Touch I/O Flash MCU

BS83A02A-4/BS83A04A-3/BS83A04A-4

Revision: V1.72 Date: November 11, 2021

www.holtek.com

Table of Contents

Features 5
CPU Features 5
Peripheral Features 5
Development Tools 5
General Description 6
Selection Table 6
Block Diagram 7
Pin Assignment 7
Pin Description 8
Absolute Maximum Ratings 9
D.C. Characteristics 10
A.C. Characteristics 11
Power-on Reset Characteristics 11
System Architecture 12
Clocking and Pipelining 12
Program Counter – PC 13
Stack 14
Arithmetic and Logic Unit – ALU 14
Flash Progam Memory 15
Structure 15
Special Vectors 15
Look-up Table 15
Table Program Example 16
In Circuit Programming 17
On-chip Debug Support – OCDS 18
RAM Data Memory 19
Structure 19
Special Function Register 21
Indirect Addressing Registers – IAR0, IAR1 21
Memory Pointers – MP0, MP1 21
Accumulator – ACC 22
Program Counter Low Register – PCL 22
Look-up Table Registers – TBLP, TBHP, TBLH 22
Status Register – STATUS 22
Systen Control Register – CTRL 24
Oscillators 25
Oscillator Overview 25
System Clock Configuratios 25
Internal RC Oscillator – HIRC 25
Internal 32kHZ Oscillator – LIRC 25

Operating Modes and System Clocks 26
System Clocks 26
Control Register 27
System Operation Modes 28
Operating Mode Switching 29
Standby Current Considerations 32
Wake-up 33
Programming Considerations 33
Watchdog Timer 34
Watchdog Timer Clock Source 34
Watchdog Timer Control Register 34
Watchdog Timer Operation 35
Reset and Initialisation 36
Reset Functions 36
Reset Initial Conditions 38
Input/Output Ports 40
Pull-high Resistors 40
Port A Wake-up 41
I/O Port Control Registers 41
I/O Pin Structures 42
Programming Considerations 42
Timer/Event Counter 43
Configuring the Timer/Event Counter Input Clock Source 43
Timer Register – TMR 43
Timer Control Register – TMRC 44
Timer/Event Counter Operation 44
Prescaler 45
Programming Consideration 45
Touch Key Function 46
Touch Key Structure 46
Touch Key Register Definition 46
Touch Key Operation 51
Touch Key Interrupt 53
Programming Considerations 53
Interrupts 54
Interrupt Registers 54
Interrupt Operation 56
External Interrupt 57
Touch Key Interrupt 57
Timer/Event Counter Interrupt 57
Time Base Interrupt 57
Interrupt Wake-up Function 58
Programming Considerations 59

Application Circuits **60**
Instruction Set **62**
Introduction 62
Instruction Timing 62
Moving and Transferring Data 62
Arithmetic Operations 62
Logical and Rotate Operation 63
Branches and Control Transfer 63
Bit Operations 63
Table Read Operations 63
Other Operations 63
Instruction Set Summary **64**
Table Conventions 64
Instruction Definition **66**
Package Information **75**
6-pin DFN (2mm×2mm) Outline Dimensions 76
6-pn SOT23-6 Outline Dimensions 77
8-pin SOP (150mil) Outline Dimensions 78
10-pin MSOP Outline Dimensions 79
16-pin NSOP (150mil) Outline Dimensions 80

Features

CPU Features

- Operating voltage:
  - f_{SYS} = 8MHz: 2.7V~5.5V (BS83A04A-3)
  - f_{SYS} = 8MHz: 2.2V~5.5V (BS83A02A-4/BS83A04A-4)
- Up to 0.5μs instruction cycle with 8MHz system clock at V_{DD} =5V
- Power down and wake-up functions to reduce power consumption
- Fully integrated low and high speed internal oscillators
  - Low Speed -- 32kHz
  - High speed -- 8MHz
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- Fully integrated internal 8MHz oscillator requires no external components
- All instructions executed in one or two instruction cycles
- Table read instruction
- 61 powerful instructions
- 4-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 1K×16
- RAM Data Memory: 96×8
- Watchdog Timer function
- Up to 8 bidirectional I/O lines
- One external interrupt pin shared with I/O pin
- Single 8-bit programmable Timer/Event Counter
- Single Time-Base function for generation of fixed time interrupt signals
- Low voltage reset function
- Up to 4 touch key functions
- Package types: 6-pin DFN, SOT23-6, 8-pin SOP and 10-pin MSOP

Development Tools

For rapid product development and to simplify device parameter setting, Holtek has provided relevant development tools which users can download from the following link:

https://www.holtek.com/esk-bs-210 (BS83A02A-4/A04A-3 only)

General Description

This series of devices are Flash Memory type 8-bit high performance RISC architecture microcontrollers with fully integrated touch key function. With the touch key function provided internally and with the convenience of Flash Memory multi-programming features, these devices have all the features to offer designers a reliable and easy means of implementing Touch Keyes within their products applications.

The touch key function is integrated completely eliminating the need for external components. In addition to the flash program memory, other memory includes an area of RAM Data Memory. Protective features such as an internal Watchdog Timer and Low Voltage Reset functions coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

The devices include fully integrated low and high speed oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption. Easy communication with the outside world is provided using the inclusion of flexible I/O programming features, Timer/Event Counters and many other features further enhance device functionality and flexibility.

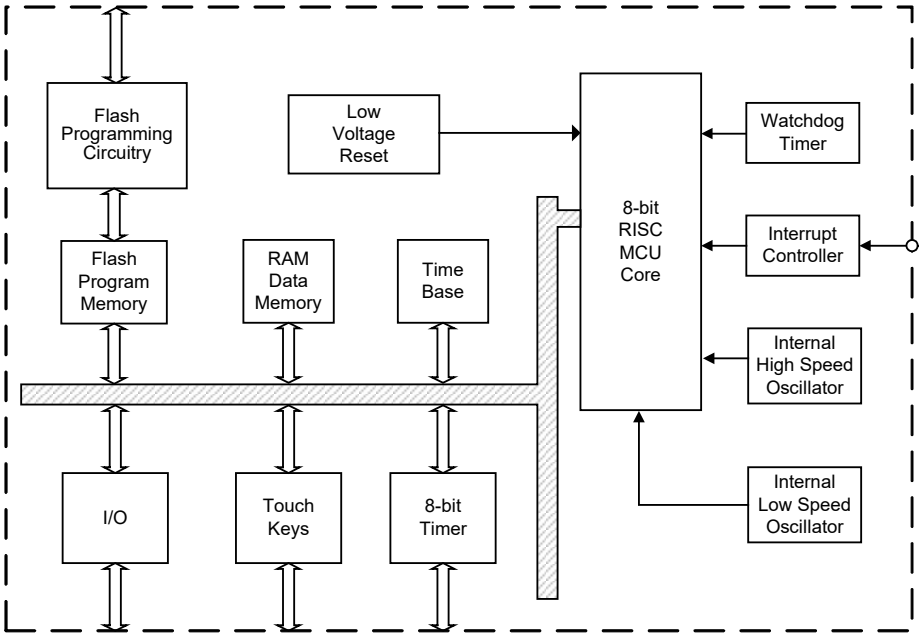
These touch key devices will find excellent use in a huge range of modern Touch Key product applications such as instrumentation, household appliances, electronically controlled tools to name but a few.

Selection Table

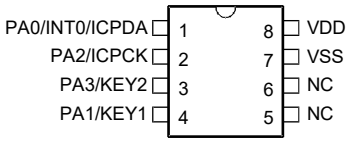
Most features are common to all devices, the main distinguishing feature is the operating voltage range, I/O count, Touch Key count, LVR value and package type. The following table summarises the main features of each device.

<table>
<tr><th>Part No.</th><th>Internal Clock</th><th>V_{DD}</th><th>System Clock</th><th>Program Memory</th><th>Data Memory</th><th>I/O</th><th>8-bit Timer</th><th>Touch Key</th><th>LVR</th><th>Stack</th><th>Package</th><th>Marking</th></tr>
<tr><td>BS83A02A-4</td><td rowspan="3">8MHz</td><td>2.2V~
5.5V</td><td rowspan="3">8MHz</td><td rowspan="3">1K×16</td><td rowspan="3">96×8</td><td>4</td><td rowspan="3">1</td><td>2</td><td>2.10V</td><td rowspan="3">4</td><td>6DFN
SOT23-6
8SOP</td><td>BS83A02A-4
A2A4 (for 6DFN)
02A-4 (for SOT23-6)
BS83A02A-4 (for 8SOP)</td></tr>
<tr><td>BS83A04A-3</td><td>2.7V~
5.5V</td><td rowspan="2">8</td><td rowspan="2">4</td><td>2.55V</td><td rowspan="2">8SOP
10MSOP</td><td>BS83A04A-3
83A04A-3 (for 8SOP)
8304A-3 (for 10MSOP)</td></tr>
<tr><td>BS83A04A-4</td><td>2.2V~
5.5V</td><td>2.10V</td><td>BS83A04A-4
83A04A-4 (for 8SOP)
8304A-4 (for 10MSOP)</td></tr>
</table>

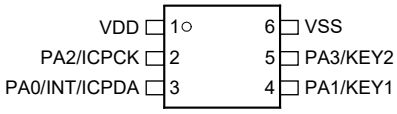
Block Diagram



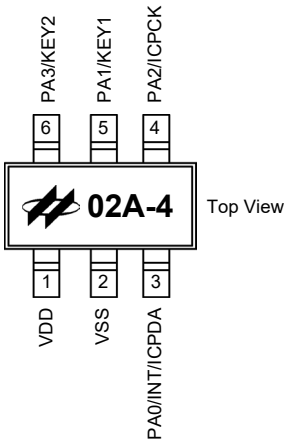
Pin Assignment



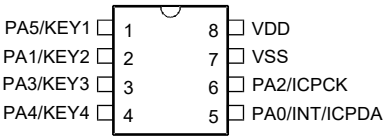
BS83A02A-4
8 SOP-A



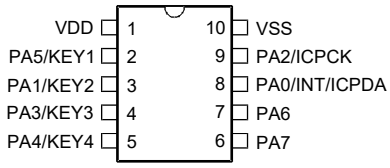
BS83A02A-4
6 DFN-A



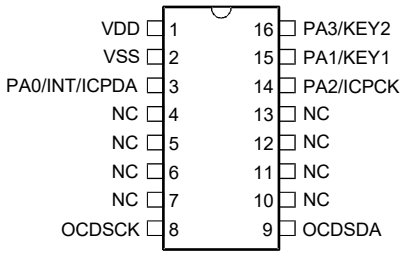
SOT23-6



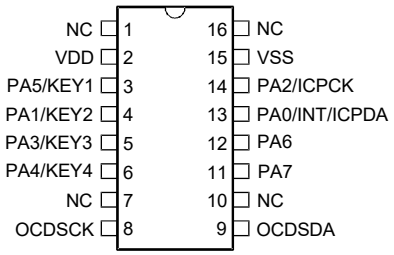
BS83A04A-3/BS83A04A-4
8 SOP-A



BS83A04A-3/BS83A04A-4
10 MSOP-A



BS83AV02A
16 NSOP-A



BS83V04A
16 NSOP-A

Note: The 16NSOP package type is only for OCDS EV chips.

Pin Description

The function of each pin is listed in the following table, however the details of each pin is contained in other sections of the datasheet.

As the Pin Description table shows the situation for the package with the most pins, not all pins in the table will be available on smaller package sizes.

BS83A02A-4

Pin Name	Function	OPT	I/T	O/T	Description
PA0/INT/ ICPDA	PA0	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	INTEG	ST	—	External interrupt
	ICPDA	—	ST	CMOS	In-circuit programming data/address pin
PA1/KEY1	PA1	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	KEY1	TKM0C1	NSI	—	Touch key input
PA2/ICPCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	ICPCK	—	ST	—	In-circuit programming clock pin
PA3/KEY2	PA3	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	KEY2	TKM0C1	NSI	—	Touch key input
OCDSCK	OCDSCK	—	ST	—	On-chip debug clock pin, for EV chip only
OCDSDA	OCDSDA	—	ST	CMOS	On-chip debug data/address pin, for EV chip only
VDD	VDD	—	PWR	—	Power voltage
VSS	VSS	—	PWR	—	Ground

Legend: I/T: Input type; O/T: Output type
OPT: Optional by register selection
PWR: Power; ST: Schmitt Trigger input
CMOS: CMOS output
NSI: Non-standard input

BS83A04A-3/BS83A04A-4

Pin Name	Function	OPT	I/T	O/T	Description
PA0/INT/ ICPDA	PA0	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	INT	INTEG	ST	—	External interrupt
	ICPDA	—	ST	CMOS	In-circuit programming data/address pin
PA1/KEY2	PA1	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	KEY2	TKM0C1	NSI	—	Touch key input
PA2/ICPCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	ICPCK	—	ST	—	In-circuit programming clock pin
PA3/KEY3	PA3	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	KEY3	TKM0C1	NSI	—	Touch key input
PA4/KEY4	PA4	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	KEY4	TKM0C1	NSI	—	Touch key input
PA5/KEY1	PA5	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
	KEY1	TKM0C1	NSI	—	Touch key input
PA6	PA6	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
PA7	PA7	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up.
OCDSCK	OCDSCK	—	ST	—	On-chip debug clock pin, for EV chip only
OCDSDA	OCDSDA	—	ST	CMOS	On-chip debug data/address pin, for EV chip only
VDD	VDD	—	PWR	—	Power voltage
VSS	VSS	—	PWR	—	Ground

Legend: I/T: Input type; O/T: Output type
OPT: Optional by register selection
PWR: Power; ST: Schmitt Trigger input
CMOS: CMOS output
NSI: Non-standard input
For the 8-pin package type, the I/O pin PA7 is not bonded to an external pin but internally bonded together with PA4, so it is recommended that the PA7 should be configured as input with pull-high resistor disabled.

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}−0.3V to V_{SS}+6.0V
Input Voltage ..... V_{SS}−0.3V to V_{DD}+0.3V
Storage Temperature ..... -50˚C to 125˚C
Operating Temperature ..... -40˚C to 85˚C
I_{OL} Total 80mA
I_{OH} Total ..... -80mA
Total Power Dissipation 500mW

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

D.C. Characteristics

Ta = 25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage (HIRC) (BS83A04A-3)	—	f_{SYS} = 8MHz	2.7	—	5.5	V
	Operating Voltage (HIRC) (BS83A02A-4/BS83A04A-4)	—	f_{SYS} = 8MHz	2.2	—	5.5	V
I_{DD}	Operating Current (HIRC) (f_{SYS}=f_H, f_S=f_{SUB}= f_{LIRC})	3V	No load, f_H = 8MHz, WDT enable	—	0.8	1.5	mA
		5V		—	1.5	3.0	mA
	Operating Current (LIRC) (f_{SYS}=f_L=f_{LIRC}, f_S=f_{SUB}= f_{LIRC})	3V	No load, WDT enable, LVR disable	—	10	20	μA
		5V		—	20	35	μA
I_{STB}	IDLE Mode Stanby Current (HIRC) (f_{SYS}=off, f_S=f_{SUB}=f_{LIRC})	3V	No load, WDT enable, f_{SYS} = 8MHz	—	1.5	3.0	μA
		5V		—	2.5	5.0	μA
	IDLE Mode Stanby Current (LIRC) (f_{SYS}=off, f_S=f_{SUB}=f_{LIRC})	3V	No load, WDT enable, f_{SYS} = 32KHz	—	1.5	3.0	μA
		5V		—	2.5	5.0	μA
V_{IL}	Input Low Voltage for I/O Ports or Input Pins	5V	—	0	—	1.5	V
		—		0	—	0.2V_{DD}	V
V_{IH}	Input High Voltage for I/O Ports or Input Pins	5V	—	3.5	—	5.0	V
		—		0.8V_{DD}	—	V_{DD}	V
V_{LVR}	Low Voltage Reset Voltage (BS83A04A-3)	—	LVR enable, V_{LVR} =2.55V	-5%	2.55	+5%	V
	Low Voltage Reset Voltage (BS83A02A-4/BS83A04A-4)	—	LVR enable, V_{LVR} =2.10V	-5%	2.10	+5%	V
I_{LVR}	Additional Power Consumption if LVR is used	3V	LVR disble→LVR enable	—	15	25	μA
		5V		—	20	30	μA
I_{OL}	I/O Ports Sink Current (BS83A02A-4)	3V	V_{OL}=0.1V_{DD}	8	16	—	mA
		5V	V_{OL}=0.1V_{DD}	16	32	—	mA
	I/O Ports Sink Current (BS83A04A-3/BS83A04A-4)	3V	V_{OL}=0.1V_{DD}	4	8	—	mA
		5V	V_{OL}=0.1V_{DD}	10	20	—	mA
I_{OH}	I/O Ports Source Current (BS83A02A-4)	3V	V_{OH} = 0.9V_{DD}	-3.75	-7.5	—	mA
		5V	V_{OH} = 0.9V_{DD}	-7.5	-15	—	mA
	I/O Ports Source Current (BS83A04A-3/BS83A04A-4)	3V	V_{OH} = 0.9V_{DD}	-2	-4	—	mA
		5V	V_{OH} = 0.9V_{DD}	-5	-10	—	mA
R_{PH}	Pull-high Resistance for I/O Ports	3V	—	20	60	100	kΩ
		5V	—	10	30	50	kΩ

A.C. Characteristics

Ta=25°C

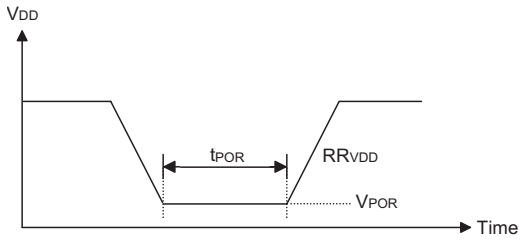
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	Operating Clock (BS83A04A-3)	—	2.7V ~ 5.5V	DC	—	8	MHz
	Operating Clock (BS83A02A-4/BS83A04A-4)	—	2.2V ~ 5.5V	DC	—	8	MHz
f_{HIRC}	System Clock (HIRC)	3V/5V	Ta=25°C	-2%	8	+2%	MHz
f_{LIRC}	System clock (LIRC)	5V	—	-10%	32	+10%	kHz
		2.2V~5.5V	Ta=-40°C ~85°C	-50%	32	+60%	kHz
t_{INT}	Interrupt pulse width	—	—	10	—	—	μs
t_{LVR}	Low Voltage Width to Reset	—	—	120	240	480	μs
t_{SST}	System start-up timer period (wake-up from HALT)	—	f_{SYS}=HIRC	—	1024	1025	t_{SYS}
			f_{SYS}=LIRC	—	1~2	3	

Note: 1. $t_{SYS}= 1/f_{SYS}$
2. To maintain the accuracy of the internal HIRC oscillator frequency, a 0.1μF decoupling capacitor should be connected between VDD and VSS and located as close to the device as possible.

Power-on Reset Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{POR}	V_{DD} Start Voltage	—	—	—	—	100	mV
RR_{VDD}	V_{DD} Raising Rate	—	—	0.035	—	—	V/ms
t_{POR}	Minimum Time for V_{DD} Stays at V_{POR} to Ensure Power-on Reset	—	—	1	—	—	ms

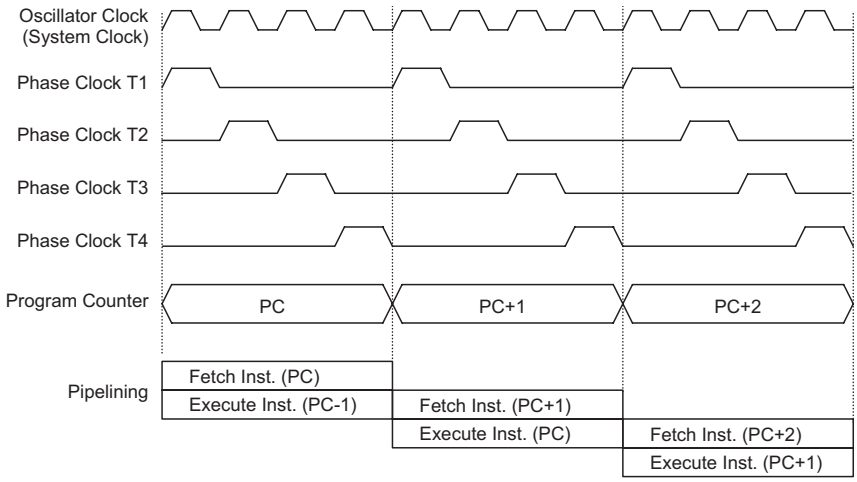


System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their 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 instruction set operations, which 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 with maximum reliability and flexibility. This makes the device suitable for low-cost, high-volume production for controller applications.

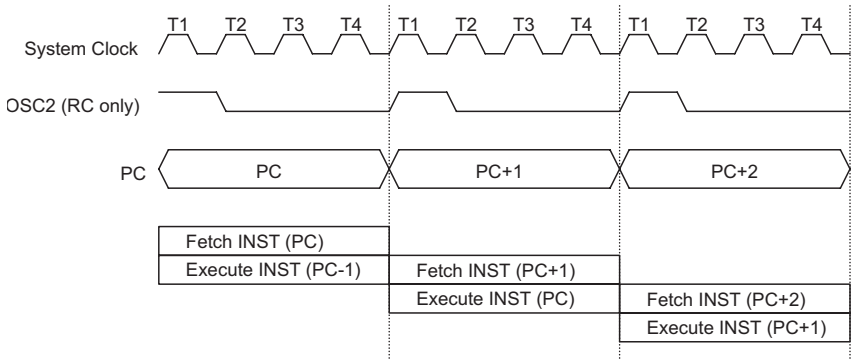
Clocking and Pipelining

The mian system clock, derived from HIRC or LIRC 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.



System Clocking and Pipelining

For instructions involving branches, such as jump or call instructions, two instruction cycles are required to complete instruction execution. An extra cycle is required as the program takes one cycle to firstly 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.



Instruction Fetching

Program Counter – PC

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. It must be noted that only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by user.

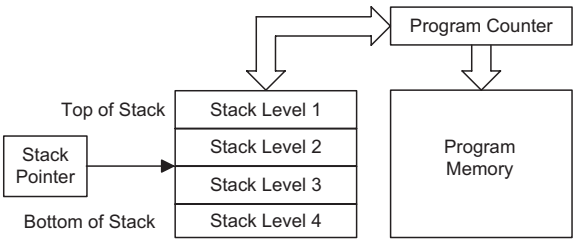
When executing instructions requiring jumping to non-consecutive addresses such as a jump instruction, a subroutine call, interrupt or reset, etc, the microcontroller manages program control by loading the required address into the Program Counter. For conditional skip instructions, once the condition 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.

Program Counter	
High Byte of Program	**Low Byte of Program**
PC9~ PC8	PCL7~ PCL0

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 in the present page of memory, which have 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.

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 4 levels and 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, 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. If the stack is overflow, the first Program Counter save in the stack will be lost.

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:

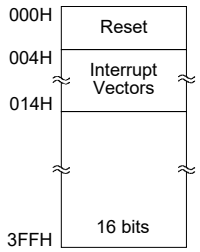
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation: RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement: INCA, INC, DECA, DEC
- Branch decision: JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI.

Flash Progam Memory

The Program Memory is the location where the user code or program is stored. For this device the Programm Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate programming tools, this Flash device offers users the flexibility to debug and develop their applications while also offering a means of field programming and updating.

Structure

The Program Memory has a capacity of 1K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries information. Table data, which can be setup in any location within the Program Memory, is addressed by separate table pointer register.



Program Memory Structure

Special Vectors

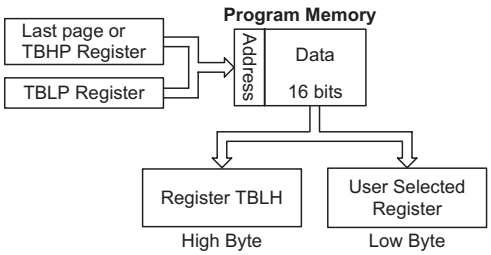
Within the Program Memory, certain locations are reerved for the reset and interrupts. The location 000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

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 address of the look up data to be retrieved in the table pointer registers, TBLP and TBHP. These registers define the total address of the look-up table.

After setting up the table pointer, the table data can be retrieved from the Program Memory using the "TABRD [m]" or "TABRDL [m]" instructions, respectively. When the instruction is 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".

The accompanying diagram illustrates the addressing data flow of the look-up table.



Instruction	Table Location Bits									
	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRD [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

b9~b0: Table address location bits

Table Program Example

The accompanying example shows how the table pointer and table data is defined and retrieved from the devices. 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 microcontroller.

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 last page if the "TABRDL [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.

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.

Table Read Program Example

```
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
         :
         :
tabrdl   tempreg1      ; transfers value in table referenced by table pointer to tempreg1
                       ; data at program 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 program 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
         :
         :
```

In Circuit Programming

The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or upgrading the program at a later stage. This enables product manufacturers to easily keep their manufactured products sup plied with the latest program releases without removal and re-insertion of the device.

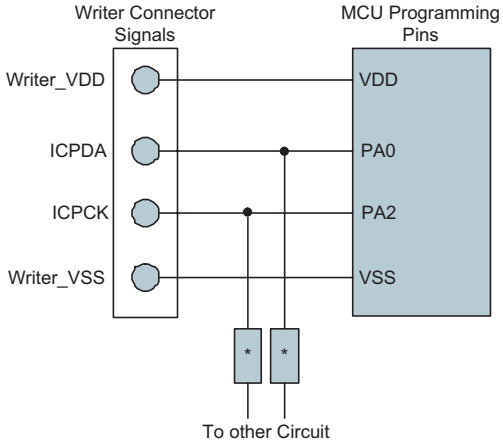
The Holtek Flash MCU to Writer Programming Pin correspondence table is as follows:

Holtek WritePins	MCU Programming Pins	Function
ICPDA	PA0	Serial Address and data – read/write
ICPCK	PA2	Programming Serial Clock
VDD	VDD	Power Supply (5.0V)
VSS	VSS	Ground

During the programming process, the user must there take care to ensure that no other outputs are connected to these two pins.

The Program Memory can be programmed serially in circuit using this 4-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Two additional lines are required for the power supply and one line for the reset. The technical details regarding the incircuit programming of the device are beyond the scope of this document and will be supplied in supplementary literature.

During the programming process the microcontroller takes control of the PA0 and PA2 I/O pins for data and clock programming purposes. The user must there take care to ensure that no other outputs are connected to these two pins.



Note: * may be resistor or capacitor. The resistance of * must be greater than 1k or the capacitance of * must be less than 1nF.

On-chip Debug Support – OCDS

An EV chip exists for the purposes of device emulation. This EV chip device also provides an "On-Chip Debug" function to debug the device during the development process. The EV chip and the actual MCU devices are almost functionally compatible except for the "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the actual MCU device will have no effect in the EV chip. For a more detailed OCDS description, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

Holtek e-Link Pins	EV Chip Pins	Pin Description
OCDSDA	OCDSDA	On-chip Debug Support Data/Address input/output
OCDSCK	OCDSCK	On-chip Debug Support Clock input
VDD	VDD	Power Supply
GND	VSS	Ground

RAM Data Memory

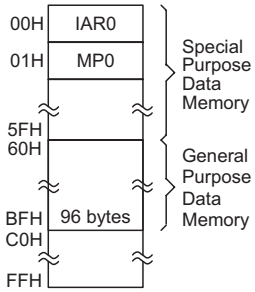
The Data Memory is an 8-bit wide RAM internal memory and is the location where temporary information is stored.

Structure

Divided into two sections, the first of these is an area of RAM, known as the Special Function Data Memory. Here are located registers which 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 known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

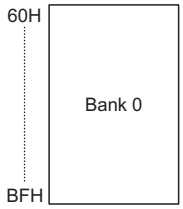
The start address of the overall Data Memory is the address 00H.



Data Memory

00H	IAR0	30H	Unused
01H	MP0	31H	Unused
02H	IAR1	32H	Unused
03H	MP1	33H	Unused
04H	Unused	34H	Unused
05H	ACC	35H	Unused
06H	PCL	36H	Unused
07H	TBLP	37H	Unused
08H	TBLH	38H	Unused
09H	TBHP	39H	Unused
0AH	STATUS	3AH	Unused
0BH	SMOD	3BH	Unused
0CH	CTRL	3CH	Unused
0DH	INTEG	3DH	Unused
0EH	INTC0	3EH	Unused
0FH	INTC1	3FH	Unused
10H	Unused	40H	Unused
11H	Unused	41H	Unused
12H	Unused	42H	Unused
13H	LVRC	43H	TKTMR
14H	PA	44H	TKC0
15H	PAC	45H	TK16DL
16H	PAPU	46H	TK16DH
17H	PAWU	47H	TKC1
18H	Unused	48H	TKM016DL
19H	Unused	49H	TKM016DH
1AH	WDTC	4AH	TKM0ROL
1BH	TBC	4BH	TKM0ROH
1CH	TMR	4CH	TKM0C0
1DH	TMRC	4DH	TKM0C1
1EH	Unused	4EH	Unused
1FH	Unused	4FH	Unused
20H	Unused	50H	Unused
21H	Unused	51H	Unused
22H	Unused	52H	Unused
23H	Unused	53H	Unused
24H	Unused	54H	Unused
25H	Unused	55H	Unused
26H	Unused	56H	Unused
27H	Unused	57H	Unused
28H	Unused	58H	Unused
29H	Unused	59H	Unused
2AH	Unused	5AH	Unused
2BH	Unused	5BH	Unused
2CH	Unused	5CH	Unused
2DH	Unused	5DH	Unused
2EH	Unused	5EH	Unused
2FH	Unused	5FH	Unused

Special Purpose Data Memory



General Purpose Data Memory

Special Function Register

Most of the Special Function Register details will be described in the relevant functional section. However several registers require a separate description in this section.

Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation is using 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 Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together access data from Bank 0 while the IAR1 and MP1 register pair can access data from any bank. 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 directly will result in no operation.

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 indirectly address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address which the microcontroller directs to is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Bank 0, while MP1 and IAR1 are used to access data from all banks. Direct Addressing can only be used with Bank 0, all other Banks must be addressed indirectly using MP1 and IAR1. Note that for these devices, the Memory Pointers, MP0 and MP1, are both 8-bit registers and used to access the Data Memory together with their corresponding indirect addressing registers IAR0 and IAR1.

The following example shows how to clear a section of four Data Memory locations already defined as locations adres1 to adres4.

Indirect Addressing Program Example

```
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, TBHP, TBLH

These three special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP and TBHP are the table pointers and indicate the location where the table data is located. Their value must be setup before any table read commands are executed. Their 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 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.

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 subroutine can corrupt the status register, precautions must be taken to correctly save it. Note that bits 3~0 of the STATUS register are both readable and writeable bits.

STATUS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TO	PDF	OV	Z	AC	C
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	×	×	×	×

"×" unknown

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TO**: Watchdog Time-Out flag
0: After power up or executing the "CLR WDT"or "HALT" instruction
1: A watchdog time-out occurred.

Bit 4 **PDF**: Power down flag
0: After power up or executing the "CLR WDT" instruction
1: By executing the "HALT" instruction

Bit 3 **OV**: Overflow flag
0: No overflow
1: An operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa.

Bit 2 **Z**: Zero flag
0: The result of an arithmetic or logical operation is not zero
1: The result of an arithmetic or logical operation is zero

Bit 1 **AC**: Auxiliary flag
0: No auxiliary carry
1: 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

Bit 0 **C**: Carry flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation
C is also affected by a rotate through carry instruction.

Systen Control Register – CTRL

CTRL Register

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	—	LVRF	LRF	WRF
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	×	0	0

"×" unknown

Bit 7 **FSYSON**: f_{SYS} control in IDLE mode
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2 **LVRF**: LVR function reset flag
0: Not active
1: Active
This bit can be cleared to "0", but can not be set to "1".

Bit 1 **LRF**: LVR Control register software reset flag
0: Not active
1: Active
This bit can be cleared to "0", but can not be set to "1".

Bit 0 **WRF**: reset caused by WE[4:0] setting
0: Not active
1: Active
This bit can be cleared to "0", but can not be set to "1".

Oscillators

Various oscillator options offer the user a wide range of functions according to their various application requirements. The flexible features of the oscillator functions ensure that the best optimisation can be achieved in terms of speed and power saving.

Oscillator Overview

The devices include two internal oscillators, a low speed oscillator and a high speed oscillator. Both can be chosen as the clock source for the main system clock however the slow speed oscillator is also used as a clock source for other functions such as the Watchdog Timer, Time Base and Timer/Event Counter. Both oscillators require no external components for their implementation. All oscillator options are selected using registers. The high speed oscillator provides higher performance but carries with it the disadvantage of higher power requirements, while the opposite is of course true for the low speed oscillator. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimise the performance/power ratio, a feature especially important in power sensitive portable applications.

Type	Name	Freq.
Internal High Speed RC	HIRC	8 MHz
Internal Low Speed RC	LIRC	32 kHz

Oscillator Types

System Clock Configuratios

There are two system oscillators, one high speed oscillator and one low speed oscillator. The high speed oscillator is a fully internal 8MHz RC oscillator. The low speed oscillator is a fully internal 32kHz RC oscillator. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK bit and CKS2~CKS0 bits in the SMOD register and as the system clock can be dynamically selected.

Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 8MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuit is used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

Internal 32kHZ Oscillator – LIRC

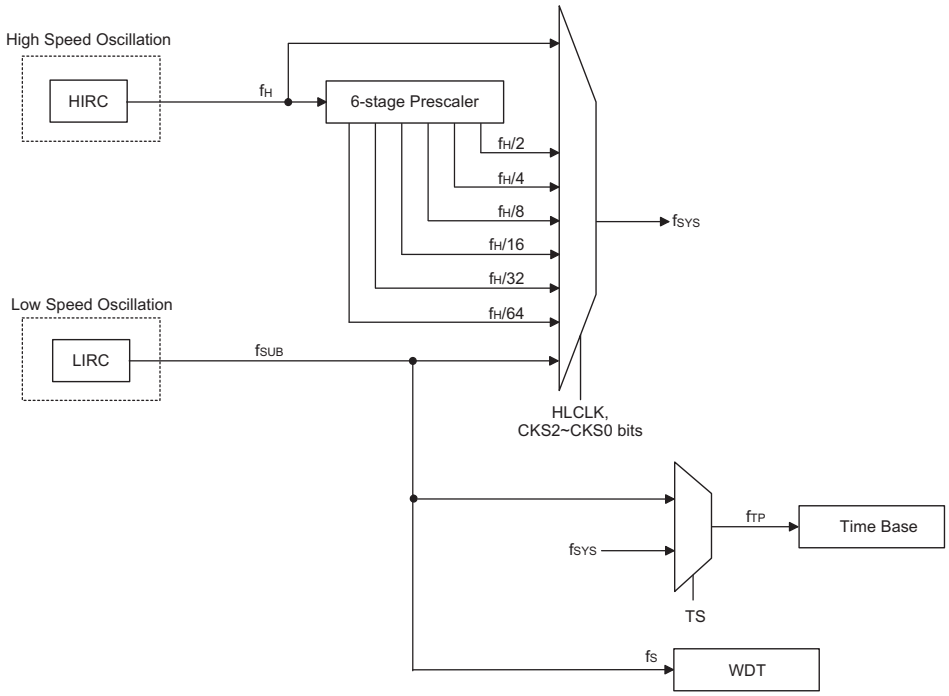
The LIRC is a fully self-contained free running on-chip RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. After power on this LIRC oscillator will be permanently enabled; there is no provision to disable the oscillator using register bits.

Operating Modes and System Clocks

Present day applications require that their microcontrollers have high performance but often still demand that they consume as little power as possible, conflicting requirements that are especially true in battery powered portable applications. The fast clocks required for high performance will by their nature increase current consumption and of course, versa, lower speed clocks reduce current consumption. As Holtek has provided these devices with both high and low speed clock sources and the means to switch between them dynamically, the user can optimise the operation of their microcontroller to achieve the best performance/power ratio.

System Clocks

The main system clock, can come from a high frequency f_H or low frequency f_{SUB} source, and is selected using the HLCLK bit and CKS2~CKS0 bits in the SMOD register. Both the high and low speed system clocks are sourced from internal RC oscillators.



System Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_{SUB} from f_H, the high speed oscillation will stop to conserve the power. Thus there is no f_H~f_H/64 clock source for use by the peripheral circuits.

Control Register

A single register, SMOD, is used for overall control of the internal clocks within the devices.

SMOD Register

Bit	7	6	5	4	3	2	1	0
Name	CKS2	CKS1	CKS0	—	LTO	HTO	IDLEN	HLCLK
R/W	R/W	R/W	R/W	—	R	R	R/W	R/W
POR	0	0	0	—	0	0	1	1

Bit 7~5 **CKS2 ~ CKS0**: The system clock selection when HLCLK is "0"
000: f_{SUB} (f_{LIRC})
001: f_{SUB} (f_{LIRC})
010: $f_H/64$
011: $f_H/32$
100: $f_H/16$
101: $f_H/8$
110: $f_H/4$
111: $f_H/2$

These three bits are used to select which clock is used as the system clock source. In addition to the system clock source, which can be LIRC, a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 Unimplemented, read as "0"

Bit 3 **LTO**: Low speed system oscillator ready flag
0: Not ready
1: Ready

This is the low speed system oscillator ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The flag will be low when in the SLEEP Mode but after a wake-up has occurred, the flag will change to a high level after 1~2 clock cycles if the LIRC oscillator is used.

Bit 2 **HTO**: High speed system oscillator ready flag
0: Not ready
1: Ready

This is the high speed system oscillator ready flag which indicates when the high speed system oscillator is stable after a wake-up has occurred. The flag will be low when in the SLEEP or IDLE0 Mode but after power on reset or a wake-up has occurred, the flag will change to a high level after 1024 clock cycles if the HIRC oscillator is used.

Bit 1 **IDLEN**: IDLE Mode Control
0: Disable
1: Enable

This is the IDLE Mode Control bit and determines what happens when the HALT instruction is executed. If this bit is high, when a HALT instruction is executed the device will enter the IDLE Mode. In the IDLE1 Mode the CPU will stop running but the system clock will continue to keep the peripheral functions operational, if FSYSON bit is high. If FSYSON bit is low, the CPU and the system clock will all stop in IDLE0 mode. If the bit is low the device will enter the SLEEP Mode when a HALT instruction is executed.

Bit 0 **HLCLK**: System Clock Selection
0: $f_H/2$ ~ $f_H/64$ or f_{SUB}
1: f_H

This bit is used to select if the f_H clock or the $f_H/2$ ~ $f_H/64$ or f_{SUB} clock is used as the system clock. When the bit is high the f_H clock will be selected and if low the $f_H/2$ ~ $f_H/64$ or f_{SUB} clock will be selected. When system clock switches from the f_H clock to the f_{SUB} clock and the f_H clock will be automatically switched off to conserve power.

System Operation Modes

There are five different modes of operation for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the NORMAL Mode and SLOW Mode. The remaining three modes, the SLEEP, IDLE0 and IDLE1 Mode are used when the microcontroller CPU is switched off to conserve power.

Operation Mode	Description			
	CPU	f_{SYS}	f_{SUB}	f_S
NORMAL Mode	On	f_H~f_H/64	On	On
SLOW Mode	On	f_{SUB}	On	On
IDLE0 Mode	Off	Off	On	On
IDLE1 Mode	Off	On	On	On
SLEEP Mode	Off	Off	On	On

NORMAL Mode

As the name suggests this is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. This mode allows the microcontroller to operate normally with a clock source that will come from the HIRC oscillator. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 and HLCLK bits in the SMOD register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be sourced from the low speed oscillator, the LIRC. Running the microcontroller in this mode allows it to run with much lower operating currents. In the SLOW Mode, f_H is off.

SLEEP Mode

The SLEEP Mode is entered when an HALT instruction is executed and the IDLEN bit in the SMOD register is low. In the SLEEP mode the CPU will be stopped. However the f_{SUB} and f_S clocks will continue to operate because the Watchdog Timer function is always enabled and its clock source is from f_{SUB}.

IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the CTRL register is low. In the IDLE0 Mode the system oscillator will be inhibited from driving the CPU but some peripheral functions will remain operational such as the Watchdog Timer and Timer/Event Counter. In the IDLE0 Mode, the system oscillator will be stopped and the Watchdog Timer clock, f_S, will be still on.

IDLE1 Mode

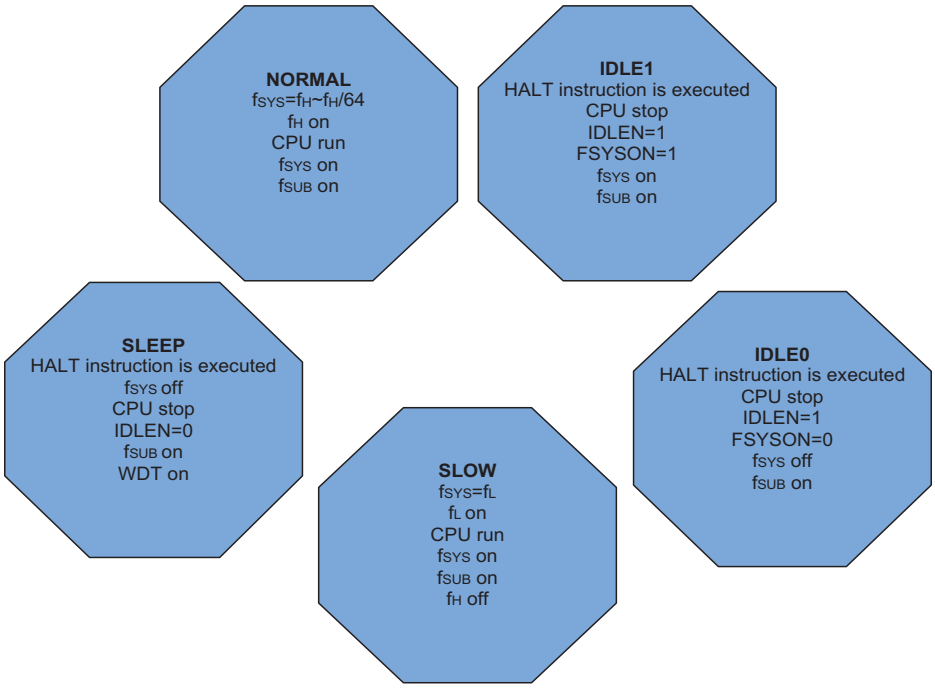
The IDLE1 Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the CTRL register is high. In the IDLE1 Mode the system oscillator will be inhibited from driving the CPU but may continue to provide a clock source to keep some peripheral functions operational such as the Watchdog Timer and Timer/Event Counter. In the IDLE1 Mode, the system oscillator will continue to run, and this system oscillator may be high speed or low speed system oscillator. In the IDLE1 Mode the Watchdog Timer clock, f_S, will be on.

Operating Mode Switching

The devices can switch between operating modes dynamically allowing the user to select the best performance/power ratio for the present task in hand. In this way microcontroller operations that do not require high performance can be executed using slower clocks thus requiring less operating current and prolonging battery life in portable applications.

In simple terms, Mode Switching between the NORMAL Mode and SLOW Mode is executed using the HLCLK bit and CKS2~CKS0 bits in the SMOD register while Mode Switching from the NORMAL/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the IDLEN bit in the SMOD register and FSYSON in the CTRL register.

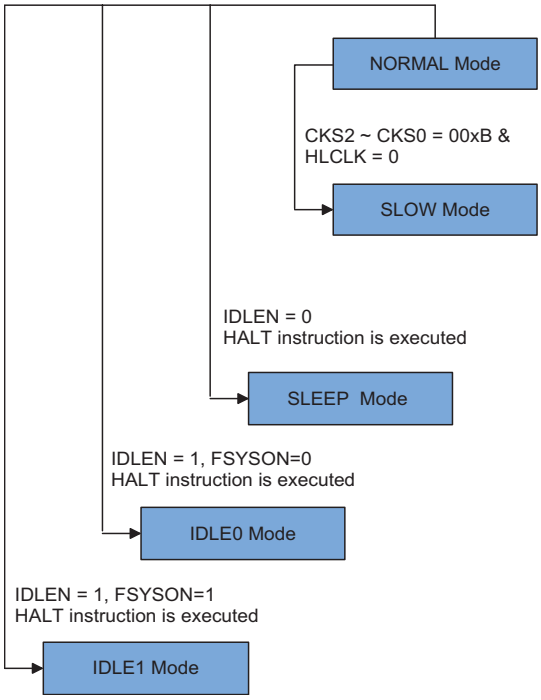
When the HLCLK bit switches to a low level, which implies that clock source is switched from the high speed clock source, f_H, to the clock source, $f_H/2$~$f_H/64$ or f_{SUB}. If the clock is from the f_{SUB}, the high speed clock source will stop running to conserve power. When this happens it must be noted that the $f_H/16$ and $f_H/64$ internal clock sources will also stop running, which may affect the operation of other internal functions such as the Timer/Event Counter. The accompanying flowchart shows what happens when the device moves between the various operating modes.



NORMAL Mode to SLOW Mode Switching

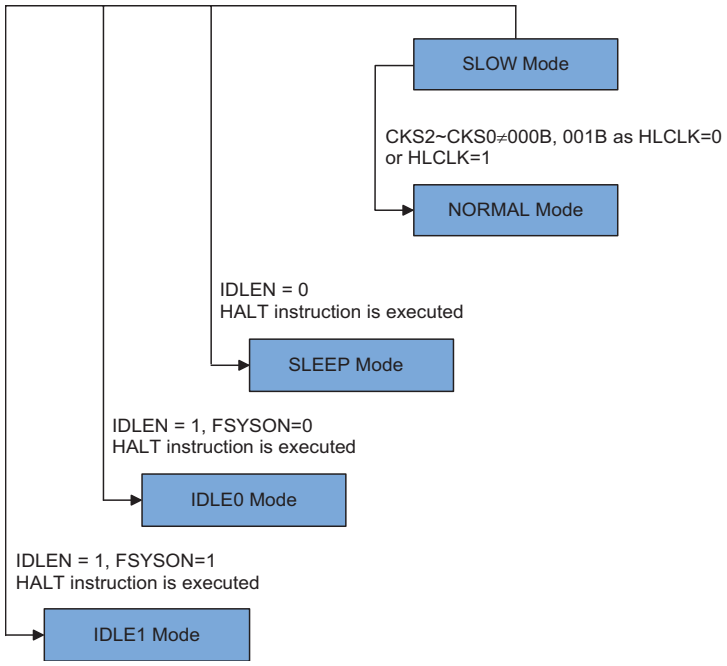
When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the HLCLK bit to "0" and set the CKS2~CKS0 bits to "000" or "001" in the SMOD register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

The SLOW Mode is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs. This is monitored using the LTO bit in the SMOD register.



SLOW Mode to NORMAL Mode Switching

In SLOW Mode the system uses the LIRC low speed system oscillator. To switch back to the NORMAL Mode, where the high speed system oscillator is used, the HLCLK bit should be set to "1" or HLCLK bit is "0", but CKS2~CKS0 is set to "010", "011", "100", "101", "110" or "111". As a certain amount of time will be required for the high frequency clock to stabilise, the status of the HTO bit is checked. The amount of time required for high speed system oscillator stabilization depends upon which high speed system oscillator type is used.



Entering the SLEEP Mode

There is only one way for the devices to enter the SLEEP Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "0" and the WDT is on. When this instruction is executed under the conditions described above, the following will occur:

- The system clock and Time Base clock will be stopped and the application program will stop at the "HALT" instruction, but the WDT will remain with the clock source coming from the f_{SUB} clock.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the IDLE0 Mode

There is only one way for the devices to enter the IDLE0 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register is equal to "1" and the FSYSON bit in CTRL register is equal to "0". When this instruction is executed under the conditions described above, the following will occur:

- The system clock will be stopped and the application program will stop at the "HALT" instruction, but the Time Base clock and f_{SUB} clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Entering the IDLE1 Mode

There is only one way for the devices to enter the IDLE1 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register is equal to "1" and the FSYSON bit in CTRL register is equal to "1". When this instruction is executed under the conditions described above, the following will occur:

- The system clock and Time Base clock and f_{SUB} clock will be on and the application program will stop at the "HALT" instruction
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting.
- The I/O ports will maintain their present conditions.
- 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 SLEEP or IDLE Mode is to keep the current consumption of the devices to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 Mode, 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. This also applies to device which has different package types, as there may be unbonded pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

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. Also note that additional standby current will also be required if using the LIRC oscillator.

In the IDLE1 Mode the system oscillator is on, if the system oscillator is from the high speed system oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

Wake-up

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

- An external falling edge on Port A
- A system interrupt
- A WDT overflow

If the devices are woken up by a WDT overflow, a Watchdog Timer reset will be initiated. Although this wake-up method 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.

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake-up the system. When a Port A 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 high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

System Oscillator	Wake-up Time (SLEEP Mode)	Wake-up Time (IDLE0 Mode)	Wake-up Time (IDLE1 Mode)
HIRC	1024 HIRC cycles		1024 HIRC cycles
LIRC	1~2 LIRC cycles		1~2 LIRC cycles

Wake-Up Times

Programming Considerations

The high speed and low speed oscillators both use the same SST counter. For example, if the system is woken up from the SLEEP Mode the HIRC oscillator needs to start-up from an off state.

If the devices are woken up from the SLEEP Mode to the NORMAL Mode, the high speed system oscillator needs an SST period. The devices will execute the first instruction after HTO is high.

Watchdog Timer

The Watchdog Timer is provided to prevent program malfunctions or sequences from jumping to unknown locations, due to certain uncontrollable external events such as electrical noise.

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock f_{SUB}, which is sourced from the LIRC oscillator. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register. The LIRC internal oscillator has an approximate period frequency of 32KHz at a supply voltage of 5V.

However, it should be noted that this specified internal clock period can vary with V_{DD}, temperature and process variations.

Watchdog Timer Control Register

A single register, WDTC, controls the required time out period.

WDTC Register

Bit	7	6	5	4	3	2	1	0
Name	WE4	WE3	WE2	WE1	WE0	WS2	WS1	WS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	0	1	1

Bit 7~3 **WE4~WE0**: WDT Software control
01010B: enable
10101B: enable (default)
Other values: MCU reset (reset will be active after 2~3 LIRC clock for debounce time).
If the MCU reset is caused by WDTC software reset, the WRF flag in CTRL register will be set after reset.

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $2^8/f_S$
001: $2^{10}/f_S$
010: $2^{12}/f_S$
011: $2^{14}/f_S$
100: $2^{15}/f_S$
101: $2^{16}/f_S$
110: $2^{17}/f_S$
111: $2^{18}/f_S$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

CTRL Register

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	—	LVRF	LRF	WRF
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	×	0	0

"×" unknown

Bit 7,2~1 Decribed in other section

Bit 6~3 Unimplemented, read as "0"

Bit 0 **WRF**: reset caused by WE[4:0] setting
0: Not active
1: Active
This bit can be cleared to "0", but can not be set to "1".

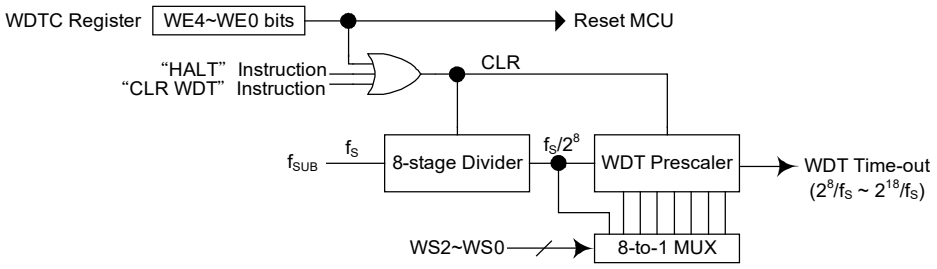
Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instructions.

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 SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDT software reset, which means a certain value is written into the WE4~WE0 bit filed except 01010B and 10101B, the second is using the Watchdog Timer software clear instructions and the third is via a HALT instruction.

To clear the Watchdog Timer is to use the single "CLR WDT" instruction. A simple execution of "CLR WDT" will clear the WDT.

The maximum time out period is when the 2^{18} division ratio is selected. As an example, with the LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 second for the 2^{18} division ratio, and a minimum timeout of 7.8ms for the 2^{8} division ration.



Watchdog Timer

Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the device can be set to some pre-determined 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.

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, is implemented in situations where the power supply voltage falls below a certain threshold.

Reset Functions

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

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.

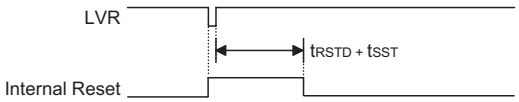
The microcontroller has an internal RC reset function, due to unstable power on conditions. This time delay created by the RC network ensures the state of the POR remains low for an extended period while the power supply stabilises. During this time, normal operation of the microcontroller is inhibited. After the state of the POR reaches a certain voltage value, the reset delay time t_{POR} is invoked to provide an extra delay time after which the microcontroller can begin normal operation.

Power-On Reset Timing Chart

Low Voltage Reset – LVR

The microcontrollers contain a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled with a specific LVR voltage, V_{LVR}. If the supply voltage of the devices drops to within a range of 0.9V~V_{LVR} such as might occur when changing a battery, the LVR will automatically reset the devices internally and the LVRF bit in the CTRL register will also be set to "1".

The LVR includes the following specifications: For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for greater than the value t_{LVR} specified in the A.C. characteristics. If the low voltage state does not exceed t_{LVR}, the LVR will ignore it and will not perform a reset function. The actual V_{LVR} is set by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by the environmental noise, the LVR will reset the device after 2~3 LIRC clock cycles. When this happens, the LRF bit in the CTRL register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the power down mode.



Low Voltage Reset Timing Chart

- **LVRC Register – BS83A04A-3**

Bit	7	6	5	4	3	2	1	0
Name	LVS7	LVS6	LVS5	LVS4	LVS3	LVS2	LVS1	LVS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	1	0	1

Bit 7 ~ 0 **LVS7 ~ LVS0**: LVR Voltage Select control
01010101: 2.55V
00110011: 2.55V
10011001: 2.55V
10101010: 2.55V
Other values: MCU reset (reset will be active after 2~3 LIRC clock for debounce time)
Note: Using S/W to write 00H~FFH can control the LVR voltage, also can reset the MCU. If the MCU reset is caused by the LVRC register, the LRF flag in the CTRL register will be set high.

- **LVRC Register – BS83A02A-4/BS83A04A-4**

Bit	7	6	5	4	3	2	1	0
Name	LVS7	LVS6	LVS5	LVS4	LVS3	LVS2	LVS1	LVS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	1	0	1	0	1	0	1

Bit 7 ~ 0 **LVS7 ~ LVS0**: LVR Voltage Select control
01010101: 2.10V
00110011: 2.10V
10011001: 2.10V
10101010: 2.10V
Other values: MCU reset (reset will be active after 2~3 LIRC clock for debounce time)
Note: Using S/W to write 00H~FFH can control the LVR voltage, also can reset the MCU. If the MCU reset is caused by the LVRC register, the LRF flag in the CTRL register will be set high.

- **CTRL Register**

Bit	7	6	5	4	3	2	1	0
Name	FSYSON	—	—	—	—	LVRF	LRF	WRF
R/W	R/W	—	—	—	—	R/W	R/W	R/W
POR	0	—	—	—	—	×	0	0

"×" unknown

Bit 7 Decribed in other section

Bit 6~3 Unimplemented, read as "0"

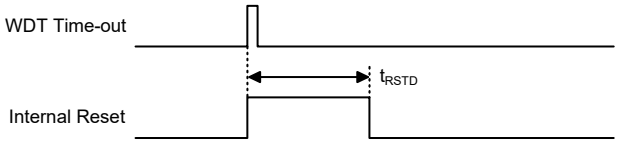
Bit 2 **LVRF**: LVR function reset flag
0: Not active
1: Active
This bit can be cleared to "0", but can not be set to "1".

Bit 1 **LRF**: LVR Control register software reset flag
0: Not active
1: Active
This bit can be cleared to "0", but can not be set to "1".

Bit 0 Decribed in other section

Watchdog Time-out Reset during Normal Operation

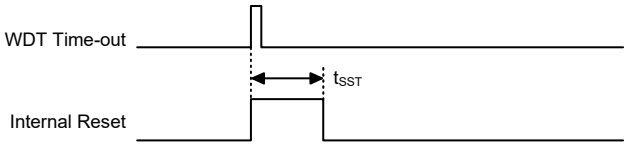
The Watchdog time-out Reset during normal operation is the same as a hardware power-on reset except that the Watchdog time-out flag TO will be set to high.



WDT Time-out Reset during Normal Operation Timing Chart

Watchdog Time-out Reset during SLEEP or IDLE Mode

The Watchdog time-out Reset during SLEEP or IDLE Mode 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 SLEEP or IDLE Timing Chart

Note: The t_{SST} is 1024~1025 clock cycles if the system clock source is provided by HIRC. The t_{SST} is 1~2 clock for LIRC.

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 SLEEP or IDLE Mode function or Watchdog Timer. The reset flags are shown in the table:

TO	PDF	RESET Conditions
0	0	Power-on reset
u	u	LVR reset during NORMAL or SLOW Mode operation
1	u	WDT time-out reset during NORMAL or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

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
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 the microcontroller internal registers.

Register		LVR & Power-on Reset	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
MP0		xxxx xxxx	uuuu uuuu	uuuu uuuu
MP1		xxxx xxxx	uuuu uuuu	uuuu uuuu
ACC		xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL		0000 0000	0000 0000	0000 0000
TBLP		xxxx xxxx	uuuu uuuu	uuuu uuuu
TBLH		xxxx xxxx	uuuu uuuu	uuuu uuuu
TBHP		---- --xx	---- --uu	---- --uu
STATUS		--00 xxxx	--1u uuuu	--11 uuuu
SMOD		000- 0011	000- 0011	uuu- uuuu
CTRL		0--- -x00	0--- -x00	u--- -uuu
INTEG		---- --00	---- --00	---- --uu
INTC0		-000 0000	-000 0000	-uuu uuuu
INTC1		--0- --0-	--0- --0-	--u- --u-
LVRC		0101 0101	0101 0101	uuuu uuuu
PA	BS83A02A-4	---- 1111	---- 1111	---- uuuu
	BS83A04A-3 BS83A04A-4	1111 1111	1111 1111	uuuu uuuu
PAC	BS83A02A-4	---- 1111	---- 1111	---- uuuu
	BS83A04A-3 BS83A04A-4	1111 1111	1111 1111	uuuu uuuu
PAPU	BS83A02A-4	---- 0000	---- 0000	---- uuuu
	BS83A04A-3 BS83A04A-4	0000 0000	0000 0000	uuuu uuuu
PAWU	BS83A02A-4	---- 0000	---- 0000	---- uuuu
	BS83A04A-3 BS83A04A-4	0000 0000	0000 0000	uuuu uuuu
WDTC		0101 0011	0101 0011	uuuu uuuu
TBC		--00 ----	--00 ----	--uu ----
TMR		0000 0000	0000 0000	uuuu uuuu
TMRC		--00 -000	--00 -000	--uu -uuu
TKTMR		0000 0000	0000 0000	uuuu uuuu
TKC0		-000 0-00	-000 0-00	-uuu u-uu
TK16DL		0000 0000	0000 0000	uuuu uuuu
TK16DH		0000 0000	0000 0000	uuuu uuuu
TKC1		---- --11	---- --11	---- --uu
TKM016DL		0000 0000	0000 0000	uuuu uuuu
TKM016DH		0000 0000	0000 0000	uuuu uuuu
TKM0ROL		0000 0000	0000 0000	uuuu uuuu
TKM0ROH		---- --00	---- --00	---- --uu
TKM0C0	BS83A02A-4	-00- 0000	-00- 0000	-uu- uuuu
	BS83A04A-3 BS83A04A-4	0000 0000	0000 0000	uuuu uuuu
TKM0C1	BS83A02A-4	0-00 --00	0-00 --00	u-uu --uu
	BS83A04A-3 BS83A04A-4	0-00 0000	0-00 0000	u-uu uuuu

Note: "-" not implement

"u" means "unchanged"

"x" means "unknown"

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. Most pins can have either an input or output designation under user program control. Additionally, as there are pull-high resistors and wake-up software configurations, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

The devices provide bidirectional input/output lines labeled with port names PA. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. 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.

I/O Register List

Device	Register Name	Bit							
		7	6	5	4	3	2	1	0
BS83A02A-4	PAWU	—	—	—	—	PAWU3	PAWU2	PAWU1	PAWU0
	PAPU	—	—	—	—	PAPU3	PAPU2	PAPU1	PAPU0
	PAC	—	—	—	—	PA3	PA2	PA1	PA0
	PA	—	—	—	—	PAC3	PAC2	PAC1	PAC0
BS83A04A-3 BS83A04A-4	PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
	PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
	PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
	PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0

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, all I/O pins, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using register PAPU etc.and are implemented using weak PMOS transistors.

PAPU Register – BS83A02A-4

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PAPU3	PAPU2	PAPU1	PAPU0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **PAPU3~PAPU0**: PA port pull-high resistor control
0: Disable
1: Enable

PAPU Register – BS83A04A-3/BS83A04A-4

Bit	7	6	5	4	3	2	1	0
Name	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PAPU7~PAPU0**: PA port pull-high resistor control
0: Disable
1: Enable

Port A Wake-up

The HALT instruction forces the microcontroller into the SLEEP or IDLE Mode which preserves power, 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 Port A pins from high to low. This function is especially suitable for applications that can be woken up via external switches. Each pin on Port A can be selected individually to have this wake-up feature using the PAWU register.

PAWU Register – BS83A02A-4

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PAWU3	PAWU2	PAWU1	PAWU0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **PAWU3~PAWU0**: PA wake-up function control
0: Disable
1: Enable

PAWU Register – BS83A04A-3/BS83A04A-4

Bit	7	6	5	4	3	2	1	0
Name	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 **PAWU7~PAWU0**: PA wake-up function control
0: Disable
1: Enable

I/O Port Control Registers

The I/O port has its own control register known as PAC, to control the input/output configuration. With this control register, each CMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O port is 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.

PAC Register – BS83A02A-4

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PAC3	PAC2	PAC1	PAC0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	1

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **PAC3~PAC0**: I/O Port bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

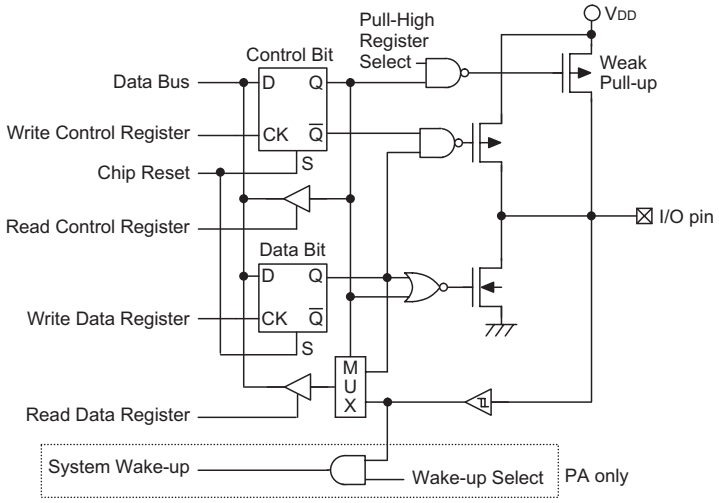
PAC Register – BS83A04A-3/BS83A04A-4

Bit	7	6	5	4	3	2	1	0
Name	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	1	1	1	1	1	1	1	1

Bit 7~0 **PAC7~PAC0**: I/O Port bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

I/O Pin Structures

The accompanying diagrams illustrate the internal structures of some generic I/O pin types. As the exact logical construction of the I/O pin will differ from these drawings, they are supplied as a guide only to assist with the functional understanding of the I/O pins. The wide range of pin-shared structures does not permit all types to be shown.



Generic Input/Output Ports

Programming Considerations

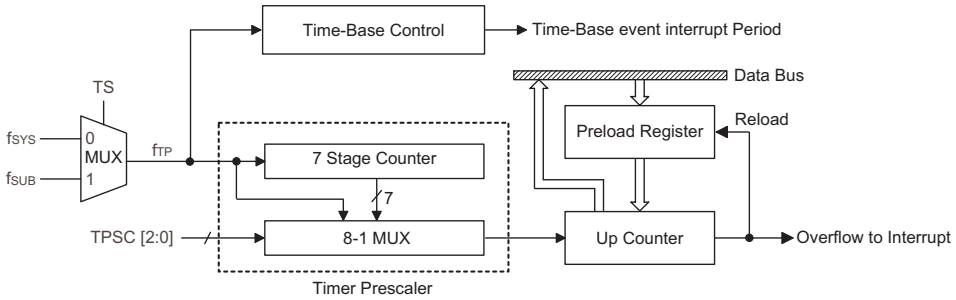
Within the user program, one of the things first to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set to high. This means that all I/O pins will be defaulted to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated port data registers are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate 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 rewrite this data back to the output ports.

Port A has the additional capability of providing wake-up functions. When the device is in the SLEEP or IDLE Mode, various methods are available to wake the device up. One of these is a high to low transition of any of the Port A pins. Single or multiple pins on Port A can be setup to have this function.

Timer/Event Counter

The provision of timers form an important part of any microcontroller, giving the designer a means of carrying out time related functions. The devices contain an 8-bit timer. And the provision of an internal prescaler to the clock circuitry on 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.



Timer/Event Counter

Configuring the Timer/Event Counter Input Clock Source

The Timer/Event Counter clock source can originate from either f_{SYS} or the f_{SUB} Oscillator, the choice of which is determined by the TS bit in the TMRC register. This internal clock source is first divided by a prescaler, the division ratio of which is conditioned by the Timer Control Register bits TPSC0~TPSC2.

Timer Register – TMR

The timer register TMR is a special function register located in the Special Purpose Data Memory and is the place where the actual timer value is stored. The value in the timer register increases by one each time an internal clock pulse is received. The timer will count from the initial value loaded by the preload register to the full count of FFH for the 8-bit Timer/Event Counter, at which point the timer overflows and an internal interrupt signal is generated. The timer value will then 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. Note that if the Timer/Event Counter is in an OFF condition and data is written to its preload register, 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 register during this period will remain in the preload register and will only be written into the actual counter the next time an overflow occurs.

Timer Control Register – TMRC

It is the Timer Control Register together with its corresponding timer register that controls 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.

The timer-on bit, which is bit 4 of the Timer Control Register and known as TON bit, 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 TMRC register determine the division ratio of the input clock prescaler. In addition, the bit TS is used to select the internal clock source.

TMRC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TS	TON	—	TPSC2	TPSC1	TPSC0
R/W	—	—	R/W	R/W	—	R/W	R/W	R/W
POR	—	—	0	0	—	0	0	0

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TS**: Timer/Event Counter Clock Source
0: f_{SYS}
1: f_{SUB}

Bit 4 **TON**: Timer/Event Counter Counting Enable
0: Disable
1: Enable

Bit 3 Unimplemented, read as "0"

Bit 2~0 **TPSC2~TPSC0**: Timer prescaler rate selection
Timer internal clock=
000: f_{TP}
001: $f_{TP}/2$
010: $f_{TP}/4$
011: $f_{TP}/8$
100: $f_{TP}/16$
101: $f_{TP}/32$
110: $f_{TP}/64$
111: $f_{TP}/128$

Timer/Event Counter Operation

The Timer/Event Counter can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter overflows. The internal clock is used as the timer clock. The timer input clock is either f_{SYS} or the f_{SUB} Oscillator. However, this timer clock source is further divided by a prescaler, the value of which is determined by the bits TPSC0~TPSC2 in the Timer Control Register. The timer-on bit, TON must be set high to enable the timer to run. Each time when an internal clock high to low transition occurs, the timer will reload the value already loaded into the preload register and continues 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 TE bit of the INTC0 register are reset to zero.

Prescaler

Bits TPSC0~TPSC2 of the TMRC 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.

Programming Consideration

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 initialized 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 initialized 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 IDLE/SLEEP Mode.

Touch Key Function

These devices provide touch key function. The touch key function is fully integrated and requires no external components, allowing touch key function to be implemented by the simple manipulation of internal registers.

Touch Key Structure

The touch keys are pin shared with the PA I/O pins, with the desired function chosen via register bits. The Touch Keys have their own control logic circuits and register set.

Device	Keys - n	Touch Key	Shared I/O Pin
BS83A02A-4	2	KEY1~KEY2	PA1, PA3
BS83A04A-3 BS83A04A-4	4	KEY1~KEY4	PA5, PA1, PA3, PA4

Touch Key Register Definition

The touch key module contains two or four touch key functions depending on the device chosen, has its own suite of eight registers. The following table shows the register set for this touch key module.

Name	Usage
TKTMR	Touch key 8-bit Timer/counter register
TKC0	Counter ON/OFF and clear control/ reference clock control/start bit
TK16DL	Touch key module 16-bit counter low byte contents
TK16DH	Touch key module 16-bit counter high byte contents
TKC1	Touch key OSC frequency select
TKM016DH	16-bit C/F counter high byte
TKM016DL	16-bit C/F counter low byte
TKM0ROL	Reference Oscillator internal capacitor select
TKM0ROH	Reference Oscillator internal capacitor select
TKM0C0	Control Register 0, Key Select
TKM0C1	Control Register 1, Touch key or I/O select

Touch Key Registers

Device	Register Name	Bit							
		7	6	5	4	3	2	1	0
All devices	TKTMR	D7	D6	D5	D4	D3	D2	D1	D0
	TKC0	—	TKRCOV	TKST	TKCFOV	TK16OV	—	TK16S1	TK16S0
	TK16DL	D7	D6	D5	D4	D3	D2	D1	D0
	TK16DH	D15	D14	D13	D12	D11	D10	D9	D8
	TKC1	—	—	—	—	—	—	TKFS1	TKFS0
	TKM016DL	D7	D6	D5	D4	D3	D2	D1	D0
	TKM016DH	D15	D14	D13	D12	D11	D10	D9	D8
	TKM0ROL	D7	D6	D5	D4	D3	D2	D1	D0
	TKM0ROH	—	—	—	—	—	—	D9	D8
BS83A02A-4	TKM0C0	—	M0MXS0	M0DFEN	—	M0SOFC	M0SOF2	M0SOF1	M0SOF0
	TKM0C1	M0TSS	—	M0ROEN	M0KOEN	—	—	M0K2IO	M0K1IO
BS83A04A-3 BS83A04A-4	TKM0C0	M0MXS1	M0MXS0	M0DFEN	M0FILEN	M0SOFC	M0SOF2	M0SOF1	M0SOF0
	TKM0C1	M0TSS	—	M0ROEN	M0KOEN	M0K4IO	M0K3IO	M0K2IO	M0K1IO

Touch Key Register Listing

TKTMR Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 Touch Key 8-bit timer/counter register
The Time slot counter overflow time setup is (256-TKTMR[7:0]) × 32

TKC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	TKRCOV	TKST	TKCFOV	TK16OV	—	TK16S1	TK16S0
R/W	—	R/W	R/W	R/W	R/W	—	R/W	R/W
POR	—	0	0	0	0	—	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **TKRCOV**: Time slot counter overflow flag
0: No overflow
1: Overflow

If time slot counter is overflow, the Touch Key Interrupt request flag, TKMF, will be set and module key oscillator and reference oscillator auto stop. All module 16-bit C/F counter, 16-bit counter, 5-bit time slot counter and 8-bit time slot timer counter will auto off.

Bit 5 **TKST**: Start Touch Key detection control bit
0:Stop
0→1:On

If this bit is 0, the 16-bit C/F counter, 16-bit counter, 5-bit time slot counter will aotu clear except for the 8-bit programmable time slot counter is not kown, its overflow time is setup by the user.

0→1: By the TKST rising edge, the 16-bit C/F counter, 16-bit counter, 5-bit time slot counter and 8-bit time slot timer counter will auto on and enable key oscillator and reference oscillator output clock to input to these counters.

Bit 4 **TKCFOV**: Touch key module 16-bit C/F counter overflow flag
0: Not overflow
1: Overflow

When the touch key module 16-bit C/F counter overflows, this bit will be set to 1. As this flag will not be automatically cleared, it has to be cleared by the application program.

Bit 3 **TK16OV**: Touch key module 16-bit counter overflow flag
0: Not overflow
1: Overflow

When the touch key module 16-bit counter overflows, this bit will be set to 1. As this flag will not be automatically cleared, it has to be cleared by the application program.

Bit 2 Unimplemented, read as "0"

Bit 1,0 **TK16S1, TK16S0**: touch key module 16-bit counter clock source selection
00: f_{SYS} /1
01: f_{SYS} /2
10: f_{SYS} /4
11: f_{SYS} /8

TKC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	TKFS1	TKFS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	1	1

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **TKFS1~TKFS0**: touch key OSC frequency selection
00: 500kHz
01: 1000kHz
10: 1500kHz
11: 2000kHz

TK16DL Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 Touch key module 16-bit counter low byte contents

TK16DH Register

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 Touch key module 16-bit counter high byte contents

TKM016DL Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 Module 0 16-bit counter low byte contents

TKM016DH Register

Bit	7	6	5	4	3	2	1	0
Name	D15	D14	D13	D12	D11	D10	D9	D8
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	0	0	0	0

Bit 7~0 Module 0 16-bit counter high byte contents

TKM0ROL Register

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	D5	D4	D3	D2	D1	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~0 Reference Oscillator internal capacitor selection
The reference oscillator is selected as TKM0RO[9:0] × 50pF / 1024.

TKM0ROH Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 Reference Oscillator internal capacitor selection
The reference oscillator is selected as TKM0RO[9:0] × 50pF / 1024.

TKM0C0 Register – BS83A02A-4

Bit	7	6	5	4	3	2	1	0
Name	—	M0MXS0	M0DFEN	—	M0SOFC	M0SOF2	M0SOF1	M0SOF0
R/W	—	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	—	0	0	—	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **M0MXS0**: Multiplexer Key Select
0: KEY1
1: KEY2

Bit 5 **M0DFEN**: Multi-frequency function control
0: Disable
1: Enable

Bit 4 Unimplemented, read as "0"

Bit 3 **M0SOFC**: C to F oscillator frequency-hopping function control
0: Controlled by the M0SOF2~M0SOF0 bits
1: Controlled by hardware, regardless of what is the state of M0SOF2~M0SOF0 bits
When controlled by hardware, the time slot counter selected by the M0SOF2~M0SOF0 bits adjust the C to F oscillator frequency automatically.

Bit 2~0 **M0SOF2~M0SOF0**: Selecting key oscillator or reference oscillator frequency as the C to F oscillator is controlled by software
000: 1380kHz
001: 1500kHz
010: 1670kHz
011: 1830kHz
100: 2000kHz
101: 2230kHz
110: 2460kHz
111: 2740kHz
The frequency mentioned here will differ with the external or internal capacitor value. If the key oscillator frequency is selected as 2MHz, users can adjust the frequency in scale when selecting other frequencies.

TKM0C0 Register – BS83A04A-3/BS83A04A-4

Bit	7	6	5	4	3	2	1	0
Name	M0MXS1	M0MXS0	M0DFEN	M0FILEN	M0SOFC	M0SOF2	M0SOF1	M0SOF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **M0MXS1, M0MXS0**: Multiplexer Key Select
00: KEY1
01: KEY2
10: KEY3
11: KEY4

Bit 5 **M0DFEN**: Multi-frequency function control
0: Disable
1: Enable

Bit 4 **M0FILEN**: Filter function control
0: Disable
1: Enable

Bit 3 **M0SOFC**: C to F oscillator frequency-hopping function control
0: Controlled by the M0SOF2~M0SOF0 bits
1: Controlled by hardware, regardless of what is the state of M0SOF2~M0SOF0 bits

When controlled by hardware, the time slot counter selected by the M0SOF2~M0SOF0 bits adjust the C to F oscillator frequency automatically.

Bit 2~0 **M0SOF2~M0SOF0**: Selecting key oscillator or reference oscillator frequency as the C to F oscillator is controlled by software
000: 1380kHz
001: 1500kHz
010: 1670kHz
011: 1830kHz
100: 2000kHz
101: 2230kHz
110: 2460kHz
111: 2740kHz

The frequency mentioned here will differ with the external or internal capacitor value. If the key oscillator frequency is selected as 2MHz, users can adjust the frequency in scale when selecting other frequencies.

TKM0C1 Register – BS83A02A-4

Bit	7	6	5	4	3	2	1	0
Name	M0TSS	—	M0ROEN	M0KOEN	—	—	M0K2IO	M0K1IO
R/W	R/W	—	R/W	R/W	—	—	R/W	R/W
POR	0	—	0	0	—	—	0	0

Bit 7 **M0TSS**: Time Slot counter Select
0: Reference Oscillator
1: f_{SYS} /4

Bit 6 Unimplemented, read as "0"

Bit 5 **M0ROEN**: Reference Oscillator control
0: Disable
1: Enable

Bit 4 **M0KOEN**: Key Oscillator control
0: Disable
1: Enable

Bit 3~2 Unimplemented, read as "0"

Bit 1 **M0K2IO**: I/O Pin or Touch Key 2 Function Select
0: I/O
1: Touch key input

Bit 0 **M0K1IO**: I/O Pin or Touch Key 1 Function Select
0: I/O
1: Touch key input

TKM0C1 Register – BS83A04A-3/BS83A04A-4

Bit	7	6	5	4	3	2	1	0
Name	M0TSS	—	M0ROEN	M0KOEN	M0K4IO	M0K3IO	M0K2IO	M0K1IO
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	0	0	0

Bit 7 **M0TSS**: Time Slot counter Select
0: Reference Oscillator
1: f_{SYS} /4

Bit 6 Unimplemented, read as "0"

Bit 5 **M0ROEN**: Reference Oscillator control
0: Disable
1: Enable

Bit 4 **M0KOEN**: Key Oscillator control
0: Disable
1: Enable

Bit 3 **M0K4IO**: I/O Pin or Touch Key 4 Function Select
0: I/O
1: Touch key input

Bit 2 **M0K3IO**: I/O Pin or Touch Key 3 Function Select
0: I/O
1: Touch key input

Bit 1 **M0K2IO**: I/O Pin or Touch Key 2 Function Select
0: I/O
1: Touch key input

Bit 0 **M0K1IO**: I/O Pin or Touch Key 1 Function Select
0: I/O
1: Touch key input

Touch Key Operation

When a finger touches or is in proximity to a touch pad, the capacitance of the pad will increase. By using this capacitance variation to change slightly the frequency of the internal sense oscillator, touch actions can be sensed by measuring these frequency changes. Using an internal programmable divider the reference clock is used to generate a fixed time period. By counting a number of generated clock cycles from the sense oscillator during this fixed time period touch key actions can be determined.

The devices contain two or four touch key inputs which are shared with logical I/O pins, with the desired function selected using register bits. The Touch Key module also has its own interrupt vectors and set of interrupt flags.

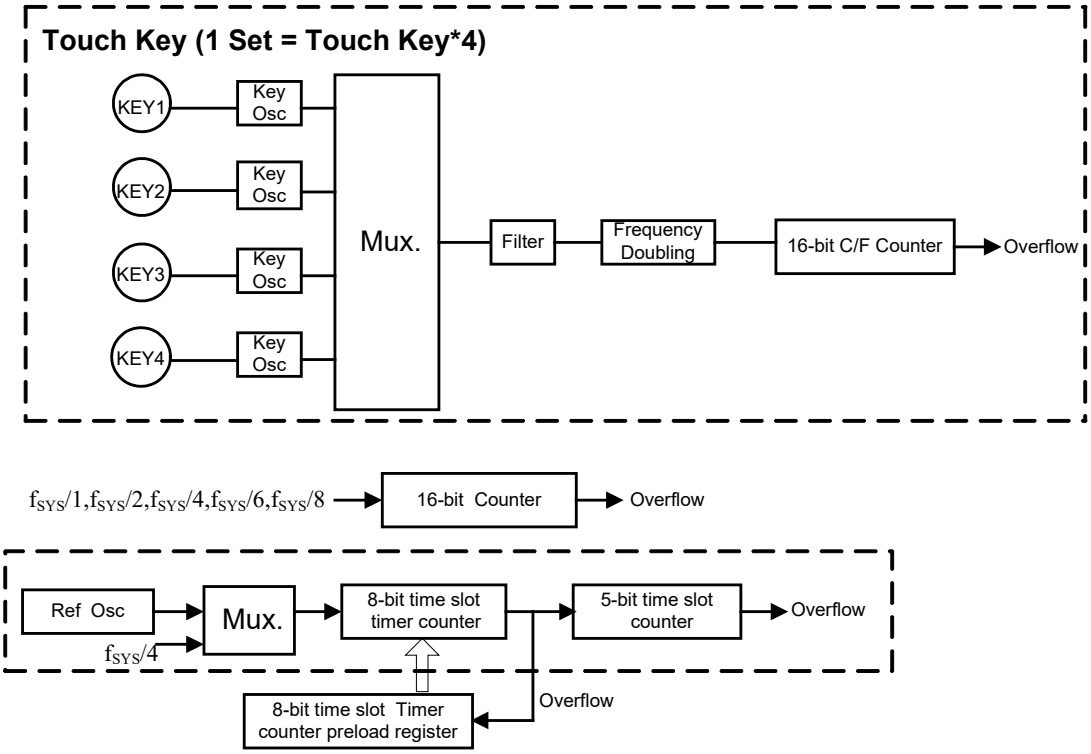
During this reference clock fixed interval, the number of clock cycles generated by the sense oscillator is measured, and it is this value that is used to determine if a touch action has been made or not. At the end of the fixed reference clock time interval, a Touch Key interrupt signal will be generated.

When a TKST rising edge occurs, the 16-bit C/F counter, 16-bit counter, 5-bit time slot counter and 8-bit time slot timer counter will auto on.While a TKST falling edge occurs, the 16-bit C/F counter, 16-bit counter and 5-bit time slot counter will auto off except that for the 8-bit programmable time slot counter, its overflow time is setup by the user.

When the 5-bit time slot counter is overflow, the key oscillator and reference oscillator auto stop and the 16-bit C/F counter, 16-bit counter, 5-bit time slot counter and 8-bit time slot timer counter will auto off. The 5-bit time slot counter and 8-bit time slot timer counter clock source derive from the reference oscillator or $f_{SYS}/4$.

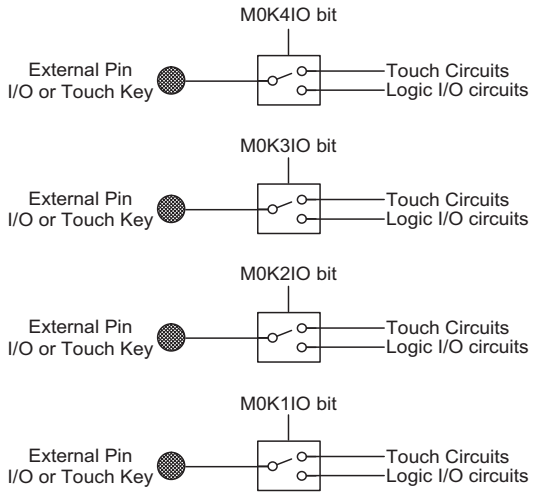
The reference oscillator is enabled by setting the bit M0ROEN in the TKM0C1 register to "1". The key oscillator is enabled by setting the bit M0KOEN in the TKM0C1 register to "1".

If the time slot counter is overflow, an interrupt will occur.



Note: 1. The dotted lines show the portions that each touch key individually has.
2. For the BS83A02A-4, there are only two touch keys in one set and there is no filter.

Touch Key Module Block Diagram



Touch Key or I/O Function Selection

Touch Key Interrupt

The touch key module, which consists of two or four touch keys, has one interrupt. If the time slot counter overflows, an interrupt will occur. At the same time, the 16-bit C/F counter, 16-bit counter, 5-bit time slot and 8-bit time slot timer counter will auto off. Only when all the enabled touch keys overflow, an interrupt will occur. More details regarding the touch key interrupt are located in the interrupt section of the datasheet.

Programming Considerations

After the relevant registers are setup, the touch key detection process is initiated the changing the TKST bit from low to high. This will enable and synchronise all relevant oscillators. The TKRCOV flag, which is the time slot counter flag will go high and remain high until the counter overflows. When this happens an interrupt signal will be generated.

When the external touch key size and layout are defined, their related capacitances will then determine the sensor oscillator frequency.

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Touch Action or Timer/Event Counter overflow 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 devices contain one external interrupt and several internal interrupt functions. The external interrupt is generated by the action of the external INT pin, while the internal interrupts are generated by various internal functions such as the Touch Keys, Timer/Event Counter and Time Base.

Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers depends upon the device chosen but fall into two categories. The first is the INTC0~INTC1 registers which setup the primary interrupts, the second is the INTEG register which setups the external interrupt trigger edge type.

Each register contains a number of enable bits to enable or disable individual interrupts as well as interrupt flags to indicate the presence of an interrupt request. The naming convention of these follows a specific pattern. First is listed an abbreviated interrupt type, then the (optional) number of that interrupt followed by either an "E" for enable/disable bit or "F" for request flag.

Function	Enable Bit	Request Flag
Global	EMI	—
INT Pin	INTE	INTF
Touch Key Module	TKME	TKMF
Timer/Event Counter	TE	TF
Time Base	TBE	TBF

Interrupt Register Bit Naming Conventions

Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	—	—	—	—	—	—	INTS1	INTS0
INTC0	—	TF	TKMF	INTF	TE	TKME	INTE	EMI
INTC1	—	—	TBF	—	—	—	TBE	—

Interrupt Register Contents

INTEG Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	INTS1	INTS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **INT2S1~INT2S0**: Interrupt edge control for INT pin
00: Disable
01: Rising edge
10: Falling edge
11: Both rising and falling edge

INTC0 Register

Bit	7	6	5	4	3	2	1	0
Name	—	TF	TKMF	INTF	TE	TKME	INTE	EMI
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7 Unimplemented, read as "0"

Bit 6 **TF**: Timer/Event Counter interrupt request Flag
0: No request
1: Interrupt request

Bit 5 **TKMF**: Touch Key module interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INTF**: INT pin interrupt request flag
0: No request
1: Interrupt request

Bit 3 **TE**: Timer/Event Counter interrupt control
0: Disable
1: Enable

Bit 2 **TKME**: Touch Key module interrupt control
0: Disable
1: Enable

Bit 1 **INTE**: INT interrupt control
0: Disable
1: Enable

Bit 0 **EMI**: Global interrupt control
0: Disable
1: Enable

INTC1 Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TBF	—	—	—	TBE	—
R/W	—	—	R/W	—	—	—	R/W	—
POR	—	—	0	—	—	—	0	—

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TBF**: Time Base interrupt request flag
0: No request
1: Interrupt request

Bit 4~2 Unimplemented, read as "0"

Bit 1 **TBE**: Time Base interrupt Control
0: Disable
1: Enable

Bit 0 Unimplemented, read as "0"

Interrupt Operation

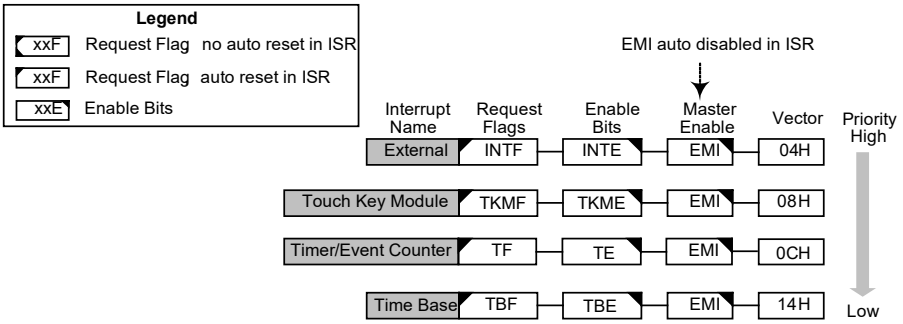
When the conditions for an interrupt event occur, such as a Touch Key Counter overflow, Timer/Event Counter overflow, etc., the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

When an interrupt is generated, 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 instruction 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", 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.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. These interrupt sources have their own individual vector. Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, 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. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake-up the devices if they are in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the devices are in SLEEP or IDLE Mode.



Interrupt Structure

External Interrupt

The external interrupt is controlled by signal transitions on the INT pin. An external interrupt request will take place when the external interrupt request flag, INTF, is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pin. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INTE, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pin is pin-shared with I/O pin, it can only be configured as external interrupt pin if the external interrupt enable bit in the corresponding interrupt register has been set. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pin will remain valid even if the pin is used as an external interrupt input.

The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

Touch Key Interrupt

For a Touch Key interrupt to occur, the global interrupt enable bit, EMI, and the corresponding Touch Key interrupt enable TKME must be first set. An actual Touch Key interrupt will take place when the Touch Key request flag, TKMF, is set, a situation that will occur when the time slot counter in the relevant Touch Key module overflows. When the interrupt is enabled, the stack is not full and the Touch Key time slot counter overflow occurs, a subroutine call to the relevant timer interrupt vector, will take place. When the interrupt is serviced, the Touch Key interrupt request flag, TKMF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

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, TE must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter request flag, TF, 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 overflow occurs, a subroutine call to the relevant timer interrupt vector, will take place. When the interrupt is serviced, the timer interrupt request flag, TF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Time Base Interrupt

Time Base Interrupt function is to provide regular time signal in the form of an internal interrupt. It is controlled by the overflow signal from its respective timer function. When this happens its respective interrupt request flag, TBF, will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bit, TBE, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TBF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Their clock sources originate from the internal clock source f_{SYS} or f_{SUB} selected by the TS bit in the TMRC register. The input clock passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TBC register to obtain longer interrupt periods whose value ranges. The clock source that generates f_{TP}, which in turn controls the Time Base interrupt period, can originate from several different sources, as shown in the System Operating Mode section.

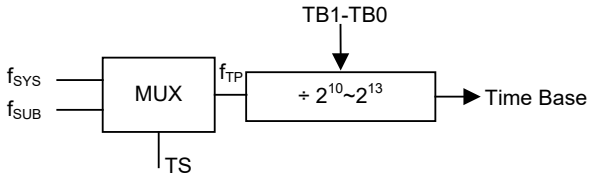
TBC Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	TB1	TB0	—	—	—	—
R/W	—	—	R/W	R/W	—	—	—	—
POR	—	—	0	0	—	—	—	—

Bit 7~6 Unimplemented, read as "0"

Bit 5~4 **TB1~TB0**: Select Time Base Time-out Period
00: $1024/f_{TP}$
01: $2048/ f_{TP}$
10: $4096/ f_{TP}$
11: $8192/ f_{TP}$

Bit 3~0 Unimplemented, read as "0"



Time Base Structure

Interrupt Wake-up Function

Each of the interrupt functions has the capability of waking up the microcontroller when in the SLEEP or IDLE Mode. A wake-up is generated when an interrupt request flag changes from low to high and is independent of whether the interrupt is enabled or not. Therefore, even though the devices are in the SLEEP or IDLE Mode and their system oscillator is stopped, situations such as external edge transitions on the external interrupt pins, a low power supply voltage or may cause their respective interrupt flag to be set high and consequently generate an interrupt. Care must therefore be taken if spurious wake-up situations are to be avoided. If an interrupt wake-up function is to be disabled then the corresponding interrupt request flag should be set high before the device enters the SLEEP or IDLE Mode. The interrupt enable bits have no effect on the interrupt wake-up function.

Programming Considerations

By disabling the relevant 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" instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. 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.

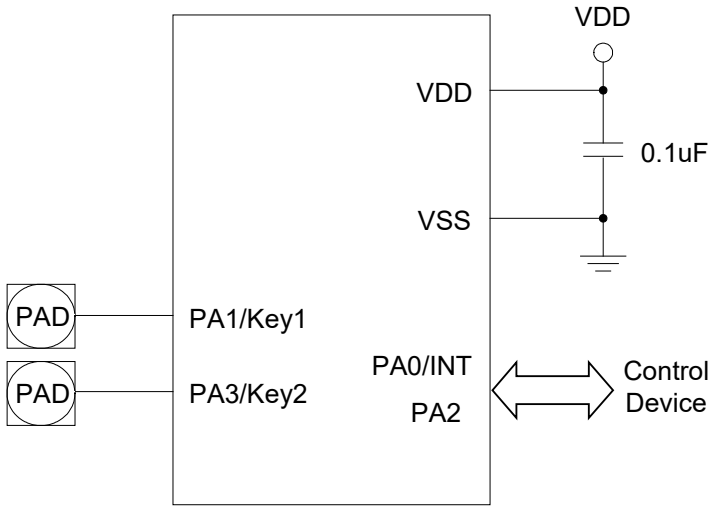
Every interrupt has the capability of waking up the microcontroller when it is in SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before entering the SLEEP or IDLE Mode.

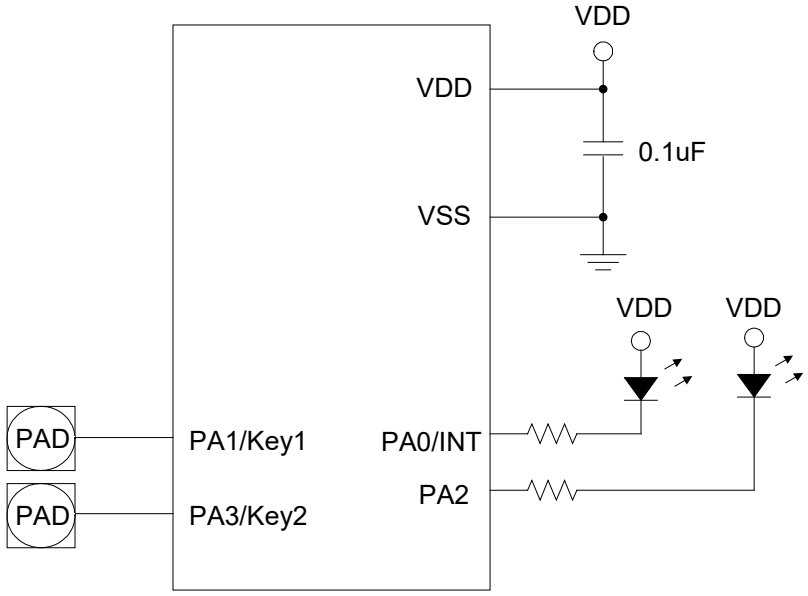
As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved in advance.

To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

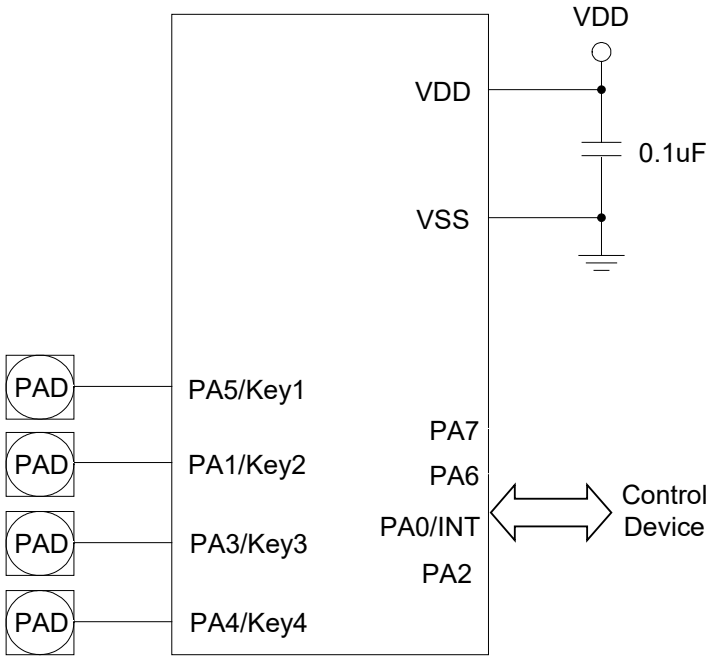
Application Circuits

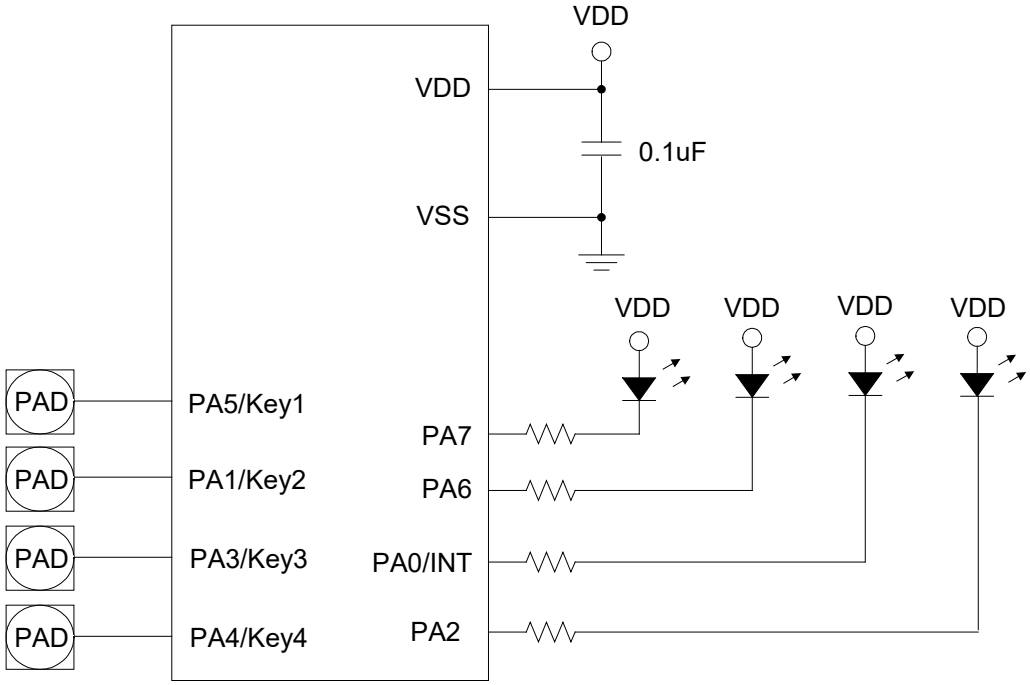
BS83A02A-4





BS83A04A-3/BS83A04A-4





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 microcontroller, 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 Operation

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 which 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 set 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
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

Mnemonic	Description	Cycles	Flag Affected
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 Operation			
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 Operation			
TABRD [m]	Read table (specific page or 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
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.

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 bit wise 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
CPL [m]	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	[m] ← $\overline{[m]}$
Affected flag(s)	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 ← $\overline{[m]}$
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] ← ACC + 00H or [m] ← ACC + 06H or [m] ← ACC + 60H or [m] ← ACC + 66H
Affected flag(s)	C

DEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	[m] ← [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 ← [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 ← 0 PDF ← 1
Affected flag(s)	TO, PDF

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

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

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

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

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

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

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

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

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

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

RET	Return from subroutine
Description	The Program Counter is restored from the stack. Program execution continues at the restored address.
Operation	Program Counter ← Stack
Affected flag(s)	None

RET A,x	Return from subroutine and load immediate data to ACC
Description	The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address.
Operation	Program Counter ← Stack ACC ← x
Affected flag(s)	None

RETI	Return from interrupt
Description	The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program.
Operation	Program Counter ← Stack EMI ← 1
Affected flag(s)	None

RL [m]	Rotate Data Memory left
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0.
Operation	[m].(i+1) ← [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) ← [m].i; (i=0~6)
[m].0 ← C
C ← [m].7

Affected flag(s): C

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

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

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

Affected flag(s): C

RR [m] Rotate Data Memory right

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

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

Affected flag(s): None

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

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

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

Affected flag(s): None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

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

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

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

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

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

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

Operation: [m] ← 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] ← [m] − 1
Skip if [m]=0

Affected flag(s): None

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

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

Operation: ACC ← [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] ← 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 ← 1

Affected flag(s): None

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

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

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

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

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

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

SWAP [m]	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	[m].3~[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	The contents of the specified Data Memory are read out and then written to the specified Data Memory again. 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
TABRD [m]	Read table (specific page or current page) to TBLH and Data Memory
Description	The low byte of the program code addressed by the table pointer (TBHP and TBLP or only TBLP if no TBHP) 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

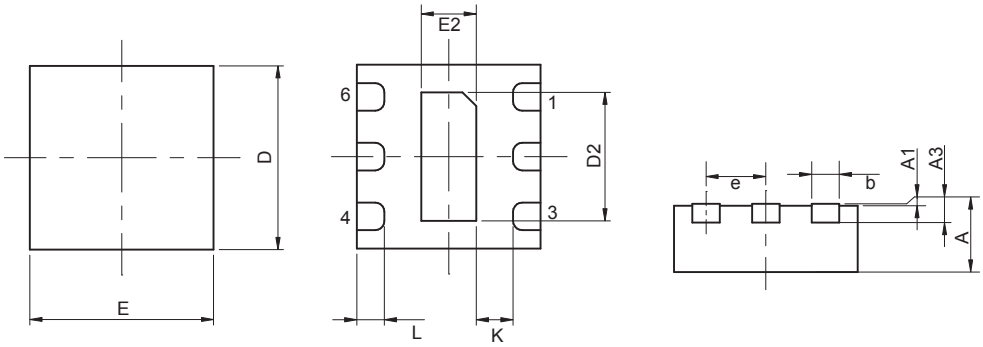
Package Information

Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

- Package Information (include Outline Dimensions, Product Tape and Reel Specifications)
- The Operation Instruction of Packing Materials
- Carton information

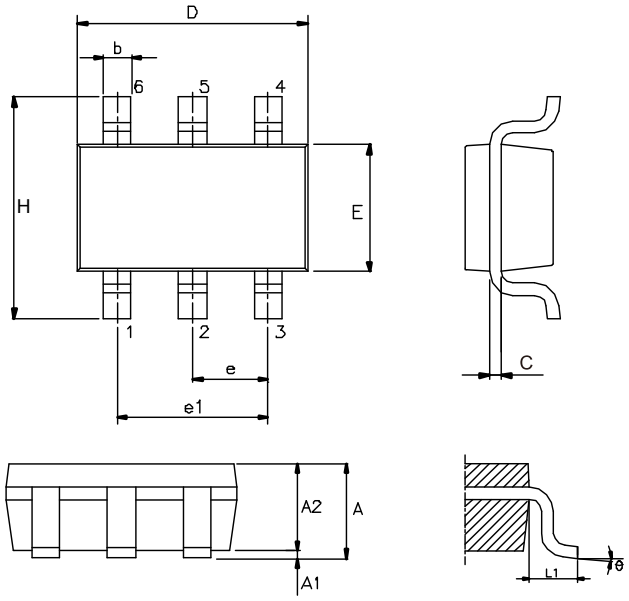
6-pin DFN (2mm×2mm) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.028	0.030	0.031
A1	0.000	0.001	0.002
A3	—	0.008 BSC	—
b	0.010	0.012	0.014
D	—	0.079 BSC	—
E	—	0.079 BSC	—
e	—	0.026 BSC	—
D2	0.053	0.055	0.057
E2	0.022	0.024	0.026
L	0.010	0.012	0.014
K	0.008	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.700	0.750	0.800
A1	0.000	0.020	0.050
A3	—	0.200 BSC	—
b	0.250	0.300	0.350
D	—	2.00 BSC	—
E	—	2.00 BSC	—
e	—	0.65 BSC	—
D2	1.350	1.400	1.450
E2	0.550	0.600	0.650
L	0.250	0.300	0.350
K	0.200	—	—

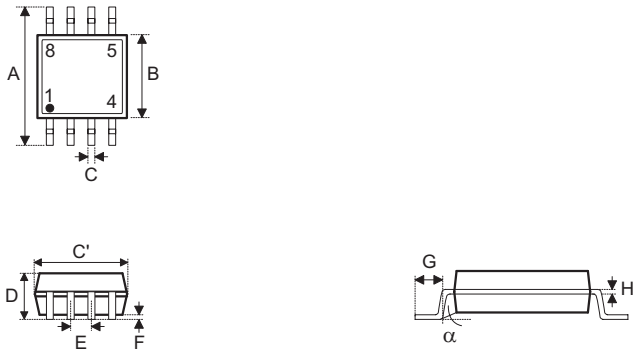
6-pn SOT23-6 Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	—	0.057
A1	—	—	0.006
A2	0.035	0.045	0.051
b	0.012	—	0.020
C	0.003	—	0.009
D	—	0.114 BSC	—
E	—	0.063 BSC	—
e	—	0.037 BSC	—
e1	—	0.075 BSC	—
H	—	0.110 BSC	—
L1	—	0.024 BSC	—
θ	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	—	1.45
A1	—	—	0.15
A2	0.90	1.15	1.30
b	0.30	—	0.50
C	0.08	—	0.22
D	—	2.90 BSC	—
E	—	1.60 BSC	—
e	—	0.95 BSC	—
e1	—	1.90 BSC	—
H	—	2.80 BSC	—
L1	—	0.60 BSC	—
θ	0°	—	8°

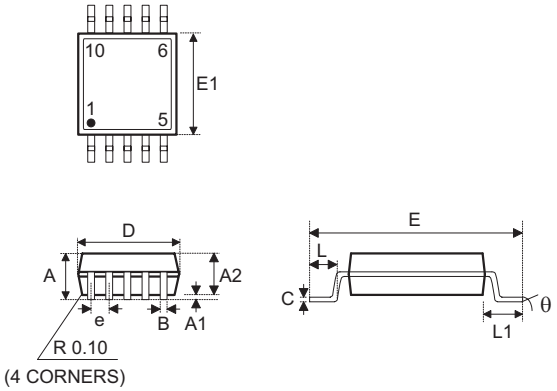
8-pin SOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C′	—	0.193 BSC	—
D	—	—	0.069
E	—	0.050 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.00 BSC	—
B	—	3.90 BSC	—
C	0.31	—	0.51
C′	—	4.90 BSC	—
D	—	—	1.75
E	—	1.27 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

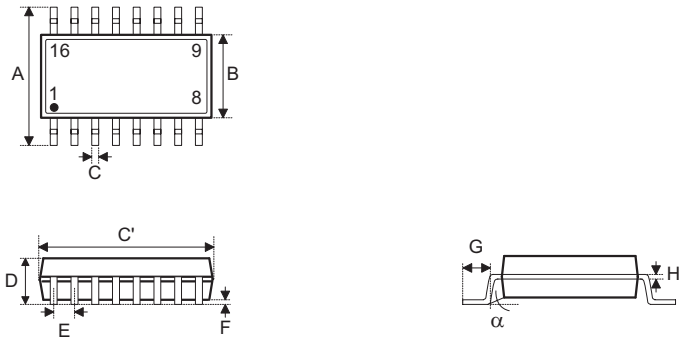
10-pin MSOP Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	—	0.043
A1	0.000	—	0.006
A2	0.030	0.033	0.037
b	0.007	—	0.013
c	0.003	—	0.009
D	—	0.118 BSC	—
E	—	0.193 BSC	—
E1	—	0.118 BSC	—
e	—	0.020 BSC	—
L	0.016	0.024	0.031
L1	—	0.037 BSC	—
y	—	0.004	—
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	—	1.100
A1	0.000	—	0.150
A2	0.750	0.850	0.950
b	0.170	—	0.330
c	0.080	—	0.230
D	—	3.000 BSC	—
E	—	4.900 BSC	—
E1	—	3.000 BSC	—
e	—	0.500 BSC	—
L	0.400	0.600	0.800
L1	—	0.950 BSC	—
y	—	0.1	—
α	0°	—	8°

16-pin NSOP (150mil) Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	—	0.236 BSC	—
B	—	0.154 BSC	—
C	0.012	—	0.020
C'	—	0.390 BSC	—
D	—	—	0.069
E	—	0.050 BSC	—
F	0.004	—	0.010
G	0.016	—	0.050
H	0.004	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	—	6.000 BSC	—
B	—	3.900 BSC	—
C	0.31	—	0.51
C'	—	9.900 BSC	—
D	—	—	1.75
E	—	1.270 BSC	—
F	0.10	—	0.25
G	0.40	—	1.27
H	0.10	—	0.25
α	0°	—	8°

Copyright© 2021 by HOLTEK SEMICONDUCTOR INC.

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