



## HT82D20R/HT82D20A

### 27MHz One Channel RX 8-Bit MCU

---

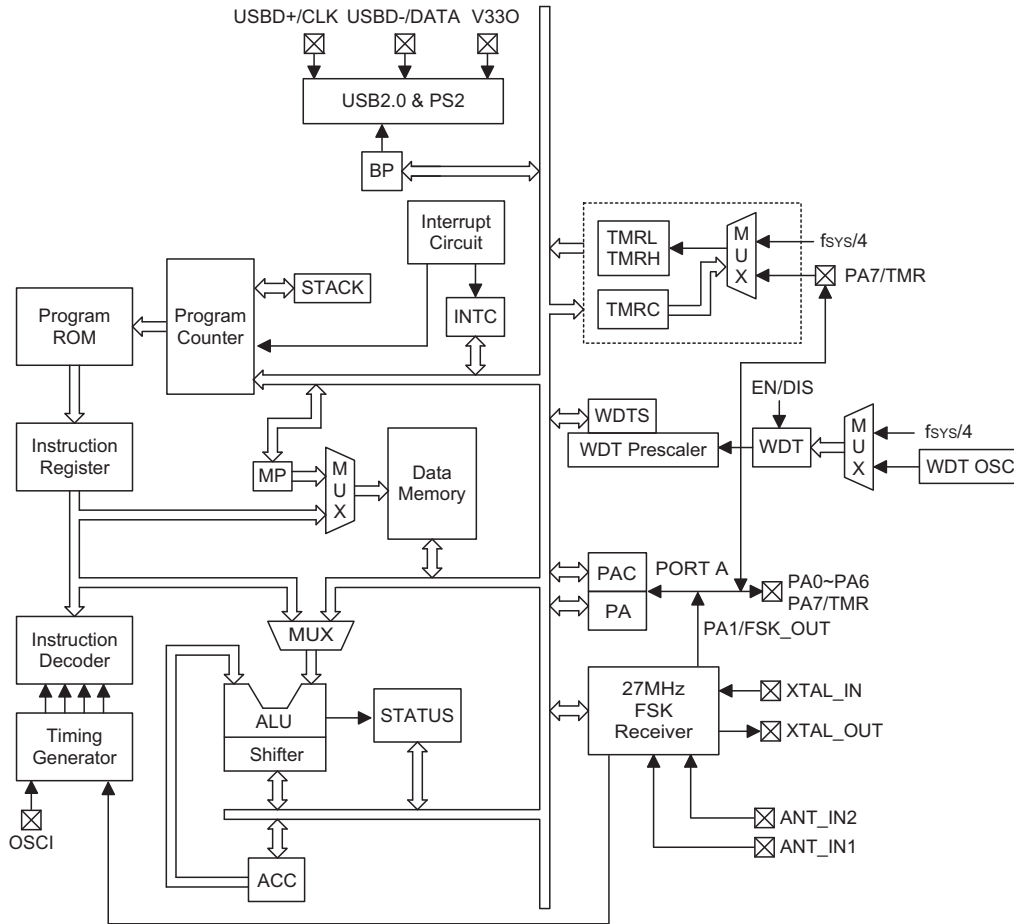
#### Features

- USB specification compliance
  - Conforms to USB specification V2.0
  - Conforms to USB HID specification V1.11
- Supports 1 Low-speed USB control endpoint and 1 interrupt endpoint
- Each endpoint has 8 byte FIFO
- Integrated USB transceiver
- 3.3V regulator output
- Built-in one 27MHz FSK receiver
- PS2 and USB modes supported
- 27MHz FSK receiver power down function
- FSK receiver frequency range 26.995~27.295MHz
- FSK receiver high sensitivity:  $\leq -90$  dBm
- RF tuner, mixer, transistors, passives, coils, and SAW filter functions integrated in the same device
- Integrated FSK receiver phase locked loop
- Eight user selectable frequencies
- Integrated FSK Receiver 6Kbps data rate
- Uses external 12MHz crystal
- 8-bit RISC microcontroller, with 2K×14 bits EPROM (000H~7FFH)
- 96 bytes RAM (20H~7FH)
- 6MHz internal MCU clock
- 4-level stack
- 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 support wake-up function
- Internal power-on reset (POR)
- Watchdog Timer (WDT)
- 8 I/O ports
- 28-pin SSOP package

#### General Description

USB Encoder Built-in one 27MHz FSK receiver MCU OTP body is suitable for USB interface and 27MHz Wireless system. Flexible total solution for applications that combine PS/2 and low-speed USB interface and 27MHz wireless system, such as mice, joysticks, and many others.

It consists of a Holtek high performance 8-bit MCU core for control unit, built-in USB SIE, 27MHz FSK Receiver, 2K×14 bits ROM and 96 bytes data RAM.

**Block Diagram**

**Pin Assignment**
[www.DataSheet4U.com](http://www.DataSheet4U.com)

VSS	1	28	VDD
V330	2	27	PA7/TMR
USBBD+/CLK	3	26	PA6
USBBD-/DATA	4	25	PA2
OSCI	5	24	PA5
RES	6	23	PA3
PA0	7	22	PA4
PA1/FSK_OUT	8	21	VSS
VSS	9	20	XTAL_OUT
LF	10	19	XTAL_IN
VREF1	11	18	VREF3
VSS	12	17	VCC
ANT_IN2	13	16	VREF2
ANT_IN1	14	15	GND

**HT82D20R/HT82D20A**  
**28 SSOP-A**

**Pin Description**

Pin Name	I/O	Configuration Option	Description
PA0 PA1/FSK_OUT PA2~6 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 configuration 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 PA1 is wire-bonded with FSK demodulated data output The TMR pin is shared with PA7
VSS	—	—	Digital negative power supply, ground
VDD	—	—	Digital positive power supply
RES	I	—	Schmitt trigger reset input. Active low.
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
LF	I	—	Loop filter for Local Oscillator (resistance 24k $\Omega$ and capacity 5.6nF) in parallel with capacity 560pF to ground.
OSCI	—	—	For test pin
VCC	I	—	Analog positive power supply
GND	—	—	Analog negative power supply, ground
VREF1	O	—	Mid rail reference voltage for the FSK receiver
VREF2	O	—	Internal positive analog supply reference voltage for the FSK receiver
VREF3	O	—	Internal positive digital supply reference voltage for the FSK receiver
ANT_IN1	I	—	Antenna input 1
ANT_IN2	I	—	Antenna input 2
XTAL_IN XTAL_OUT	I O	—	XTAL_IN, XTAL_OUT are connected to a 12MHz crystal

**Absolute Maximum Ratings**

Supply Voltage .....	$V_{SS}-0.3V$ to $V_{SS}+6.0V$	Storage Temperature .....	$-50^{\circ}C$ to $125^{\circ}C$
Input Voltage .....	$V_{SS}-0.3V$ to $V_{DD}+0.3V$	Operating Temperature .....	$0^{\circ}C$ to $70^{\circ}C$

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

**D.C. Characteristics**

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	Operating Voltage	—	—	4.0	—	5.5	V
I <sub>DD</sub>	Operating Current	5V	No load, f <sub>XTAL</sub> =12MHz	—	25	—	mA
I <sub>STB1</sub>	Standby Current	5V	No load, system HALT, USB suspend**, FSK receiver power down	—	1.5	—	mA
I <sub>STB2</sub>	Standby Current	5V	No load, system HALT, input/output mode, set SUSPEND2 [1CH].4, FSK receiver power down	—	1.0	—	mA
V <sub>IL1</sub>	Input Low Voltage for I/O Ports	5V	—	0	—	0.8	V
V <sub>IH1</sub>	Input High Voltage for I/O Ports	5V	—	2	—	V <sub>DD</sub>	V
V <sub>IL2</sub>	Input Low Voltage (RES)	—	—	0	—	0.4V <sub>DD</sub>	V
V <sub>IH2</sub>	Input High Voltage (RES)	—	—	0.9V <sub>DD</sub>	—	V <sub>DD</sub>	V
I <sub>OL</sub>	Output Sink Current for PA4~PA7	5V	V <sub>OL</sub> =0.4V	2	4	—	mA
I <sub>OH</sub>	Output Source Current for PA4~PA7	5V	V <sub>OH</sub> =3.4V	-2.5	-4	—	mA
I <sub>OL2</sub>	Output Sink Current for PA0~PA3	5V	V <sub>OL</sub> =0.4V	10	15	—	mA
I <sub>OH2</sub>	Output Source Current for PA0~PA3	5V	V <sub>OH</sub> =3.4V	8	12	—	mA
R <sub>PD</sub>	Pull-down Resistance for PA0~PA3	5V	—	10	30	50	kΩ
R <sub>PH1</sub>	Pull-high Resistance for DATA*	—	—	1.3	1.5	2.0	kΩ
R <sub>PH2</sub>	Pull-high Resistance for CLK	—	—	2.0	4.7	6.0	kΩ
R <sub>PH3</sub>	Pull-high Resistance for PA0~PA7	—	—	30	50	70	kΩ
V <sub>LVR</sub>	Low Voltage Reset	—	—	2.4	2.7	3	V

Note: "\*" The DATA pull-high is implemented using an external 1.5kΩ resistor.

 "\*" includes 15kΩ load on the USB<sub>D</sub>+, USB<sub>D</sub>- line at the host terminal.

**A.C. Characteristics**

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
f <sub>SYS</sub>	System Clock	5V	—	—	6	—	MHz
f <sub>XTAL</sub>	Quartz xtal Frequency	—	—	—	12	—	MHz
Xppm	Quartz xtal Frequency Tolerance	—	—	—	±50	—	ppm
C <sub>XTAL</sub>	Xtal Shunt Capacitance	—	—	—	47	—	pF
f <sub>RCSYS</sub>	RC Clock with 8-bit Prescaler Register	5V	—	16	32	48	kHz
t <sub>WDT</sub>	Watchdog Time-out Period (System Clock)	—	Without WDT prescaler	1024	—	—	t <sub>RCSYS</sub>
t <sub>RF</sub>	USB <sub>D+</sub> , USB <sub>D-</sub> Rising & Falling Time	—	—	75	—	300	ns
t <sub>RES</sub>	External Reset Low Pulse Width	—	—	1	—	—	ms
t <sub>SST</sub>	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	t <sub>SYS</sub>
t <sub>OSC</sub>	Crystal Setup	—	—	—	5	10	ms

 Note: Power-on period=t<sub>WDT</sub>+t<sub>SST</sub>+t<sub>OSC</sub>

 WDT Time-out in the normal mode=1/f<sub>RCSYS</sub>×256×WDTS+t<sub>WDT</sub>

 WDT Time-out in the HALT mode=1/f<sub>RCSYS</sub>×256×WDTS+t<sub>SST</sub>+t<sub>OSC</sub>
**RF Characteristics**

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>CC</sub>	Conditions				
f <sub>CHN</sub>	Channel Spacing	5V	—	—	50	—	kHz
f <sub>COM</sub>	Communication Spacing	5V	PB 4 3 2 = 000 001 010 011 100 101 110 111	—	26.995 27.045 27.095 27.145 27.195 27.245 27.295 26.945	—	MHz
R <sub>IN</sub>	Input Resistance	5V	DC Differential @27MHz	—	< 100 8	—	kΩ
C <sub>IN</sub>	Input Capacitance	5V	Differential @27MHz	—	5	—	pF
A <sub>SENS</sub>	Antenna Input Sensitivity	5V	Antenna Input 50Ω to 8kΩ Impedance Transform	-90	—	—	dbm
AC <sub>REJ</sub>	Adjacent Channel Rejection	5V	50kHz offset, 100kHz & 150kHz offsets	—	1.5 28	—	dB
f <sub>DEV</sub>	Frequency Deviation	5V	—	3.2	—	—	kHz
DR <sub>FSK</sub>	Data Rate	5V	—	—	6K	—	bit/sec
V <sub>REF1</sub>	Internal Mid-rail Reference	5V	—	—	1.6	—	V
V <sub>REF2</sub>	Internal Supply Voltage Reference	5V	—	—	3.3	—	V
t <sub>PU</sub>	Power Up Settling Time	5V	—	—	3	—	ms

[www.DataSheet4U.com](http://www.DataSheet4U.com)

## Functional Description

### Execution Flow

The External crystal must use 12MHZ but the system clock for the microcontroller is derived from 6MHZ internal clock. 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

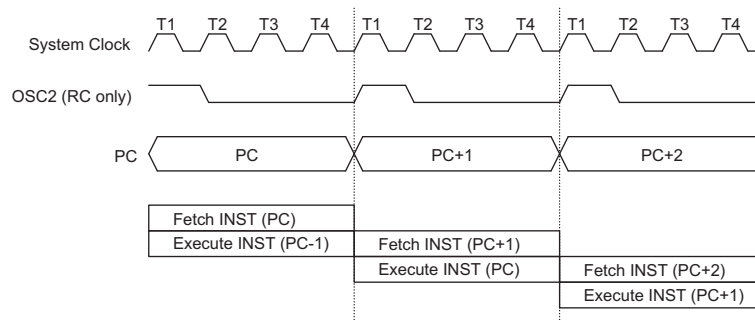
incremented by one. The program counter then points to the memory word containing the next instruction code.

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**

www.DataSheet4U.com

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**

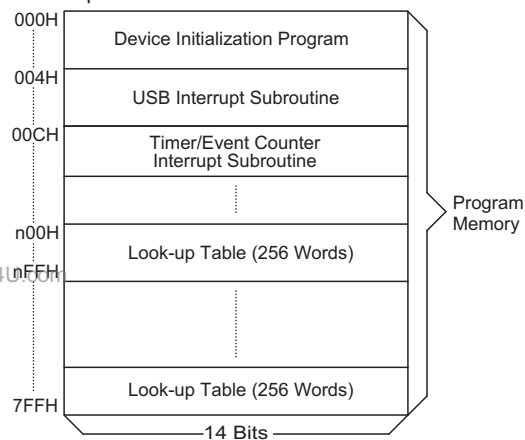
The program memory is used to store the program instructions which are to be executed. It also contains data, table, and interrupt entries, and is organized into 2048×14 bits, addressed by the program counter and table pointer.

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



Note: n ranges from 0 to 7

**Program Memory**

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 configuration option TBHP is disabled (default).
- ♦ The instructions "TABRDC [m]", where the table locations is defined by registers TBLP (07H) and TBHP (01FH). And the configuration 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 configuration 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 configuration option TBHP is disabled, the instruction "TABRDC [m]" reads the ROM data as defined by TBLP and the current program counter bits.

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  
@7~@0: TBLP bits

P10~P8: Current program counter bits when TBHP is disabled  
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 bits). Most are read/write, but some are read only.

The unused space before 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.

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).

Bank 0	
00H	IAR0
01H	MP0
02H	IAR1
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	General Purpose DATA MEMORY (96 Bytes)
21H	
22H	
23H	
24H	
25H	
26H	
27H	
7FH	

**Bank 0 RAM Mapping**



**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.

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

Bank 1	
00H	IAR0
01H	MP0
02H	IAR1
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**

**Indirect Addressing Register**

Locations 00H and 02H are indirect addressing registers (IAR0:00H; IAR1:02H) 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	—	Unused bit, read as "0"
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.

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	—	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)
5	—	Unused bit, read as "0"
6	TF	Internal timer/event counter request flag (1:active; 0:inactive)
7	—	Unused bit, read as "0"

**INTC (0BH) Register**

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

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, 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 device receives a USB Suspend signal from the Host PC, the suspend line (bit0 of the USC) is set and a USB interrupt is also triggered.

When the device receives a Resume signal from the Host PC, the resume line (bit3 of the USC) 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/event 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 "RET" 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.

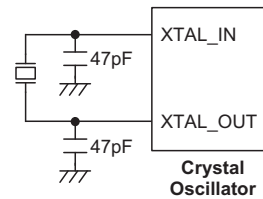
Interrupts, occurring in the interval between the rising edges of two consecutive T2 pulses, will be serviced on the latter of the two T2 pulses, if the corresponding interrupts are enabled. In the case of simultaneous requests the following table shows the priority that is applied. These can be masked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

Interrupt Source	Priority	Vector
USB interrupt	1	04H
Timer/Event Counter overflow	2	0CH

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.



**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 XTAL\_IN and XTAL\_OUT 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 XTAL\_IN and XTAL\_OUT to get a frequency reference, but two external capacitors in XTAL\_IN and XTAL\_OUT are required.

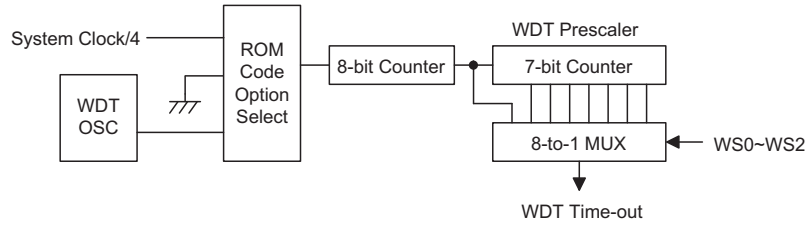
The external crystal must use 12MHz but it operate in 6MHz system clock. The USB SIE function also operate in 6MHz.

Both of XTAL\_IN and XTAL\_OUT pin must be connect a 47pF capacitor to ground.

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 configuration 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 configuration option. This timer is designed to prevent a software malfunction or sequence from jumping to an unknown location with unpredictable results. The Watchdog



**Watchdog Timer**

Timer can be disabled by configuration 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 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 de-

pending on the configuration 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 re-counted 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 func-

tion 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**

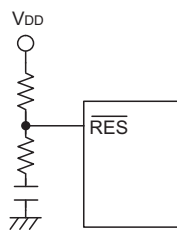
There are four ways in which a reset can occur:

- $\overline{RES}$  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 Counter and Stack Pointer, leaving the other circuits in their original state. Some registers remain unchanged during other reset conditions. Most registers are reset to the "initial condition" when the reset conditions are met. By examining the PDF and TO flags, the program can distinguish between different "chip resets".

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

Note: "u" stands for "unchanged"



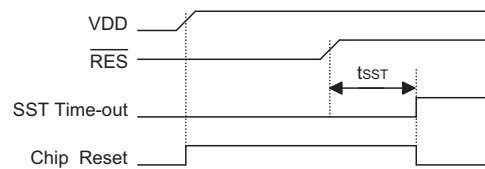
**Reset Circuit**

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.

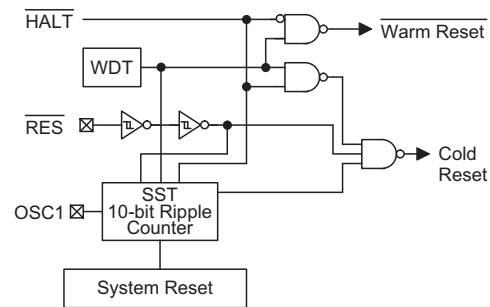
When a system reset occurs, the SST delay is added during the reset period. Any wake-up from HALT will enable the SST delay.

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



**Reset Timing Chart**



**Reset Configuration**

The registers status are summarized in the following table.

Register	Reset (Power On)	WDT Time-out (Normal Operation)	RES Reset (Normal Operation)	RES Reset (HALT)	WDT Time-Out (HALT)*	USB-Reset (Normal)	USB-Reset (HALT)
TMRH	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMRL	xxxx xxxx	0000 0000	0000 0000	0000 0000	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	--00 uuuu	--11 uuuu	--uu uuuu	--01 uuuu
INTC	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu	-000 0000	-000 0000
WDS	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
USC	11xx 0000	uuxx uuuu	11xx 0000	11xx 0000	uuxx uuuu	1100 0u00	1100 0u00
USR	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	u0uu 0000	u0uu 0000
SCC	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	uu00 u000	uu00 u000
TBHP	---- -xxx	---- -uuu	---- -uuu	---- -uuu	---- -uuu	---- -uuu	---- -uuu
PIPE_CTL	0000 0010	0000 uuuu	0000 0010	0000 0010	0000 uuuu	0000 0010	0000 0010
AWR	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
STALL	0000 0010	0000 uuuu	0000 0010	0000 0010	0000 uuuu	0000 0010	0000 0010
PIPE	0000 0000	xxxx xxxx	0000 0000	0000 0000	xxxx xxxx	0000 0000	0000 0000
SIES	0xxx xx00	uuuu xxuu	0xxx xx00	0xxx xx00	uuuu xxuu	0xxx xx00	0xxx xx00
MISC	0x00 0000	uuuu uuuu	0x00 0000	0x00 0000	uuuu uuuu	0x00 0000	0x00 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

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,

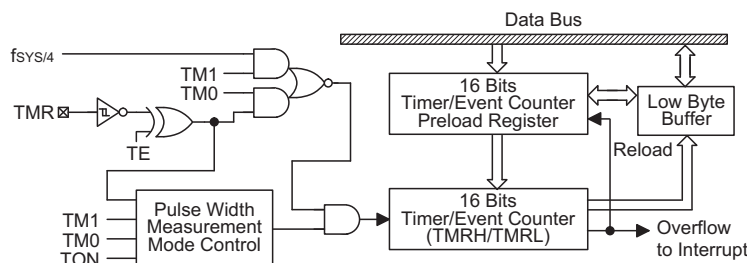
which means that the clock source comes from an external (TMR) pin. The timer mode functions as a normal timer with the clock source coming from the  $f_{SYS}/4$  (Timer). The pulse width measurement mode can be used to count the high or low level duration of the external signal (TMR). The counting is 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 No.	Label	Function
0~2	—	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)
5	—	Unused bit, read as "0"
6	TM0 TM1	Defines the operating mode 01=Event count mode (external clock) 10=Timer mode (internal clock) 11=Pulse width measurement mode 00=Unused
7		

**TMRC (11H) Register**



**Timer/Event Counter**

bit (TON; bit 4 of TMRC) should be set to 1. In the pulse width measurement mode, the TON will be cleared automatically after the measurement cycle is completed. But in the other two modes the TON can only be reset by instructions. The overflow of the timer/event counter 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 8 bidirectional input/output lines in the microcontroller, labeled from PA, which are mapped to the data memory of [12H] 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). 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) 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 con-

trol 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. These control registers are mapped to locations 13H.

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) instructions.

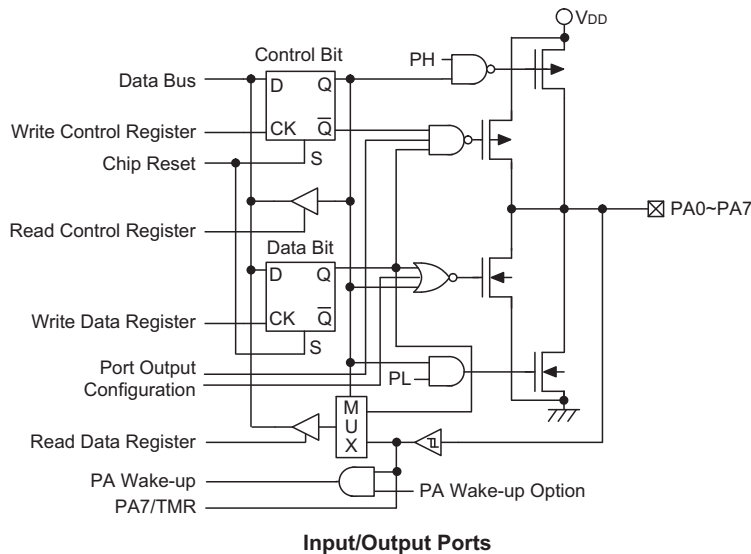
Some instructions first input data and then follow the output operations. For example, "SET [m].i", "CLR [m].i", "CPL [m]", "CPLA [m]" read the entire port states into the CPU, execute the defined operations (bit-operation), and then write the results back to the latches or the accumulator.

Each line of port A has the capability of waking-up the device.

There are pull-high/low 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.

PA1 is wire-bonded with FSK demodulation data output, if want read FSK demodulation data PA1 must set to input mode.





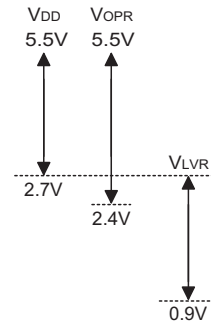
**Low Voltage Reset – LVR**

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.

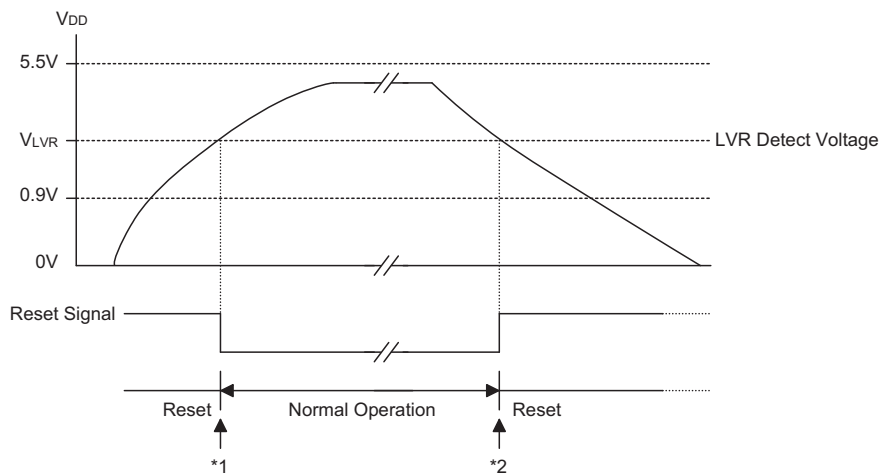
The LVR includes the following specifications:

- 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 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.

**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 and only for HT82D20A body. 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
6~2	Reserved bit	—	
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.
—	—	Bit 2~Bit 6 Reserved bit
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 devices have two 8x8 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:

The 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 devices allow a maximum of 8 bytes of data in each packet.

The 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 devices allow 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

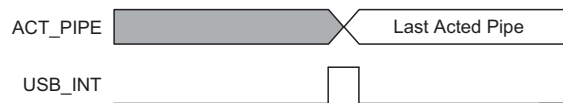
Note: \*: There are 2μs gap existing between 2 reading actions or between 2 writing actions

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 devices 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 devices should jump to the suspend state.

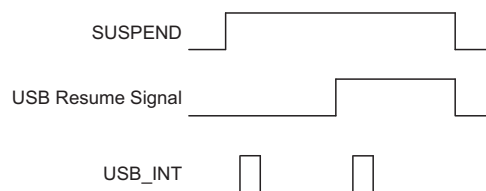
Disable the USB clock by clearing the USBCKEN (bit3 of the SCC) to "0". The suspend current is 1.5mA.

The user want to decrease the suspend current to about 1mA 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 devices will be reset.

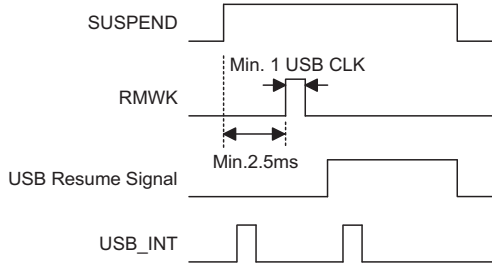
When the resume signal is sent out by the host, the devices will wake-up the MCU by USB interrupt and the Resume line (bit 3 of the USC) is set. In order to make the device function properly, the programmer must set the USBCKEN (bit 3 of the SCC) to 1 and clear the

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 device, it will send a Resume signal to the device. The timing is as follows:



**To Configure as an PS2 Device**

The devices 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 device is defined as a 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 device is defined as a USB interface. Both the USBD- and USBD+ are driven by the USB SIE of the device. User only writes or reads the USB data through the corresponding FIFO.

Both SPS2 and SUSB default is "0".

**27MHz FSK Receiver Function**

There is a single channel integrated RF transceiver designed for human interface devices (HID). Operating at 27MHz, it provide frequency selection from 8 discrete channels via a parallel control bit2~0 [14H].2 ~[14H].4, the frequency range is 26.945~26.995MHz and channel spacing is 50kHz.

It provide power down to reduce power consumption by 27MHz FSK Receiver power down bit [14H].7 .The optimized receiver design enables reception up to 3kHz per channel. It supply demodulation output data with PA1.

- 27MHz FSK Receiver Control Register 1 (14H) – PB

Register	Bits	Labels	Read/Write	Functions
PB (14H)	0~1	PB0~PB1	—	Reserved bit.
	2~4	PB2~PB4	R/W	Parallel control bit 2~bit 0 Communication spacing control 000: 26.995 MHz 001: 27.045 MHz 010: 27.095 MHz 011: 27.145 MHz 100: 27.195 MHz 101: 27.245 MHz 110: 27.295 MHz 111: 26.945 MHz
	5~6	PB5~PB6	—	Reserved bit.
	7	PB7	R/W	27MHz FSK Receiver power down bit When 1 indicate FSK Receiver for power down mode, otherwise for normal mode. Default value 0

- 27MHz FSK Receiver Control Register 2 (15H) – PBC  
Bit2~bit4 & bit7 must be "0", the other bits are reserved

**I/O Port Special Registers Definition**

- Port-A (12H) – PA

Register	Bits	Labels	Read/Write	Functions
PA (12H)	0	PA0	R/W	I/O (R/W) has pull-low and pull-high configuration option. Has falling edge wake-up configuration option.
	1	PA1	R/W	I/O (R/W) has pull-low and pull-high option. Has falling edge wake-up option. Read FSK demodulation output data for input mode
	2	PA2	R/W	I/O (R/W) has pull-low and pull-high option. Has falling edge and rising edge wake-up option.
	3	PA3	R/W	I/O (R/W) has pull-low and pull-high option. Has falling edge and rising edge wake-up option.
	4	PA4	R/W	I/O (R/W) has pull-high option. Has falling edge wake-up option.
	5	PA5	R/W	I/O (R/W) has pull-high option. Has falling edge wake-up option.
	6	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.

- Port-A Control (13H) – PAC

This port configure the input or output mode of Port-A

**USB/PS2 Status and Control Register USC (Address 1AH)**

Register	Bits	Labels	Read/Write	Functions
USC (1AH)	0	SUSPEND	R	USB suspend mode status bit. When 1, indicates that the USB system entry is in suspend mode.
	1	RMOT_WK	W	USB remote wake-up signal. Default value is 0.
	2	URST_FLAG	R/W	USB bus reset event flag. Default value is 0.
	3	RESUME_O	R	When RESUME_OUT EVENT, RESUME_O is set to 1. Default value is 0.
	4	PS2_DAI	R	USBD-/DATA input
	5	PS2_CKI	R	USBD+/CLK input
	6	PS2_DAO	W	Output for driving USBD-/DATA pin, when working under 3D PS2 mouse function. Default value is 1.
	7	PS2_CKO	W	Output for driving USBD-/DATA pin, when working under 3D PS2 mouse function. Default value is 1.

**Endpoint Interrupt Status Register USR (Address 1BH)**

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".

Register	Bits	Labels	Read/Write	Functions
USR (1BH)	0	EP0IF	R/W	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. Default value is "0".
	1	EP1IF	R/W	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. Default value is "0".
	2~3	—	R/W	Reserved bit, set to "0"
	4	SELPS2	R/W	When set to "1", indicates that the chip is working under PS2 mode. Default value is "0".
	5	SELUSB	R/W	When set to "1", indicates that the chip is working under USB mode. Default value is 0.
	6	—	R/W	Reserved bit, set to "0"
	7	USB_flag	R/W	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".

**Clock Control Register SCC (Address 1CH)**

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).

Register	Bits	Labels	Read/Write	Functions
SCC (1CH)	2~0	—	R/W	Reserved
	3	USBCKEN	R/W	USB clock control bit. When set to "1", indicates a USBCK ON, else USBCK OFF. Default value is "0".
	4	SUSPEND2	R/W	This bit is used to reduce power consumption in the suspend mode. In the normal mode this bit must be cleared to zero (Default="0"). In the HALT mode this bit should be set high to reduce power consumption and LVR with no function. In the USB mode this bit cannot be set high.
	5	—	R/W	Reserved
	6	—	R/W	Reserved bit, set to "1"
	7	PS2_flag	R/W	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".

Note: The user must set bit 6 in the SCC register high after a power-on reset.





## Instruction Set

### Introduction

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

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

### Instruction Timing

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

### Moving and Transferring Data

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

### Arithmetic Operations

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

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

### Logical and Rotate Operations

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

### Branches and Control Transfer

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

### Bit Operations

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

### Table Read Operations

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

### Other Operations

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

### Instruction Set Summary

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

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0-7 number of bits

addr: Program memory address

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

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

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

**Instruction Definition**

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

www.DataSheet4U.com

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

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

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



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

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

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

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

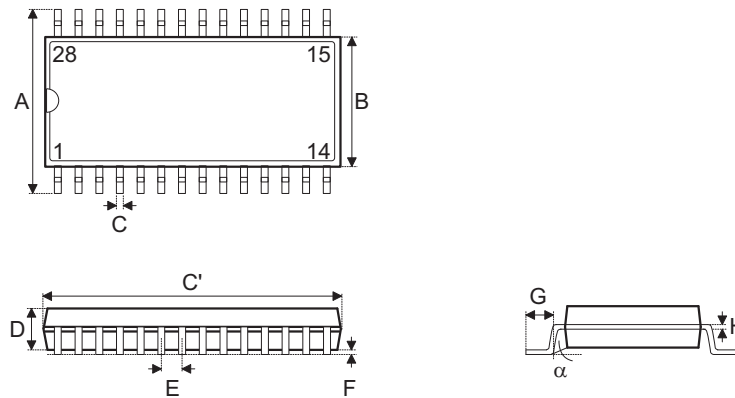
www.DataSheet4U.com

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

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

**Package Information**

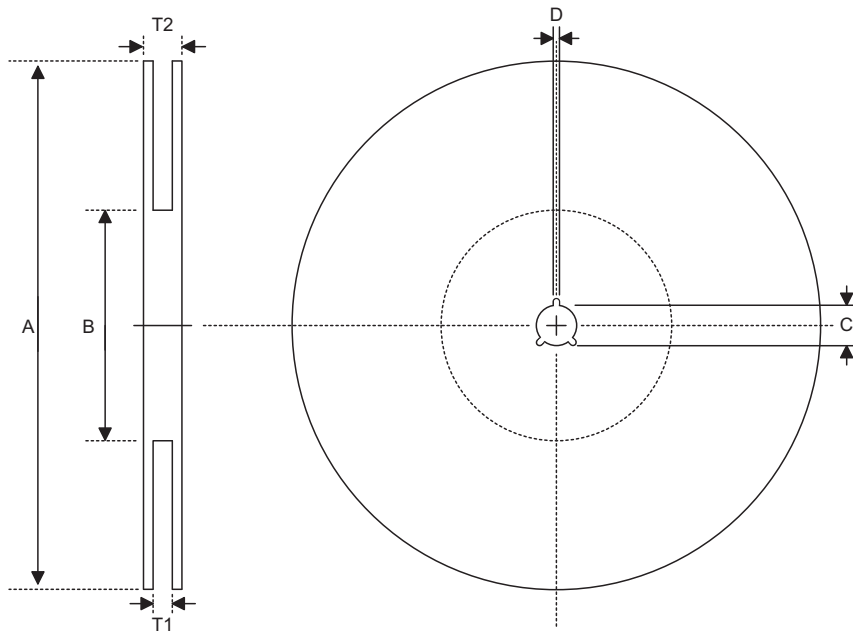
**28-pin SSOP (150mil) Outline Dimensions**



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	228	—	244
B	150	—	157
C	8	—	12
C'	386	—	394
D	54	—	60
E	—	25	—
F	4	—	10
G	22	—	28
H	7	—	10
$\alpha$	0°	—	8°

**Product Tape and Reel Specifications**

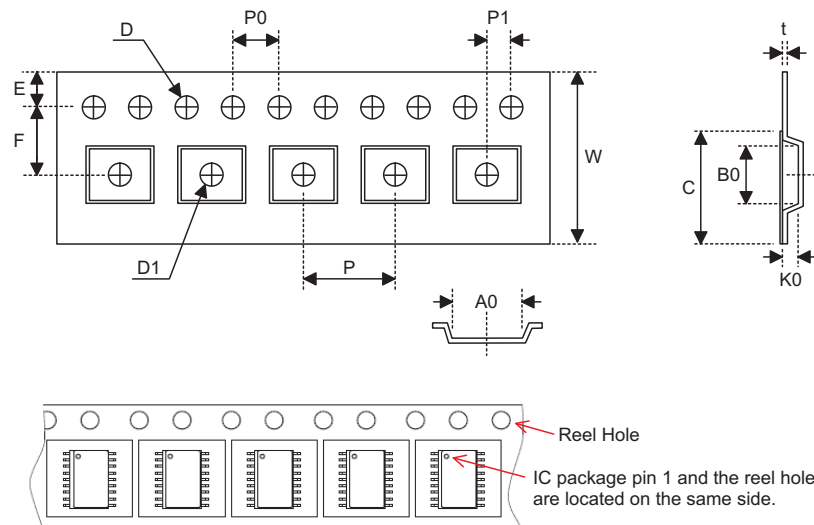
**Reel Dimensions**



SSOP 28S (150mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	62.0±1.5
C	Spindle Hole Diameter	13.0 <sup>+0.5/-0.2</sup>
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	16.8 <sup>+0.3/-0.2</sup>
T2	Reel Thickness	22.2±0.2



**Carrier Tape Dimensions**

**SSOP 28S (150mil)**

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0±0.3
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	7.5±0.1
D	Perforation Diameter	1.55 <sup>+0.10/-0.00</sup>
D1	Cavity Hole Diameter	1.50 <sup>+0.25/-0.00</sup>
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	10.3±0.1
K0	Cavity Depth	2.1±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

**Holtek Semiconductor Inc. (Headquarters)**

No.3, Creation Rd. II, Science Park, Hsinchu, Taiwan  
Tel: 886-3-563-1999  
Fax: 886-3-563-1189  
<http://www.holtek.com.tw>

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

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

www.DataSheet4U.com

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

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

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

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

Copyright © 2009 by HOLTEK SEMICONDUCTOR INC.

The information appearing in this Data Sheet is believed to be accurate at the time of publication. However, Holtek assumes no responsibility arising from the use of the specifications described. The applications mentioned herein are used solely for the purpose of illustration and Holtek makes no warranty or representation that such applications will be suitable without further modification, nor recommends the use of its products for application that may present a risk to human life due to malfunction or otherwise. Holtek's products are not authorized for use as critical components in life support devices or systems. Holtek reserves the right to alter its products without prior notification. For the most up-to-date information, please visit our web site at <http://www.holtek.com.tw>.