HT48R01A
Small Package I/O Type 8-Bit OTP MCU

Technical Document

- Tools Information
- FAQs
- Application Note
  - HA0003E Communicating between the HT48 & HT46 Series MCUs and the HT93LC46 EEPROM
  - HA0016E Writing and Reading to the HT24 EEPROM with the HT48 MCU Series
  - HA0018E Controlling the HT1621 LCD Controller with the HT48 MCU Series
  - HA0049E Read and Write Control of the HT1380
  - HA0075E MCU Reset and Oscillator Circuits Application Note

Features

- Operating voltage:
  f_{SYS}= 4MHz: 2.2V~5.5V
  f_{SYS}= 8MHz: 3.3V~5.5V
  f_{SYS}= 12MHz: 4.5V~5.5V
- 7 bidirectional I/O lines and 1 input line
- External interrupt input shared with an I/O line
- Single 8-bit programmable Timer/Event Counter with overflow interrupt and 7-stage prescaler
- External crystal system oscillator
- Fully integrated internal RC oscillator available with three frequencies: 4MHz, 8MHz or 12MHz
- Watchdog Timer function
- Buzzer function

- Power down and wake-up functions to reduce power consumption
- Up to 0.33μs instruction cycle with 12MHz system clock at V_{DD}= 5V
- 4-level subroutine nesting
- Bit manipulation instruction
- Table read instructions
- 63 powerful instructions
- All instructions executed in one or two machine cycles
- Low voltage reset function
- 10-pin ultra-small MSOP package

General Description

The HT48R01A is 8-bit high performance, RISC architecture microcontroller specifically designed for a wide range of applications. The usual Holtek microcontroller features of low power consumption, I/O flexibility, timer functions, oscillator options, power down and wake-up functions, watchdog timer and low voltage reset, combine to provide devices with a huge range of functional options while still maintaining cost effectiveness. Their fully integrated system oscillator, with three frequency selections, and being supplied in an extremely small outline MSOP 10-pin package opens up a huge range of new application possibilities for these devices, some of which may include industrial control, consumer products, household appliances subsystem controllers, etc.
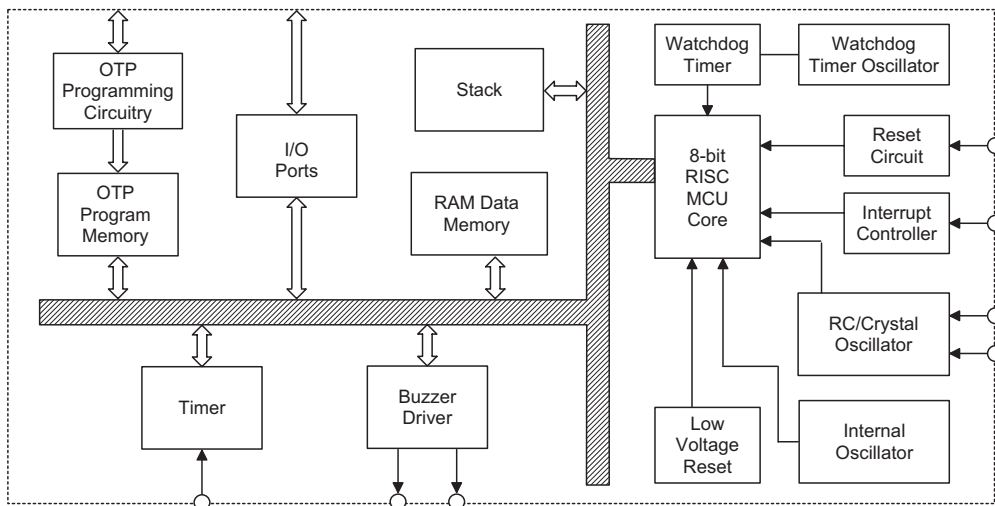
Device Markings

A suffix with the number 1, 2 or 3 will be added to each of the part numbers and printed on the package. The suffix number denotes the fixed oscillation frequency of the internal RC oscillator as shown.
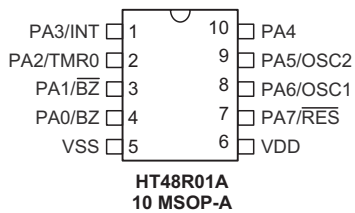
Part No.	Device Marking	Internal RC Oscillator Frequency
HT48R01A	48R01A1	4MHz
	48R01A2	8MHz
	48R01A3	12MHz

Block Diagram

The following block diagram illustrates the main functional blocks.

Pin Assignment

HT48R01A
10 MSOP-A

Pin Description

Pin Name	I/O	Configuration Options	Description
PA0/BZ PA1/$\overline{BZ}$	I/O	—	Bidirectional 2 line I/O. Each pin can be setup as a wake-up input using the wake-up register. Software instructions determine if the pin is a CMOS output or a Schmitt Trigger input. A pull-high resistor can be connected using the pull-high register. PA0/PA1 are pin-shared with the BZ/$\overline{BZ}$ input pins, the function of which is chosen via a configuration option.
PA2/TMR0	I/O	—	Bidirectional single line I/O. The pin can be setup as a wake-up input using the wake-up register. Software instructions determine if the pin is a CMOS output or a Schmitt Trigger input. A pull-high resistor can be connected using the pull-high register. PA2 is pin-shared with the TMR0 input.
PA3/INT	I/O	—	Bidirectional single line I/O. The pin can be setup as a wake-up input using the wake-up register. Software instructions determine if the pin is a CMOS output or a Schmitt Trigger input. A pull-high resistor can be connected using the pull-high register. PA3 is pin-shared with the INT input.
PA4	I/O	—	Bidirectional single line I/O. The pin can be setup as a wake-up input using the wake-up register. Software instructions determine if the pin is a CMOS output or a Schmitt Trigger input. A pull-high resistor can be connected using the pull-high register.
PA6/OSC1 PA5/OSC2	I/O	RC, Crystal, RTC or I/O	Bidirectional 2-line I/O. The pins can be setup as wake-up inputs using the wake-up register. Software instructions determine if the pin is a CMOS output or a Schmitt Trigger input. A pull-high resistors can be connected using the pull-high register. Configuration options determine if the pins are to be used as oscillator pins or I/O pins. Configuration options also determine which oscillator mode is selected. The four oscillator modes are: 1. Internal RC OSC: both pins configured as I/Os. 2. External crystal OSC: both pins configured as OSC1/OSC2. 3. Internal RC + RTC OSC: both pins configured as OSC1, OSC2. 4. External RC OSC+PA5: PA6 configured as OSC1 pin, PA5 configured as I/O. If the internal RC OSC is selected, the frequency will be fixed at either 4MHz, 8MHz or 12MHz, dependent upon which device is chosen.
PA7/$\overline{RES}$	I	PA7 or $\overline{RES}$	Active low Schmitt trigger reset input or PA7 input. A configuration option determines which function is selected. PA7 has a wake-up function but does not have a pull-high function.
VDD	—	—	Positive power supply
VSS	—	—	Negative power supply, ground

Note: Each pin on PA except PA7 can be selected to have a pull-high resistor.

Absolute Maximum Ratings

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

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

I_{OL} Total 150mA

Total Power Dissipation 500mW

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

Operating Temperature..........................−40°C to 85°C

I_{OH} Total ....−100mA

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=4MHz	2.2	—	5.5	V
		—	f_{SYS}=8MHz	3.3	—	5.5	V
		—	f_{SYS}=12MHz	4.5	—	5.5	V
I_{DD1}	Operating Current (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=4MHz	—	1	2	mA
		5V		—	2.5	5	mA
I_{DD2}	Operating Current (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=8MHz	—	2	4	mA
		5V		—	4	8	mA
I_{DD3}	Operating Current (Crystal OSC, RC OSC)	5V	No load, f_{SYS}=12MHz	—	6	12	mA
I_{DD4}	Operating Current (Internal RC+RTC OSC, Normal Mode)	3V	No load, f_{SYS}=4MHz	—	1	2	mA
		5V		—	2.5	5	mA
I_{DD5}	Operating Current (Internal RC+RTC OSC, Normal Mode)	3V	No load, f_{SYS}=8MHz	—	2	4	mA
		5V		—	4	8	mA
I_{DD6}	Operating Current (Internal RC+RTC OSC, Normal Mode)	5V	No load, f_{SYS}=12MHz	—	6	12	mA
I_{DD7}	Operating Current (Internal RC+RTC OSC, Slow Mode)	3V	No load, f_{SYS}=32768Hz	—	20	30	μA
		5V		—	40	60	μA
I_{STB1}	Standby Current (WDT Enabled, RTC Off)	3V	No load, system HALT	—	—	5	μA
		5V		—	—	10	μA
I_{STB2}	Standby Current (WDT Disabled, RTC Off)	3V	No load, system HALT	—	—	1	μA
		5V		—	—	2	μA
I_{STB3}	Standby Current (WDT Disabled, RTC On)	3V	No load, system HALT QOSC=1	—	—	5	μA
		5V		—	—	10	μA
V_{IL1}	Input Low Voltage for PA0~PA7, TMR0 and INT	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage forPA0~PA7, TMR0 and INT	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
V_{LVR1}	Low Voltage Reset 1	—	Configuration option: 4.2V	3.98	4.2	4.42	V

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
		V_{DD}	Conditions				
V_{LVR2}	Low Voltage Reset 2	—	Configuration option: 3.15V	2.98	3.15	3.32	V
V_{LVR3}	Low Voltage Reset 3	—	Configuration option: 2.1V	1.98	2.1	2.22	V
I_{OL}	I/O Port Sink Current	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V		10	20	—	mA
I_{OH}	I/O Port Source Current	3V	V_{OH}=0.9V_{DD}	−2	−4	—	mA
		5V		−5	−10	—	mA
R_{PH}	Pull-high Resistance	3V	—	20	60	100	kΩ
		5V		10	30	50	kΩ

A.C. Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
		V_{DD}	Conditions				
f_{SYS1}	System Clock (Crystal OSC, RC OSC)	—	2.2V~5.5V	400	—	4000	kHz
		—	3.3V~5.5V	400	—	8000	kHz
		—	4.5V~5.5V	400	—	12000	kHz
f_{SYS2}	System Clock (Internal RC OSC) (±5%)	4.5V~5.5V	12MHz, Ta=25°C	11400	12000	12600	kHz
		3.3V~5.5V	8MHz, Ta=25°C	7600	8000	8400	kHz
		2.7V~5.5V	4MHz, Ta=25°C	3800	4000	4200	kHz
f_{SYS3}	System Clock (32768 Crystal)	—	—	—	32768	—	Hz
f_{TIMER}	Timer I/P Frequency (TMR)	—	2.2V~5.5V	0	—	4000	kHz
		—	3.3V~5.5V	0	—	8000	kHz
		—	4.5V~5.5V	0	—	12000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	3V	—	45	90	180	μs
		5V	—	32	65	130	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Wake-up from Power Down	—	1024	—	t_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1	2	ms
V_{POR}	VDD Start Voltage to Ensure Power-on Reset	—	—	—	—	100	mV
R_{POR}	VDD Rise Time to Ensure Power-on Reset	—	—	0.035	—	—	V/ms

Note: t_{SYS}=1/f_{SYS1}, 1/f_{SYS2} or 1/f_{SYS3}

System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to the internal system architecture. The range of devices take advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one cycle, with the exception of branch or call instructions. An 8-bit wide ALU is used in practically all operations of the instruction set. It carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O and A/D control system with maximum reliability and flexibility.

Clocking and Pipelining

The main system clock, derived from either a Crystal/Resonator or RC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the decoding and execution functions. In this way, one T1~T4 clock cycle forms one instruction cycle. Although the fetching and execution of instructions takes place in consecutive instruction cycles, the pipelining structure of the microcontroller ensures that instructions are effectively executed in one instruction cycle. The exception to this are instructions where the contents of the Program Counter are changed, such as subroutine calls or jumps, in which case the instruction will take one more instruction cycle to execute.

For instructions involving branches, such as jump or call instructions, two machine cycles are required to complete instruction execution. An extra cycle is required as the program takes one cycle to first obtain the actual jump or call address and then another cycle to actually execute the branch. The requirement for this extra cycle should be taken into account by programmers in timing sensitive applications
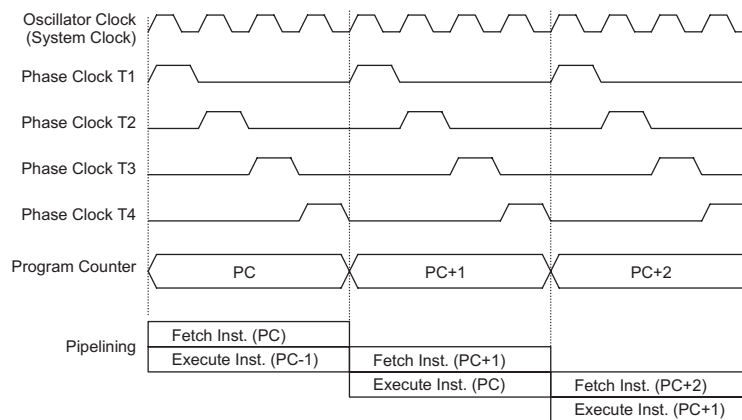
System Clocking and Pipelining
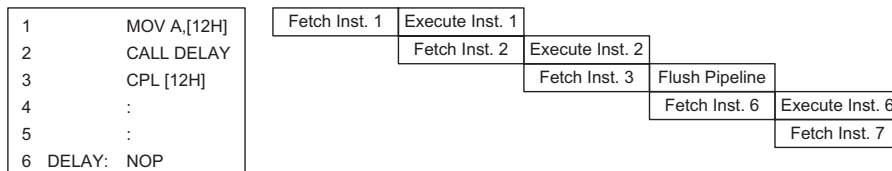
Instruction Fetching

Program Counter

During program execution, the Program Counter is used to keep track of the address of the next instruction to be executed. It is automatically incremented by one each time an instruction is executed except for instructions, such as ″JMP″ or ″CALL″ that demand a jump to a non-consecutive Program Memory address. Note that the Program Counter width varies with the Program Memory capacity depending upon which device is selected. However, it must be noted that only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by user.

When executing instructions requiring jumps to non-consecutive addresses such as a jump instruction, a subroutine call, interrupt or reset, etc., the microcontroller manages program control by loading the required address into the Program Counter. For conditional skip instructions, once the condition has been met, the next instruction, which has already been fetched during the present instruction execution, is discarded and a dummy cycle takes its place while the correct instruction is obtained.

The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly, however, as only this low byte is available for manipulation, the jumps are limited to the present page of memory, that is 256 locations. When such program jumps are executed it should also be noted that a dummy cycle will be inserted.

The lower byte of the Program Counter is fully accessible under program control. Manipulating the PCL might cause program branching, so an extra cycle is needed to pre-fetch. Further information on the PCL register can be found in the Special Function Register section.

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack has 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 device reset, the Stack Pointer will point to the top of the stack.

If the stack is full and an enabled 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. However, when the stack is full, a CALL subroutine instruction can still be executed which will result in a stack overflow. Precautions should be taken to avoid such cases which might cause unpredictable program branching.
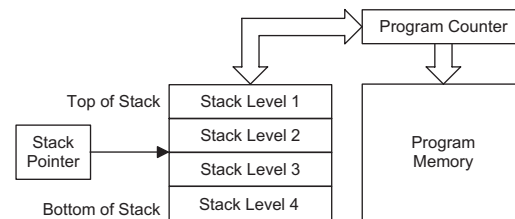
Mode	Program Counter Bits									
	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
Initial Reset	0	0	0	0	0	0	0	0	0	0
External Interrupt	0	0	0	0	0	0	0	1	0	0
Timer/Event Counter Overflow	0	0	0	0	0	0	1	0	0	0
Skip	Program Counter + 2									
Loading PCL	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

Note: PC9~PC8: Current Program Counter bits
        @7~@0: PCL bits
        #9~#0: Instruction code address bits
        S9~S0: Stack register bits

Arithmetic and Logic Unit − ALU

The arithmetic-logic unit or ALU is a critical area of the microcontroller that carries out arithmetic and logic operations of the instruction set. Connected to the main microcontroller data bus, the ALU receives related instruction codes and performs the required arithmetic or logical operations after which the result will be placed in the specified register. As these ALU calculation or operations may result in carry, borrow or other status changes, the status register will be correspondingly updated to reflect these changes. The ALU supports the following functions:

- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA

- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA

- Rotation RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC

- Increment and Decrement INCA, INC, DECA, DEC

- Branch decision, JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

Program Memory

The Program Memory is the location where the user code or program is stored. The device is supplied with One-Time Programmable, OTP, memory where users can program their application code into the device. By using the appropriate programming tools, OTP devices offer users the flexibility to freely develop their applications which may be useful during debug or for products requiring frequent upgrades or program changes. OTP devices are also applicable for use in applications that require low or medium volume production runs.

Structure

The Program Memory has a capacity of 1K by 14. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by separate table pointer registers.

Special Vectors

Within the Program Memory, certain locations are reserved for special usage such as reset and interrupts.

- Location 000H
  This vector is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

- Location 004H
  This vector is used by the external interrupt. If the external interrupt pin on the device receives an edge transition, the program will jump to this location and

begin execution if the external interrupt is enabled and the stack is not full. The external interrupt active edge transition type, whether high to low, low to high or both is specified in the CTRL1 register.

- Location 008H
  This internal vector is used by the Timer/Event Counter 0. If a counter overflow occurs, the program will jump to this location and begin execution if the timer/event counter interrupt is enabled and the stack is not full.
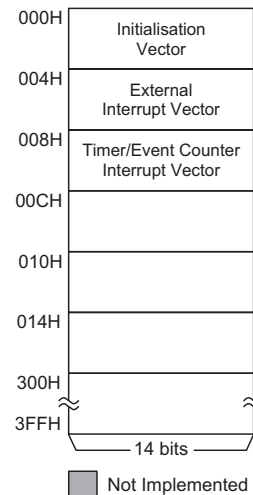
Program Memory Structure

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be setup by placing the lower order address of the look up data to be retrieved in the table pointer register, TBLP. This register defines the lower 8-bit address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the current Program Memory page or last Program Memory page using the ″TABRDC[m]″ or ″TABRDL [m]″ instructions, respectively. When these instructions are executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. Any unused bits in this transferred higher order byte will be read as ″0″.

Instruction	Table Location Bits									
	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC [m]	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: PC9~PC8: Current Program Counter bits

@7~@0: Table Pointer TBLP bits

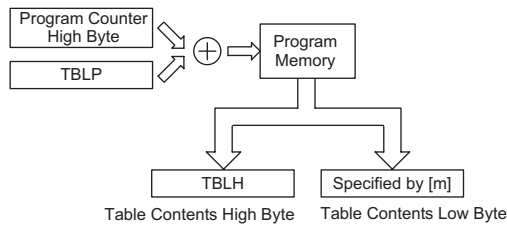The following diagram illustrates the addressing/data flow of the look-up table:

Table Contents High Byte   Table Contents Low Byte

Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the device. This example uses raw table data located in the last page

which is stored there using the ORG statement. The value at this ORG statement is ″300H″ which refers to the start address of the last page within the 1K Program Memory of the HT48R01A microcontrollers. The table pointer is setup here to have an initial value of ″06H″. This will ensure that the first data read from the data table will be at the Program Memory address ″306H″ or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the ″TABRDC [m]″ instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the ″TABRDL [m]″ instruction is executed.

```
tempreg1    db      ?        ; temporary register #1
tempreg2    db      ?        ; temporary register #2
            :
            :
mov         a,06h            ; initialise table pointer - note that this address
                             ; is referenced
mov         tblp,a           ; to the last page or present page
            :
            :
tabrdl      tempreg1         ; transfers value in table referenced by table pointer
                             ; to tempregl
                             ; data at prog. memory address ″306H″ transferred to
                             ; tempreg1 and TBLH
dec         tblp             ; reduce value of table pointer by one
tabrdl      tempreg2         ; transfers value in table referenced by table pointer
                             ; to tempreg2
                             ; data at prog.memory address ″305H″ transferred to
                             ; tempreg2 and TBLH
                             ; in this example the data ″1AH″ is transferred to
                             ; tempreg1 and data ″0FH″ to register tempreg2
                             ; the value ″00H″ will be transferred to the high byte
                             ; register TBLH
            :
            :
org         300h                ; sets initial address of last page
dc          00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
            :
            :
```

Because the TBLH register is a read-only register and cannot be restored, care should be taken to ensure its protection if both the main routine and Interrupt Service Routine use the table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of TBLH and subsequently cause errors if used again by the main routine. As a rule it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

Data Memory

The Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored. Divided into two sections, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is reserved for general purpose use. All locations within this area are read and write accessible under program control.

Structure

The two sections of Data Memory, the Special Purpose and General Purpose Data Memory are located at consecutive locations. All are implemented in RAM and are 8 bits wide but the length of each memory section is dictated by the type of microcontroller chosen. The start address of the Data Memory for all devices is the address ″00H″. Registers which are common to all microcontrollers, such as ACC, PCL, etc., have the same Data Memory address.
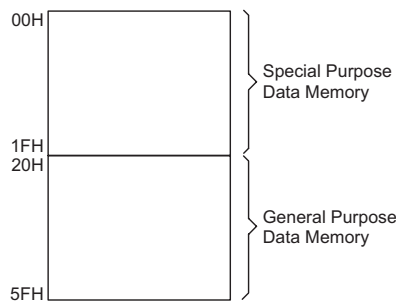
Data Memory Structure

Note: Most of the Data Memory bits can be directly manipulated using the ″SET [m].i″ and ″CLR [m].i″ with the exception of a few dedicated bits. The Data Memory can also be accessed through the memory pointer registers.

General Purpose Data Memory

All microcontroller programs require an area of read/write memory where temporary data can be stored and retrieved for use later. It is this area of RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user program for both read and write operations. By using the ″SET [m].i″ and ″CLR [m].i″ instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.
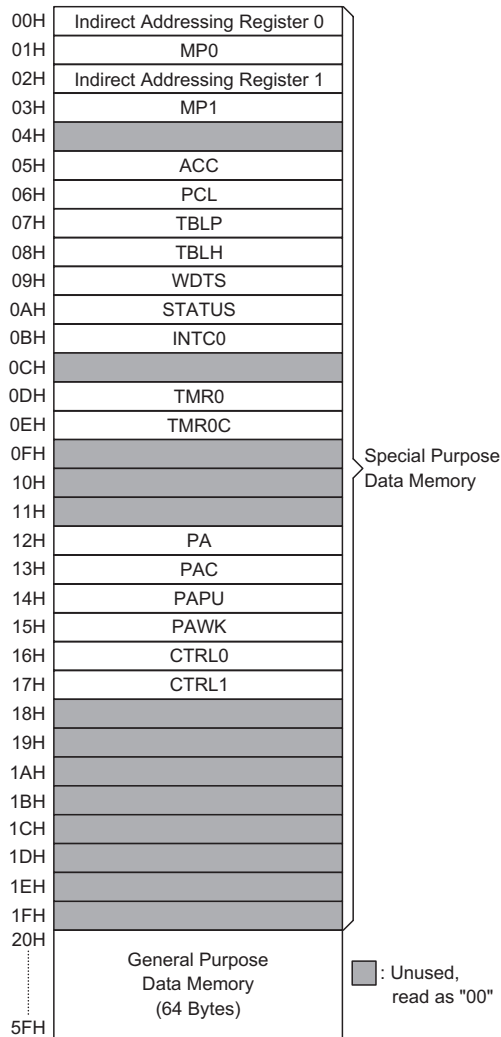
Special Purpose Data Memory

This area of Data Memory is where registers, necessary for the correct operation of the microcontroller, are stored. Most of the registers are both readable and writable but some are protected and are readable only, the details of which are located under the relevant Special Function Register section. Note that for locations that are unused, any read instruction to these addresses will return the value ″00H″.

Special Function Registers

To ensure successful operation of the microcontroller, certain internal registers are implemented in the Data Memory area. These registers ensure correct operation of internal functions such as timers, interrupts, etc., as well as external functions such as I/O data control and A/D converter operation. The location of these registers within the Data Memory begins at the address 00H. Any unused Data Memory locations between these special function registers and the point where the General Purpose Memory begins is reserved and attempting to read data from these locations will return a value of 00H.

Indirect Addressing Registers − IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointer, MP0 or MP1. Acting as a pair, IAR0 with MP0 and IAR1 with MP1 can together access data from the Data Memory. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of ″00H″ and writing to the registers indirectly will result in no operation.

00H	Indirect Addressing Register 0
01H	MP0
02H	Indirect Addressing Register 1
03H	MP1
04H	
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	WDTS
0AH	STATUS
0BH	INTC0
0CH	
0DH	TMR0
0EH	TMR0C
0FH	
10H	
11H	
12H	PA
13H	PAC
14H	PAPU
15H	PAWK
16H	CTRL0
17H	CTRL1
18H	
19H	
1AH	
1BH	
1CH	
1DH	
1EH	
1FH	
20H	General Purpose Data Memory (64 Bytes)
5FH	

Special Purpose Data Memory

: Unused, read as "00"

Special Purpose Data Memory

Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to, is the address specified by the related Memory Pointer, bit 7 of the Memory Pointers is not required to address the full memory space. It must be noted that when bit 7 of the Memory Pointers for these devices is read, a value of ″1″ will be returned. The following example shows how to clear a section of four Data Memory locations already defined as locations adres1 to adres4.

```
data .section  'data'
adres1        db ?
adres2        db ?
adres3        db ?
adres4        db ?
block         db ?
code .section at 0 'code'
org   00h

start:
        mov a,04h            ; setup size of block
        mov block,a
        mov a,offset adres1; Accumulator loaded with first RAM address
        mov mp0,a           ; setup memory pointer with first RAM address

loop:
        clr IAR0            ; clear the data at address defined by MP0
        inc mp0             ; increment memory pointer
        sdz block           ; check if last memory location has been cleared
        jmp loop

continue:
```

The important point to note here is that in the example shown above, no reference is made to specific Data Memory addresses.

Accumulator – ACC

The Accumulator is central to the operation of any microcontroller and is closely related with operations carried out by the ALU. The Accumulator is the place where all intermediate results from the ALU are stored. Without the Accumulator it would be necessary to write the result of each calculation or logical operation such as addition, subtraction, shift, etc., to the Data Memory resulting in higher programming and timing overheads. Data transfer operations usually involve the temporary storage function of the Accumulator; for example, when transferring data between one user defined register and another, it is necessary to do this by passing the data through the Accumulator as no direct transfer between two registers is permitted.

Program Counter Low Register – PCL

To provide additional program control functions, the low byte of the Program Counter is made accessible to programmers by locating it within the Special Purpose area of the Data Memory. By manipulating this register, direct jumps to other program locations are easily implemented. Loading a value directly into this PCL register will cause a jump to the specified Program Memory location, however, as the register is only 8-bit wide, only jumps within the current Program Memory page are permitted. When such operations are used, note that a dummy cycle will be inserted.

Look-up Table Registers – TBLP, TBLH

These two special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP is the table pointer and indicates the location where the table data is located. Its value must be setup before any table read commands are executed. Its value can be changed, for example using the ″INC″ or ″DEC″ instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.
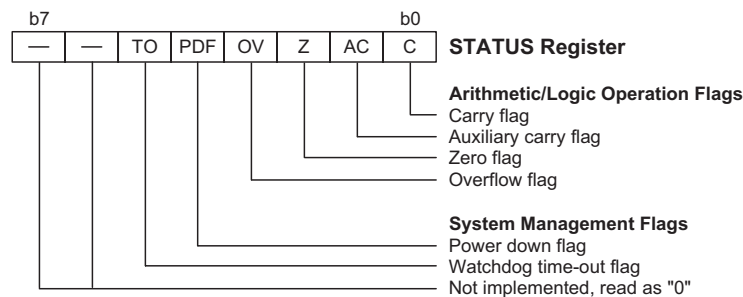
Status Register – STATUS

This 8-bit register 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). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

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

- **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.

- **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.

- **Z** is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.

- **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.

- **PDF** is cleared by a system power-up or executing the ″CLR WDT″ instruction. PDF is set by executing the ″HALT″ instruction.

- **TO** is cleared by a system power-up or executing the ″CLR WDT″ or ″HALT″ instruction. TO is set by a WDT time-out.

Status Register

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

Interrupt Control Registers − INTC0

The 8-bit register, known as INTC0, control the operation of both external, and internal timer interrupts. By setting various bits within this register using standard bit manipulation instructions, the enable/disable function of each interrupt can be independently controlled. A master interrupt bit within this register, the EMI bit, acts like a global enable/disable and is used to set all of the interrupt enable bits on or off. This bit is cleared when an interrupt routine is entered to disable further interrupt and is set by executing the ″RETI″ instruction.

Timer/Event Counter Register

The device contains one integrated 8-bit Timer/Event Counter. An associated register known as TMR0 is the location where the timer's 8-bit value is located. An associated control register, known as TMR0C, contains the setup information for this timer.

Input/Output Ports and Control Registers

Within the area of Special Function Registers, the port PA data I/O register and its associated control register PAC play a prominent role. These registers are mapped to specific addresses within the Data Memory as shown in the Data Memory table. The PA data I/O register, is used to transfer the appropriate output or input data on the PA port. The PAC control register specifies which pins of PA are set as inputs and which are set as outputs. To setup a pin as an input, the corresponding bit of the control register must be set high, for an output it must be set low. During program initialisation, it is important to first setup the control registers to specify which pins are outputs and which are inputs before reading data from or writing data to the I/O ports. One flexible

feature of these registers is the ability to directly program single bits using the ″SET [m].i″ and ″CLR [m].i″ instructions. The ability to change I/O pins from output to input and vice versa by manipulating specific bits of the I/O control registers during normal program operation is a useful feature of these devices.

System Control Register − CTRL0

This register is used to provide control over certain internal functions including certain system clock options, the buzzer selection control and an RTC Oscillator quick start up function.

When the buzzer function, BZ or $\overline{BZ}$, is disabled, then these two pins will have a normal I/O function. If enabled, the BZ or $\overline{BZ}$ outputs will still be controlled by the related I/O port control and data settings.
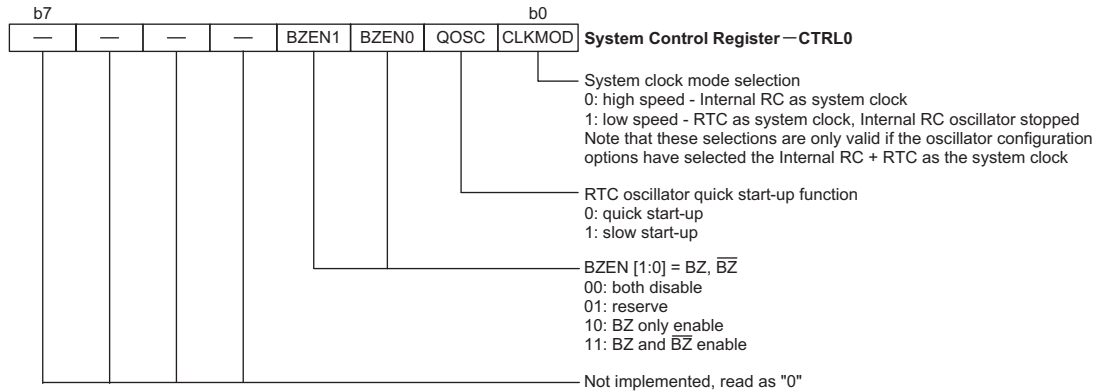
System Control Register − CTRL1

This register is used to provide control over certain internal functions including the External Interrupt edge trigger type and the Watchdog Timer control function.

Wake-up Function Register − PAWK

When the microcontroller enters the Power Down Mode, various methods exist to wake the device up and continue with normal operation. One method is to allow a low going edge on the I/O pins to have a wake-up function. This register is used to select which I/O pins are used to have this wake-up function.

Pull-high Register − PAPU

The I/O pins, if configured as inputs, can have internal pull-high resistors connected, which eliminates the need for external pull-high resistors. This register selects which I/O pins are connected to internal pull-high resistors.

System Control Register － CTRL0
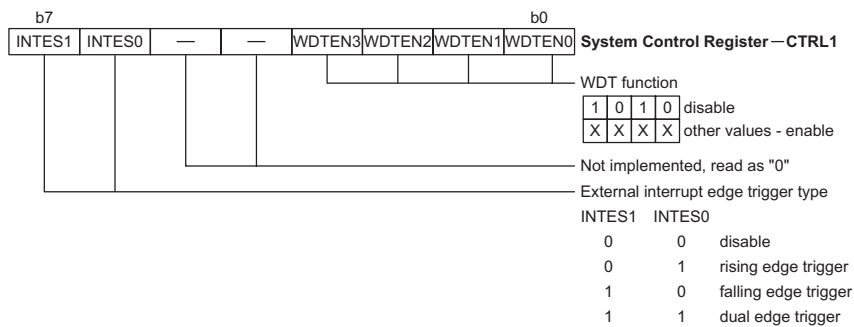
System Control Register － CTRL1

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. There are seven I/O pins whose input or output designation is under user program control and an additional input pin. Additionally, as there are pull-high resistor and wake-up software configurations for each pin, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

Each device has a single I/O port known as Port A, which has a corresponding data register known as PA. This register is mapped to the Data Memory with an address as shown in the Special Purpose Data Memory table. Seven of these I/O lines can be used for input and output operations and one line as an input only. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction ″MOV A,[m]″, where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, I/O pins PA0~PA6, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selectable via a register known as PAPU, located in the Data Memory. The pull-high resistors are implemented using weak PMOS transistors.

Port A Wake-up

If the HALT instruction is executed, the device will enter the Power Down Mode, where the system clock will stop resulting in power being conserved, a feature that is important for battery and other low-power applications. Various methods exist to wake-up the microcontroller, one of which is to change the logic condition on one of the PA0~PA7 pins from high to low. After a HALT instruction forces the microcontroller into entering the Power Down Mode, the processor will remain idle or in a low-power state until the logic condition of the selected wake-up pin on Port A changes from high to low. This function is especially suitable for applications that can be woken up via external switches. Note that pins PA0 to PA7 can be selected individually to have this wake-up feature using an internal register known as PAWK, located in the Data Memory.

I/O Port Control Registers

Port A has its own control register, known as PAC, which controls the input/output configuration. With this control register, each PA0~PA6 I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. Pins PA0 to PA6 port are directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a ″1″. This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a ″0″, the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

Pin-shared Functions

The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. For some pins, the chosen function of the multi-function I/O pins is set by configuration options while for others the function is set by application program control.

- External Interrupt Input
  The external interrupt pin, INT, is pin-shared with the I/O pin PA3. To use the pin as an external interrupt input the correct bits in the INTCO register must be programmed. The pin must also be setup as an input by setting the appropriate bit in the Port Control Register. A pull-high resistor can also be selected via the appropriate port pull-high resistor register. Note that even if the pin is setup as an external interrupt input the I/O function still remains.

- External Timer/Event Counter Input
  The device contains a single Timer/Event Counter. The Timer/Event Counter has an external input pin, known as TMR0 which is pin-shared with the I/O pin PA2. For the shared pin to be used as a Timer/Event Counter input, the Timer/Event Counter must be configured to be in the Event Counter or Pulse Width Measurement Mode. This is achieved by setting the appropriate bits in the Timer/Event Counter Control Register. The pin must also be setup as an input by setting the appropriate bit in the Port Control Register. Pull-high resistor options can also be selected via the port pull-high resistor register. Note that even if the pin is setup as an external timer input the I/O function still remains.

- Buzzer
  Buzzer output pins BZ and $\overline{BZ}$ are pin-shared with I/O pins PA0 and PA1. The buzzer function is selected by setting the appropriate bits in the CTRL0 register. Note that the corresponding bits of the port control register, PAC, must setup the pins as outputs to enable the buzzer outputs. If the PAC port control register has setup the pins as inputs, then the pins will functions as normal logic inputs with the usual pull-high options, even if the buzzer function has been selected.

I/O Pin Structures

The diagrams illustrate the I/O pin internal structures. As the exact logical construction of the I/O pin may differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins.

Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, the PA data register and PAC port control register will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high options have been selected. If the PAC port control register, is then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated PA port data register is first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct value into the port control register or by programming individual bits in the port control register using the ″SET [m].i″ and ″CLR [m].i″ instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then re-write this data back to the output ports.


Read/Write Timing
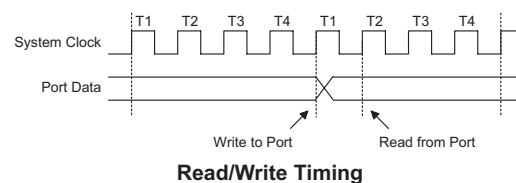
Pins PA0 to PA7 each have a wake-up functions, selected via the PAWK register. When the device is in the Power Down Mode, various methods are available to wake the device up. One of these is a high to low transition of any of the these pins. Single or multiple pins on Port A can be setup to have this function.
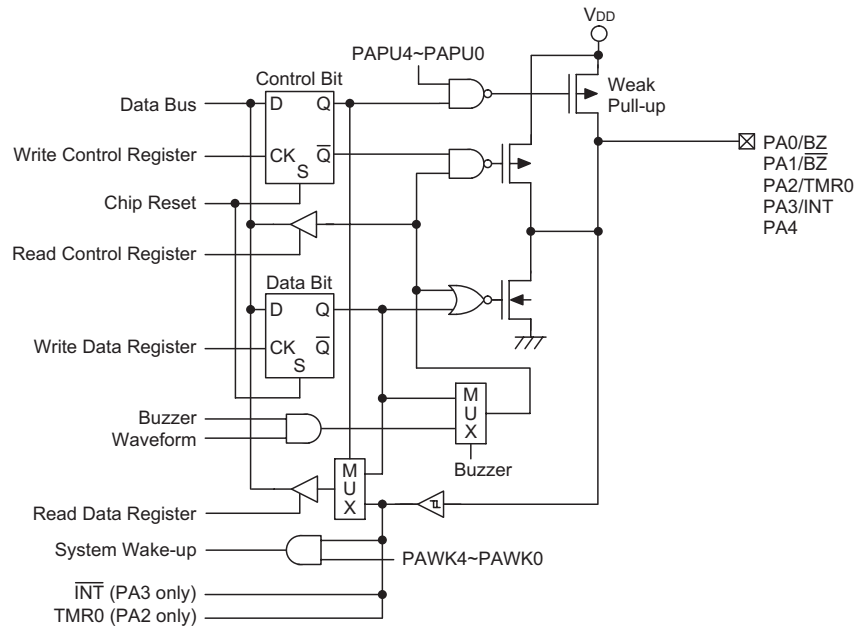
PA0~PA4 Input/Output Ports
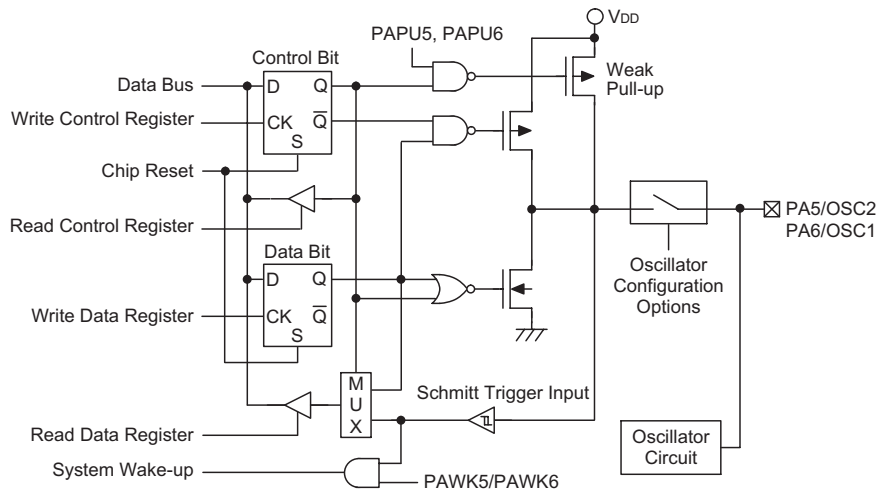
PA5~PA6 Input/Output Ports
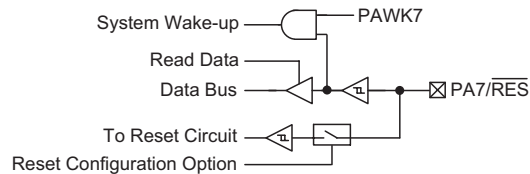
PA7 Input Port

Timer/Event Counters

The provision of timers form an important part of any microcontroller, giving the designer a means of carrying out time related functions. The device contains one count-up timer of 8-bit capacity. As the timer has three different operating modes, it can be configured to operate as a general timer, an external event counter or as a pulse width measurement device. The provision of an internal prescaler to the clock circuitry gives added range to the timer.

There are two types of registers related to the Timer/Event Counters. The first is the register that contains the actual value of the timer and into which an initial value can be preloaded. Reading from this register retrieves the contents of the Timer/Event Counter. The second type of associated register is the Timer Control Register which defines the timer options and determines how the timer is to be used. The device can have the timer clock configured to come from the internal clock source. In addition, the timer clock source can also be configured to come from an external timer pin.

An external clock source is used when the timer is in the event counting mode, the clock source being provided on the external timer pin, known as TMR0. These external timer pins are pin-shared with other I/O pins. Depending upon the condition of the T0E bit in the Timer Control Register, each high to low, or low to high transition on the external timer input pin will increment the counter by one.

Configuring the Timer/Event Counter Input Clock Source

The internal timer's clock can originate from various sources, depending upon which device and which timer is chosen. The system clock input timer source is used when the timer is in the timer mode or in the pulse width measurement mode. For Timer/Event Counter 0 this system clock timer source is first divided by a prescaler, the division ratio of which is conditioned by the Timer Control Register bits T0PSC0~T0PSC2.

An external clock source is used when the timer is in the event counting mode, the clock source being provided on an external timer pin TMR0 or TMR1, depending upon which device and which timer is used. Depending

upon the condition of the T0E bit, each high to low, or low to high transition on the external timer pin will increment the counter by one.

Timer Registers – TMR0

The timer registers are special function registers located in the Special Purpose Data Memory and is the place where the actual timer value is stored. This register is known as TMR0. The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reset with the initial preload register value and continue counting.

Note that to achieve a maximum full range count of FFH, the preload register must first be cleared to all zeros. It should be noted that after power-on, the preload registers will be in an unknown condition. Note that if the Timer/Event Counters are in an OFF condition and data is written to their preload registers, this data will be immediately written into the actual counter. However, if the counter is enabled and counting, any new data written into the preload data registers during this period will remain in the preload registers and will only be written into the actual counter the next time an overflow occurs.

Timer Control Registers – TMR0C

The flexible features of the Holtek microcontroller Timer/Event Counters enable them to operate in three different modes, the options of which are determined by the contents of their respective control register.

The Timer Control Register is known as TMR0C. It is the Timer Control Register together with its corresponding timer register that control the full operation of the Timer/Event Counter. Before the timer can be used, it is essential that the Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

To choose which of the three modes the timer is to operate in, either in the timer mode, the event counting mode
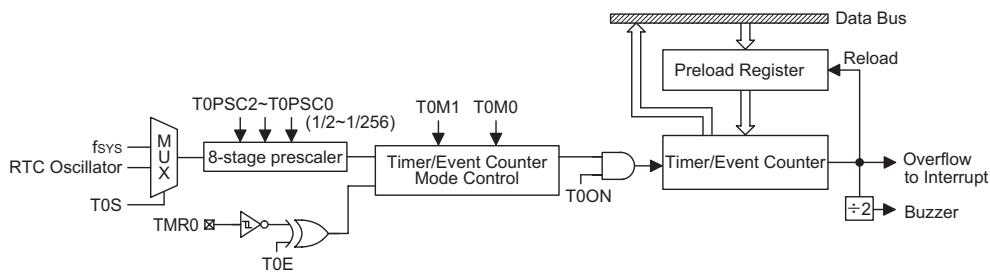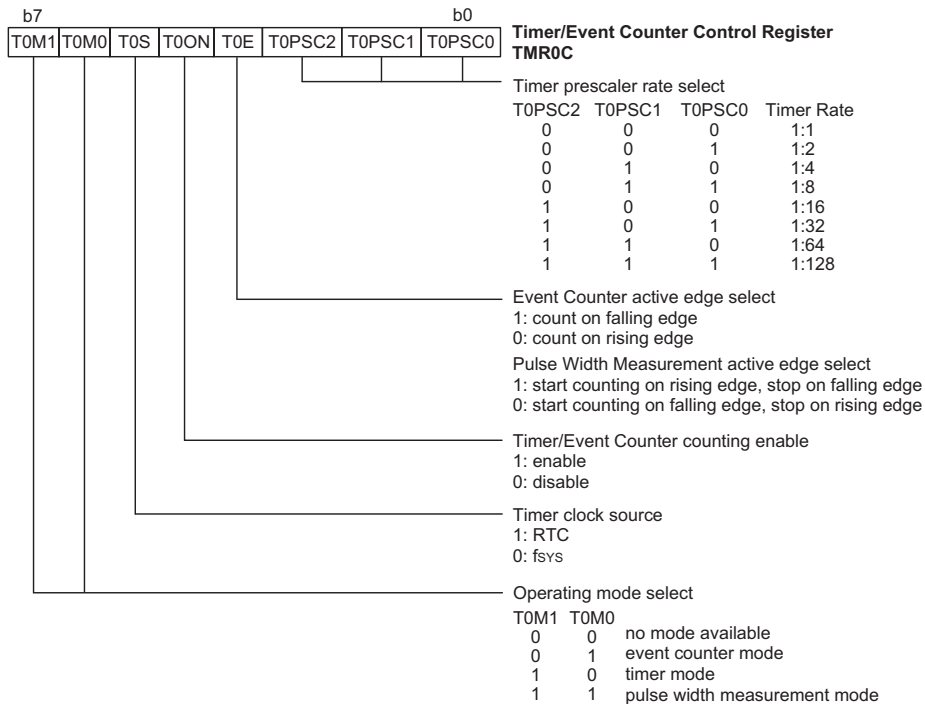

8-bit Timer/Event Counter Structure

or the pulse width measurement mode, bits 7 and 6 of the Timer Control Register, which are known as the bit pair T0M1/T0M0, must be set to the required logic levels. The timer-on bit, which is bit 4 of the Timer Control Register and known as T0ON, provides the basic on/off control of the respective timer. Setting the bit high allows the counter to run, clearing the bit stops the counter. Bits 0~2 of the Timer Control Register determine the division ratio of the input clock prescaler. The prescaler bit settings have no effect if an external clock source is used. If the timer is in the event count or pulse width measurement mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as T0E.
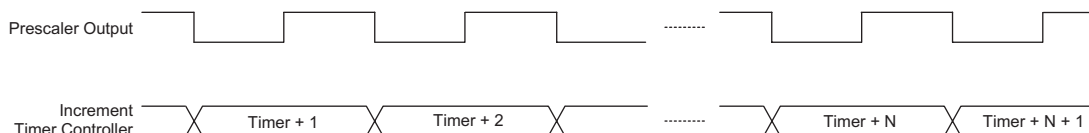
Configuring the Timer Mode

In this mode, the timer can be utilized to measure fixed time intervals, providing an internal interrupt signal each time the counter overflows. To operate in this mode, the

bit pair, T0M1/T0M0, must be set to 1 and 0 respectively. In this mode the internal clock is used as the timer clock. The timer input clock source is either fsys or the RTC oscillator. However, this timer clock source is further divided by a prescaler, the value of which is determined by the bits T0PSC2~T0PSC0 in the Timer Control Register. The timer-on bit, T0ON must be set high to enable the timer to run. Each time an internal clock high to low transition occurs, the timer increments by one; when the timer is full and overflows, an interrupt signal is generated and the timer will preload the value already loaded into the preload register and continue counting. A timer overflow condition and corresponding internal interrupt is one of the wake-up sources, however, the internal interrupts can be disabled by ensuring that the ET0I bits of the INTC0 register are reset to zero.

b7							b0
T0M1	T0M0	T0S	T0ON	T0E	T0PSC2	T0PSC1	T0PSC0

Timer/Event Counter Control Register TMR0C

Timer prescaler rate select

T0PSC2	T0PSC1	T0PSC0	Timer Rate
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

Event Counter active edge select
1: count on falling edge
0: count on rising edge

Pulse Width Measurement active edge select
1: start counting on rising edge, stop on falling edge
0: start counting on falling edge, stop on rising edge

Timer/Event Counter counting enable
1: enable
0: disable

Timer clock source
1: RTC
0: fsys

Operating mode select

T0M1	T0M0	
0	0	no mode available
0	1	event counter mode
1	0	timer mode
1	1	pulse width measurement mode

TMR0C Register

Prescaler Output

Increment Timer Controller — Timer + 1 — Timer + 2 — .... — Timer + N — Timer + N + 1
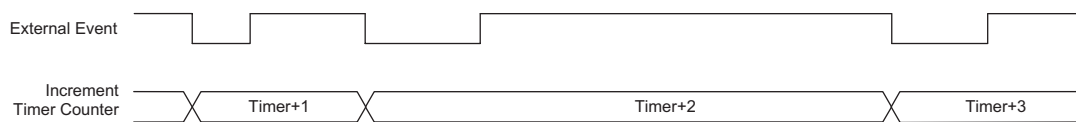
Timer Mode Timing Chart

Configuring the Event Counter Mode

In this mode, a number of externally changing logic events, occurring on the external timer pin, can be recorded by the internal timer. For the timer to operate in the event counting mode, the bit pair, T0M1/T0M0 must be set to 0 and 1 respectively. The timer-on bit T0ON or T1ON, depending upon which timer is used, must be set high to enable the timer to count. If T0E is low, the counter will increment each time the external timer pin receives a low to high transition. If T0E is high, the counter will increment each time the external timer pin receives a high to low transition. As in the case of the other two modes, when the counter is full, the timer will overflow and generate an internal interrupt signal. The counter will then preload the value already loaded into the preload register. As the external timer pins are pin-shared with other I/O pins, to ensure that the pin is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the T0M1/T0M0 bits place the Timer/Event Counter in the event counting mode, the second is to ensure that the port control register configures the pin as an input. It should be noted that a timer overflow is one of the interrupt and wake-up sources. Note that the timer interrupts can be disabled by ensuring that the ET0I bits in the INTC0 register are reset to zero.

Configuring the Pulse Width Measurement Mode

In this mode, the width of external pulses applied to the external timer pin can be measured. In the Pulse Width Measurement Mode the timer clock source is supplied by the internal clock. For the timer to operate in this mode, the bit pair, T0M1/T0M0 must both be set high. If the T0E bit is low, once a high to low transition has been received on the external timer pin, the timer will start counting until the external timer pin returns to its original high level. At this point the T0ON bit will be automatically

reset to zero and the timer will stop counting. If the T0E bit is high, the timer will begin counting once a low to high transition has been received on the external timer pin and stop counting when the external timer pin returns to its original low level. As before, the T0ON bit will be automatically reset to zero and the timer will stop counting. It is important to note that in the Pulse Width Measurement Mode, the T0ON bit is automatically reset to zero when the external control signal on the external timer pin returns to its original level, whereas in the other two modes the T0ON bit can only be reset to zero under program control. The residual value in the timer, which can now be read by the program, therefore represents the length of the pulse received on the external timer pin. As the T0ON bit has now been reset, any further transitions on the external timer pin, will be ignored. Not until the T0ON bit is again set high by the program can the timer begin further pulse width measurements. In this way, single shot pulse measurements can be easily made. It should be noted that in this mode the counter is controlled by logical transitions on the external timer pin and not by the logic level. As in the case of the other two modes, when the counter is full, the timer will overflow and generate an internal interrupt signal. The counter will also be reset to the value already loaded into the preload register. If the external timer pin is pin-shared with other I/O pins, to ensure that the pin is configured to operate as a pulse width measuring input pin, two things have to happen. The first is to ensure that the T0M1/T0M0 bits place the Timer/Event Counter in the pulse width measuring mode, the second is to ensure that the port control register configures the pin as an input. It should be noted that a timer overflow and corresponding timer interrupt is one of the wake-up sources. Note that the timer interrupts can be disabled by ensuring that the ET0I bits in the INTC0 register are reset to zero.

Event Counter Mode Timing Chart
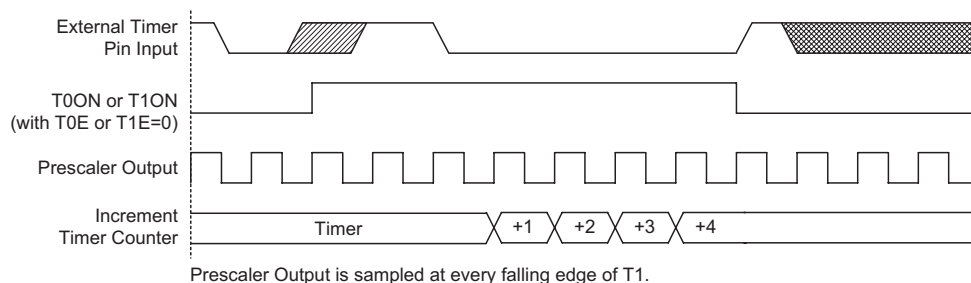
Prescaler Output is sampled at every falling edge of T1.

Pulse Width Measure Mode Timing Chart

Prescaler

Bits T0PSC0~T0PSC2 of the TMR0C register can be used to define a division ratio for the internal clock source of the Timer/Event Counter enabling longer time out periods to be setup.

Buzzer Function

Operating similar to a programmable frequency divider, the buzzer function within the microcontroller provides a means of producing a variable frequency output suitable for applications, such as piezo-buzzer driving or other interfaces requiring a precise frequency generator.

BZ and $\overline{BZ}$ pins are a complimentary pair and pin-shared with I/O pins, PA0 and PA1. The function is selected via the CTRL0 register, however, if not selected, the pins can operate as normal I/O pins. Note that the $\overline{BZ}$ pin is the inverse of the BZ pin generating a kind of differential output and supplying more power to connected interfaces such as buzzers. The CTRL0 register can select a single BZ pin or the complimentary pair BZ and $\overline{BZ}$.

The timer overflow signal is the clock source for the buzzer circuit. The output frequency is controlled by loading the required values into the timer prescaler and timer registers to give the required division ratio. The counter will begin to count-up from this preload register value until full, at which point an overflow signal is generated, causing both the BZ and $\overline{BZ}$ outputs to change state. The counter will then be automatically reloaded with the preload register value and continue counting- up.

If the CTRL0 register has selected the buzzer function, then for both buzzer outputs to operate, it is essential that the Port A control register PAC bit 0 and PAC bit 1 are setup as outputs. If only one pin is setup as an output, the other pin can still be used as a normal data input pin. However, if both pins are setup as inputs then the buzzer will not function. The buzzer outputs will only be activated if bit PA0 is set to "1". This output data bit is used as the on/off control bit for the buzzer outputs. Note that the BZ and $\overline{BZ}$ outputs will both be low if the PA0 output data bit is cleared to "0". The condition of

data bit PA1 has no effect on the overall control of the BZ and $\overline{BZ}$ pins.

Using this method of frequency generation, and if a crystal oscillator is used for the system clock, very precise values of frequency can be generated.

I/O Interfacing

The Timer/Event Counter, when configured to run in the event counter or pulse width measurement mode, requires the use of an external timer pin for its operation. As this pin is a shared pin it must be configured correctly to ensure that it is setup for use as a Timer/Event Counter input pin. This is achieved by ensuring that the mode select bits in the Timer/Event Counter control register, select either the event counter or pulse width measurement mode. Additionally the corresponding PAC Port Control Register bit must be set high to ensure that the pin is setup as an input. Any pull-high resistor connected to this pin will remain valid even if the pin is used as a Timer/Event Counter input.

Programming Considerations

When configured to run in the timer mode, the internal system clock is used as the timer clock source and is therefore synchronised with the overall operation of the microcontroller. In this mode when the appropriate timer register is full, the microcontroller will generate an internal interrupt signal directing the program flow to the respective internal interrupt vector. For the pulse width measurement mode, the internal system clock is also used as the timer clock source but the timer will only run when the correct logic condition appears on the external timer input pin. As this is an external event and not synchronised with the internal timer clock, the microcontroller will only see this external event when the next timer clock pulse arrives. As a result, there may be small differences in measured values requiring programmers to take this into account during programming. The same applies if the timer is configured to be in the event counting mode, which again is an external event and not synchronised with the internal system or timer clock.

Buzzer Function
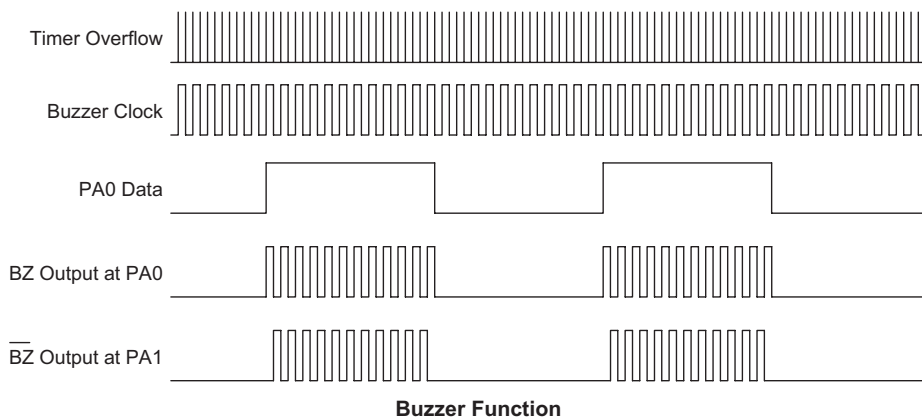
When the Timer/Event Counter is read, or if data is written to the preload register, the clock is inhibited to avoid errors, however as this may result in a counting error, this should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialised before using them for the first time. The associated timer enable bits in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The edge select, timer mode and clock source control bits in timer control register must also be correctly set to ensure the timer is properly configured for the required application. It is also important to ensure that an initial value is first loaded into the timer registers before the timer is switched on; this is because after power-on the initial values of the timer registers are unknown. After the timer has been initialised the timer can be turned on and off by controlling the enable bit in the timer control register.

When the Timer/Event Counter overflows, its corresponding interrupt request flag in the interrupt control register will be set. If the Timer/Event Counter interrupt is enabled this will in turn generate an interrupt signal. However irrespective of whether the interrupts are enabled or not, a Timer/Event Counter overflow will also generate a wake-up signal if the device is in a Power-down condition. This situation may occur if the Timer/Event Counter is in the Event Counting Mode and if the external signal continues to change state. In such a case, the Timer/Event Counter will continue to count these external events and if an overflow occurs the device will be woken up from its Power-down condition. To prevent such a wake-up from occurring, the timer interrupt request flag should first be set high before issuing the ″HALT″ instruction to enter the Power Down Mode.

Timer Program Example

The program shows how the Timer/Event Counter registers are setup along with how the interrupts are enabled and managed. Note how the Timer/Event Counter is turned on, by setting bit 4 of the Timer Control Register. The Timer/Event Counter can be turned off in a similar way by clearing the same bit. This example program sets the Timer/Event Counters to be in the timer mode, which uses the internal system clock as their clock source.

```
:                       :
org 04h                 ; external interrupt vector

org 08h                 ; Timer Counter 0 interrupt vector
jmp tmr0int             ; jump here when Timer 0 overflows
:                       :
org 20h                 ; main program
:                       :
;internal Timer 0 interrupt routine
tmr0int:
:
; Timer 0 main program placed here
:

:
begin:
;setup Timer 0 registers
mov a,09bh              ; setup Timer 0 preload value
mov tmr0,a
mov a,081h              ; setup Timer 0 control register
mov tmr0c,a             ; timer mode and prescaler set to /2
;setup interrupt register
mov a,00dh              ; enable master interrupt and both timer interrupts
mov intc0,a
:                       :
set tmr0c.4             ; start Timer 0
:                       :
```
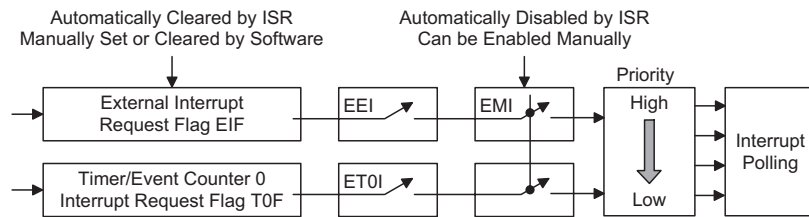
Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer/Event Counter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains a single external interrupt and a single internal interrupt function. The external interrupt is controlled by the action of the external interrupt pin, while the internal interrupt is controlled by the Timer/Event Counter overflow.
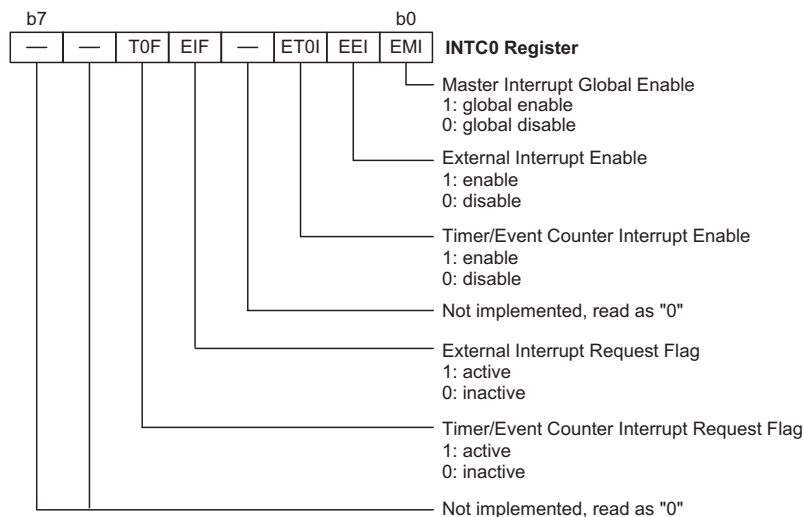
Interrupt Register

Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by a single INTC0 register, which is located in the Data Memory. By controlling the appropriate enable bits in this registers each individual interrupt can be enabled or disabled. Also when an interrupt occurs, the corresponding request flag will be set by the microcontroller. The global enable flag if cleared to zero will disable all interrupts.

Interrupt Operation

A Timer/Event Counter overflow or an active edge on the external interrupt pin will all generate an interrupt request by setting their corresponding request flag, if their appropriate interrupt enable bit is set. When this happens, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a JMP statement which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI statement, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

Interrupt Scheme

INTC0 Register

The various interrupt enable bits, together with their associated request flags, are shown in the following diagram with their order of priority.

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded. If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, 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 Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full.

Interrupt Priority

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 case of simultaneous requests, the following table shows the priority that is applied. These can be masked by resetting the EMI bit.

Interrupt Source	HT48R01A
External Interrupt	1
Timer/Event Counter 0 Overflow	2

In cases where both external and internal interrupts are enabled and where an external and internal interrupt occurs simultaneously, the external interrupt will always have priority and will therefore be serviced first. Suitable masking of the individual interrupts using the interrupt registers can prevent simultaneous occurrences.

External Interrupt

For an external interrupt to occur, the global interrupt enable bit, EEI, and external interrupt enable bit, EEI, must first be set. An actual external interrupt will take place when the external interrupt request flag, EIF, is set, a situation that will occur when an edge transition appears on the external INT line. The type of transition that will trigger an external interrupt, whether high to low, low to high or both is determined by the INTES0 and INTES1 bits, which are bits 6 and 7 respectively, in the CTRL1 control register. These two bits can also disable the external interrupt function.

INTES1	INTES0	Edge Trigger Type
0	0	Disable
0	1	Rising Edge Trigger
1	0	Falling Edge Trigger
1	1	Dual Edge Trigger

The external interrupt pin is pin-shared with the I/O pin PA3 and can only be configured as an external interrupt pin if the corresponding external interrupt enable bit in the INTC0 register has been set and the edge trigger type has been selected using the CTRL1 register. The pin must also be setup as an input by setting the corresponding PAC.3 bit in the port control register. When the interrupt is enabled, the stack is not full and a transition appears on the external interrupt pin, a subroutine call to the external interrupt vector at location 04H, will take place. When the interrupt is serviced, the external interrupt request flag, EIF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor connections on this pin will remain valid even if the pin is used as an external interrupt input.

Timer/Event Counter Interrupt

For a Timer/Event Counter interrupt to occur, the global interrupt enable bit, EMI, and the corresponding timer interrupt enable bit, ET0I, must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, T0F, is set, a situation that will occur when the relevant Timer/Event Counter overflows. When the interrupt is enabled, the stack is not full and a Timer/Event Counter 0 overflow occurs, a subroutine call to the timer interrupt vector at location 08H, will take place. When the interrupt is serviced, the timer interrupt request flag, T0F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Programming Considerations

By disabling the interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt register until the corresponding interrupt is serviced or until the request flag is cleared by a software instruction.

It is recommended that programs do not use the ″CALL subroutine″ instruction 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 the interrupt is not well controlled, the original control sequence will be damaged once a ″CALL subroutine″ is executed in the interrupt subroutine.

All of these interrupts have the capability of waking up the processor when in the Power Down Mode.

Only the Program Counter is pushed onto the stack. If the contents of the register or status register are altered by the interrupt service program, which may corrupt the desired control sequence, then the contents should be saved in advance.

Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the microcontroller. In this case, internal circuitry will ensure that the microcontroller, after a short delay, will be in a well defined state and ready to execute the first program instruction. After this power-on reset, certain important internal registers will be set to defined states before the program commences. One of these registers is the Program Counter, which will be reset to zero forcing the microcontroller to begin program execution from the lowest Program Memory address.

In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{\text{RES}}$ line is forcefully pulled low. In such a case, known as a normal operation reset, some of the microcontroller registers remain unchanged allowing the microcontroller to proceed with normal operation after the reset line is allowed to return high. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup.

Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the $\overline{\text{RES}}$ reset is implemented in situations where the power supply voltage falls below a certain threshold.

Reset Functions

There are five ways in which a microcontroller reset can occur, through events occurring both internally and externally:

- Power-on Reset
  The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. All the I/O port and port control registers will power up in a high condition ensuring that all pins will be first set to inputs.
  Although the microcontroller has an internal RC reset function, if the VDD power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{\text{RES}}$ pin, whose additional time delay will ensure that the $\overline{\text{RES}}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{\text{RES}}$ line reaches a certain voltage value, the reset delay time t_{RSTD} is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.
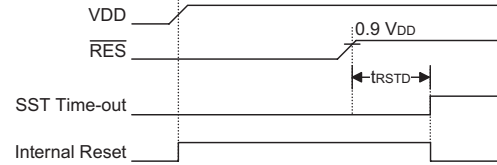
Power-On Reset Timing Chart

For most applications a resistor connected between VDD and the $\overline{\text{RES}}$ pin and a capacitor connected between VSS and the $\overline{\text{RES}}$ pin will provide a suitable external reset circuit. Any wiring connected to the $\overline{\text{RES}}$ pin should be kept as short as possible to minimise any stray noise interference.
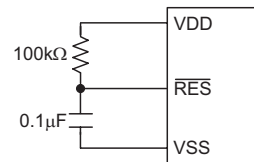
Basic Reset Circuit

For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.
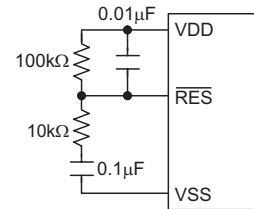
Enhanced Reset Circuit

More information regarding external reset circuits is located in Application Note HA0075E on the Holtek website.

- $\overline{\text{RES}}$ Pin Reset
  This type of reset occurs when the microcontroller is already running and the $\overline{\text{RES}}$ pin is forcefully pulled low by external hardware such as an external switch. In this case as in the case of other reset, the Program Counter will reset to zero and program execution initiated from this point.
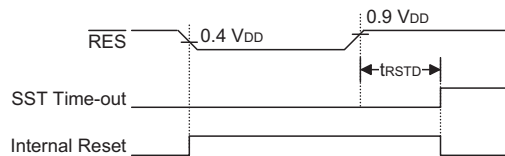
$\overline{\text{RES}}$ Reset Timing Chart

- Low Voltage Reset − LVR
  The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is selected via a configuration option. If the supply voltage of the device drops to within a range of $0.9V\sim V_{LVR}$ such as might occur when changing the battery, the LVR will automatically reset the device internally. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between $0.9V\sim V_{LVR}$ must exist for a time greater than that specified by t_{LVR} in the A.C. characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual V_{LVR} value can be selected via configuration options.
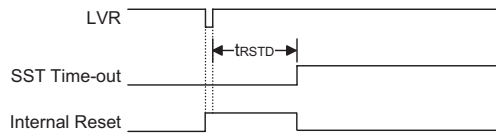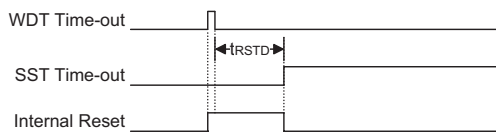

Low Voltage Reset Timing Chart

- Watchdog Time-out Reset during Normal Operation
  The Watchdog time-out Reset during normal operation is the same as a hardware $\overline{RES}$ pin reset except that the Watchdog time-out flag TO will be set to "1".

WDT Time-out Reset during Normal Operation Timing Chart

- Watchdog Time-out Reset during Power Down
  The Watchdog time-out Reset during Power Down is a little different from other kinds of reset. Most of the conditions remain unchanged except that the Program Counter and the Stack Pointer will be cleared to "0" and the TO flag will be set to "1". Refer to the A.C. Characteristics for t_{SST} details.
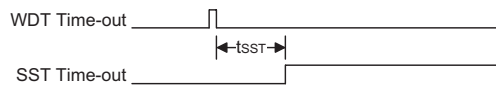
WDT Time-out Reset during Power Down Timing Chart

Note: The SST can be chosen to be either 1024 or 2 clock cycles via configuration option if the system clock source is not provided by an external crystal.

Reset Initial Conditions

The different types of reset described affect the reset flags in different ways. These flags, known as PDF and TO are located in the status register and are controlled by various microcontroller operations, such as the Power Down function or Watchdog Timer. The reset flags are shown in the table:

TO	PDF	RESET Conditions
0	0	$\overline{RES}$ reset during power-on
u	u	$\overline{RES}$ or LVR reset during normal operation
1	u	WDT time-out reset during normal operation
1	1	WDT time-out reset during Power Down

Note: "u" stands for unchanged

The following table indicates the way in which the various components of the microcontroller are affected after a power-on reset occurs.

Item	Condition After RESET
Program Counter	Reset to zero
Interrupts	All interrupts will be disabled
WDT	Clear after reset, WDT begins counting
Timer/Event Counter	Timer Counter will be turned off
Prescaler	The Timer Counter Prescaler will be cleared
Input/Output Ports	I/O ports will be setup as inputs
Stack Pointer	Stack Pointer will point to the top of the stack

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects each of the microcontroller internal registers.

Register	Reset (Power-on)	RES or LVR Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
MP0	1 x x x x x x x	u u u u u u u u	1 u u u u u u u	1 u u u u u u u
MP1	1 x x x x x x x	u u u u u u u u	1 u u u u u u u	1 u u u u u u u
ACC	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
TBLP	x x x x x x x x	u u u u u u u u	u u u u u u u u	u u u u u u u u
TBLH	– – x x x x x x	– – u u u u u u	– – u u u u u u	– – u u u u u u
WDTS	– – – – – 1 1 1	– – – – – 1 1 1	– – – – – 1 1 1	– – – – – u u u
STATUS	– – 0 0 x x x x	– – u u u u u u	– – 1 u u u u u	– – 1 1 u u u u
INTC0	– 0 0 0 0 0 0 0	– 0 0 0 0 0 0 0	– 0 0 0 0 0 0 0	– u u u u u u u
TMR0	x x x x x x x x	x x x x x x x x	x x x x x x x x	u u u u u u u u
TMR0C	0 0 0 0 1 0 0 0	0 0 0 0 1 0 0 0	0 0 0 0 1 0 0 0	u u u u u u u u
PA	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	u u u u u u u u
PAC	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	1 1 1 1 1 1 1 1	u u u u u u u u
PAPU	– 0 0 0 0 0 0 0	– 0 0 0 0 0 0 0	– 0 0 0 0 0 0 0	– u u u u u u u
PAWK	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	u u u u u u u u
CTRL0	– – – – 0 0 0 0	– – – – 0 0 0 0	– – – – 0 0 0 0	– – – – u u u u
CTRL1	1 0 – – 1 0 1 0	1 0 – – 1 0 1 0	1 0 – – 1 0 1 0	u u – – u u u u

Note: ″*″ means ″warm reset″
      ″-″ not implemented
      ″u″ means ″unchanged″
      ″x″ means ″unknown″

Oscillator

Various oscillator options offer the user a wide range of functions according to their various application requirements. Four types of system clocks can be selected while various clock source options for the Watchdog Timer are provided for maximum flexibility. All oscillator options are selected through the configuration options.

The four methods of generating the system clock are:

- External crystal/resonator oscillator
- External RC oscillator
- Internal RC oscillator
- Combined Internal RC oscillator and Real Time Clock

One of these four methods must be selected using the configuration options.

More information regarding the oscillator is located in Application Note HA0075E on the Holtek website.

External Crystal/Resonator Oscillator

The simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation, and will normally not require external capacitors. However, for some crystals and most resonator types, to ensure oscillation and accurate frequency generation, it may be necessary to add two
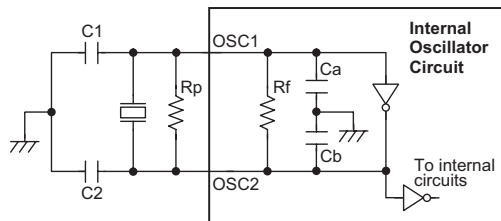
Note: 1. Rp is normally not required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

Crystal/Resonator Oscillator

small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. The external parallel feedback resistor, Rp, is normally not required but in some cases may be needed to assist with oscillation start up.

Internal Ca, Cb, Rf Typical Values @ 5V, 25°C		
Ca	**Cb**	**Rf**
11~13pF	13~15pF	270kΩ

Oscillator Internal Component Values

Crystal Oscillator C1 and C2 Values			
Crystal Frequency	**C1**	**C2**	**CL**
12MHz	TBD	TBD	TBD
8MHz	TBD	TBD	TBD
4MHz	TBD	TBD	TBD
1MHz	TBD	TBD	TBD

Note: 1. C1 and C2 values are for guidance only.
2. CL is the crystal manufacturer specified load capacitor value.

Crystal Recommended Capacitor Values

Resonator C1 and C2 Values		
Resonator Frequency	**C1**	**C2**
3.58MHz	TBD	TBD
1MHz	TBD	TBD
455kHz	TBD	TBD

Note: C1 and C2 values are for guidance only.

Resonator Recommended Capacitor Values

External RC Oscillator

Using the external system RC oscillator requires that a resistor, with a value between 24kΩ and 1.5MΩ, is connected between OSC1 and VDD, and a capacitor is connected to ground. Although this is a cost effective oscillator configuration, the oscillation frequency can vary with VDD, temperature and process variations and is therefore not suitable for applications where timing is critical or where accurate oscillator frequencies are required. For the value of the external resistor R_{OSC} refer to the Holtek website for typical RC Oscillator vs. Temperature and VDD characteristics graphics Here only the OSC1 pin is used, which is shared with I/O pin PA6, leaving pin PA5 free for use as a normal I/O pin. Note that it is the only microcontroller internal circuitry together with the external resistor, that determine the frequency of the oscillator. The external capacitor shown on the diagram does not influence the frequency of oscillation.
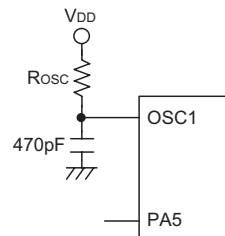
External RC Oscillator

Internal RC Oscillator

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of either 4MHz, 8MHz or 12MHz, the choice of which is indicated by the suffix marking next to the part number of the device used. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins PA5 and PA6 are free for use as normal I/O pins. Refer to the Appendix section for more information on the actual internal oscillator frequency vs. Temperature and VDD characteristics graphics.
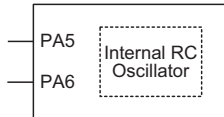
Note: PA5/PA6 used as normal I/Os

Internal RC Oscillator

Internal RC Oscillator + External RTC Oscillator

When the microcontroller enters the Power Down Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the Power Down Mode. To do this,
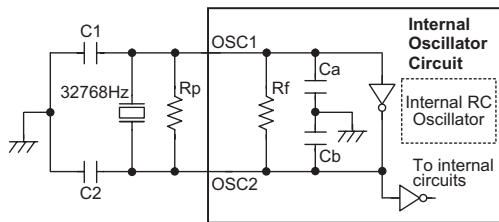
Note: 1. Rp is normally not required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

Internal RC Oscillator + External RTC Oscillator

another clock, independent of the system clock, must be provided. To do this a configuration option exists to allow the Internal System RC Oscillator to be used in conjunction with a Real Time Clock - RTC oscillator. Here the OSC1 and OSC2 pins, which are shared with I/O pins PA6 and PA5, should be connected to a 32768Hz crystal to implement this internal RTC oscillator. However, for some crystals, to ensure oscillation and accurate frequency generation, it may be necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. The external parallel feedback resistor, Rp, is normally not required but in some cases may be needed to assist with oscillation start up.

Internal Ca, Cb, Rf Typical Values @ 5V, 25°C		
Ca	**Cb**	**Rf**
11~13pF	13~15pF	270kΩ

RTC Oscillator Internal Component Values

| RTC Oscillator C1 and C2 Values			
Crystal Frequency	**C1**	**C2**	**CL**
32768Hz	TBD	TBD	TBD
Note: 1. C1 and C2 values are for guidance only. 2. CL is the crystal manufacturer specified load capacitor value.			

32768 Hz Crystal Recommended Capacitor Values

For applications using the RTC oscillator, the system clock can be chosen to be either the Internal System RC Oscillator or the RTC oscillator itself. This selection is made using the CLKMOD bit in the CTRL0 register. If this bit is set high then the 32768Hz external crystal will also provide the system clock source. If the bit is low then the system clock source will be the Internal RC Oscillator. When the system enters the Power Down Mode, the system clock, irrespective of whether the CLKMOD bit has selected the RTC or Internal RC Oscillator as its source, will always stop running. The accompanying table shows the relationship between the CLKMOD bit and the various oscillators.

Operating Mode	CLKMOD Bit	Internal RC Osc.	RTC	System Clock
Normal Running	0	On	On	RC Oscillator
	1	Off	On	32768Hz
Power Down	X	Off	On	Stopped

During power up there is a time delay associated with the RTC oscillator waiting for it to start up. Bit 1 of the CTRL0 register, known as the QOSC bit, is provided to give a quick start-up function and can be used to minimize this delay. During a power up condition, this bit will be cleared to 0 which will initiate the RTC oscillator quick start-up function. However, as there is additional power consumption associated with this quick start-up function, to reduce power consumption after start up takes place, it is recommended that the application program should set the QOSC bit high about 2 seconds after power on. It should be noted that, no matter what condition the QOSC bit is set to, the RTC oscillator will always function normally, only there is more power consumption associated with the quick start-up function.

Watchdog Timer Oscillator

The WDT oscillator is a fully self-contained free running on-chip RC oscillator with a typical period of $65\mu s$ at 5V requiring no external components. When the device enters the Power Down Mode, the system clock will stop running but the WDT oscillator continues to free-run and to keep the watchdog active. However, to preserve power in certain applications the WDT oscillator can be disabled via a configuration option.

Power Down Mode and Wake-up

Power Down Mode

All of the Holtek microcontrollers have the ability to enter a Power Down Mode. When the device enters this mode, the normal operating current, will be reduced to an extremely low standby current level. This occurs because when the device enters the Power Down Mode, the system oscillator is stopped which reduces the power consumption to extremely low levels, however, as the device maintains its present internal condition, it can be woken up at a later stage and continue running, without requiring a full reset. This feature is extremely important in application areas where the MCU must have its power supply constantly maintained to keep the device in a known condition but where the power supply capacity is limited such as in battery applications.

Entering the Power Down Mode

There is only one way for the device to enter the Power Down Mode and that is to execute the ″HALT″ instruction in the application program. When this instruction is executed, the following will occur:

- The system oscillator will stop running and the application program will stop at the ″HALT″ instruction.
- If the RTC oscillator configuration option is enabled then the RTC clock will keep running.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT clock source is selected to come from the WDT or RTC oscillator. The WDT will stop if its clock source originates from the system clock.
- The I/O ports will maintain their present condition.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Standby Current Considerations

As the main reason for entering the Power Down Mode is to keep the current consumption of the MCU to as low a value as possible, perhaps only in the order of several micro-amps, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised.

Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs.

If the configuration options have enabled the Watchdog Timer internal oscillator then this will continue to run when in the Power Down Mode and will thus consume some power. For power sensitive applications it may be therefore preferable to use the system clock source for the Watchdog Timer. The RTC, if configured for use, will also consume a limited amount of power, as it continues to run when the device enters the Power Down Mode. To keep the RTC power consumption to a minimum level the QOSC bit in the CTRL0 register, which controls the quick start up function, should be set high.

Wake-up

After the system enters the Power Down Mode, it can be woken up from one of various sources listed as follows:

- An external reset
- An external falling edge on PA0 to PA7
- A system interrupt
- A WDT overflow

If the system is woken up by an external reset, the device will experience a full system reset, however, if the device is woken up by a WDT overflow, a Watchdog Timer reset will be initiated. Although both of these wake-up methods will initiate a reset operation, the actual source of the wake-up can be determined by examining the TO and PDF flags. The PDF flag is cleared by a system power-up or executing the clear Watchdog Timer instructions and is set when executing the ″HALT″ instruction. The TO flag is set if a WDT time-out occurs, and causes a wake-up that only resets the Program Counter and Stack Pointer, the other flags remain in their original status.

Pins PA0 to PA7 can be setup via the PAWUK register to permit a negative transition on the pin to wake-up the system. When a PA0 to PA7 pin wake-up occurs, the program will resume execution at the instruction following the ″HALT″ instruction.

If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the ″HALT″ instruction. In this situation, the interrupt which woke-up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is

where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set to "1" before entering the Power Down Mode, the wake-up function of the related interrupt will be disabled.

No matter what the source of the wake-up event is, once a wake-up situation occurs, a time period equal to 1024 system clock periods will be required before normal system operation resumes. However, if the wake-up has originated due to an interrupt, the actual interrupt subroutine execution will be delayed by an additional one or more cycles. If the wake-up results in the execution of the next instruction following the "HALT" instruction, this will be executed immediately after the 1024 system clock period delay has ended.

Watchdog Timer

The Watchdog Timer, also known as the WDT, is provided to inhibit program malfunctions caused by the program jumping to unknown locations due to certain uncontrollable external events such as electrical noise. It operates by providing a device reset when the Watchdog Timer counter overflows. Note that if the Watchdog Timer function is not enabled, then any instructions related to the Watchdog Timer will result in no operation.

Setting up the various Watchdog Timer options are controlled via the configuration options and two internal registers WDTS and CTRL1. Enabling the Watchdog Timer can be controlled by both a configuration option and the WDTEN bits in the CTRL1 internal register in the Data Memory.

Configuration Option	CTRL1 Register	WDT Function
Disable	Disable	OFF
Enable	Disable	ON
Disable	Enable	ON
Enable	Enable	ON

Watchdog Timer On/Off Control

The Watchdog Timer will be disabled if bits WDTEN3~WDTEN0 in the CTRL1 register are written with the binary value 1010B and WDT configuration option is disable. This will be the condition when the device is powered up. Although any other data written to WDTEN3~WDTEN0 will ensure that the Watchdog Timer is enabled, for maximum protection it is recommended that the value 0101B is written to these bits.

The Watchdog Timer clock can emanate from three different sources, selected by configuration option. These are its own fully integrated dedicated internal oscillator, the RTC or $f_{SYS}/4$. The Watchdog Timer dedicated internal clock source is an internal oscillator which has an approximate period of 65μs at a supply voltage of 5V. However, it should be noted that this specified internal clock period can vary with VDD, temperature and process variations. The other Watchdog Timer clock source options are the $f_{SYS}/4$ clock and the RTC. It is important to note that when the system enters the Power Down Mode the instruction clock is stopped, therefore if the configuration options have selected $f_{SYS}/4$ as the Watchdog Timer clock source, the Watchdog Timer will cease to function. For systems that operate in noisy environments, using the internal Watchdog Timer internal oscillator or the RTC as the clock source is therefore the recommended choice. No matter which clock source is selected, it is further divided by 256 via an internal 8-bit counter and then by a 7-bit prescaler to give longer time-out periods. The division ratio of the prescaler is determined by bits 0, 1 and 2 of the WDTS register, known as WS0, WS1 and WS2. If the Watchdog Timer internal clock source is selected and with the WS0, WS1 and WS2 bits of the WDTS register all set high, the prescaler division ratio will be 1:128, which will give a maximum time-out period of about 2.1s.
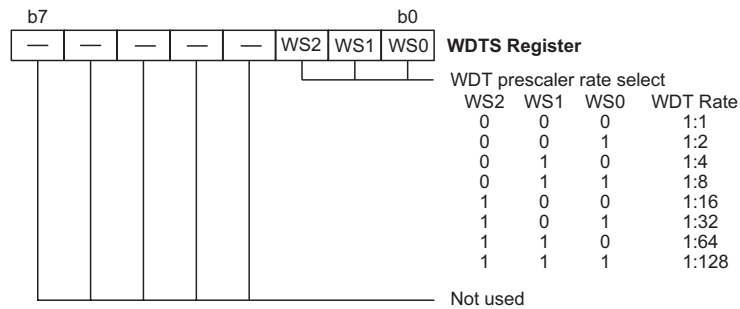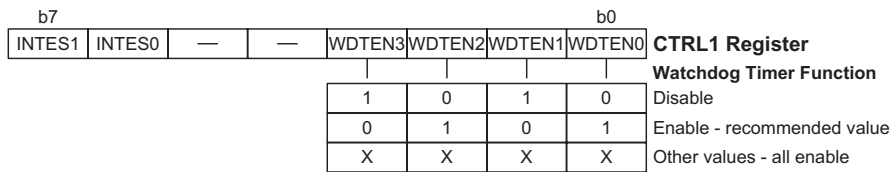
Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the Power Down Mode, when a Watchdog Timer time-out occurs, the device will be woken up, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be
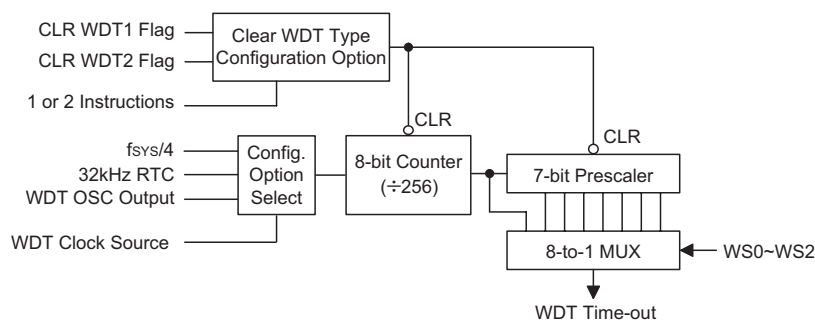
Watchdog Timer

reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is an external hardware reset, which means a low level on the external reset pin, the second is using the Clear Watchdog Timer software instructions and the third is when a HALT instruction is executed. There are two methods of using software instructions to clear the Watchdog Timer, one of which must be chosen by configuration option. The first option is to use the single ″CLR WDT″ instruction while the second is to use the two commands ″CLR WDT1″ and ″CLR WDT2″. For the first option, a simple execution of ″CLR WDT″ will clear the Watchdog Timer

while for the second option, both ″CLR WDT1″ and ″CLR WDT2″ must both be executed to successfully clear the Watchdog Timer. Note that for this second option, if ″CLR WDT1″ is used to clear the Watchdog Timer, successive executions of this instruction will have no effect, only the execution of a ″CLR WDT2″ instruction will clear the Watchdog Timer. Similarly after the ″CLR WDT2″ instruction has been executed, only a successive ″CLR WDT1″ instruction can clear the Watchdog Timer.

b7							b0	
INTES1	INTES0	—	—	WDTEN3	WDTEN2	WDTEN1	WDTEN0	**CTRL1 Register**

Watchdog Timer Function

WDTEN3	WDTEN2	WDTEN1	WDTEN0	
1	0	1	0	Disable
0	1	0	1	Enable - recommended value
X	X	X	X	Other values - all enable

b7							b0	
—	—	—	—	—	WS2	WS1	WS0	**WDTS Register**

WDT prescaler rate select

WS2	WS1	WS0	WDT Rate
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

Not used

Configuration Options

Configuration options refer to certain options within the MCU that are programmed into the OTP Program Memory device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later by the application software. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
1	Watchdog Timer: enable or disable
2	Watchdog Timer clock source: WDT internal oscillator, $f_{SYS}/4$ or RTC
3	CLRWDT instructions: 1 or 2 instructions
4	System oscillator: Internal RC, Internal RC with external RTC, External Crystal, External RC
5	LVR function: enable or disable
6	LVR voltage: 2.1V, 3.15V or 4.2V
7	$\overline{RES}$ or PA7
8	SST: enable (1024 clocks) or disable (2 clocks)

Application Circuits
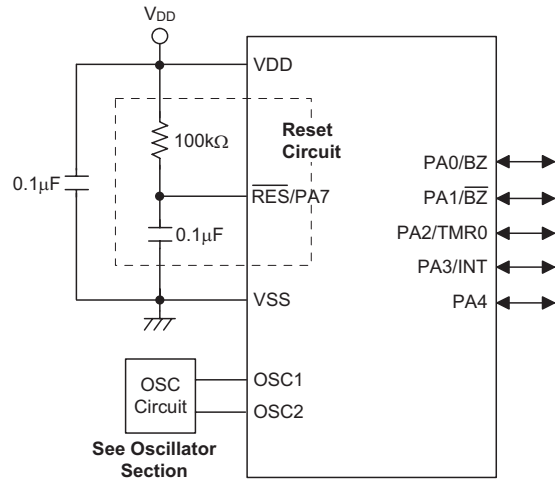
Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

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

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

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

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

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

Instruction Definition

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

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

Operation   ACC ← ACC + [m] + C

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

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

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

Operation   [m] ← ACC + [m] + C

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

ADD A,[m]   Add Data Memory to ACC

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

Operation   ACC ← ACC + [m]

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

ADD A,x   Add immediate data to ACC

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

Operation   ACC ← ACC + x

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

ADDM A,[m]   Add ACC to Data Memory

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

Operation   [m] ← ACC + [m]

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

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

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

Operation   ACC ← ACC ″AND″ [m]

Affected flag(s)   Z

AND A,x   Logical AND immediate data to ACC

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

Operation   ACC ← ACC ″AND″ x

Affected flag(s)   Z

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

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

Operation   [m] ← ACC ″AND″ [m]

Affected flag(s)   Z

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None

CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None

CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None

CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

CLR WDT1	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

CLR WDT2	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

CPL [m]   Complement Data Memory

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

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

Affected flag(s)   Z

CPLA [m]   Complement Data Memory with result in ACC

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

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

Affected flag(s)   Z

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

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

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

Affected flag(s)   C

DEC [m]   Decrement Data Memory

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

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

Affected flag(s)   Z

DECA [m]   Decrement Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] - 1$

Affected flag(s)   Z

HALT   Enter power down mode

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

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

Affected flag(s)   TO, PDF

INC [m]   Increment Data Memory

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

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

Affected flag(s)   Z

INCA [m]   Increment Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] + 1$

Affected flag(s)   Z

JMP addr   Jump unconditionally

Description   The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation   Program Counter $\leftarrow$ addr

Affected flag(s)   None

MOV A,[m]   Move Data Memory to ACC

Description   The contents of the specified Data Memory are copied to the Accumulator.

Operation $ACC \leftarrow [m]$

Affected flag(s)   None

MOV A,x   Move immediate data to ACC

Description   The immediate data specified is loaded into the Accumulator.

Operation $ACC \leftarrow x$

Affected flag(s)   None

MOV [m],A   Move ACC to Data Memory

Description   The contents of the Accumulator are copied to the specified Data Memory.

Operation $[m] \leftarrow ACC$

Affected flag(s)   None

NOP   No operation

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

Operation   No operation

Affected flag(s)   None

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

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

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

Affected flag(s)   Z

OR A,x   Logical OR immediate data to ACC

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

Operation   ACC ← ACC ″OR″ x

Affected flag(s)   Z

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

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

Operation   [m] ← ACC ″OR″ [m]

Affected flag(s)   Z

RET   Return from subroutine

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

Operation   Program Counter ← Stack

Affected flag(s)   None

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

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

Operation   Program Counter ← Stack
ACC ← x

Affected flag(s)   None

RETI   Return from interrupt

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

Operation   Program Counter ← Stack
EMI ← 1

Affected flag(s)   None

RL [m]   Rotate Data Memory left

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

Operation   [m].(i+1) ← [m].i; (i = 0~6)
[m].0 ← [m].7

Affected flag(s)   None

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

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

Operation   ACC.(i+1) ← [m].i; (i = 0~6)
ACC.0 ← [m].7

Affected flag(s)   None

RLC [m] Rotate Data Memory left through Carry

Description The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.

Operation $[m].(i+1) \leftarrow [m].i; (i = 0\sim6)$
$[m].0 \leftarrow C$
$C \leftarrow [m].7$

Affected flag(s) C

RLCA [m] Rotate Data Memory left through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation $ACC.(i+1) \leftarrow [m].i; (i = 0\sim6)$
$ACC.0 \leftarrow C$
$C \leftarrow [m].7$

Affected flag(s) C

RR [m] Rotate Data Memory right

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

Operation $[m].i \leftarrow [m].(i+1); (i = 0\sim6)$
$[m].7 \leftarrow [m].0$

Affected flag(s) None

RRA [m] Rotate Data Memory right with result in ACC

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

Operation $ACC.i \leftarrow [m].(i+1); (i = 0\sim6)$
$ACC.7 \leftarrow [m].0$

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

Description The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.

Operation $[m].i \leftarrow [m].(i+1); (i = 0\sim6)$
$[m].7 \leftarrow C$
$C \leftarrow [m].0$

Affected flag(s) C

RRCA [m] Rotate Data Memory right through Carry with result in ACC

Description Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

Operation $ACC.i \leftarrow [m].(i+1); (i = 0\sim6)$
$ACC.7 \leftarrow C$
$C \leftarrow [m].0$

Affected flag(s) C

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

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

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

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

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

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

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

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

SDZ [m]   Skip if decrement Data Memory is 0

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

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

Affected flag(s)   None

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

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

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

Affected flag(s)   None

SET [m]   Set Data Memory

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

Operation $[m] \leftarrow FFH$

Affected flag(s)   None

SET [m].i   Set bit of Data Memory

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

Operation $[m].i \leftarrow 1$

Affected flag(s)   None

SIZ [m] Skip if increment Data Memory is 0

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

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

Affected flag(s) None

SIZA [m] Skip if increment Data Memory is zero with result in ACC

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

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

Affected flag(s) None

SNZ [m].i Skip if bit i of Data Memory is not 0

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

Operation Skip if $[m].i \neq 0$

Affected flag(s) None

SUB A,[m] Subtract Data Memory from ACC

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

Operation $ACC \leftarrow ACC - [m]$

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

SUBM A,[m] Subtract Data Memory from ACC with result in Data Memory

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

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

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

SUB A,x Subtract immediate data from ACC

Description The immediate data specified by the code is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.

Operation $ACC \leftarrow ACC - x$

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

SWAP [m]   Swap nibbles of Data Memory

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

Operation   [m].3~[m].0 ↔ [m].7 ~ [m].4

Affected flag(s)   None

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

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

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

Affected flag(s)   None

SZ [m]   Skip if Data Memory is 0

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

Operation   Skip if [m] = 0

Affected flag(s)   None

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

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

Operation   ACC ← [m]
                          Skip if [m] = 0

Affected flag(s)   None

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

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

Operation   Skip if [m].i = 0

Affected flag(s)   None

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

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

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

Affected flag(s)   None

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

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

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

Affected flag(s)   None

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

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

Operation   ACC ← ACC ″XOR″ [m]

Affected flag(s)   Z

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

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

Operation   [m] ← ACC ″XOR″ [m]

Affected flag(s)   Z

XOR A,x   Logical XOR immediate data to ACC

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

Operation   ACC ← ACC ″XOR″ x

Affected flag(s)   Z

Device Characteristic Graphics

The following characteristic graphics depicts typical device behavior. The data presented here is a statistical summary of data gathered from different units. This is for information only and the figures were not tested during manufacturing.

In some of the graphs, the data exceeding the specified operating range are shown for information purposes only. The device will operate properly only within the specified range.

Noramlized frequency is the ratio of actual frequency(F) to reference frequency(Fr) at V_{DD}=5V, Ta=25°C

4MHZ IRC Frequency vs. VDD
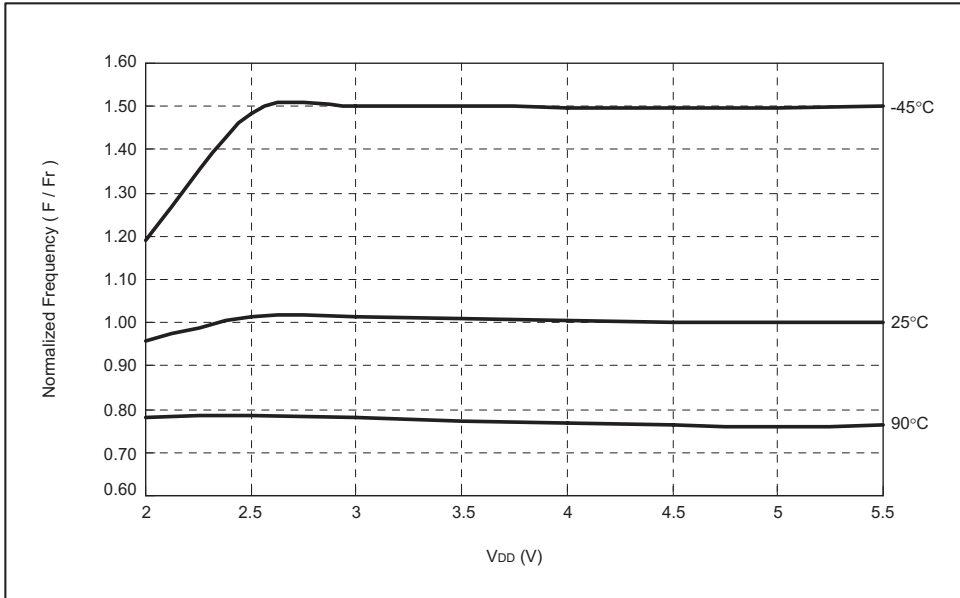
8MHZ IRC Frequency vs. VDD
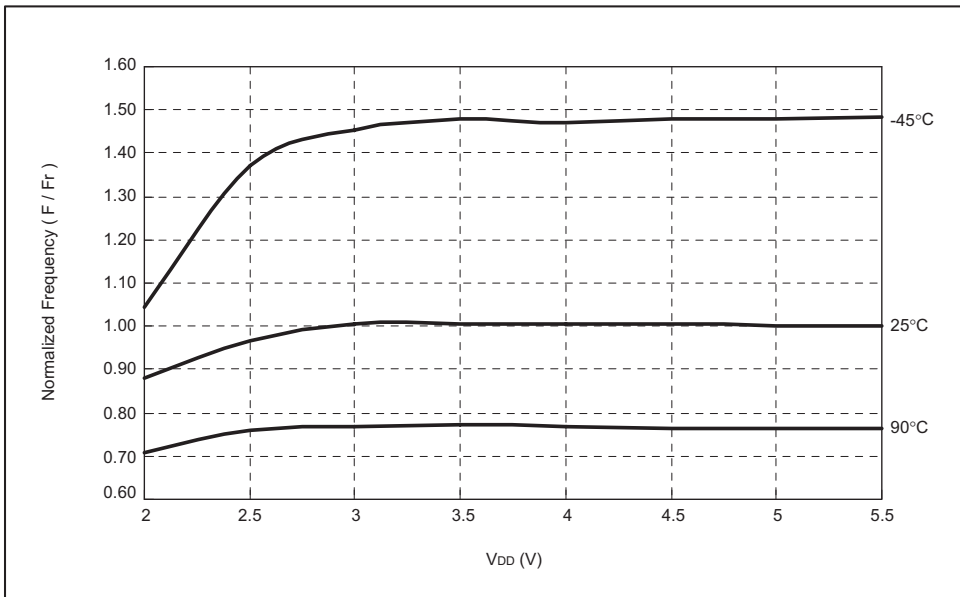
12MHZ IRC Frequency vs. VDD
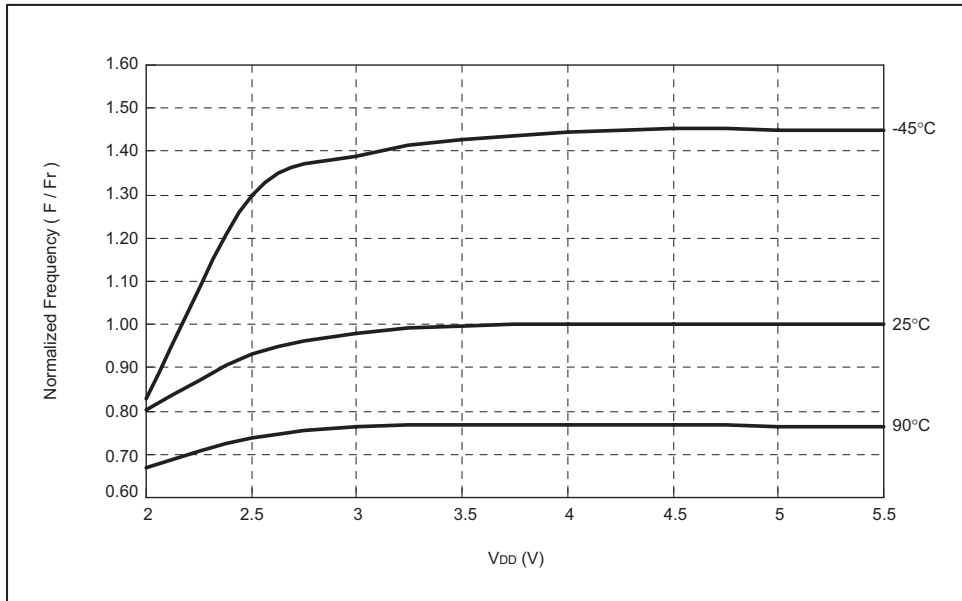
Package Information

10-pin MSOP Outline Dimensions
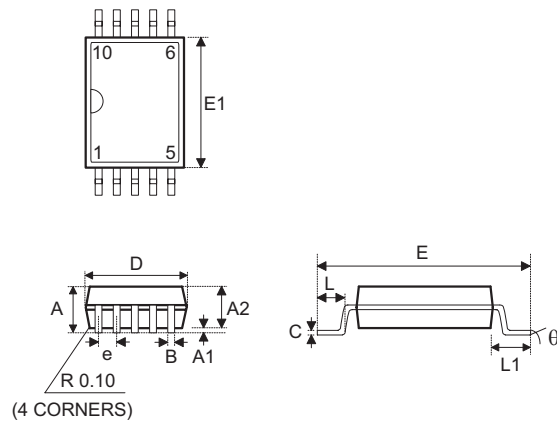
Symbol	Dimensions in mm		
	Min.	**Nom.**	**Max.**
A	—	—	1.1
A1	0	—	0.15
A2	0.75	—	0.95
B	0.17	—	0.27
C	—	—	0.25
D	—	3	—
E	—	4.9	—
E1	—	3	—
e	—	0.5	—
L	0.4	—	0.8
L1	—	0.95	—
θ	0°	—	8°

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

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

Holtek Semiconductor Inc. (Shanghai Sales Office)
G Room, 3 Floor, No.1 Building, No.2016 Yi-Shan Road, Minhang District, Shanghai, China 201103
Tel: 86-21-5422-4590
Fax: 86-21-5422-4705
http://www.holtek.com.cn

Holtek Semiconductor Inc. (Shenzhen Sales Office)
5F, Unit A, Productivity Building, Gaoxin M 2nd, Middle Zone Of High-Tech Industrial Park, ShenZhen, China 518057
Tel: 86-755-8616-9908, 86-755-8616-9308
Fax: 86-755-8616-9722

Holtek Semiconductor Inc. (Beijing Sales Office)
Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031
Tel: 86-10-6641-0030, 86-10-6641-7751, 86-10-6641-7752
Fax: 86-10-6641-0125

Holtek Semiconductor Inc. (Chengdu Sales Office)
709, Building 3, Champagne Plaza, No.97 Dongda Street, Chengdu, Sichuan, China 610016
Tel: 86-28-6653-6590
Fax: 86-28-6653-6591

Holtek Semiconductor (USA), Inc. (North America Sales Office)
46729 Fremont Blvd., Fremont, CA 94538
Tel: 1-510-252-9880
Fax: 1-510-252-9885
http://www.holtek.com