The instructions "TABRDC [m]" (the current page, one page=256words), where the table locations is defined by TBLP (07H) in the current page. And the ROM code option TBHP is disabled (default).
  - The instructions "TABRDC [m]", where the table locations is defined by registers TBLP (07H) and TBHP (01FH). And the ROM code option TBHP is enabled.
  - The instructions "TABRDL [m]", where the table locations is defined by Registers TBLP (07H) in the last page (0700H~07FFH).

  Only the destination of the lower-order byte in the table is well-defined, the other bits of the table word are transferred to the lower portion of TBLH, and the remaining 1-bit words are read as "0". The Table Higher-order byte register (TBLH) is read only. The table pointer (TBLP, TBHP) is a read/write register (07H, 1FH), which indicates the table location. Before accessing the table, the location must be placed in the TBLP and TBHP (If the OTP option TBHP is disabled, the value in TBHP has no effect). The TBLH is read only and cannot be restored. If the main routine and the ISR (Interrupt Service Routine) both employ the table read instruction, the contents of the TBLH in the main routine are likely to be changed by the table read instruction used in the ISR. Errors can occur. In other words, using the table read instruction in the main routine and the ISR simultaneously should be avoided. However, if the table read instruction has to be applied in both the main routine and the ISR, the interrupt should be disabled prior to the table read instruction. It will not be enabled until the TBLH has been backed up. All table related instructions require two cycles to complete the operation. These areas may function as normal program memory depending on the requirements.

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



Note: n ranges from 0 to 7

Program Memory

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

Table Location

Note: *10~*0: Table location bits

P10~P8: Current program counter bits when TBHP is disabled

@7~@0: TBLP bits

TBHP register bit2~bit0 when TBHP is enabled

Stack Register – STACK

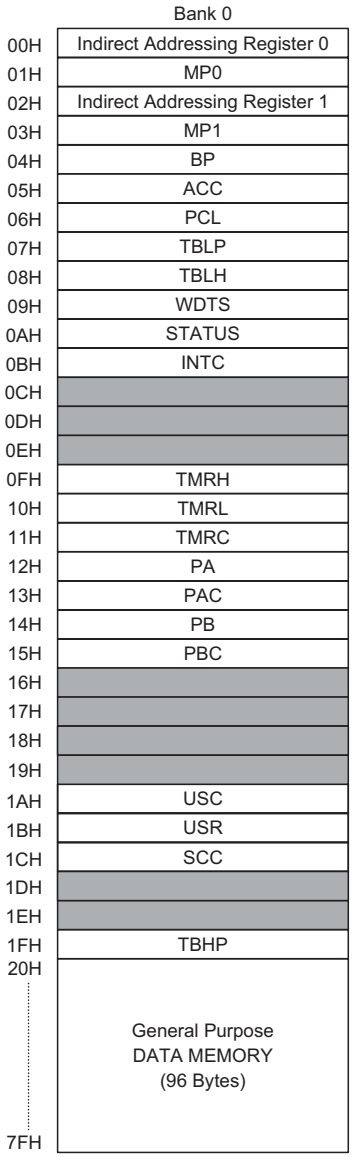
This is a special part of the memory which is used to save the contents of the program counter only. The stack is organized into 4 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the stack pointer (SP) and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

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

Data Memory – RAM for Bank 0

The data memory is designed with 96×8 bits. The data memory is divided into two functional groups: special function registers and general purpose data memory (96×8). 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 higher order byte register (TMRH;0FH), Timer/Event Counter lower order byte register (TMRL;10H), Timer/Event Counter control register (TMRC;11H), program counter lower-order byte register (PCL;06H), memory pointer registers (MP0;01H, MP1;03H), accumulator (ACC;05H), table pointers (TBLP;07H, TBHP;1FH), 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), I/O control registers (PAC;13H, PBC;15H). USB/PS2 status and control register (USC;1AH), USB endpoint interrupt status register (USR;1BH), system clock control register (SCC;1CH). The remaining space before the 20H is reserved for future expanded usage and reading these locations will get "00H". The general purpose data memory, addressed from 20H to 7FH, is used for data and control information under instruction commands.



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

	Bank 1
00H	Indirect Addressing Register 0
01H	MP0
02H	Indirect Addressing Register 1
03H	MP1
04H	BP
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	WDTS
0AH	STATUS
0BH	INTC
0CH	
0DH	
0EH	
0FH	TMRH
10H	TMRL
11H	TMRC
12H	PA
13H	PAC
14H	PB
15H	PBC
16H	
17H	
18H	
19H	
1AH	USC
1BH	USR
1CH	SCC
1DH	
1EH	
1FH	TBHP
20H	
41H	Pipe_ctrl
42H	AWR
43H	STALL
44H	PIPE
45H	SIES
46H	MISC
47H	
48H	FIFO 0
49H	FIFO 1

Bank 1 RAM Mapping

Address 00~1FH in RAM Bank0 and Bank1 are located in the same Registers

Indirect Addressing Register

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

The indirect addressing pointer (MP0) always points 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 to the value of BP which is set to ″0″ or ″1″ respectively.

The memory pointer registers (MP0 and MP1) are 7-bit registers.

Accumulator

The accumulator is closely related to ALU operations. It is also mapped to location 05H of the data memory and can carry out immediate data operations. The data movement between two data memory locations must pass through the accumulator.

Arithmetic and Logic Unit – ALU

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

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

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

Status Register – STATUS

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

With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like most other registers. Any data written into the status register will not change the TO or PDF flag. In addition, operations related to the status register may give different results from those intended.

Bit No.	Label	Function
0	C	C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
1	AC	AC is set if an operation results in a carry out of the low nibbles in addition or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
2	Z	Z is set if the result of an arithmetic or logic operation is zero; otherwise Z is cleared.
3	OV	OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
4	PDF	PDF is cleared by a system power-up or executing the ″CLR WDT″ instruction. PDF is set by executing the ″HALT″ instruction.
5	TO	TO is cleared by a system power-up or executing the ″CLR WDT″ or ″HALT″ instruction. TO is set by a WDT time-out.
6~7	—	Unused bit, read as ″0″

Status (0AH) Register

The TO flag can be affected only by a system power-up, a WDT time-out or executing the ″CLR WDT″ or ″HALT″ instruction. The PDF flag can be affected only by executing the ″HALT″ or ″CLR WDT″ instruction or during a system power-up.

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

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

Interrupt

The 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 a 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.

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

- Access of the corresponding USB FIFO from PC
- The USB suspend signal from PC
- The USB resume signal from PC
- USB Reset signal

Bit No.	Label	Function
0	EMI	Controls the master (global) interrupt (1=enable; 0=disable)
1	EUI	Controls the USB interrupt (1=enable; 0= disable)
2, 5, 7	—	Unused bit, read as ″0″
3	ETI	Controls the Timer/Event Counter interrupt (1=enable; 0=disable)
4	USBF	USB interrupt request flag (1=active; 0=inactive)
6	TF	Internal timer/event counter request flag (1:active; 0:inactive)

INTC (0BH) Register

When the interrupt is enabled, the stack is not full and the external interrupt is active, a subroutine call to location 04H will occur. The interrupt request flag (USBF) and EMI bits will be cleared to disable other interrupts.

When the PC Host access the FIFO of the HT82M99EE/HT82M99AE, the corresponding request bit of the USR is set, and a USB interrupt is triggered. So user can easily decide which FIFO is accessed. When the interrupt has been served, the corresponding bit should be cleared by firmware. When the HT8M99E receives a USB Suspend signal from the Host PC, the suspend line (bit0 of the USC) of the HT8M99E is set and a USB interrupt is also triggered.

When the HT82M99EE/HT82M99AE receives a Resume signal from the Host PC, the resume line (bit3 of the USC) of the HT82M99EE/HT82M99AE are set and a USB interrupt is triggered.

Whenever a USB reset signal is detected, the USB interrupt is triggered and URST_Flag bit of the USC register is set. When the interrupt has been served, the bit should be cleared by firmware.

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

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

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.

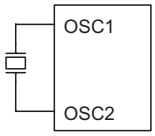
No.	Interrupt Source	Priority	Vector
a	USB interrupt	1	04H
b	Timer/Event Counter overflow	2	0CH

The timer/event counter interrupt request flag (TF), USB interrupt request flag (USBF), enable timer/event counter interrupt bit (ETI), 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 and ETI are used to control the enabling/disabling of interrupts. These bits prevent the requested interrupt from being serviced. Once the interrupt request flags (TF, USBF) are set, they will remain in the INTC register until the interrupts are serviced or cleared by a software instruction.

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

Oscillator Configuration

There is an oscillator circuit 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.

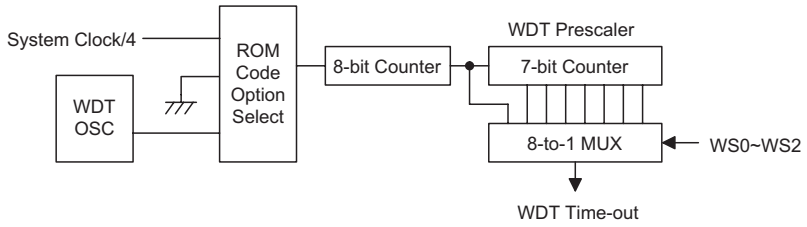
The HT82M99EE/HT82M99AE can operate in 6MHz or 12MHz system clocks. In order to make sure that the USB SIE functions properly, user should correctly configure the SCLKSEL bit of the SCC Register. The default system clock is 12MHz.

The WDT oscillator is a free running on-chip RC oscillator, and no external components are required. Even if the system enters the power down mode, the system clock is stopped, but the WDT oscillator still works within a period of approximately 31μs. The WDT oscillator can be disabled by ROM code option to conserve power.

Watchdog Timer – WDT

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

Once the internal WDT oscillator (RC oscillator with a period of 31μs/5V normally) is selected, it is first divided by 256 (8-stage) to get the nominal time-out period of



Watchdog Timer

8ms/5V. This time-out period may vary with temperatures, VDD and process variations. By invoking the WDT prescaler, longer time-out periods can be realized. Writing data to WS2, WS1, WS0 (bits 2, 1, 0 of the WDTS) can give different time-out periods. If WS2, WS1, and WS0 are all equal to 1, the division ratio is up to 1:128, and the maximum time-out period is 1s/5V. If the WDT oscillator is disabled, the WDT clock may still come from the instruction clock and operates in the same manner except that in the HALT state the WDT may stop counting and lose its protecting purpose. In this situation the logic can only be restarted by external logic. The high nibble and bit 3 of the WDTS are reserved for user defined flags, which can only be set to "10000" (WDTS.7~WDTS.3).

If the device operates in a noisy environment, using the on-chip 32kHz RC oscillator (WDT OSC) is strongly recommended, since the HALT will stop the system clock.

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

WDTS (09H) Register

The WDT overflow under normal operation will initialize a "chip reset" and set the status bit "TO". But in the HALT mode, the overflow will initialize a "warm reset" and only the program counter and SP are reset to zero. To clear the contents of the WDT (including the WDT prescaler), three methods are adopted; external reset (a low level to $\overline{RES}$), software instruction and a "HALT" instruction. The software instruction include "CLR WDT" and the other set – "CLR WDT1" and "CLR WDT2". Of these two types of instruction, only one can be active depending on the ROM code option – "CLR WDT times selection option". If the "CLR WDT" is selected (i.e. CLRWDT times is equal to 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 is equal to two), these two instructions must be executed to clear the WDT; otherwise, the WDT may reset the chip as a result of time-out.

Power Down Operation – HALT

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

- The system oscillator will be turned off but the WDT oscillator remains running (if the WDT oscillator is selected).
- The contents of the on-chip RAM and registers remain unchanged.
- The 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 remain in their original status.
- The PDF flag is set and the TO flag is cleared.

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

The port A wake-up and interrupt methods can be considered as a continuation of normal execution. Each bit in port A can be independently selected to wake-up the device by mask option. Awakening from an I/O port stimulus, the program will resume execution of the next instruction. If it awakens from an interrupt, two sequence may occur. If the related interrupt is disabled or the interrupt is enabled but the stack is full, the program will resume execution at the next instruction. If the interrupt is enabled and the stack is not full, the regular interrupt response takes place. If an interrupt request flag is set to "1" before entering the HALT mode, the wake-up function of the related interrupt will be disabled. Once a wake-up event occurs, it takes 1024 t_{SYS} (system clock period) to resume normal operation. In other words, a dummy period will be inserted after a wake-up. If the wake-up results from an interrupt acknowledge signal, the actual interrupt subroutine execution will be delayed

by one or more cycles. If the wake-up results in the next instruction execution, this will be executed immediately after the dummy period is finished.

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

Reset

There are four ways in which a reset can occur:

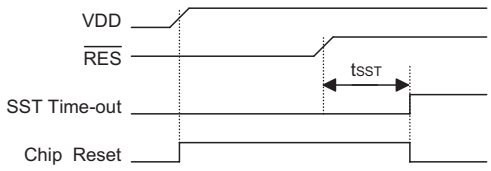
- $\overline{RES}$ reset during normal operation
- $\overline{RES}$ reset during HALT
- WDT time-out reset during normal operation
- USB reset

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

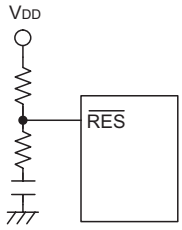
TO	PDF	RESET Conditions
0	0	$\overline{RES}$ reset during power-up
u	u	$\overline{RES}$ reset during normal operation
0	1	$\overline{RES}$ wake-up HALT
1	u	WDT time-out during normal operation
1	1	WDT wake-up HALT

Note: "u" stands for "unchanged"

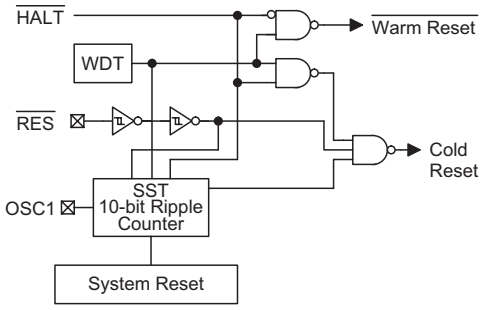
To guarantee that the system oscillator is started and stabilized, the SST (System Start-up Timer) provides an extra delay of 1024 system clock pulses when the system resets (power-up, WDT time-out or $\overline{RES}$ reset) or the system awakes from the HALT state.



Reset Timing Chart



Reset Circuit



Reset Configuration

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

The functional unit chip reset status are shown below.

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

The registers status are summarized in the following 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)
TMRH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMRL	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMRC	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	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu	1uuu uuuu	1uuu uuuu	1uuu uuuu
MP1	1xxx xxxx	1uuu uuuu	1uuu uuuu	1uuu uuuu	1uuu uuuu	1uuu uuuu	1uuu 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	1xx1 11xx	1xx1 11xx	1xx1 11xx	1xx1 11xx	uuuu uuuu	1xx1 11xx	1xx1 11xx
PBC	1xx1 11xx	1xx1 11xx	1xx1 11xx	1xx1 11xx	uuuu uuuu	1xx1 11xx	1xx1 11xx
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
SIES	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
MISC	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
FIFO0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
USC	11xx 0000	uuxx uuuu	11xx 0000	11xx 0000	uuxx uuuu	1100 0u00	1100 0u00
USR	0100 0000	uuuu uuuu	0100 0000	0100 0000	uuuu uuuu	u1uu 0000	u1uu 0000
SCC	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	uu00 u000	uu00 u000

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

Timer/Event Counter

A timer/event counter (TMR) is implemented in the microcontroller.

The timer/event counter contains a 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 the timer/event counter. The internal clock source is coming from $f_{SYS}/4$. The external clock input allows the user to count external events, measure time intervals or pulse widths.

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

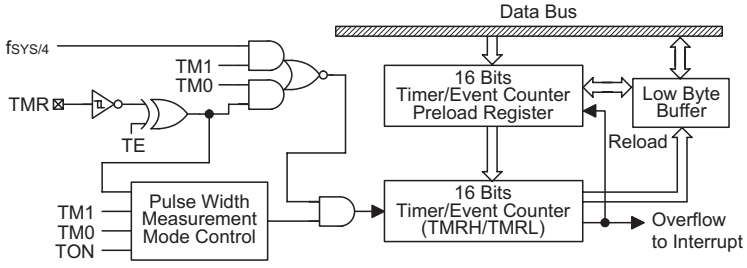
The TM0, TM1 bits define the operating mode. The event count mode is used to count external events, which means that the clock source comes from an external (TMR) pin. The timer mode functions as a normal timer with the clock source coming from the $f_{SYS}/4$ (Timer). The pulse width measurement mode can be used to count the high or low level duration of the external signal (TMR). The counting is based on the $f_{SYS}/4$.

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

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

Bit No.	Label	Function
0~2, 5	—	Unused bit, read as "0"
3	TE	Defines the TMR active edge of the timer/event counter (0=active on low to high; 1=active on high to low)
4	TON	Enable/disable the timer counting (0=disable; 1=enable)
6 7	TM0 TM1	Defines the operating mode 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused

TMRC (11H) Register



Timer/Event Counter

tomatically 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 is one of the wake-up sources. No matter what the operation mode is, writing a ″0″ to ET can disable the corresponding interrupt services.

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

Input/Output Ports

There are 12 bidirectional input/output lines in the microcontroller, labeled from PA to PB, which are mapped to the data memory of [12H] and [14H] 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 or 14H). 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 and PBC) 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 under software control. To function as an input, the corresponding latch of the control register must write a ″1″. The input source also depends on the control register. If the control register bit is ″1″, the input will read the pad state. If the control register bit is ″0″, the contents of the latches will move to the internal bus. The latter is possible in the ″read-modify-write″ instruction. For output function, CMOS/NMOS/PMOS configurations can be selected (NMOS and PMOS are available for PA only). These control registers are mapped to locations 13H and 15H.

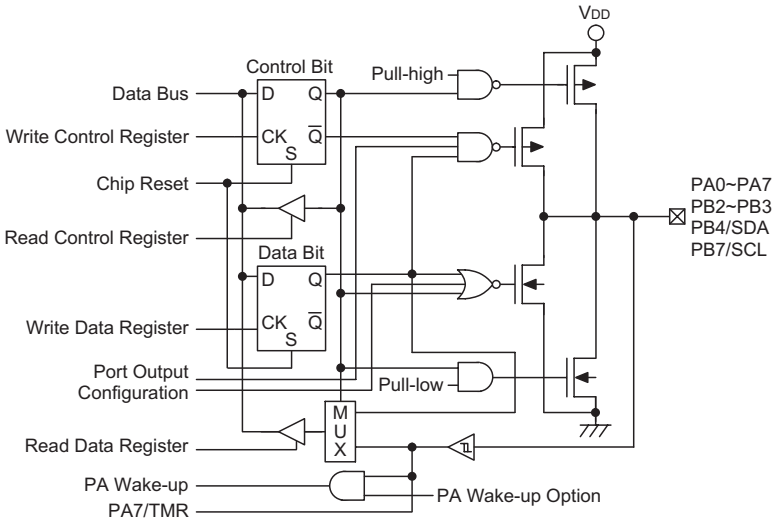
After a chip reset, these input/output lines remain at high levels or in a floating state (depending on the pull-high/low options). Each bit of these input/output latches can be set or cleared by ″SET [m].i″ and ″CLR [m].i″ (m=12H or 14H) instructions.

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

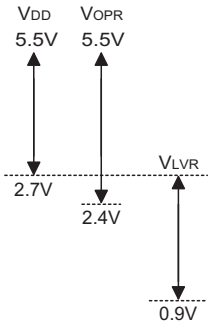
Low Voltage Reset – LVR

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

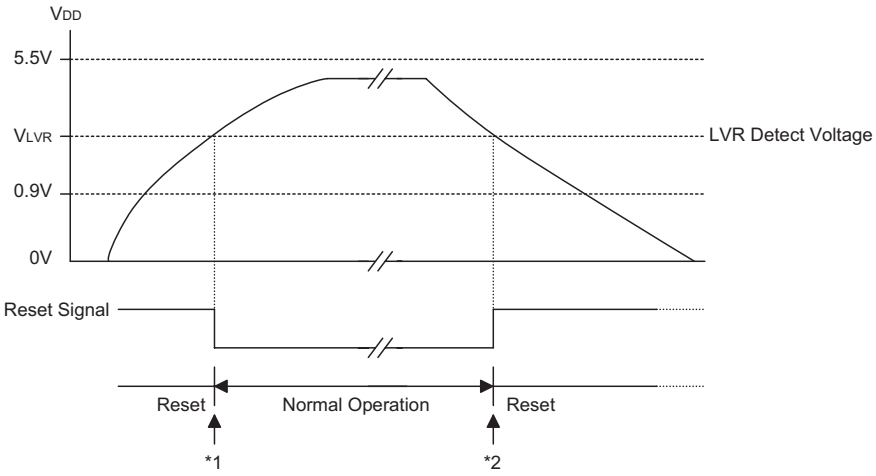
The LVR includes the following specifications:

- For a valid LVR signal, a low voltage (0.9V~V_{LVR}) must exist for more than 1ms. If the low voltage state does not exceed 1ms, the LVR will ignore it and will not perform a reset function.
- The LVR uses the "OR" function with the external $\overline{RES}$ signal to perform a chip reset.

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



Note: V_{OPR} is the voltage range for proper chip operation at 6MHz or 12MHz 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: A low voltage has to exist for more than 1ms, after that 1ms delay, the device enters a reset mode.

Data EEPROM Functional Description

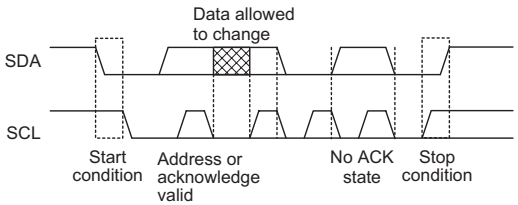
- Serial clock (SCL)
  The SCL input is used for positive edge clock data into each EEPROM device and negative edge clock data out of each device.
- Serial data (SDA)
  The SDA pin is bidirectional for serial data transfer. The pin is open-drain driven and may be wired-OR with any number of other open-drain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

Memory Organization

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

Device Operations

- Clock and data transition
  Data transfer may be initiated only when the bus is not busy. During data transfer, the data line must remain stable whenever the clock line is high. Changes in data line while the clock line is high will be interpreted as a START or STOP condition.
- Start condition
  A high-to-low transition of SDA with SCL high is a start condition which must precede any other command (refer to Start and Stop Definition Timing diagram).
- Stop condition
  A low-to-high transition of SDA with SCL high is a stop condition. After a read sequence, the stop command will place the EEPROM in a standby power mode (refer to Start and Stop Definition Timing Diagram).
- Acknowledge
  All addresses and data words are serially transmitted to and from the EEPROM in 8-bit words. The EEPROM sends a zero to acknowledge that it has received each word. This happens during the ninth clock cycle.



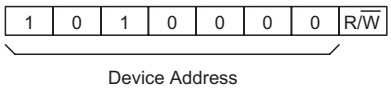
Device Addressing

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

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

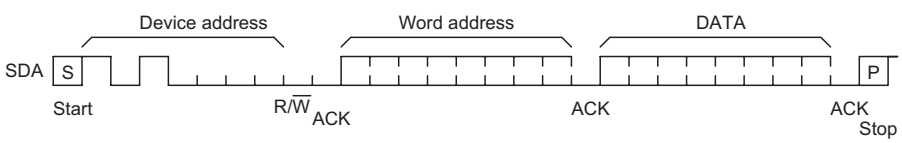
The 8th bit of device address is the read/write operation select bit. A read operation is initiated if this bit is high and a write operation is initiated if this bit is low.

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

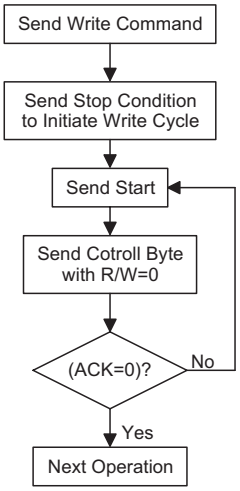


Write Operations

- Byte write
  A write operation requires an 8-bit data word address following the device address word and acknowledgment. Upon receipt of this address, the EEPROM will again respond with a zero and then clock in the first 8-bit data word. After receiving the 8-bit data word, the EEPROM will output a zero and the addressing device, such as a microcontroller, must terminate the write sequence with a stop condition. At this time the EEPROM enters an internally-timed write cycle to the non-volatile memory. All inputs are disabled during this write cycle and EEPROM will not respond until the write is completed (refer to Byte write timing).
- Acknowledge polling
  To maximise bus throughput, one technique is to allow the master to poll for an acknowledge signal after the start condition and the control byte for a write command have been sent. If the device is still busy implementing its write cycle, then no ACK will be returned. The master can send the next read/write command when the ACK signal has finally been received.

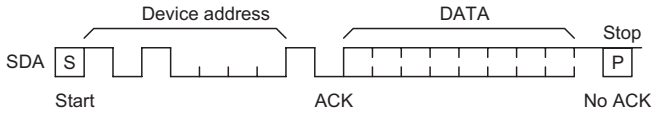


Byte Write Timing

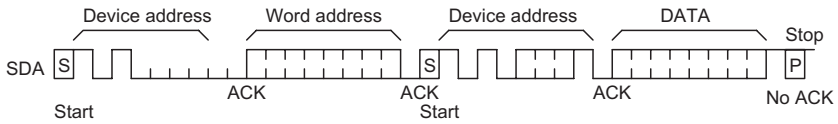


Acknowledge Polling Flow

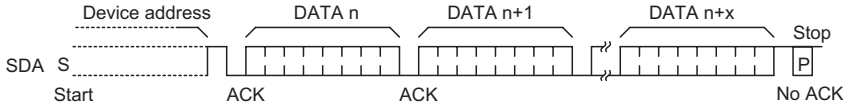
- Read operations
  The data EEPROM supports three read operations, namely, current address read, random address read and sequential read. During read operation execution, the read/write select bit should be set to ″1″.
- Current address read
  The internal data word address counter maintains the last address accessed during the last read or write operation, incremented by one. This address stays valid between operations as long as the chip power is maintained. The address roll over during read from the last byte of the last memory page to the first byte of the first page. The address roll over during write from the last byte of the current page to the first byte of the same page. Once the device address with the read/write select bit set to one is clocked in and acknowledged by the EEPROM, the current address data word is serially clocked out. The microcontroller should respond a No ACK (High) signal and following stop condition (refer to Current read timing).
- Random read
  A random read requires a dummy byte write sequence to load in the data word address which is then clocked in and acknowledged by the EEPROM. The microcontroller must then generate another start condition. The microcontroller now initiates a current address read by sending a device address with the read/write select bit high. The EEPROM acknowledges the device address and serially clocks out the data word. The microcontroller should respond with a ″no ACK″ signal (high) followed by a stop condition. (refer to Random read timing).
- Sequential read
  Sequential reads are initiated by either a current address read or a random address read. After the microcontroller receives a data word, it responds with an acknowledgment. As long as the EEPROM receives an acknowledgment, it will continue to increment the data word address and serially clock out sequential data words. When the memory address limit is reached, the data word address will roll over and the sequential read continues. The sequential read operation is terminated when the microcontroller responds with a ″no ACK″ signal (high) followed by a stop condition.



Current Read Timing

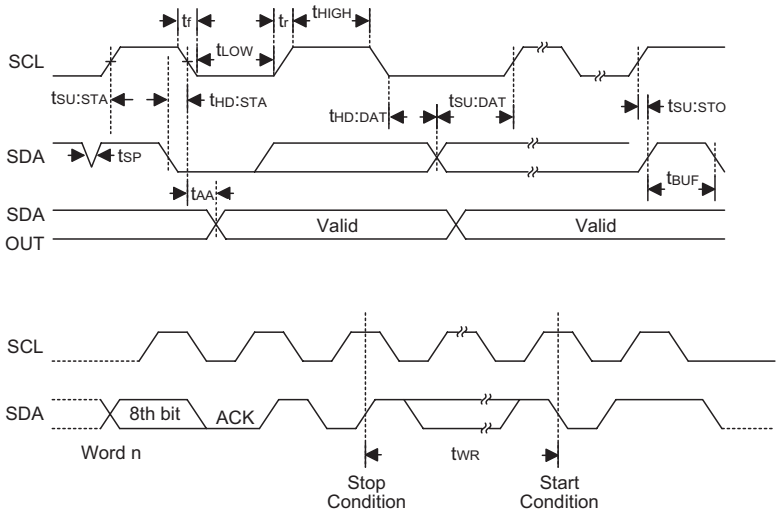


Random Read Timing



Sequential Read Timing

Data EEPROM Timing Diagrams



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

USB with MCU Interface

There are eight registers, including Pipe_ctrl, Address+Remote_WakeUp, STALL, PIPE, SIES, MISC, FIFO 0 and FIFO 1 in this buffer function.

Register Name	Pipe_ctrl	Addr.+Remote	STALL	PIPE	SIES	MISC	FIFO 0	FIFO 1
Mem. Addr.	41H	42H	43H	44H	45H	46H	48H	49H
Reserved Addr.	Bank 1, Address 40H, 4AH, 4FH							

Register Memory Mapping

Address+Remote_WakeUp register represents current address and remote wake-up function. The initial value is ″00000000″ from MSB to LSB.

Register Address	R/W	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
01000010B	R/W	Address value Default value=00000000							Remote Wake-up Function 0: Not this function 1: The function exists

Address+Remote_WakeUp Register

The Pipe_ctrl, STALL and PIPE are bitmap ones. The Pipe_ctrl Register is used for configuring IN (Bit=1) or OUT (Bit=0) Pipe. The default is defined as IN Pipe. The PIPE register represents whether the corresponding endpoint is accessed by host or not. After a USB interrupt signal is being sent out, the MCU can check which endpoint had been accessed. This register is set only after the host accessed the corresponding endpoint. The STALL register shows whether the corresponding endpoint works or not. As soon as the endpoint works improperly, the corresponding bit must be set. The bitmaps are listed as follows:

Register Name	R/W	Register Address	Bit7~Bit2 Reserved	Bit 1	Bit 0	Default Value
Pipe_ctrl	R/W	01000001B	—	Pipe 1	Pipe 0	00000011
STALL	R/W	01000011B	—	Pipe 1	Pipe 0	00000000
PIPE	R	01000100B	—	Pipe 1	Pipe 0	00000000

STALL (43H) and PIPE (44H) Registers

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

Bit No.	Function	Read/Write	Register Address
7	MNI	R/W	01000101B
6	EOT	R	
5	CRC_ERR	R/W	
4	NAK	R	
3	IN	R	
2	OUT	R/W	
1	F0_ERR	R/W	
0	Adr_set	R/W	

SIES (45H) Registers Table

Function Name	Read/Write	Description
Adr_set	R/W	This bit is used to configure the USB SIE to automatically change the device address with the value of the Address+Remote_WakeUp Register (42H). When this bit is set to 1 by F/W, the USB SIE will update the device address with the value of the Address+Remote_WakeUp Register (42H) after the PC Host has successfully read the data from the device by the IN operation. The USB SIE will clear the bit after updating the device address. Otherwise, when this bit is cleared to ″0″, the USB SIE will update the device address immediately after an address is written to the Address+Remote_WakeUp Register (42H).
F0_Err	R/W	This bit is used to indicate when there are some errors that occurred when the FIFO0 is accessed. This bit is set by the USB SIE and cleared by F/W.
Out	R/W	This bit is used to indicate that there are OUT token (except for the OUT zero) that has been received. The F/W clears the bit after the OUT data has been read. Also, this bit will be cleared by the USB SIE after the next valid SETUP token is received.
IN	R	This bit is used to indicate that the current USB receiving signal from the PC Host is IN token.
NAK	R	This bit is used to indicate that the USB SIE has transmitted the NAK signal to the Host in response to the PC Host IN or OUT token.
CRC_err	R/W	This bit indicates that there are CRC error (bit=1). The programmer must do something to save the device and keep it alive. This bit is set by the USB SIE and cleared by F/W.
EOT	R	End of transient flag, normal status is ″1″. If suspend=″1″ line & EOT=″0″ indicates that something is wrong in the USB Interface. The programmer must do something to save the device and keep it alive.
MNI	R/W	This bit is for masking the NAK interrupt when MNI=″1″, the default value=″0″

SIES Function Table

The MISC register is actually a command + status to control the desired FIFO action and to show the status of the desired FIFO. Every bit's meaning and usage are listed as follows:

Bit No.	Function	Read/Write	Register Address
7	Len0	R/W	01000110B
6	Ready	R	
5	Set CMD	R/W	
4	Sel_pipe1	R/W	
3	Sel_pipe0	R/W	
2	Clear	R/W	
1	Tx	R/W	
0	Request	R/W	

MISC (46H) Registers Table

Function Name	Read/Write	Description
Request	R/W	After setting the other desired status, FIFO can be requested by setting this bit high active. After work has been done, this bit must be set low.
Tx	R/W	Represents the direction and transition end of the MCU accesses. When being set as logic 1, the MCU wants to write data to FIFO. After work has been done, this bit must be set to logic 0 before terminating the request to represent a transition end. For reading action, this bit must be set to logic 0 to indicate that the MCU wants to read and must be set to logic 1 after work is done.
Clear	R/W	Represents MCU clear requested FIFO, even if FIFO is not ready.
Sel_pipe1 Sel_pipe0	R/W	Determines which FIFO is desired, ″00″ for FIFO 0, ″01″ for FIFO 1
Set CMD	R/W	Shows that the data in FIFO is setup as command. This bit will be cleared by firmware. So, even if the MCU is busy, nothing is missed by the SETUP command from the host.
Ready	R	Indicates that the desired FIFO is ready to work.
Len0	R/W	Indicates that the host sent a 0-sized packet to the MCU. This bit must be cleared by a read action to the corresponding FIFO. Also, this bit will be cleared by the USB SIE after the next valid SETUP token is received.

MISC Function Table

The HT82M99EE/HT82M99AE have two 8×8 bidirectional FIFO for the two endpoints (control and Interrupt). User can easily read/write the FIFO data by accessing the corresponding FIFO pointer register (FIFO0, FIFO1). The following are two examples for reading and writing the FIFO data:

HT82M99EE/HT82M99AE FIFO is read by packet. To read from FIFO, the following should be followed:

- Select one set of FIFO, set in the read mode (MISC TX bit = 0), and set the REQ bit to ″1″.
- Check the ready bit until the status = 1
- Read through the FIFO pointer register, and record the data number that has been read.
- Repeat steps 2 and 3 until the ready bit becomes 0 which indicates the end of the FIFO data reading.
- Set MISC TX bit = 1
- Clear the REQ bit to 0. Complete reading.

User reads the data through the FIFO pointer register, user has to record the number of bytes to be read. The HT82M99EE/HT82M99AE allows a maximum of 8 bytes of data in each packet.

The HT82M99EE/HT82M99AE FIFO is written by packet. To write to FIFO, the following should be followed:

- Select a set of FIFO, set in the write mode (MISC TX bit = 1), and set the REQ bit to ″1″
- Check the ready bit until the status = 1
- Write through the FIFO pointer register and take down the data number that has been written
- Repeat steps 2 and 3 until writing is complete or the ready bit becomes 0 which indicates that the FIFO no longer allows any data writing.
- Set MISC TX bit = 0
- Clear the REQ bit to 0. Complete writing.

User writes the data through the FIFO pointer register, user has to record the number of bytes that have been written. The HT8M99E allows a maximum of 8 bytes of data in each packet.

There are some timing constrains and usages illustrated here. By setting the MISC register, the 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 of 2μs, check 41H→read* from FIFO0 register and check if not ready (01H)→03H→02H
Write FIFO1 sequence	0AH→0BH→delay of 2μs, check 4BH→write* to FIFO1 register and check if not ready (0BH)→09H→08H
Check whether FIFO0 can be read or not	00H→01H→delay of 2μs, check 41H (if ready) or 01H (if not ready) →00H
Check whether FIFO1 can be written to or not	0AH→0BH→delay of 2μs, check 4BH (if ready) or 0BH (if not ready) →0AH
Write 0-sized packet sequence to FIFO 0	02H→03H→delay of 2μs, check 43H→01H→00H

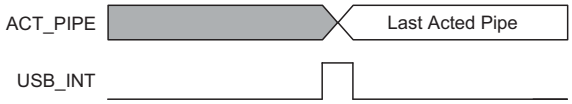
Note: *: There are 2μs gap existing between 2 reading actions or between 2 writing actions

Register Name	R/W	Register Address	Bit7~Bit0
FIFO 0	R/W	01001000B	Data7~Data0
FIFO 1	R/W	01001001B	Data7~Data0

FIFO Register Address Table

USB Active Pipe Timing

The USB active pipe accessed by the host cannot be used by the MCU simultaneously. When the host finishes its work, the signal, a USB_INT will be produced to tell the MCU that the pipe can be used and the acted pipe No. will be shown in the signal, ACT_PIPE as well. The timing is illustrated in the Figure below.



USB Active Pipe Timing

Suspend Wake-Up and Remote Wake-Up

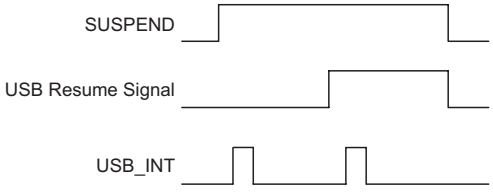
If there is no signal on the USB bus for over 3ms, the HT8M99E will go into a suspend mode. The Suspend line (bit 0 of the USC) will be set to 1 and a USB interrupt is triggered to indicate that the HT8M99E should jump to the suspend state to meet the 500μA USB suspend current spec.

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

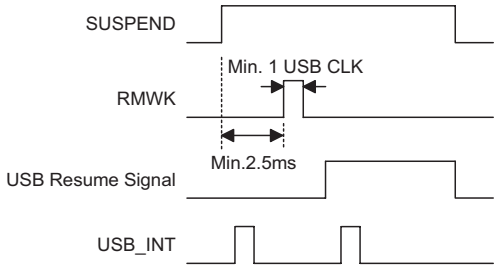
The user can also further decrease the suspend current to 250μA by setting the SUSP2 (bit4 of the SCC). But if the SUSP2 is set, the user has to make sure not to enable the LVR OPT option, otherwise the HT8M99E will be reset.

When the resume signal is sent out by the host, the HT8M99E will wake-up the MCU by USB interrupt and the Resume line (bit 3 of the USC) is set. In order to make the HT8M99E function properly, the programmer must set the USBCKEN (bit 3 of the SCC) to 1 and clear the SUSP2 (bit4 of the SCC). Since the Resume signal will be cleared before the Idle signal is sent out by the host and the Suspend line (bit 0 of the USC) is going to ″0″. So when the MCU is detecting the Suspend line (bit0 of the USC), the Resume line should be remembered and taken into consideration.

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



The device with remote wake-up function can wake-up the USB Host by sending a wake-up pulse through RMWK (bit 1 of USC). Once the USB Host receive the wake-up signal from the HT8M99E, it will send a Resume signal to the device. The timing is as follows:



To Configure the HT8M99E as PS2 Device

The HT8M99E can be defined as a USB interface or a PS2 interface by configuring the SPS2 (bit 4 of the USR) and SUSB (bit 5 of the USR). If SPS2=1, and SUSB=0, the HT8M99E 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 easily read or write to the PS2 Data or PS2 Clk pin by accessing the corresponding bit PS2DAI (bit 4 of the USC), PS2CKI (bit 5 of the USC), PS2DAO (bit 6 of the USC) and S2CKO (bit 7 of the USC) respectively.

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

If SPS2=0, and SUSB=1, the HT8M99E is defined as a USB interface. Both the USBD- and USBD+ are driven by the USB SIE of the HT8M99E. User only writes or reads the USB data through the corresponding FIFO.

Both SPS2 and SUSB default is ″0″.

I/O Port Special Registers Definition

- Port-A (12H) – PA

Bit No.	Label	Read/Write	Option	Functions
0	PA0	R/W	—	I/O (R/W) has pull-low and pull-high ROM code option. Has falling edge wake-up ROM code option.
1	PA1	R/W	—	I/O (R/W) has pull-low and pull-high option. Has falling edge wake-up option.
2~3	PA2~PA3	R/W	—	I/O (R/W) has pull-low and pull-high option. Has falling edge and rising edge wake-up option.
4~6	PA4~PA6	R/W	—	I/O (R/W) has pull-high option. Has falling edge wake-up option.
7	PA7	R/W	—	I/O (R/W) has pull-high option. Has falling edge wake-up option, pin-shared with timer input pin.

PA (12H) Register

- Port-A Control (13H) – PAC
  This port configure the input or output mode of Port-A

- Port-B Control (14H) – PB

Bit No.	Label	Read/Write	Option	Functions
0~1, 5~6	PB0~PB1, PB5~PB6	—	—	Reserved bit.
2~3	PB2~PB3	R/W	—	I/O (R/W), has pull-low and pull-high option, ADC input.
4	PB4	R/W	—	I/O (R/W), has pull-high option, can wake-up, ADC input.
7	PB7	R/W	—	I/O (R/W), has pull-high option, ADC input, VRH input for ADC external mode, has wake-up capability.

PB (14H) Register

- Port-B Control (15H) – PBC
  This port configures the input or output mode of Port-B for I/O mode

USB/PS2 Status and Control Register – USC

Bit No.	Label	Read/Write	Option	Functions
0	PE0	R	SUSPEND	USB suspend mode status bit. When 1, indicates that the USB system entry is in suspend mode.
1	PE1	W	RMOT_WK	USB remote wake-up signal. The default value is ″0″.
2	PE2	R/W	URST_FLAG	USB bus reset event flag. The default value is ″0″.
3	PE3	R	RESUME_O	When RESUME_OUT EVENT, RESUME_O is set to ″1″. The default value is ″0″.
4	PE4	R	PS2_DAI	USBD-/DATA input
5	PE5	R	PS2_CKI	USBD+/CLK input
6	PE6	W	PS2_DAO	Output for driving USBD-/DATA pin, when working under 3D PS2 mouse function. The default value is ″1″.
7	PE7	W	PS2_CKO	Output for driving USBD-/DATA pin, when working under 3D PS2 mouse function. The default value is ″1″.

USC (0X1A) Register

Endpoint Interrupt Status Register – USR

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

Bit No.	Label	Read/Write	Option	Functions
0	PEC0	R/W	EP0IF	When set to ″1″, indicates an endpoint 0 interrupt event. Must wait for the MCU to process the interrupt event and clear this bit by firmware. This bit must be ″0″, then the next interrupt event will be processed. The default value is ″0″.
1	PEC1	R/W	EP1IF	When set to ″1″, indicates an endpoint 1 interrupt event. Must wait for the MCU to process the interrupt event, then clear this bit by firmware. This bit must be ″0″, then the next interrupt event will be processed. The default value is ″0″.
2~3, 6	PEC2~PEC3, PEC6	R/W	—	Reserved bit, set to ″0″
4	PEC4	R/W	SELPS2	When set to ″1″, indicates that the chip is working under PS2 mode. The default value is ″0″.
5	PEC5	R/W	SELUSB	When set to ″1″, indicates that the chip is working under USB mode. The default value is ″0″.
7	PEC7	R/W	USB_flag	This flag is used to show that the MCU is in USB mode (Bit=1). This bit is R/W by FW and will be cleared to zero after power-on reset. The default value is ″0″.

USR (0X1B) Register

Clock Control Register – SCC

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 (SUSPEND2) and system clock selection (SCLKSEL).

Bit No.	Label	Read/Write	Option	Functions
2~0, 5	PF2~PF0, PF5	R/W	—	Reserved
3	PF3	R/W	USBCKEN	USB clock control bit. When set to "1", indicates a USBCK ON, else USBCK OFF. The default value is "0".
4	PF4	R/W	SUSPEND2	When set to "1", enables a 7.5kΩ resistor connected to D-pin to 5V VDD. The default value is "0".
6	PF6	R/W	SCLKSEL	System clock 6MHz or 12MHz option, when working on external oscillator mode. The default value is "0". 0: Operating at external 12MHz mode 1: Operating at external 6MHz mode The default value is "0".
7	PF7	R/W	PS2_flag	This flag is used to show that the MCU is in PS2 mode (Bit=1). This bit is R/W by FW and will be cleared to zero after power-on reset. The default value is "0".

SCC (0X1C) Register

Table High Byte Pointer for Current Table Read – TBHP

Bit No.	Label	Read/Write	Option	Functions
2~0	PGC2~PG0	R/W	—	Store current table read bit10~bit8 data

TBHP (0X1F) Register

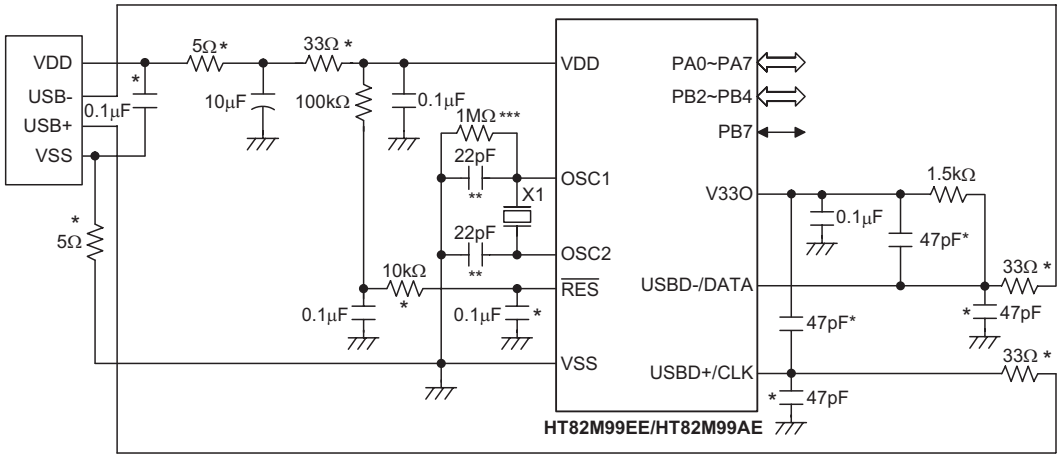
Options

No.	Option
1	WDT clock source: RC (system/4) (default: T1)
2	WDT clock source: enable/disable for normal mode (default: disable)
3	PA0~PA7 ,PB4, PB7 wake-up by bit (PA2, PA3 both wake-up by falling or rising edge) (default: non wake-up)
4	PA0~PA7 pull-high by bit (default: Pull-high)
5	PB pull-high by nibble (default: Pull-high)
6	2.7 V (error 0.3V) LVR enable/disable (default: enable)
7	PA0~PA3, PB2, PB3 Pull-low by bit (default: non pull-low 30kΩ)
8	"CLR WDT", 1 or 2 instructions
9	TBHP enable/disable (default: disable)
10	PA output mode (CMOS/NMOS/PMOS) by bit (default: CMOS)

Note: The LVR voltage is define as 2.7V±0.3V and default is enable.

Application Circuits

Crystal or Ceramic Resonator for Multiple I/O Applications



Note: The resistance and capacitance for the 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.

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

Components with * are used for EMC issue.

Components with ** are used for resonator only.

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

Instruction Set 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](5)	Read ROM code (locate by TBLP and TBHP) to data memory and TBLH	2(1)	None
TABRDC [m](6)	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,PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO(4),PDF(4)
CLR WDT2	Pre-clear Watchdog Timer	1	TO(4),PDF(4)
SWAP [m]	Swap nibbles of data memory	1(1)	None
SWAPA [m]	Swap nibbles of data memory with result in ACC	1	None
HALT	Enter power down mode	1	TO,PDF

Note: x: Immediate data

m: Data memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

√: Flag is affected

–: Flag is not affected

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

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

(3): (1) and (2)

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

(5): ″ROM code TBHP option″ is enabled

(6): ″ROM code TBHP option″ is disabled

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)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation [m] ← ACC+[m]+C

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation ACC ← ACC+[m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

ADD A,x Add immediate data to the accumulator

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

Operation ACC ← ACC+x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation [m] ← ACC+[m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

Operation ACC ← ACC ″AND″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

AND A,x Logical AND immediate data to the accumulator

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

Operation ACC ← ACC ″AND″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation [m] ← ACC ″AND″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

CALL addr Subroutine call

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

Operation Stack ← Program Counter+1
Program Counter ← addr

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m] Clear data memory

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

Operation [m] ← 00H

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR [m].i Clear bit of data memory

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

Operation [m].i ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

CLR WDT Clear Watchdog Timer

Description The WDT is cleared (clears the WDT). The power down bit (PDF) and time-out bit (TO) are cleared.

Operation WDT ← 00H
PDF and TO ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
0	0	—	—	—	—

CLR WDT1 Preclear Watchdog Timer

Description Together with CLR WDT2, clears the WDT. PDF and TO are also cleared. Only execution of this instruction without the other preclear instruction just sets the indicated flag which implies this instruction has been executed and the TO and PDF flags remain unchanged.

Operation WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CLR WDT2 Preclear Watchdog Timer

Description Together with CLR WDT1, clears the WDT. PDF and TO are also cleared. Only execution of this instruction without the other preclear instruction, sets the indicated flag which implies this instruction has been executed and the TO and PDF flags remain unchanged.

Operation WDT ← 00H*
PDF and TO ← 0*

Affected flag(s)

TO	PDF	OV	Z	AC	C
0*	0*	—	—	—	—

CPL [m] Complement data memory

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

Operation [m] ← $\overline{[m]}$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation ACC ← $\overline{[m]}$

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

DAA [m] Decimal-Adjust accumulator for addition

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

Operation If ACC.3~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)

TO	PDF	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)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation ACC ← [m]−1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

HALT Enter power down mode

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

Operation Program Counter ← Program Counter+1
PDF ← 1
TO ← 0

Affected flag(s)

TO	PDF	OV	Z	AC	C
0	1	—	—	—	—

INC [m] Increment data memory

Description Data in the specified data memory is incremented by 1

Operation [m] ← [m]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation ACC ← [m]+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

JMP addr Directly jump

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

Operation Program Counter ←addr

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation ACC ← [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

MOV A,x Move immediate data to the accumulator

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

Operation ACC ← x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation [m] ←ACC

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

NOP No operation

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

Operation Program Counter ← Program Counter+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

OR A,x Logical OR immediate data to the accumulator

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

Operation ACC ← ACC ″OR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation [m] ←ACC ″OR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

RET Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RETI Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RL [m] Rotate data memory left

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RLC [m] Rotate data memory left through carry

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

RR [m] Rotate data memory right

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

RRC [m] Rotate data memory right through carry

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SDZ [m] Skip if decrement data memory is 0

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m] Set data memory

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

Operation [m] ← FFH

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SET [m]. i Set bit of data memory

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

Operation [m].i ← 1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SIZ [m] Skip if increment data memory is 0

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation Skip if [m].i≠0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SUB A,x Subtract immediate data from the accumulator

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

Operation ACC ← ACC+$\overline{x}$+1

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	√	√	√	√

SWAP [m] Swap nibbles within the data memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

SZ [m] Skip if data memory is 0

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

Operation Skip if [m]=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation Skip if [m]=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation Skip if [m].i=0

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

TABRDC [m] Move the ROM code (locate by TBLP and TBHP) to TBLH and data memory (ROM code TBHP is enabled)

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

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

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	—	—	—

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

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

Operation ACC ← ACC ″XOR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

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

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

Operation [m] ← ACC ″XOR″ [m]

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

XOR A,x Logical XOR immediate data to the accumulator

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

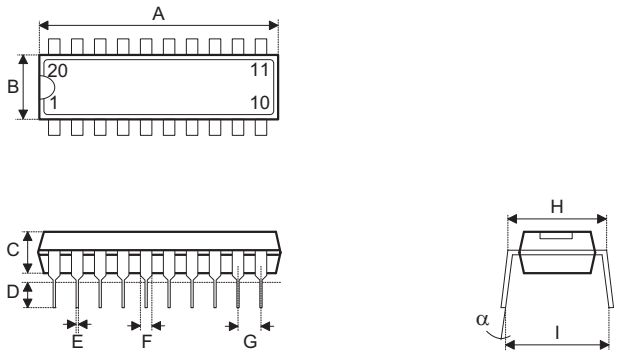
Operation ACC ← ACC ″XOR″ x

Affected flag(s)

TO	PDF	OV	Z	AC	C
—	—	—	√	—	—

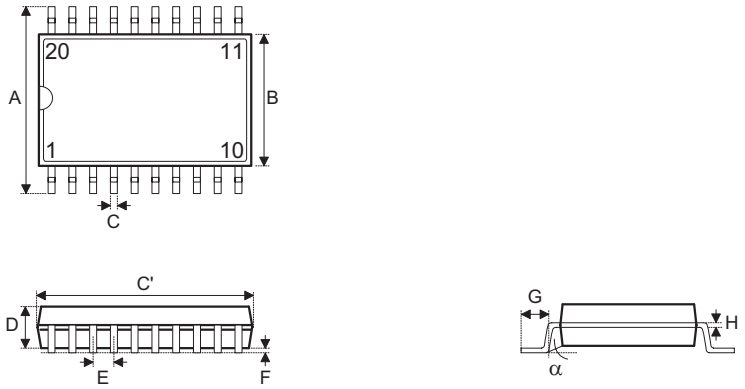
Package Information

20-pin DIP (300mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1020	—	1045
B	240	—	260
C	125	—	135
D	125	—	145
E	16	—	20
F	50	—	70
G	—	100	—
H	295	—	315
I	335	—	375
α	0°	—	15°

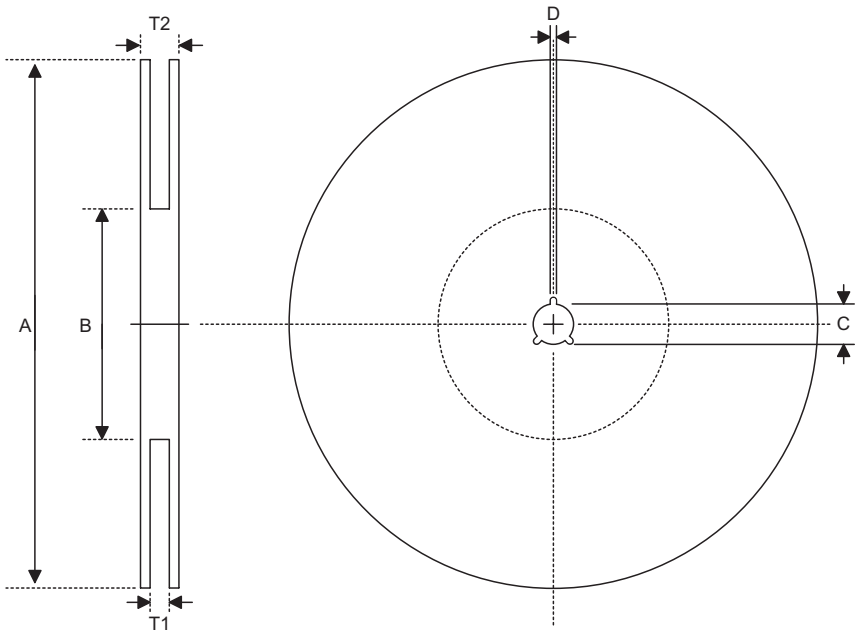
20-pin SOP (300mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	394	—	419
B	290	—	300
C	14	—	20
C′	490	—	510
D	92	—	104
E	—	50	—
F	4	—	—
G	32	—	38
H	4	—	12
α	0°	—	10°

Product Tape and Reel Specifications

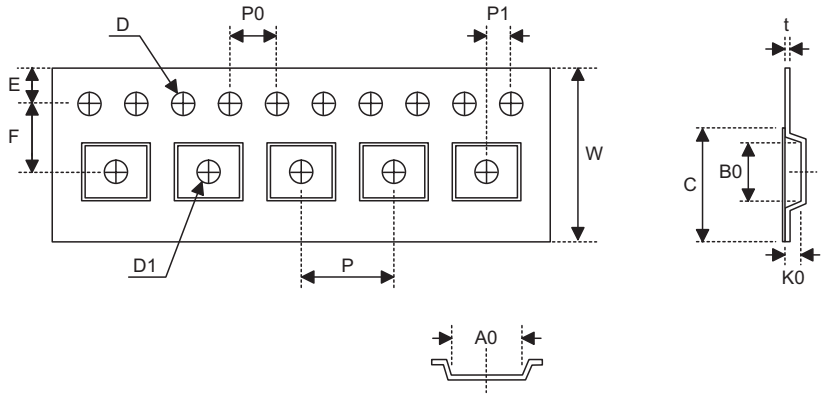
Reel Dimensions



SOP 20W

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

Carrier Tape Dimensions



SOP 20W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24+0.3 −0.1
P	Cavity Pitch	12±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	1.5+0.1
D1	Cavity Hole Diameter	1.5+0.25
P0	Perforation Pitch	4±0.1
P1	Cavity to Perforation (Length Direction)	2±0.1
A0	Cavity Length	10.8±0.1
B0	Cavity Width	13.3±0.1
K0	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.3±0.05
C	Cover Tape Width	21.3

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

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

Holtek Semiconductor Inc. (Shanghai Sales Office)
7th Floor, Building 2, No.889, Yi Shan Rd., Shanghai, China 200233
Tel: 021-6485-5560
Fax: 021-6485-0313
http://www.holtek.com.cn

Holtek Semiconductor Inc. (Shenzhen Sales Office)
43F, SEG Plaza, Shen Nan Zhong Road, Shenzhen, China 518031
Tel: 0755-8346-5589
Fax: 0755-8346-5590
ISDN: 0755-8346-5591

Holtek Semiconductor Inc. (Beijing Sales Office)
Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031
Tel: 010-6641-0030, 6641-7751, 6641-7752
Fax: 010-6641-0125

Holmate Semiconductor, Inc. (North America Sales Office)
46712 Fremont Blvd., Fremont, CA 94538
Tel: 510-252-9880
Fax: 510-252-9885
http://www.holmate.com