HT82B60R
I/O MCU with USB Interface

Features

- Operating voltage:
  f_{SYS}=6M/12MHz: 3.3V~5.5V
- Low voltage reset function
- 42 bidirectional I/O lines (max.)
- 8-bit programmable timer/event counter with overflow interrupt
- 16-bit programmable timer/event counter and overflow interrupts
- Watchdog Timer
- PS2 and USB modes supported
- USB 2.0 low speed function
- 4 endpoints supported -- endpoint 0 included
- 8192×16 program memory
- 216×8 data memory RAM
- Integrated 1.5kΩ resistor between V33O and USBPDN pins for USB applications
- Fully integrated 6MHz or 12MHz oscillator
- All I/O pins have wake-up functions
- Power-down function and wake-up feature reduce power consumption
- Serial Interface Module -- I^2C and SPI functions
- 4 COM lines for LCD display driving
- External interrupt pin
- 8-level subroutine nesting
- Up to 0.33μs instruction cycle with 12MHz system clock at V_{DD}=5V
- Bit manipulation instruction
- 15-bit table read instruction
- 63 powerful instructions
- All instructions in one or two machine cycles
- 20/28/48-pin SSOP, 32-pin QFN and 48-pin LQFP packages

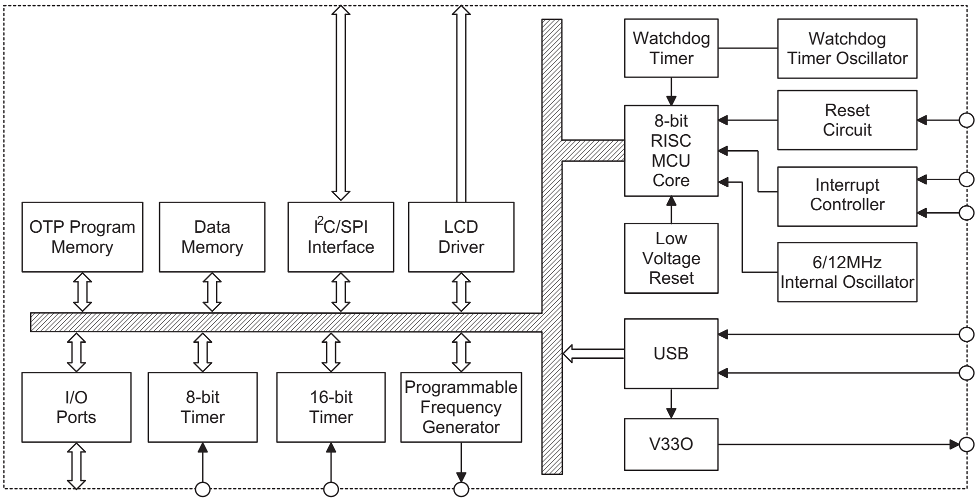
General Description

The HT82B60R is a high performance, RISC architecture microcontroller device specifically designed for multiple I/O control product applications.

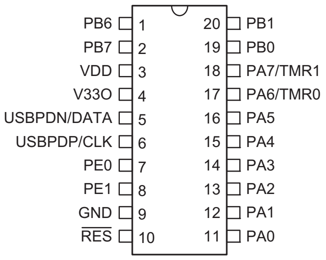
The advantages of low power consumption, I/O flexibility, timer functions, integrated USB interface, serial interfaces, LCD drive capability, power down and wake-up functions, watchdog timer etc, make the device extremely suitable for use in computer peripheral product applications as well as many other applications such as industrial control, consumer products, subsystem controllers, etc.

These wide range of functions, together with a fully integrated 6MHz or 12MHz oscillator, ensure that products can be implemented with a minimum of external components and smaller circuit board areas, providing users with the benefits of lower overall product costs.

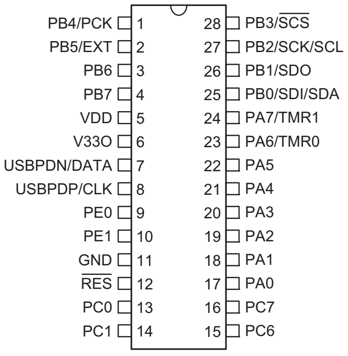
Block Diagram



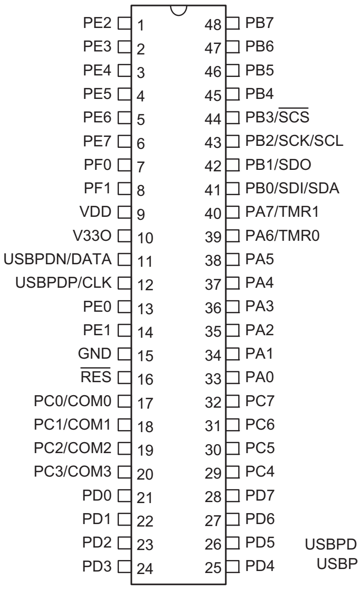
Pin Assignment



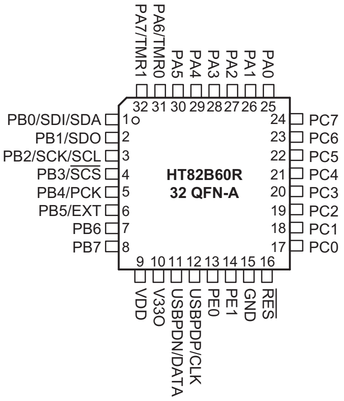
HT82B60R
20 SSOP-A

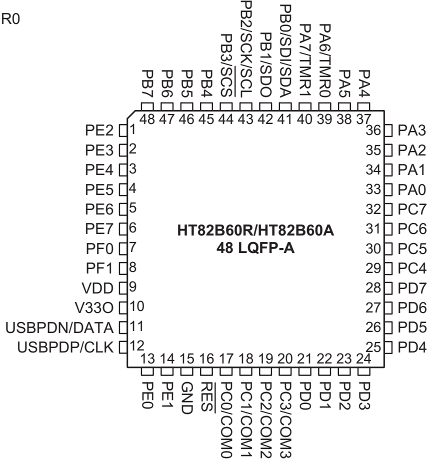


HT82B60R
28 SSOP-A



HT82B60R
48 SSOP-A





Pin Description

Pin Name	I/O	Options	Description
PA0~PA5 PA6/TMR0 PA7/TMR1	I/O	Pull-high Wake-up NMOS/CMOS/PMOS	Bidirectional 8-bit input/output port. Each pin can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or NMOS, PMOS or Schmitt Trigger input. Configuration options determine if the structures are CMOS, NMOS or PMOS types. Configuration options determine if the pins have pull-high resistors. TMR0 and TMR1 are pin-shared with PA6 and PA7, respectively.
PB0/SDI/SDA PB1/SDO PB2/$\overline{\text{SCK}}$/SCL PB3/$\overline{\text{SCS}}$ PB4/PCK PB5/INT PB6~PB7	I/O	Pull-high Wake-up	Bidirectional 8-bit input/output port. Each nibble can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine if the pins have pull-high resistors. The power supply for I/O pins PB0~PB7 can be selected to be VDD or V33O using a configuration option. Pins PB0~PB3 are pin-shared with the Serial Interface pins. Pin PB4 is pin-shared with the peripheral clock output and PB5 is shared with the external interrupt pin.
PC0/COM0 PC1/COM1 PC2/COM2 PC3/COM3 PC4~PC7	I/O	Pull-high Wake-up	Bidirectional 8-bit input/output port. Each nibble can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine if the pins have pull-high resistors. PC0~PC3 are pin-shared with COM0~COM3
PD0~PD7	I/O	Pull-high Wake-up	Bi-directional 8-bit input/output port. Each nibble can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine if the pins have pull-high resistors.
PE0~PE7	I/O	Pull-high Wake-up	Bidirectional 8-bit input/output port. Each nibble can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine if the pins have pull-high resistors.
PF0, PF1	I/O	Pull-high Wake-up	Bidirectional 2-bit input/output port. Each pin can be configured as a wake-up input by a configuration option. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine if the pins have pull-high resistors.
USBPDP/CLK	I/O	—	USBPDP line. USB function is controlled by software control registers.
USBPDN/DATA	I/O	—	USBPDN line. USB function is controlled by software control registers.
$\overline{\text{RES}}$	I	—	Schmitt trigger reset input. Active low
GND	—	—	Digital negative power supply, ground
VDD	—	—	Digital positive power supply
V33O	O	—	3.3V regulator output

Note: As the Pin Description table applies to the largest package size not all pin may exist on smaller packages.

Absolute Maximum Ratings

Supply Voltage	V_{SS}–0.3V to V_{SS}+6.0V
Input Voltage	V_{SS}–0.3V to V_{DD}+0.3V
I_{OL} Total	150mA
Total Power Dissipation	500mW
Storage Temperature	–50°C to 125°C
Operating Temperature	–40°C to 85°C
I_{OH} Total	–100mA

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage (Integrated Oscillator)	—	f_{SYS}=6MHz or 12MHz	3.3	—	5.5	V
I_{DD}	Operating Current	5V	No load, f_{SYS}=6MHz	—	6.5	12	mA
			No load, f_{SYS}=12MHz	—	7.5	16	mA
I_{STB1}	Standby Current	5V	No load, system HALT, USB mode, USR.5=1 USR.4=0, LVR disable, WDT disable, Clr D_SR [SCC.2], Clr USBCKEN [SCC.3], Clr BGOFF [SCC.4], Set CLK_adj [SCC.7]	—	—	400	μA
I_{STB2}	Standby Current	5V	No load, system HALT, PS2 mode, USR.5=0 USR.4=1, LVR disable, WDT disable, Clr D_SR [SCC.2], Clr USBCKEN [SCC.3], Set BGOFF [SCC.4], Set CLK_adj [SCC.7]	—	—	10	μA
V_{IL}	Input Low Voltage for PA, PC, PD, PE, PF0~PF1	5V	where V_{DDIO}=V_{DD} or V33O by option for PB	0	—	0.8	V
	Input Low Voltage for PB			0	—	$0.3V_{DDIO}$	V
	Input Low Voltage for $\overline{\text{RES}}$ pin			0	—	$0.4V_{DD}$	V
V_{IH}	Input High Voltage for PA, PC, PD, PE, PF0~PF1	5V	where V_{DDIO}=V_{DD} or V33O by option for PB	2	—	5	V
	Input High Voltage for PB			$0.8V_{DDIO}$	—	V_{DDIO}	V
	Input High Voltage for $\overline{\text{RES}}$ pin			$0.9V_{DD}$	—	V_{DD}	V
V_{LVR}	Low Voltage Reset	5V	—	2.0	2.6	3.2	V
V_{V33O}	3.3V Regulator Output for USB SIE	5V	I_{V33O}=70mA	3.0	3.3	3.6	V
I_{OL}	Output Sink Current for I/O Port	5V	V_{OL}=0.4V	2	4	—	mA
I_{OH}	Output Source Current for I/O Port	5V	V_{OH}=3.4V	−2	−4	—	mA
I_{LCD_BIAS}	VDD/2 Bias current for LCD	5V	LCDC. RSEL[1:0]=00	17.5	25	32.5	μA
			LCDC. RSEL[1:0]=01	35	50	65	μA
			LCDC. RSEL[1:0]=10	70	100	130	μA
			LCDC. RSEL[1:0]=11	140	200	260	μA
V_{COM}	VDD/2 voltage for LCD COM port	5V	No load	0.475	0.500	0.525	V_{DD}
R_{PH}	Pull-high Resistance for CLK, DATA	5V	—	—	4.7	—	kΩ
	Pull-high Resistance for PA, PB, PC, PD, PE and PF0~PF1			20	50	70	kΩ

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{RCSYS}	RC Clock with 8-bit Prescaler Register	5V	—	14	32	48	kHz
t_{WDT}	Watchdog Time-out Period (System Clock)	—	—	1024	—	—	$1/f_{RCSYS}$
t_{USB}	USBPDP, USBPDN Rising & Falling Time	—	—	75	—	300	ns
t_{OST}	Oscillation Start-up Timer Period	—	—	—	1024	—	t_{SYS}
$t_{OSCsetup}$	Crystal Setup	—	—	—	5	—	ms
$f_{INO125V}$	Internal Oscillator Frequency for 12MHz	4.0V~ 5.5V	—	10.80	12.00	13.20	MHz
$f_{INO123V}$	Internal Oscillator Frequency for 12MHz	3.0~ 4.0V	—	10.56	12.00	13.44	MHz
f_{INOUSB}	Internal Oscillator Frequency with USB Mode	4.2~ 5.5V	—	11.82	12.00	12.18	MHz

Note: $t_{SYS}=1/f_{SYS}$

Power_on period = $t_{WDT} + t_{OST} + t_{OSCsetup}$

WDT Time_out in Normal Mode = $1/ f_{RCSYS} \times 256 \times WDTS + t_{WDT}$

WDT Time_out in Power Down Mode = $1/ f_{RCSYS} \times 256 \times WDTS + t_{OST} + t_{OSCsetup}$

Trimmed for 5V operation using factory trim values. Frequency Trim to 12MHz ±3%

System Architecture

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

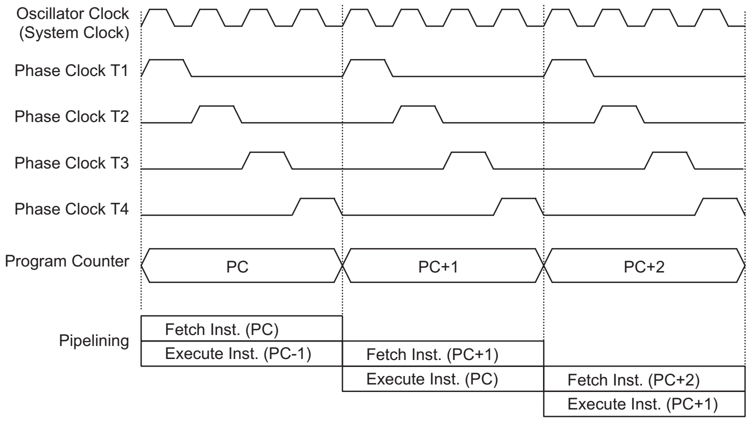
Clocking and Pipelining

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

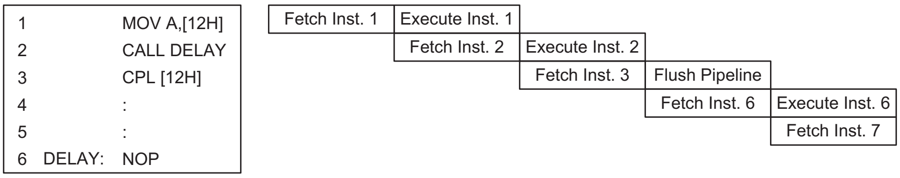
For instructions involving branches, such as jump or call instructions, two machine cycles are required to complete instruction execution. An extra cycle is required as the program takes one cycle to first obtain the actual jump or call address and then another cycle to actually execute the branch. The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

Program Counter

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



System Clocking and Pipelining



```
1          MOV A,[12H]
2          CALL DELAY
3          CPL [12H]
4          :
5          :
6  DELAY:  NOP
```

Instruction Fetching

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

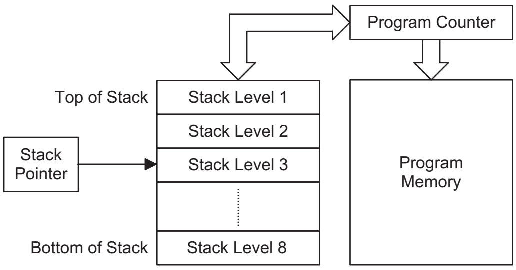
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly, however, as only this low byte is available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

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

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack has 8 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, SP, and is neither readable nor writeable. 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.



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

Mode	Program Counter Bits												
	*12	*11	*10	*9	*8	*7	*6	*5	*4	*3	*2	*1	*0
Initial Reset	0	0	0	0	0	0	0	0	0	0	0	0	0
USB Interrupt	0	0	0	0	0	0	0	0	0	0	1	0	0
Timer/Event Counter 0 Overflow	0	0	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter 1 Overflow	0	0	0	0	0	0	0	0	0	1	1	0	0
SPI/I^{2}C Interrupt	0	0	0	0	0	0	0	0	1	0	0	0	0
External Interrupt	0	0	0	0	0	0	0	0	1	0	1	0	0
Skip	Program Counter + 2												
Loading PCL	*12	*11	*10	*9	*8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#12	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S12	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

Note: *12~*0: Program Counter bits
#12~#0: Instruction code bits
@7~@0: PCL bits
S12~S0: Stack register bits

- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement INCA, INC, DECA, DEC
- Branch decision, JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI

Program Memory

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

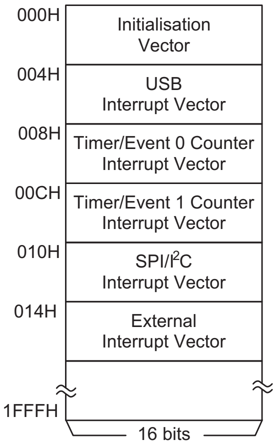
Structure

The Program Memory has a capacity of 8K by 16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by separate table pointer registers.

Special Vectors

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

- Location 000H
  This area is reserved for program initialization. After chip reset, the program always begins execution at location 000H.
- Location 004H
  This area is reserved for the USB interrupt service program. If the USB interrupt is activated, the interrupt is enabled and the stack is not full, the program jumps to this location and begins execution.
- Location 008H
  This area is reserved for the Timer/Event Counter 0 interrupt service program. If a timer interrupt results from a Timer/Event Counter 0 overflow, and if the interrupt is enabled and the stack is not full, the program jumps to this location and begins execution.
- Location 00CH
  This area is reserved for the Timer/Event Counter 1 interrupt service program. If a timer interrupt results from a Timer/Event Counter 1 overflow, and the interrupt is enabled and the stack is not full, the program jumps to this location and begins execution.
- Location 010H
  This internal vector is used by the SPI/I^{2}C interrupt. When either an SPI or I^{2}C bus, dependent upon which one is selected, requires data transfer, the program will jump to this location and begin execution if the SPI/I^{2}C interrupt is enabled and the stack is not full.
- Location 014H
  This vector is used by the external interrupt. If the external interrupt pin receives an active edge, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full.



Program Memory Structure

Instruction	Table Location Bits												
	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TABRDC [m]	PC12	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
TABRDL [m]	1	1	1	1	1	@7	@6	@5	@4	@3	@2	@1	@0

Table Location

Note: PC12~PC8: Current Program Counter bits @7~@0: Table Pointer TBLP bits
TBHP register Bit 4~Bit 0 when TBHP is enabled.

- Table location

  Any location in the program memory can be used as look-up tables. There are three methods to read the Program Memory data using two table read instructions: ″TABRDC″ and ″TABRDL″, transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH.

  The three methods are shown as follows:

  - Using the instruction ″TABRDC [m]″ for the current Program Memory page, where one page= 256words, where the table location is defined by TBLP in the current page. This is where the configuration option has disabled the TBHP register.
  - Using the instruction ″TABRDC [m]″, where the table location is defined by registers TBLP and TBHP. Here the configuration option has enabled the TBHP register.
  - Using the instruction ″TABRDL [m]″, where the table location is defined by registers TBLP in the last page which has the address range 1F00H~ 1FFFFH.

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

  Once TBHP is enabled, the instruction ″TABRDC [m]″ reads the Program Memory data as defined by the TBLP and TBHP values. If the Program Memory code option has disabled TBHP, the instruction ″TABRDC [m]″ reads the Program Memory data as defined by TBLP only in the current Program Memory page.

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be setup by placing the lower order address of the look up data to be retrieved in the TBLP register and the higher order address in the TBHP register. These two registers define the full address of the look-up table. Using the TBHP must be selected by configuration option, if not used table data can still be accessed but only the lower byte address in the current page or last page can be defined.

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

Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. 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 ″1F00H″ which refers to the start address of the last page within the 8K Program Memory of device. 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 ″1F06H″ or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the ″TABRDC [m]″ instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the ″TABRDL [m]″ instruction is executed.

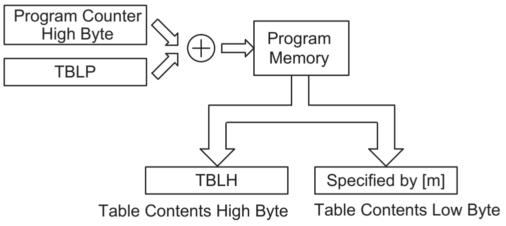


Table Read – TBLP only

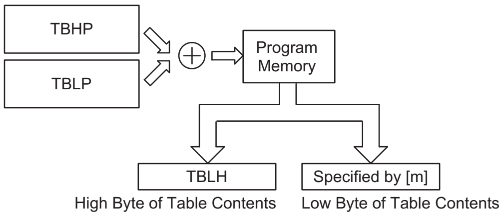


Table Read – TBLP/TBHP

```
tempreg1 db    ?   ; temporary register #1
tempreg2 db    ?   ; temporary register #2
     :
     :
mov  a,06h         ; initialise table pointer - note that this address is referenced

mov  tblp,a        ; to the last page or present page
     :
     :
tabrdl   tempreg1  ; transfers value in table referenced by table pointer to tempregl
                   ; data at prog. memory address ″1F06H″ transferred to tempreg1 and TBLH

dec  tblp          ; reduce value of table pointer by one

tabrdl   tempreg2  ; transfers value in table referenced by table pointer to tempreg2
                   ; data at prog.memory address ″1F05H″ 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  1F00h         ; sets initial address of last page

dc   00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
     :
     :
```

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

Data Memory

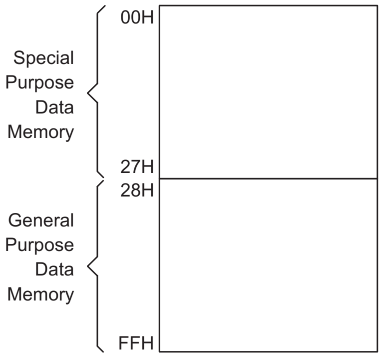
The Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored. Divided into two sections, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct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

Structure

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

General Purpose Data Memory

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



Data Memory Structure

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

Special Purpose Data Memory

This area of Data Memory is where registers, necessary for the correct operation of the microcontroller, are stored. It is divided into two banks, Bank 0 and Bank1. Most of the registers are both readable and writeable but some are protected and are readable only, the details of which are located under the relevant Special Function Register section. Note that for locations that are unused, any read instruction to these addresses will return the value ″00H″.

The Special Purpose Registers for the USB interface are stored in Bank 1 which can only be accessed by first setting the Bank Pointer to a value of 01H and then using Indirect Addressing Register IAR1 and Memory Pointer MP1. Bank 1 can only be accessed indirectly using the MP1 Memory Pointer, direct addressing is not possible.

	Bank0		Bank1
00H	IAR0	40H	USB_STAT
01H	MP0	41H	PIPE_CTRL
02H	IAR1	42H	AWR
03H	MP1	43H	STALL
04H	BP	44H	PIPE
05H	ACC	45H	SIES
06H	PCL	46H	MISC
07H	TBLP	47H	ENDPT_EN
08H	TBLH	48H	FIFO0
09H	WDTS	49H	FIFO1
0AH	STATUS	4AH	FIFO2
0BH	INTC0	4BH	FIFO3
0CH	▨		
0DH	TMR0		
0EH	TMR0C		
0FH	TMR1H		
10H	TMR1L		
11H	TMR1C		
12H	PA		
13H	PAC		
14H	PB		
15H	PBC		
16H	PC		
17H	PCC		
18H	PD		
19H	PDC		
1AH	PE		
1BH	PEC		
1CH	PF		
1DH	PFC		
1EH	INTC1		
1FH	TBHP		
20H	USC		
21H	USR		
22H	SCC		
23H	LCDC		
24H	SIMCTL0		
25H	SIMCTL1		
26H	SIMDIR		
27H	SIMAR/SIMCTL2		

▨ : Unused read as "0"

Special Purpose Data Memory

Special Function Registers

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

Indirect Addressing Register – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointer, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together only access data from Bank 0, while the IAR1 and MP1 register pair can access data from both Bank 0 and Bank 1. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of ″00H″ and writing to the registers indirectly will result in no operation.

Memory Pointer – MP0, MP1

For all devices, two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to, is the address specified by the related Memory Pointer. MP0 can only access data in Bank 0 while MP1 can access both banks.

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

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

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

continue:
```

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

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBLH, TBHP

These two 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 values 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.

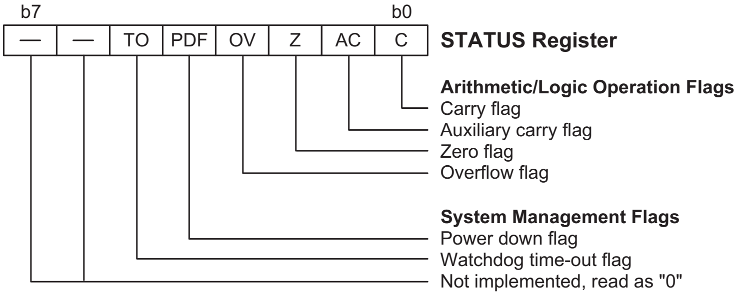
Watchdog Timer Register – WDTS

The Watchdog feature of the microcontroller provides an automatic reset function giving the microcontroller a means of protection against spurious jumps to incorrect Program Memory addresses. To implement this, a timer is provided within the microcontroller which will issue a reset command when its value overflows. To provide variable Watchdog Timer reset times, the Watchdog Timer clock source can be divided by various division ratios, the value of which is set using the WDTS register. By writing directly to this register, the appropriate division ratio for the Watchdog Timer clock source can be setup. Note that only the lower 3 bits are used to set division ratios between 1 and 128.

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.



Status Register

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

- **C** is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- **AC** is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
- **Z** is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- **OV** is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
- **PDF** is cleared by a system power-up or executing the ″CLR WDT″ instruction. PDF is set by executing the ″HALT″ instruction.
- **TO** is cleared by a system power-up or executing the ″CLR WDT″ or ″HALT″ instruction. TO is set by a WDT time-out.

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

Interrupt Control Registers – INTC0, INTC1

The microcontrollers provide two internal timer/event counter overflow interrupts, one USB interrupt, a combined SPI/I^{2}C interrupt and an external pin interrupt. By setting various bits within these registers using standard bit manipulation instructions, the enable/disable function of each interrupt can be independently controlled. A master interrupt bit within this register, the EMI bit, acts like a global enable/disable and is used to set all of the interrupt enable bits on or off. This bit is cleared when an interrupt routine is entered to disable further interrupt and is set by executing the ″RETI″ instruction.

Timer/Event Counter Registers – TMR0, TMR0C, TMR1H, TMR1L, TMR1C

Both devices possess a single internal 8-bit count-up timer. An associated register known as TMR0 is the location where the timers 8-bit value is located. This register can also be preloaded with fixed data to allow different time intervals to be setup. An associated control register, known as TMR0C, contains the setup information for this timer, which determines in what mode the timer is to be used as well as containing the timer on/off control function.

All devices possess one internal 16-bit count-up timer. An associated register pair known as TMR1L/TMR1H is the location where the timer 16-bit value is located. This register can also be preloaded with fixed data to allow different time intervals to be setup. An associated control register, known as TMR1C, contains the setup information for this timer, which determines in what mode the timer is to be used as well as containing the timer on/off control function.

Input/Output Ports and Control Registers

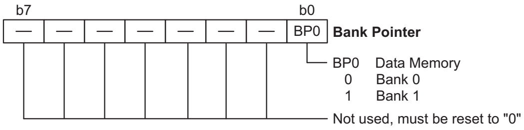
Within the area of Special Function Registers, the I/O registers and and their associated control registers play a prominent role. All I/O ports have a designated register correspondingly labeled as PA, PB, PC, PD, PE and PF0~PF1. These labeled I/O registers are mapped to specific addresses within the Data Memory as shown in the Data Memory table, which are used to transfer the appropriate output or input data on that port. With each I/O port there is an associated control register labeled PAC, PBC, PCC, PDC, PEC and PFC, also mapped to specific addresses with the Data Memory.

The control register specifies which pins of that port are set as inputs and which are set as outputs. To setup a pin as an input, the corresponding bit of the control register must be set high, for an output it must be set low. During program initialisation, it is important to first setup the control registers to specify which pins are outputs and which are inputs before reading data from or writing data to the I/O ports. One flexible feature of these registers is the ability to directly program single bits using the

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

Bank Pointer − BP

The Special Purpose Data Memory is divided into two Banks, Bank 0 and Bank 1. The USB control registers are located in Bank 1, while all other registers are located in Bank 1. The Bank Pointer selects which bank data is to be accessed from. If Bank 0 is to be accessed then BP must be set to a value of 00H, while if Bank 1 is to be accessed then BP must be set to a value of 01H.



Bank Pointer

Serial Interface Registers

The device contains two serial interfaces, an SPI and an I^2C interface. The SIMCTL0, SIMCTL1, SIMCTL2 and SIMAR are the control registers for the Serial Interface function while the SIMDR is the data register for the Serial Interface Data.

Software COM Register − SCOMC

The pins PC0~PC3 on Port C can be used as COM lines to drive an external LCD panel. To implement this function, the LCDC register is used to setup the correct bias voltages on these pins.

Input/Output Ports

Holtek microcontrollers offer considerable flexibility on their I/O ports. With the input or output designation of every pin fully under user program control, pull-high options for all ports and wake-up options on certain pins, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

Depending upon which package is chosen, the microcontroller provides up to 42 bidirectional input/output lines labeled with port names PA, PB, PC, PD, PE and PF0~PF1.

These I/O ports are mapped to the Data Memory with addresses as shown in the Special Purpose Data Memory table. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction ″MOV A,[m]″, where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, I/O pins, when configured as an input have the capability of being connected to an internal pull-high resistor. The pull-high resistors are selectable via configuration options and are implemented using weak PMOS transistors. A pin or nibble option on the I/O ports can be selected to select pull-high Resistors.

Port A CMOS/NMOS/PMOS Structure

The pins on Port A can be setup via configuration option to be either CMOS, NMOS or PMOS types.

Port B VDD/V33O Option Structure

The power supply for the Port B pins can be setup via configuration option to be either VDD or V33O.

Port Pin Wake-up

If the HALT instruction is executed, the device will enter the Power Down Mode, where the system clock will stop resulting in power being conserved, a feature that is important for battery and other low-power applications. Various methods exist to wake-up the microcontroller, one of which is to change the logic condition on one of the port pins from high to low. After a HALT instruction forces the microcontroller into entering the Power Down Mode, the processor will remain in a low-power state until the logic condition of the selected wake-up pin on the port pin changes from high to low. This function is especially suitable for applications that can be woken up via external switches. Each pin on PA, PB, PC, PD, PE and PF0~PF1 has the capability to wake-up the device on an external falling edge. Note that some pins can only be setup nibble wide whereas other can be bit selected to have a wake-up function.

I/O Port Control Registers

Each I/O port has its own control register PAC, PBC, PCC, PDC, PEC and PFC, to control the input/output configuration. With this control register, each CMOS output or input with or without pull-high resistor structures can be reconfigured dynamically under software control. Each of the I/O ports is directly mapped to a bit in its associated port control register. Note that several pins can be setup to have NMOS outputs using configuration options.

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 an output. If the pin is currently setup as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

Pin-shared Functions

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

- External Interrupt Input
  The external interrupt pin, INT is pin-shared with the I/O pin PB5. For applications not requiring an external interrupt input, the pin-shared external interrupt pin can be used as a normal I/O pin, however to do this, the external interrupt enable bit in the INTC1 register must be disabled.
- External Timer0 Clock Input
  The external timer pin TMR0 is pin-shared with the I/O pin PA6. To configure this pin to operate as timer input, the corresponding control bits in the timer control register must be correctly set. For applications that do not require an external timer input, this pin can be used as a normal I/O pin. Note that if used as a normal I/O pin the timer mode control bits in the timer control register must select the timer mode, which has an internal clock source, to prevent the input pin from interfering with the timer operation.
- External Timer1 Clock Input
  The external timer pin TMR1 is pin-shared with the I/O pin PA7. To configure this pin to operate as timer input, the corresponding control bits in the timer control register must be correctly set. For applications that do not require an external timer input, this pin can be used as a normal I/O pin. Note that if used as a normal I/O pin the timer mode control bits in the timer control register must select the timer mode, which has an internal clock source, to prevent the input pin from interfering with the timer operation.

External Interrupt Input

The external interrupt pin INT is pin-shared with the I/O pin PB5. For applications not requiring an external interrupt input, the pin-shared external interrupt pin can be used as a normal I/O pin, however to do this, the external interrupt enable bits in the INTC1 register must be disabled.

COM Driver Pins

Pins PC0~PC3 on Port C can be used as LCD COM driver pins. This function is controlled using the LCDC register which will generate the necessary 1/2 bias signals on these four pins.

Serial Interface Module

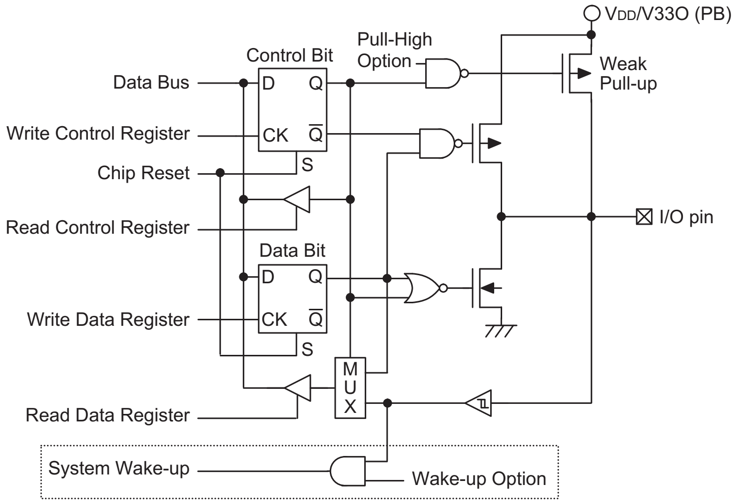
The device pins, PB0~PB3, are pin-shared with pins SDA, SCL, SCS, SCK, SDI, SDO. The choice of which function is used is selected using the SIMCTL0 register.

I/O Pin Structures

The diagram illustrates a generic I/O pin internal structures. As the exact logical construction of the I/O pin will differ and as the pin-shared structures are not illustrated this diagram is supplied as a guide only to assist with the functional understanding of the I/O pins.

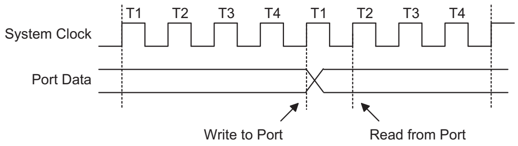
Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the data and port control register will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high options have been selected. If the PAC, PBC,



Input/Output Ports

PCC, PDC, PEC and PFC port control register, are then programmed to setup some pins as outputs, these output pins will have an initial high output value unless the associated PA, PB, PC, PD, PE and PF port data registers are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct value into the port control register or by programming individual bits in the port control register using the ″SET [m].i″ and ″CLR [m].i″ instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.



Read/Write Timing

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

Timer/Event Counters

The provision of timers form an important part of any microcontroller, giving the designer a means of carrying out time related functions. This device contains two count-up timers of 8-bit and 16-bit capacities respectively. As each timer has three different operating modes, they can be configured to operate as a general timer, an external event counter or as a pulse width measurement device.

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/Event Counter and into which an initial value can be preloaded, and is known as TMR0, TMR1H or TMR1L. 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/Event Counter is to be used, and has the name TMR0C or TMR1C. This device can have the timer clocks configured to come from the internal clock sources. In addition, the timer clock source can also be configured to come from the external timer pins.

The external clock source is used when the Timer/Event Counter is in the event counting mode, the clock source being provided on the external timer pin. The pin has the name TMR0 or TMR1 and is pin-shared with an I/O pin. Depending upon the condition of the T0E or T1E bit in the Timer Control Register, each high to low, or low to high transition on the external timer input pin will increment the Timer/Event Counter by one.

Configuring the Timer/Event Counter Input Clock Source

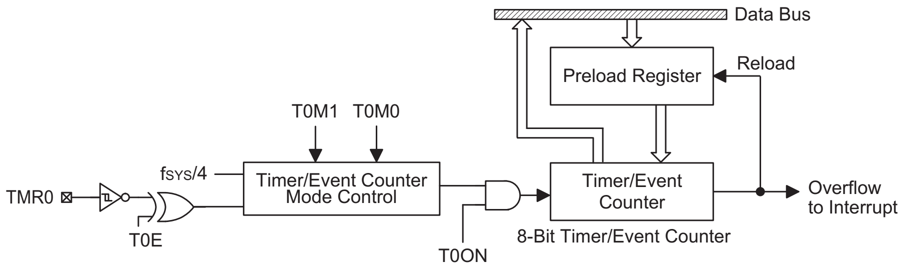
The Timer/Event Counter′s clock can originate from various sources. The system clock source is used when the Timer/Event Counter 0 is in the timer mode or in the pulse width measurement mode. The instruction clock source (system clock source divided by 4) is used when the Timer/Event Counter 1 is in the timer mode or in the pulse width measurement mode. The external clock source is used when the Timer/Event Counter is in the event counting mode, the clock source being provided on the external timer pin, TMR0 or TMR1. Depending upon the condition of the T0E or T1E bit, each high to low, or low to high transition on the external timer pin will increment the counter by one.

Timer Register – TMR0, TMR1L/TMR1H

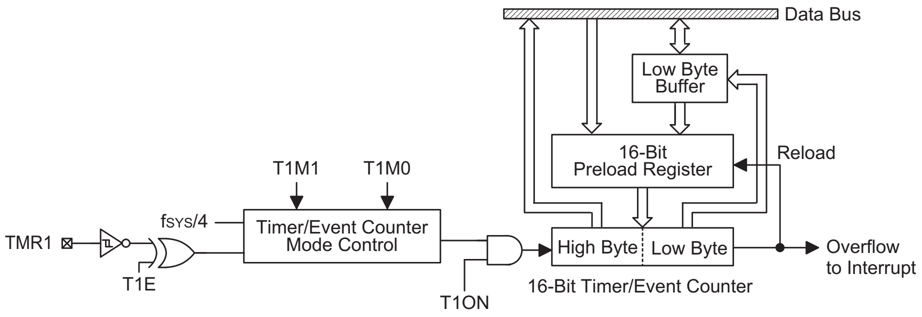
The timer registers are special function registers located in the Special Purpose RAM Data Memory and are the places where the actual timer values are stored. For 8-bit Timer/Event Counter 0, this register is known as TMR0. For 16-bit Timer/Event Counter 1, the timer registers are known as TMR1L and TMR1H. The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH for the 8-bit timer or FFFFH for the 16-bit timer at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reset with the initial preload register value and continue counting.

To achieve a maximum full range count of FFH for the 8-bit timer or FFFFH for the 16-bit timer, the preload registers must first be cleared to all zeros. It should be noted that after power-on, the preload register will be in an unknown condition. Note that if the Timer/Event Counter is switched off and data is written to its preload registers, this data will be immediately written into the actual timer registers. However, if the Timer/Event Counter is enabled and counting, any new data written into the preload data registers during this period will remain in the preload registers and will only be written into the timer registers the next time an overflow occurs.

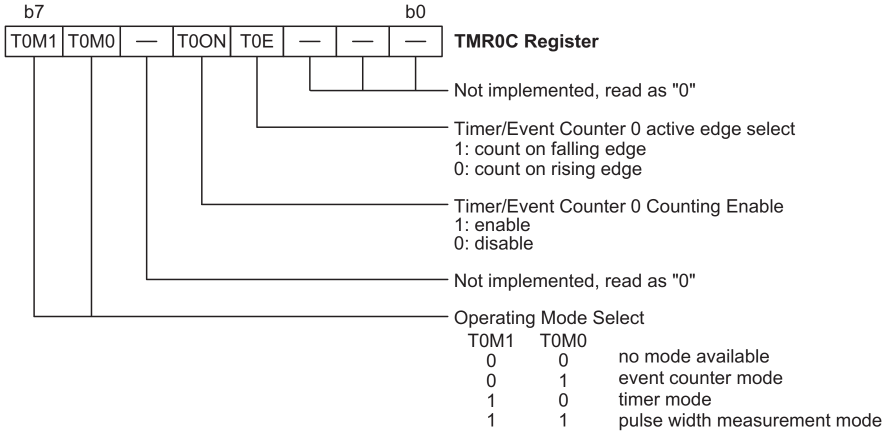
For the 16-bit Timer/Event Counter which has both low byte and high byte timer registers, accessing these registers is carried out in a specific way. It must be note when using instructions to preload data into the low byte timer register, namely TMR1L, the data will only be placed in a low byte buffer and not directly into the low byte timer register. The actual transfer of the data into



8-bit Timer/Event Counter 0 Structure



16-bit Timer/Event Counter 1 Structure



b7							b0	
T0M1	T0M0	—	T0ON	T0E	—	—	—	**TMR0C Register**

Not implemented, read as "0"

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

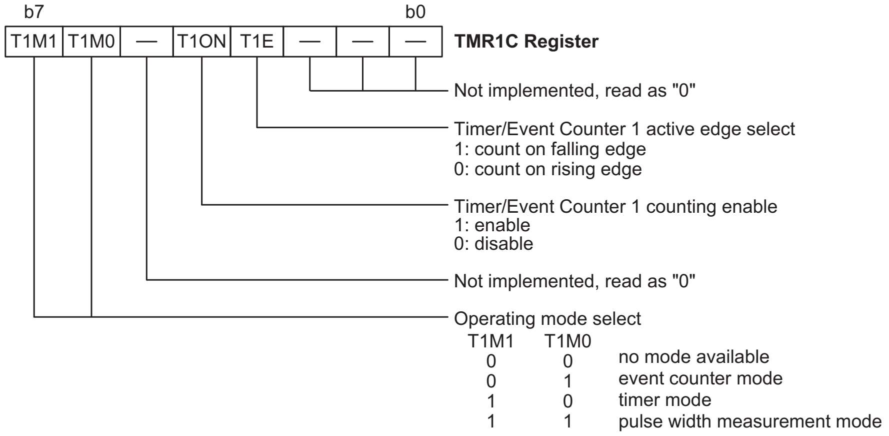
Timer/Event Counter 0 Counting Enable
1: enable
0: disable

Not implemented, read as "0"

Operating Mode Select

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

Timer/Event Counter 0 Control Register



b7							b0	
T1M1	T1M0	—	T1ON	T1E	—	—	—	**TMR1C Register**

Not implemented, read as "0"

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

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

Not implemented, read as "0"

Operating mode select

T1M1	T1M0	
0	0	no mode available
0	1	event counter mode
1	0	timer mode
1	1	pulse width measurement mode

Timer/Event Counter 1 Control Register

the low byte timer register is only carried out when a write to its associated high byte timer register, namely TMR1H, is executed. On the other hand, using instructions to preload data into the high byte timer register will result in the data being directly written to the high byte timer register. At the same time the data in the low byte buffer will be transferred into its associated low byte timer register. For this reason, the low byte timer register should be written first when preloading data into the 16-bit timer registers. It must also be noted that to read the contents of the low byte timer register, a read to the high byte timer register must be executed first to latch the contents of the low byte timer register into its associated low byte buffer. After this has been done, the low byte timer register can be read in the normal way. Note that reading the low byte timer register will result in reading the previously latched contents of the low byte buffer and not the actual contents of the low byte timer register.

Timer Control Register – TMR0C/TMR1C

The flexible features of the Holtek microcontroller Timer/Event Counters enable them to operate in three different modes, the options of which are determined by the contents of their respective control register. For devices are two timer control registers known as TMR0C, TMR1C . It is the timer control register together with its corresponding timer registers that control the full operation of the Timer/Event Counters. Before the timers can be used, it is essential that the appropriate timer control register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialization.

To choose which of the three modes the timer is to operate in, either in the timer mode, the event counting mode or the pulse width measurement mode, bits 7 and 6 of the Timer Control Register, which are known as the bit pair T0M1/T0M0 or T1M1/T1M0 respectively, depending upon which timer is used, must be set to the required logic levels. The timer-on bit, which is bit 4 of the Timer Control Register and known as T0ON or T1ON, depending upon which timer is used, 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. If the timer is in the event count or pulse width measurement mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as T0E or T1E, depending upon which timer is used.

Configuring the Timer Mode

In this mode, the Timer/Event Counter can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter overflows. To operate in this mode, the Operating Mode Select bit pair, T0M1/T0M0 or T1M1/T1M0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode Select Bits for the Timer Mode

Bit7	Bit6
1	0

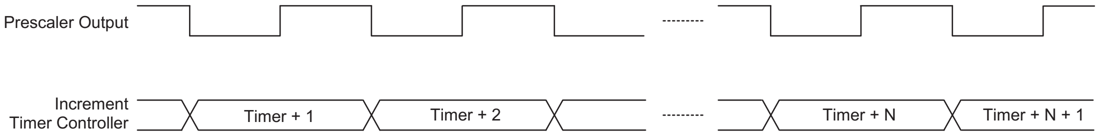
In this mode the internal clock, $f_{SYS}/4$ is used as the internal clock for the Timer/Event Counters. After the other bits in the Timer Control Register have been setup, the enable bit T0ON or T1ON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run.Each time an internal clock cycle occurs, the Timer/Event Counter increments by one. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the Interrupt Control Register, INTC0, is reset to zero.

Configuring the Event Counter Mode

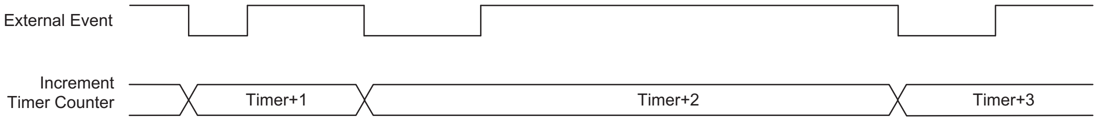
In this mode, a number of externally changing logic events, occurring on the external timer pin, can be recorded by the Timer/Event Counter. To operate in this mode, the Operating Mode Select bit pair, T0M1/T0M0 or T1M1/T1M0, in the Timer Control Register must be set to the correct value as shown.

Control Register Operating Mode Select Bits for the Event Counter Mode

Bit7	Bit6
0	1



Timer Mode Timing Chart



Event Counter Mode Timing Chart

In this mode, the external timer pin, TMR0 or TMR1, is used as the Timer/Event Counter clock source, however it is not divided by the internal prescaler. After the other bits in the Timer Control Register have been setup, the enable bit T0ON or T1ON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. If the Active Edge Select bit T0E or T1E, which is bit 3 of the Timer Control Register, is low, the Timer/Event Counter will increment each time the external timer pin receives a low to high transition. If the Active Edge Select bit is high, the counter will increment each time the external timer pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the Interrupt Control Register, INTC0, is reset to zero.

As the external timer pin is shared with an I/O pin, to ensure that the pin is configured to operate as an event counter input pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the Event Counting Mode, the second is to ensure that the port control register configures the pin as an input. It should be noted that in the event counting mode, even if the microcontroller is in the Power Down Mode, the Timer/Event Counter will continue to record externally changing logic events on the timer input pin. As a result when the timer overflows it will generate a timer interrupt and corresponding wake-up source.

Configuring the Pulse Width Measurement Mode

In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin. To operate in this mode, the Operating Mode Select bit pair, T0M1/T0M0 or T1M1/T1M0, in the Timer Control Register must be set to the correct valueas shown.

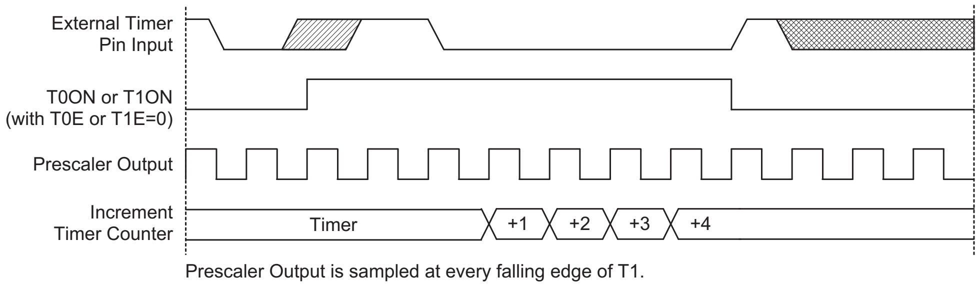
	Bit7	Bit6
Control Register Operating Mode Select Bits for the Pulse Width Measurement Mode	1	1

In this mode the internal clock, $f_{SYS}/4$ is used as the internal clock for the Timer/Event Counters. After the other bits in the Timer Control Register have been setup, the enable bit T0ON or T1ON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter, however it will not actually start counting until an active edge is received on the external timer pin.

If the Active Edge Select bit T0E or T1E, which is bit 3 of the Timer Control Register, is low, once a high to low transition has been received on the external timer pin, TMR0 or TMR1, the Timer/Event Counter will start counting until the external timer pin returns to its original high level. At this point the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. If the Active Edge Select bit is high, the Timer/Event Counter will begin counting once a low to high transition has been received on the external timer pin and stop counting when the external timer pin returns to its original low level. As before, the enable bit will be automatically reset to zero and the Timer/Event Counter will stop counting. It is important to note that in the Pulse Width Measurement Mode, the enable bit is automatically reset to zero when the external control signal on the external timer pin returns to its original level, whereas in the other two modes the enable bit can only be reset to zero under program control.

The residual value in the Timer/Event Counter, which can now be read by the program, therefore represents the length of the pulse received on the external timer pin. As the enable bit has now been reset, any further transitions on the external timer pin will be ignored. Not until the enable bit is again set high by the program can the timer begin further pulse width measurements. In this way, single shot pulse measurements can be easily made.

It should be noted that in this mode the Timer/Event Counter is controlled by logical transitions on the external timer pin and not by the logic level. When the Timer/Event Counter is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by



Pulse Width Measure Mode Timing Chart

ensuring that the Timer/Event Counter Interrupt Enable bit in the Interrupt Control Register, INTC0, is reset to zero.

As the external timer pin is shared with an I/O pin, to ensure that the pin is configured to operate as a pulse width measurement pin, two things have to happen. The first is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the Pulse Width Measurement Mode, the second is to ensure that the port control register configures the pin as an input.ot by the logic level.

I/O Interfacing

The Timer/Event Counter, when configured to run in the event counter or pulse width measurement mode, require the use of the external TMR0 and TMR1 pins for correct operation. As these pins are shared pins they must be configured correctly to ensure they are setup for use as Timer/Event Counter inputs and not as a normal I/O pins. This is implemented by ensuring that the mode select bits in the Timer/Event Counter control register, select either the event counter or pulse width measurement mode. Additionally the Port Control Register bits for these pins must be set high to ensure that the pin is setup as an input. Any pull-high resistor configuration option on these pins will remain valid even if the pin is used as a Timer/Event Counter input.

Programming Considerations

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

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

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

Timer Program Example

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

```
org 04h             ; USB interrupt vector
reti
org 08h             ; Timer/Event Counter interrupt vector
jmp tmr0int         ; jump here when Timer0 overflows
:
org 20h             ; main program
;internal Timer/Event Counter 0 interrupt routine
Tmr0int:
:
; Timer/Event Counter 0 main program placed here
:
reti
:
:
begin:
;setup Timer registers
mov a,09bh          ; setup Timer preload value
mov tmr0,a;
mov a,080h          ; setup Timer control register
mov tmr0c,a         ; timer mode
; setup interrupt register
mov a,005h          ; enable master interrupt and timer interrupt
 INTC0,a
set tmr0c.4         ; start Timer/Event Counter - note mode bits must be previously setup
```

Interrupts

Interrupts are an important part of any microcontroller system. When an internal function such as a Timer/Event Counter overflow or a USB interrupt occur or an external event or SPI/I^{2}C interrupt occur , their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The external interrupt is controlled by the action of the external interrupt pin, while the internal interrupts are controlled by the Timer/Event Counter overflow, USB interrupt or reception.

Interrupt Register

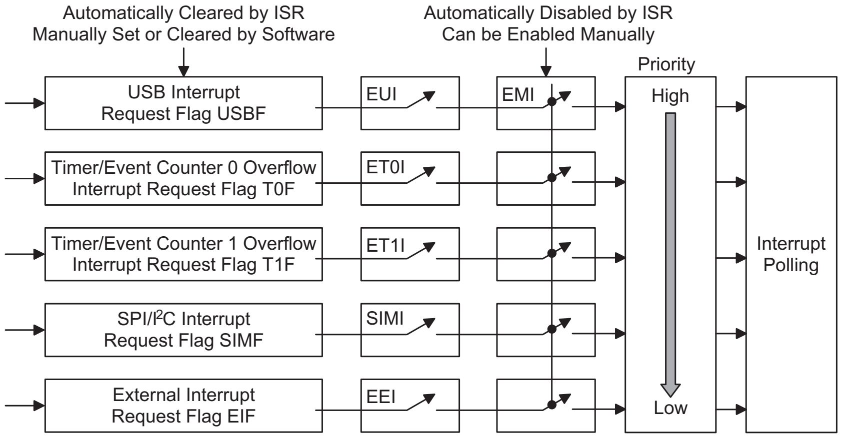
Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by two interrupt control registers. By controlling the appropriate enable bits in these registers each individual interrupt can be enabled or disabled. Also when an interrupt occurs, the corresponding request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

Interrupt Operation

When an interrupt occurs, if their appropriate interrupt enable bit is set, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a JMP statement which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI statement, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

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

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



Interrupt Structure

Interrupt Priority

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

Interrupt Source	Priority	Vector
USB Interrupt	1	0004H
Timer/Event Counter 0 Overflow Interrupt	2	0008H
Timer/Event Counter 1 Overflow Interrupt	3	000CH
SPI/I^2C Interrupt	4	0010H
External Interrupt	5	0014H

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

USB Interrupt

The USB interrupts are triggered by the following USB events causing the related interrupt request flag, USBF, to be set.

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

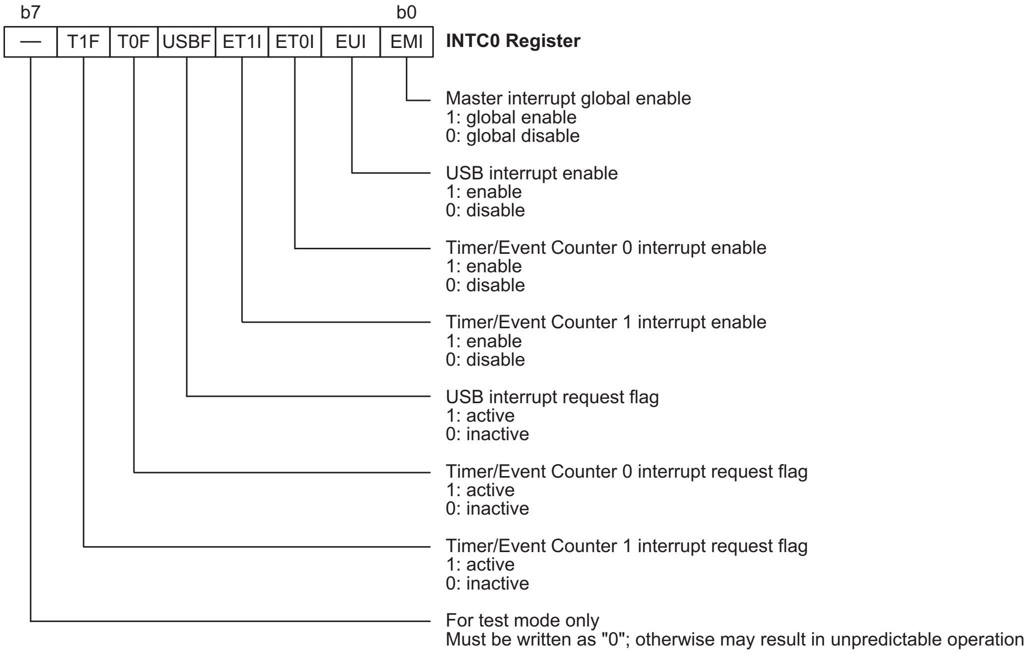
When the interrupt is enabled, the stack is not full and the USB interrupt is active, a subroutine call to location 04H will occur. The interrupt request flag, USBF, and the EMI bit will be cleared to disable other interrupts.

When the PC Host accesses the FIFO of the device, the corresponding request bit, USR, is set, and a USB interrupt is triggered. So the user can easy determine which FIFO has been accessed. When the interrupt has been served, the corresponding bit should be cleared by firmware. When the device receive a USB Suspend signal from Host PC, the suspend line (bit0 of USC) is set and a USB interrupt is also triggered.

Also when device receive a Resume signal from Host PC, the resume line (bit3 of USC) is set and a USB interrupt is triggered.

Timer/Event Counter Interrupt

For a Timer/Event Counter interrupt to occur, the global interrupt enable bit, EMI, and the corresponding timer interrupt enable bit, ET0I/ET1I, must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter interrupt request flag, T0F/T1F, is set, a situation that will occur when the 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 timer interrupt vector at location 08H/0CH, will take place. When the interrupt is serviced, the timer interrupt request flag, T0F/T1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.



Master interrupt global enable
1: global enable
0: global disable

USB interrupt enable
1: enable
0: disable

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

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

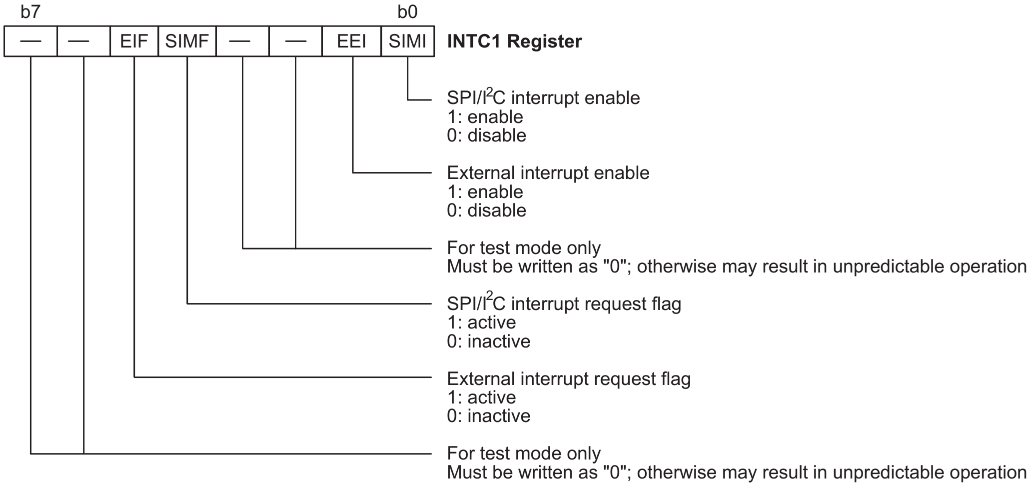
USB interrupt request flag
1: active
0: inactive

Timer/Event Counter 0 interrupt request flag
1: active
0: inactive

Timer/Event Counter 1 interrupt request flag
1: active
0: inactive

For test mode only
Must be written as "0"; otherwise may result in unpredictable operation

INTC0 Register



SPI/I^{2}C interrupt enable
1: enable
0: disable

External interrupt enable
1: enable
0: disable

For test mode only
Must be written as "0"; otherwise may result in unpredictable operation

SPI/I^{2}C interrupt request flag
1: active
0: inactive

External interrupt request flag
1: active
0: inactive

For test mode only
Must be written as "0"; otherwise may result in unpredictable operation

INTC0 Register

SPI/I^{2}C Interface Interrupt

For an SPI/I^{2}C interrupt to occur, the global interrupt enable bit, EMI, and the corresponding interrupt enable bit, SIMI must be first set. An actual SPI/I^{2}C interrupt will take place when the SPI/I^{2}C interrupt request flag, SIMF, is set, a situation that will occur when a byte of data has been transmitted or received by the SPI/I^{2}C interface or when an I^{2}C address match occurs. When the interrupt is enabled, the stack is not full and a byte of data has been transmitted or received by the SPI/I^{2}C interface or an I^{2}C address match occurs, a subroutine call to the SPI/I^{2}C interrupt vector, will take place. When the interrupt is serviced, the SPI/I^{2}C request flag, SIMF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

External Interrupt

For an external interrupt to occur, the global interrupt enable bit, EMI, and external interrupt enable bit, EEI, must first be set. An actual external interrupt will take place when the external interrupt request flag, EIF, is set, a situation that will occur when a negative edge transition appears on the INT pin. The external interrupt pin is pin-shared with the I/O pin PB5 and can only be configured as an external interrupt pin if its corresponding external interrupt enable bit in the INTC1 register has been set. The pin must also be setup as an input by setting the corresponding PBC.5 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 flags, EIF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on this pin will remain valid even if the pin is used as an external interrupt input.

Programming Considerations

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

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

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

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

Reset and Initialisation

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

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

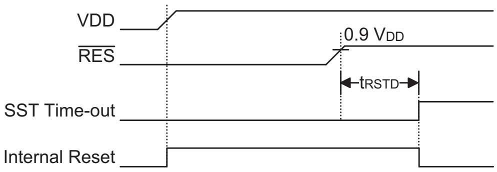
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, similar to the RES reset is implemented in situations where the power supply voltage falls below a certain threshold.

Reset Functions

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

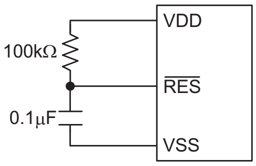
- Power-on Reset
  The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. All the I/O port and port control registers will power up in a high condition ensuring that all pins will be first set to inputs.
  Although the microcontroller has an internal RC reset function, if the VDD power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing a

proper reset operation. In such cases it is recommended that an external RC network is connected to the $\overline{\text{RES}}$ pin, whose additional time delay will ensure that the $\overline{\text{RES}}$ pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the $\overline{\text{RES}}$ line reaches a certain voltage value, the reset delay time t_{RSTD} is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Timer.



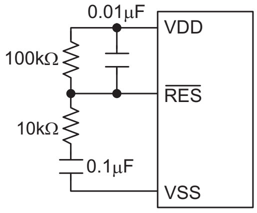
Power-On Reset Timing Chart

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



Basic Reset Circuit

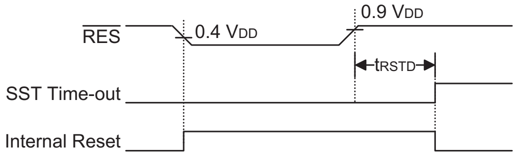
For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.



Enhanced Reset Circuit

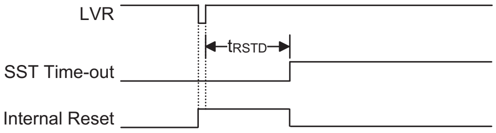
More information regarding external reset circuits is located in Application Note HA0075E on the Holtek website.

- $\overline{\text{RES}}$ Pin Reset
  This type of reset occurs when the microcontroller is already running and the $\overline{\text{RES}}$ pin is forcefully pulled low by external hardware such as an external switch. In this case as in the case of other reset, the Program Counter will reset to zero and program execution initiated from this point. Note that as the external reset pin is also pin-shared with PA7, if it is to be used as a reset pin, the correct reset configuration option must be selected.



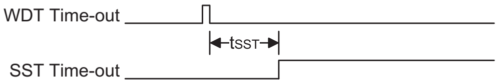
$\overline{\text{RES}}$ Reset Timing Chart

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



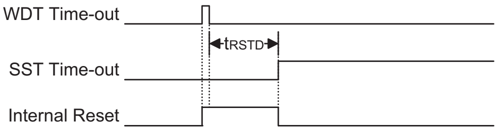
Low Voltage Reset Timing Chart

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



WDT Time-out Reset during Power Down Timing Chart

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



WDT Time-out Reset during Normal Operation Timing Chart

Reset Initial Conditions

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

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

Note: "u" stands for unchanged

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

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

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

Register	Reset (Power-on)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (HALT)*	USB Reset (Normal)	USB Reset (HALT)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
BP	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000H	0000H	0000H	0000H	0000H	0000H	0000H
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
WDTS	1000 0111	1000 0111	1000 0111	1000 0111	uuuu uuuu	1000 0111	1000 0111
STATUS	--00 xxxx	--1u uuuu	--00 uuuu	--00 uuuu	--11 uuuu	--uu uuuu	--01 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu	-000 0000	-000 0000
INTC1	--00 --00	--00 --00	--00 --00	--00 --00	--uu --uu	--00 --00	--00 --00
TMR0	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu	00-0 1000	00-0 1000
TMR1H	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR1L	xxxx xxxx	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMR1C	00-0 1---	00-0 1---	00-0 1---	00-0 1---	uu-u u---	uu-u u---	uu-u u---
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111

Register	Reset (Power-on)	WDT Time-out (Normal Operation)	$\overline{\text{RES}}$ Reset (Normal Operation)	$\overline{\text{RES}}$ Reset (HALT)	WDT Time-out (HALT)*	USB Reset (Normal)	USB Reset (HALT)
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PE	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PEC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu	1111 1111	1111 1111
PF	---- --11	---- --11	---- --11	---- --11	---- --uu	---- --11	---- --11
PFC	---- --11	---- --11	---- --11	---- --11	---- --uu	---- --11	---- --11
TBHP	---x xxxx	---u uuuu	---u uuuu	---u uuuu	---u uuuu	---u uuuu	---u uuuu
USC	11xx 0000	11xx xu0x	11xx 0000	11xx 0000	uuxx xuux	uu00 0u00	uu00 0u00
SCC	0000 0000	uuu0 uuu0	0000 0000	0000 0000	uuuu uuxu	0uu0 u000	0uu0 u000
USR	0000 0000	u0uu 0000	0000 0000	0000 0000	u0uu 0000	u1uu 0000	u1uu 0000
USB_STAT	--xx 0000	--xx 0000	--xx 0000	--xx 0000	--xx 0000	--xx 0000	--xx 0000
PIPE_CTRL	0000 1110	uuuu uuuu	0000 1110	0000 1110	uuuu uuuu	0000 1110	0000 1110
AWR	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
PIPE	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
STALL	0000 1110	uuuu uuuu	0000 1110	0000 1110	uuuu uuuu	0000 1110	0000 1110
SIES	0x0x x000	uuuu uuuu	0x0x x000	0x0x x000	uuuu uuuu	0x0x x000	0x0x x000
MISC	0000 0000	uuuu uuuu	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
ENDPT_EN	0000 0111	uuuu uuuu	0000 0111	0000 0111	uuuu uuuu	0000 0111	0000 0111
FIFO0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
FIFO3	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu	0000 0000	0000 0000
SIMCTL0	1110 0000	1110 0000	1110 0000	1110 0000	uuuu uuu0	1110 0000	1110 0000
SIMCTL1	1000 0001	1000 0001	1000 0001	1000 0001	xxxu uxux	1000 0001	1000 0001
SIMDR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu	xxxx xxxx	xxxx xxxx
SIMAR/ SIMCTL2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	0000 0000	0000 0000
LCDC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu	uuuu uuuu	uuuu uuuu

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

Oscillator

The clock source for these devices is provided by an integrated oscillator requiring no external components. This oscillator has two fixed frequencies of either 6MHz, or 12MHz, the selection of which is made by the SYSCLK bit in the SCC register.

Watchdog Timer Oscillator

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

Power Down Mode and Wake-up

Power Down Mode

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

Entering the Power Down Mode

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

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

Standby Current Considerations

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

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

If the configuration options have enabled the Watchdog Timer internal oscillator then this will continue to run when in the Power Down Mode and will thus consume some power. For power sensitive applications it may be therefore preferable to use the system clock source for the Watchdog Timer. If any I/O pins are configured as A/D analog inputs using the channel configuration bits in the ADCR register, then the A/D converter will be turned on and a certain amount of power will be consumed. It may be therefore desirable before entering te Power Down Mode to ensure that the A/D converter is powered down by ensuring that any A/D input pins are setup as normal logic inputs with pull-high resistors.

Wake-up

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

- An external reset
- An external all I/O ports
- A system interrupt
- A WDT overflow

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

Each pin on all I/O ports can be setup via an individual configuration option to permit a negative transition on the pin to wake-up the system. When a I/O ports pin wake-up occurs, the program will resume execution at the instruction following the ″HALT″ instruction.

If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the ″HALT″ instruction. In this situation, the interrupt which woke-up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set to ″1″ before entering the Power Down Mode, the wake-up function of the related interrupt will be disabled.

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

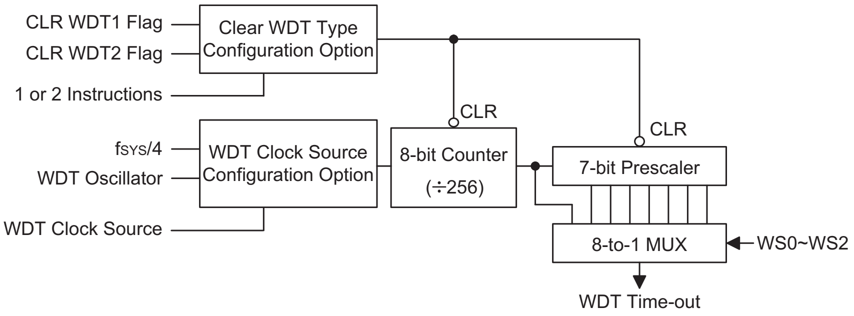
Watchdog Timer

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

Once the internal WDT oscillator (RC oscillator normally with a period of 78μs) is selected, it is first divided by 256 (8-stages) to get the nominal time-out period of approximately 20ms. This time-out period may vary with temperature, VDD and process variations. By using the WDT prescaler, longer time-out periods can be realized. Writing data to WS2, WS1, WS0 (bit 2, 1, 0 of the WDTS) can give different time-out periods. If WDTS2, WDTS1, WDTS0 are all equal to ″1″, the division ratio is up to 1:128, and the maximum time-out period is 2.6s.

If the WDT oscillator is disabled, the WDT clock source may still come from the instruction clock and operate in the same manner except that in the Power down Mode state the WDT may stop counting and lose its protecting purpose. In this situation the WDT logic can be restarted by external logic. The high nibble and bit 3 of the WDTS are reserved for user defined flags, which can be used to indicate some specified status.

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



Watchdog Timer

Bit No.	Label	Function
0 1 2	WDTS0 WDTS1 WDTS2	Watchdog Timer division ratio selection bits Bit 2,1,0 = 000, division ratio = 1:1 Bit 2,1,0 = 001, division ratio = 1:2 Bit 2,1,0 = 010, division ratio = 1:4 Bit 2,1,0 = 011, division ratio = 1:8 Bit 2,1,0 = 100, division ratio = 1:16 Bit 2,1,0 = 101, division ratio = 1:32 Bit 2,1,0 = 110, division ratio = 1:64 Bit 2,1,0 = 111, division ratio = 1:128
3	WDTS3	Bit3=1, USBPDP, and USBPDN connected to 510kΩ pull-high resistor Bit3=0, No pull-high - default at MCU reset
4~6	—	Not used
7	—	Must be set ″1″

WDTS Register

Suspend Wake-Up and Remote Wake-Up

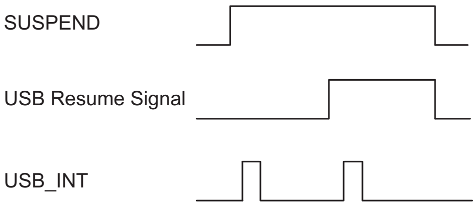
If there is no signal on the USB bus for over 3ms, the device will go into a suspend mode. The Suspend line (bit 0 of the USC register) will be set to ″1″ and a USB interrupt is triggered to indicate that the devices should jump to the suspend state to meet the 500μA USB suspend current spec.

In order to meet the 500μA suspend current, the firmware should disable the USB clock by clearing the USBCKEN bit which is bit3 of the SCC register to ″0″. The suspend current is 400μA.

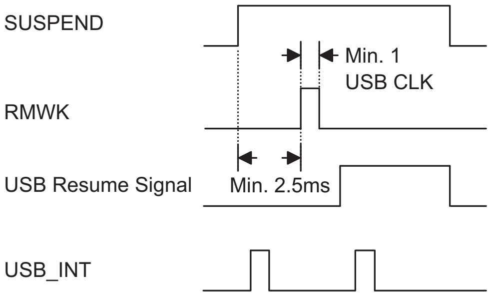
The user can further decrease the suspend current to 250μA by setting the BGOFF bit which is bit4 of the SCC register. If in the USB mode set this bit LVR OPT must disable.

When the resume signal is sent out by the host, the devices will wake up the MCU with a USB interrupt and the Resume line (bit 3 of the USC register) is set. In order to make the device function properly, the firmware must set the USBCKEN (bit 3 of the SCC register) to ″1″ and clear the BGOFF (bit4 of the SCC register). Since the Resume signal will be cleared before the Idle signal is sent out by the host, the Suspend line (bit 0 of the USC register) will be set to ″0″. So when the MCU is detecting the Suspend line (bit0 of USC register), the Resume line condition should be noted and taken into consideration.

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



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



Configure a PS2 Device

The devices can also be configured as a USB interface or PS2 interface device, by configuring MODE_CTRL 0~1 (bit 4~5 of the USR register). If MODE_CTRL 0=1, and MODE_CTRL 1=0, the device will be configured as a PS2 interface, pin USBPDN is configured as a PS2 Data pin and USBPDP is configured as a PS2 Clk pin. The user can read or write to the PS2 Data or PS2 Clk pin by accessing the corresponding bit PS2_DAI (bit 4 of the USC register), PS2_CKI (bit 5 of the USC register), PS2_DAO (bit 6 of the USC register) and SP2_CKO (bit 7 of the USC register) respectively.

The user should make sure that in order to read the data properly, the corresponding output bit must be set to ″1″. For example, if it is desired to read the PS2 Data by reading PS2_DAI, then PS2_DAO should set to ″1″. Otherwise it is always read as ″0″.

If MODE_CTRL 0=0, and MODE_CTRL 1=1, the device is configured as a USB interface. Both the USBPDN and USBPDP is driven by the SIE of the HT82B60R. The user can only write or read the USB data through the corresponding FIFO. Both the MODE_CTRL 0~1 default is ″0″.

USB Interface

The device includes a USB interface function permitting simplified implementation of USB type products. As the full scope of the USB specification is beyond the reach of this document, users are request to consult external documentation for further details.

USB Registers

There are several registers used for the USB function. The AWR register contains the current address and a remote wake up function control bit. The initial value of AWR is ″00H″. The address value extracted from the USB command is not to be loaded into this register until the SETUP stage is completed.

Bit No.	Label	R/W	Function
0	WKEN	W	Remote wake-up enable/disable
7~1	AD6~AD0	W	USB device address

AWR (42H) Register

Bit No.	Label	R/W	Function
0	SUSPEND	R	Read only, USB suspend indication. When this bit is set to ″1″ (set by the SIE), it indicates that the USB bus has entered the suspend mode. The USB interrupt is also triggered on any change of this bit.
1	RMOT_WK	W	USB remote wake up command. Set by the MCU to force the USB host to leave the suspend mode. When this bit is set to ″1″, a 2μs delay for clearing this bit to ″0″ is needed to insure the RMWK command is accepted by SIE.
2	URST_FLAG	R/W	USB reset indication. This bit is set/cleared by the USB SIE. This bit is used to detect which bus (PS2 or USB) is attached. When the URST is set to ″1″, this indicates that a USB reset has occurred (the attached bus is USB) and a USB interrupt will be initialised.
3	RESUME_O	R	USB resume indication. When the USB leaves the suspend mode, this bit is set to ″1″ (set by the SIE). This bit will appear for 20ms waiting for the MCU to detect. When the RESUME is set by the SIE, an interrupt will be generated to wake-up the MCU. In order to detect the suspend state, the MCU should set the USBCKEN and clear SUSP2 (in the SCC register) to enable the SIE detect function. The RESUME will be cleared while SUSP is going to ″0″. When the MCU is detecting the SUSP, the condition of RESUME (which wakes-up the MCU) should be noted and taken into consideration.
4	PS2_DAI	R	Read only, USBPDN/DATA input
5	PS2_CKI	R	Read only, USBPDP/CLK input
6	PS2_DAO	W	Data for driving the USBPDN/DATA pin when working under 3D PS2 mouse function. (Default=″1″)
7	PS2_CKO	W	Data for driving the USBPDP/CLK pin when working under 3D PS2 mouse function. (Default=″1″)

USC (20H) Register

The USR (USB endpoint interrupt status register) register is used to indicate which endpoint is accessed and to select the serial bus, PS2 or USB. The endpoint request flags, EP0IF, EP1IF, EP2IF and EP3IF, are used to indicate which endpoints are accessed. If an endpoint is accessed, the related endpoint request flag will be set to ″1″ and the USB interrupt will occur, if the USB interrupt is enabled and the stack is not full. When the active endpoint request flag is served, the endpoint request flag has to be cleared to ″0″.

Bit No.	Label	R/W	Function
0	EP0_INT	R/W	When this bit is set to ″1″ (set by the SIE), it indicates that endpoint 0 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
1	EP1_INT	R/W	When this bit is set to ″1″ (set by the SIE), it indicates that endpoint 1 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
2	EP2_INT	R/W	When this bit is set to ″1″ (set by the SIE), it indicates that endpoint 2 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
3	EP3_INT	R/W	When this bit is set to ″1″ (set by the SIE), it indicates that endpoint 3 is accessed and a USB interrupt will occur. When the interrupt has been served, this bit should be cleared by firmware.
4 5	MODE_CTRL0 MODE_CTRL1	R/W	00 : Non-USB mode, turn-off V33O, both USBPDP and USBPDN can be read and write - default 01 : Non-USB mode, has 200Ω between VDD and V33O, both USBPDP and USBPDN can be read and write 10 : USB mode, 1.5kΩ between USBPDN and V33O, V33O output 3.3V, both USBPDP and USBPDN are read only 11 : Non-USB mode, V33O output 3.3V, both USBPDP and USBPDN can be read and write
6	—	—	Reserved
7	USB_flag	R/W	This flag is used to indicate that the MCU is in the USB mode - Bit=1 This bit is R/W by FW and will be cleared to ″0″ after power-on reset - Default=″0″

USR (21H) Register

There is a system clock control register implemented to select the clock used in the MCU. This register consists of the USB clock control bit, USBCKEN, and a system clock selection bit, SYSCLK. The PS2 mode indicate bit, PS2_flag, and a system clock adjust control bit, CLK_adj.

Bit No.	Label	R/W	Function
0	—	—	Reserved bit - read as ″0″
1	—	—	Reserved bit - read as ″0″
2	D_SR	R/W	When set to ″1″, a 7.5kΩ resistor will be connected between VDD and USBPDN, and the 1.5kΩ between USBPDN and V33O will be removed, default value is ″0″.
3	USBCKEN	R/W	USB clock control bit. When this bit is set to ″1″, it indicates that the USB clock is enabled. Otherwise, the USB clock is turned-off (default ″0″).
4	BGOFF	R/W	When set to ″1″, the Band-gap circuit will be switched off. Default value is ″0″.
5	PS2_flag	R/W	This flag is used to indicate that the MCU is in the PS2 mode (Bit=1). This bit is R/W by FW and will be cleared to ″0″ after power-on reset (default ″0″).
6	SYSCLK	R/W	This bit is used to specify the system oscillator frequency used by the MCU. If an Integrated 6MHz oscillator is used, this bit should be set to ″1″. If an Integrated 12MHz oscillator is used, this bit should be cleared to ″0″ (default).
7	CLK_adj	R/W	This bit is used to adjust the system clock for the USB mode for temperature changes. In the Power-down Mode this bit should be set high to reduce power consumption. 0: enable (default) 1: disable

SCC (22H) Register

STALL and PIPE, PIPE_CTRL, Endpt_EN Registers

The PIPE register represents whether the corresponding endpoint is accessed by the host or not. After an ACT_EN signal has been sent out, the MCU can check which endpoint had been accessed. This register is set only after the a time when the host is accessing the corresponding endpoint.

The STALL register shows whether the corresponding endpoint works or not. As soon as the endpoint works improperly, the corresponding bit must be set.

The PIPE_CTRL Register is used for configuring the IN (Bit=1) or OUT (Bit=0) Pipe. The default is define IN pipe. Bit0 (DATA0) of the PIPE_CTRL Register is used to set the data toggle of any endpoint (except endpoint 0) using data toggles to the value DATA0. Once the user wants any endpoint (except endpoint 0) using data toggles to the value DATA0. the user can output a LOW pulse to this bit. The LOW pulse period must at least 10 instruction cycles.

The Endpt_EN Register is used to enable or disable the corresponding endpoint (except endpoint 0) Enable Endpoint (Bit=1) or disable Endpoint (Bit=0)

The bitmaps are list are shown in the following table:

Register Name	R/W	Register Address	Bit7~Bit4 Reserved	Bit 3	Bit 2	Bit 1	Bit 0	Default Value
Endpt_EN	R/W	47H	—	Pipe 3	Pipe 2	Pipe 1	Pipe 0	00001111
PIPE_CTRL	R/W	41H	—	Pipe 3	Pipe 2	Pipe 1	DATA0	00001110
STALL	R/W	43H	—	Pipe 3	Pipe 2	Pipe 1	Pipe 0	00001110
PIPE	R	44H	—	Pipe 3	Pipe 2	Pipe 1	Pipe 0	00000000

PIPE_CTRL (41H), STALL (43H), PIPE (44H) and Endpt_EN (47H) Registers

The USB_STAT Register (40H) is used to indicate the present USB signal state.

Bit No.	Function	Read/Write	Register Address
0	EOP	R/W	01000000B
1	J_state	R/W	
2	K_state	R/W	
3	SE0	R/W	
4	SE1	R/W	
5~7	—	—	

USB_STAT (40H) Register Table

Func. Name	R/W	Description
EOP	R/W	This bit is used to indicate the SIE has detected a EOP USB signal in the USB Bus. This bit is set by SIE and cleared by F/W.
J_state	R/W	This bit is used to indicate the SIE has detected a J_state USB signal in the USB Bus. This bit is set by SIE and cleared by F/W.
K_state	R/W	This bit is used to indicate the SIE has detected a K_state USB signal in the USB Bus. This bit is set by SIE and cleared by F/W.
—	—	Unused bit, read as ″0″
SE0	R/W	This bit is used to indicate the SIE has detected a SE0 noise in the USB Bus. This bit is set by SIE and cleared by F/W.
SE1	R/W	This bit is used to indicate the SIE has detected a SE1 noise in the USB Bus. This bit is set by SIE and cleared by F/W.

USB_STAT Function Table

The SIES Register is used to indicate the present signal state in which the SIE receives and also defines whether the SIE has to change the device address automatically.

Bit No.	Function	Read/Write	Register Address
0	Adr_set	R/W	01000001B
1	F0_ERR	R/W	
2~6	—	—	
7	NMI	R/W	

SIES (45H) Register Table

Func. Name	R/W	Description
Adr_set	R/W	This bit is used to configure the SIE to automatically change the device address with the value of the Address+Remote_WakeUp Register. When this bit is set to ″1″ by F/W, the SIE will update the device address with the value of the Address+Remote_WakeUp Register after the PC Host has successfully read the data from the device by the IN operation. The SIE will clear the bit after updating the device address. Otherwise, when this bit is cleared to ″0″, the SIE will update the device address immediately after an address is written to the Address+Remote_WakeUp Register. Default 0.
F0_ERR	R/W	This bit is used to indicate that some errors have occurred when accessing the FIFO0. This bit is set by SIE and cleared by F/W. Default 0
—	—	Unused bit, read as ″0″
NMI	R/W	This bit is used to control whether the USB interrupt is output to the MCU in a NAK response to the PC Host IN or OUT token. Only for Endpoint0 1: has only USB interrupt, data is transmitted to the PC host or data is received from the PC Host 0: always has USB interrupt if the USB accesses FIFO0 Default 0

SIES Function Table

The MISC register combines a command and status to control desired endpoint FIFO action and to show the status of the desired endpoint FIFO. The MISC will be cleared by the USB reset signal.

Bit No.	Label	R/W	Function
0	REQ	R/W	After setting the other status of the desired one in the MISC, endpoint FIFO can be requested by setting this bit to ″1″. After the task is completed, this bit must be cleared to ″0″.
1	TX	R/W	This bit defines the direction of data transferring between the MCU and endpoint FIFO. When the TX is set to ″1″, this means that the MCU wants to write data to the endpoint FIFO. After the task is completed, this bit must be cleared to ″0″ before terminating the request to represent the end of transferring. For a read action, this bit has to be cleared to ″0″ to represent that MCU wants to read data from the endpoint FIFO and has to be set to ″1″ after completion.
2	CLEAR	R/W	Clear the requested endpoint FIFO, even if the endpoint FIFO is not ready.
4 3	SELP1 SELP0	R/W	Defines which endpoint FIFO is selected, SELP1,SELP0: 00: endpoint FIFO0 01: endpoint FIFO1 10: endpoint FIFO2 11: endpoint FIFO3
5	SCMD	R/W	Used to show that the data in the endpoint FIFO is a SETUP command. This bit has to be cleared by firmware. That is to say, even if the MCU is busy, the device will not miss any SETUP commands from the host.
6	READY	R	Read only status bit, this bit is used to indicate that the desired endpoint FIFO is ready for operation.
7	LEN0	R/W	Used to indicate that a 0-sized packet has been sent from a host to the MCU. This bit should be cleared by firmware.

MISC (46H) Register

The MCU can communicate with the endpoint FIFO by setting the corresponding registers, of which the address is listed in the following table. After reading the current data, the next data will show after 2μs, this is used to check the endpoint FIFO status and response to the MISC register, if the read/write action is still going on.

Registers	R/W	Address	Bit7~Bit0
FIFO0	R/W	48H	Data7~Data0
FIFO1	R/W	49H	Data7~Data0
FIFO2	R/W	4AH	Data7~Data0
FIFO3	R/W	4BH	Data7~Data0

There are some timing constrains and usages illustrated here. By setting the MISC register, the MCU can perform reading, writing and clearing actions. There are some examples shown in the following table for endpoint FIFO reading, writing and clearing.

Actions	MISC Setting Flow and Status
Read FIFO0 sequence	00H→01H→delay 2μs, check 41H→read* from FIFO0 register and check not ready (01H)→03H→02H
Write FIFO0 sequence	02H→03H→delay 2μs, check 43H→write* to FIFO0 register and check not ready (03H)→01H→00H
Check whether FIFO0 can be read or not	00H→01H→delay 2μs, check 41H (ready) or 01H (not ready)→00H
Check whether FIFO0 can be written or not	02H→03H→delay 2μs, check 43H (ready) or 03H (not ready)→02H
Write 0-sized packet sequence to FIFO0	02H→03H→delay 2μs, check 43H→01H→00H
Clear FIFO1 sequence	01H→delay 2μs→05H→delay 2μs→00H

Note: *: There is a 2μs time between 2 read actions or between 2 write actions.

Register	Bits	Read/Write	Functions
TBHP	0~4	R	Store current table read bit12~bit8 data

TBHP Register

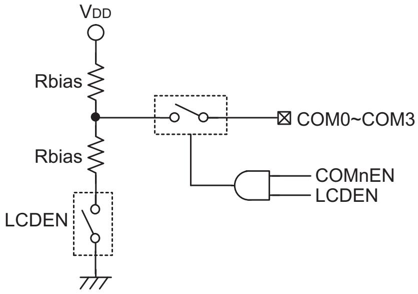
LCD Driver

The devices have the capability of driving external LCD panels. The common pins for LCD driving, COM0~COM3, are pin shared with certain pin on the PC port.

LCD Driver Operation

An external LCD panel can be driven using by configuring certain pins on the PC as common pins and using other output lines as segment pins. The LCD driver function is controlled using the LCDC register which in addition to controlling the overall on/off function also controls the bias voltage setup function. This enables the LCD driver to generate the necessary VDD/2 voltage levels for LCD 1/2 bias operation.

The register bits, control the overall on/off and bias current selection as well as selecting which PC pins are to be used as LCD common drivers.



LCD COM Bias

The LCDEN bit in the LCDC register is the overall master control for the LCD Driver, however this bit is used in conjunction with the COMnEN bits to select which Port C pins are used for LCD driving. Note that the Port Control register does not need to first setup the pins as outputs to enable the LCD driver operation.

LCDEN	COMnEN	PCC	Pin Function	O/P Level
0	X	X	I/O	0 or 1
1	0	X	I/O	0 or 1
1	1	X	COM	$V_{DD}/2$

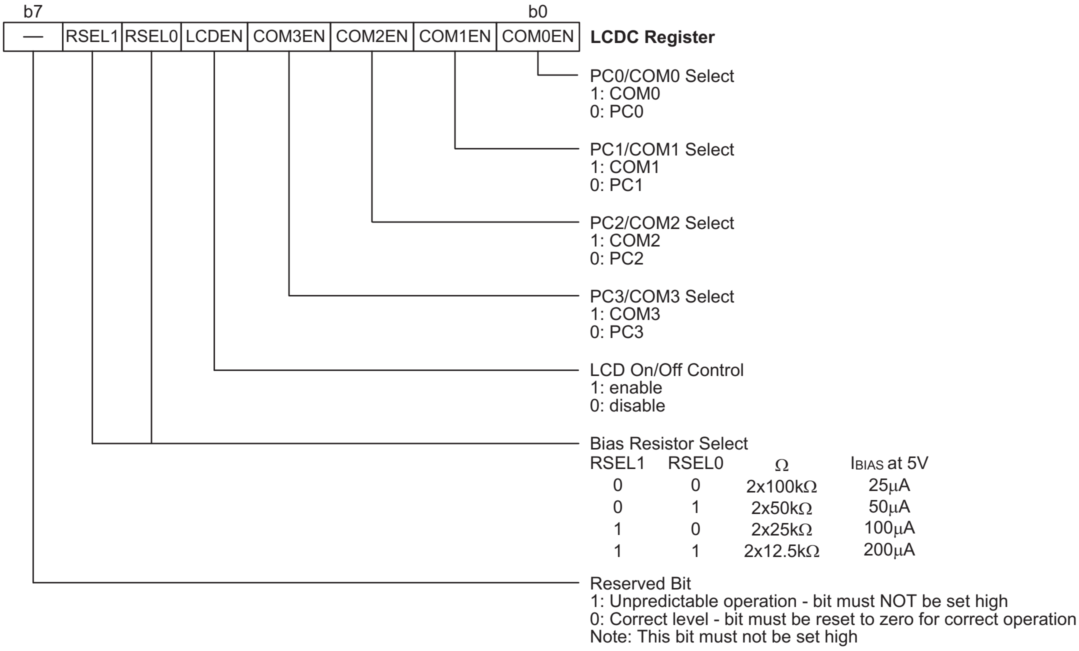
COM Pin Output Control

LCD Bias Control

The LCD COM driver enables a range of selections to be provided to suit the requirement of the LCD panel which is being used. The bias resistor choice is implemented using the RSEL0 and RSEL1 bits in the LCDC register.

Bit	7	6	5	4	3	2	1	0
Name	—	RSEL1	RSEL0	LCDEN	COM3EN	COM2EN	COM1EN	COM0EN
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

LCD Register



PC0/COM0 Select
1: COM0
0: PC0

PC1/COM1 Select
1: COM1
0: PC1

PC2/COM2 Select
1: COM2
0: PC2

PC3/COM3 Select
1: COM3
0: PC3

LCD On/Off Control
1: enable
0: disable

Bias Resistor Select

RSEL1	RSEL0	Ω	IBIAS at 5V
0	0	2x100kΩ	25μA
0	1	2x50kΩ	50μA
1	0	2x25kΩ	100μA
1	1	2x12.5kΩ	200μA

Reserved Bit
1: Unpredictable operation - bit must NOT be set high
0: Correct level - bit must be reset to zero for correct operation
Note: This bit must not be set high

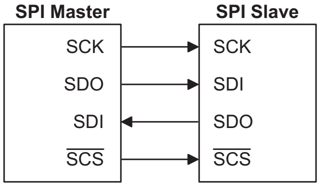
LCDC Register

Serial Interface Function

The device contains a Serial Interface Function, which includes both the four line SPI interface and the two line I^2C interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI or I^2C based hardware such as sensors, Flash or EEPROM memory, etc. The SIM interface pins are pin-shared with other I/O pins therefore the SIM interface function must first be selected using a configuration option. As both interface types share the same pins and registers, the choice of whether the SPI or I^2C type is used is made using a bit in an internal register.

SPI Interface

The SPI interface is often used to communicate with external peripheral devices such as sensors, Flash or EEPROM memory devices etc. Originally developed by Motorola, the four line SPI interface is a synchronous serial data interface that has a relatively simple communication protocol simplifying the programming requirements when communicating with external hardware devices.



SPI Master/Slave Connection

The communication is full duplex and operates as a slave/master type, where the MCU can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, here, as only a single select pin, $\overline{\text{SCS}}$, is provided only one slave device can be connected to the SPI bus.

- SPI Interface Operation
  The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and $\overline{\text{SCS}}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, SCK is the Serial Clock line and $\overline{\text{SCS}}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I^2C function pins, the SPI interface must first be enabled by selecting the SIM enable configuration option and setting the correct bits in the SIMCTL0/SIMCTL2 register. After the SPI configuration option has been configured it can also be additionally disabled or enabled using the SIMEN bit in the SIMCTL0 register. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{\text{SCS}}$ pin only one slave device can be utilised.

The SPI function in this device offers the following features:

- Full duplex synchronous data transfer
- Both Master and Slave modes
- LSB first or MSB first data transmission modes
- Transmission complete flag
- Rising or falling active clock edge
- WCOL and CSEN bit enabled or disable select

The status of the SPI interface pins is determined by a number of factors such as whether the device is in the master or slave mode and upon the condition of certain control bits such as CSEN, SIMEN and $\overline{SCS}$. There are several configuration options associated with the SPI interface. One of these is to enable the SIM function which selects the SIM pins rather than normal I/O pins. Note that if the configuration option does not select the SIM function then the SIMEN bit in the SIMCTL0 register will have no effect. Another two SIM configuration options determine if the CSEN and WCOL bits are to be used.

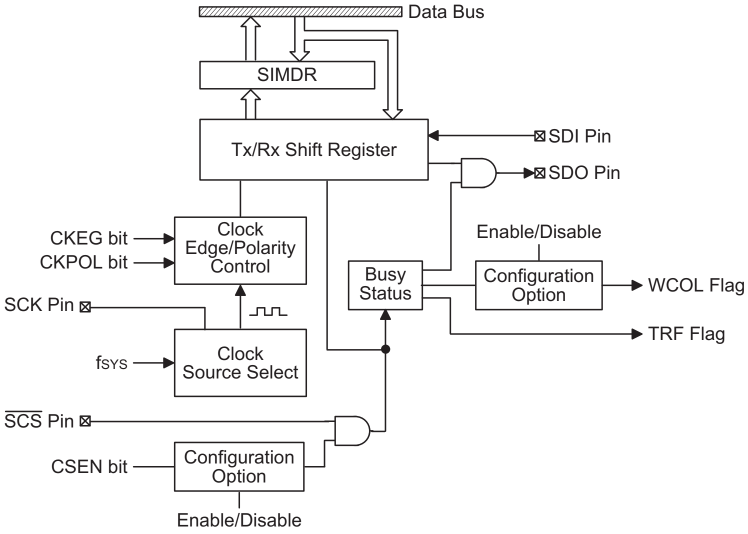
Configuration Option	Function
SIM Function	SIM interface or I/O pins
SPI CSEN bit	Enable/Disable
SPI WCOL bit	Enable/Disable

SPI Interface Configuration Options

SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMDR data register and two control registers SIMCTL0 and SIMCTL2. Note that the SIMCTL1 register is only used by the I^{2}C interface.

Configuration Option		Register		I/O Status		Note
		SIMCTL0	SIMCTL2			
SIM Function	SPI_CSEN	SIMEN	CSEN	SPI	$\overline{SCS}$	
0	x	x	x	I/O	I/O	
1	x	0	x	I/O	I/O	
1	0	1	x	SPI	I/O	$\overline{SCS}$ not floating
1	1	1	0	SPI	I/O	$\overline{SCS}$ not floating
1	1	1	1	SPI	$\overline{SCS}$	



SPI Block Diagram

The SIMDR register is used to store the data being transmitted and received. The same register is used by both the SPI and I^2C functions. Before the microcontroller writes data to the SPI bus, the actual data to be transmitted must be placed in the SIMDR register. After the data is received from the SPI bus, the microcontroller can read it from the SIMDR register. Any transmission or reception of data from the SPI bus must be made via the SIMDR register.

Bit	7	6	5	4	3	2	1	0
Label	SD7	SD6	SD5	SD4	SD3	SD2	SD1	SD0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	X	X	X	X	X	X	X	X

There are also two control registers for the SPI interface, SIMCTL0 and SIMCTL2. Note that the SIMCTL2 register also has the name SIMAR which is used by the I^2C function. The SIMCTL1 register is not used by the SPI function, only by the I^2C function. Register SIMCTL0 is used to control the enable/disable function and to set the data transmission clock frequency. Although not connected with the SPI function, the SIMCTL0 register is also used to control the Peripheral Clock prescaler. Register SIMCTL2 is used for other control functions such as LSB/MSB selection, write collision flag etc.

The following gives further explanation of each SIMCTL0 register bit:

- SIMEN
  The bit is the overall on/off control for the SPI interface. When the SIMEN bit is cleared to zero to disable the SPI interface, the SDI, SDO, SCK and $\overline{SCS}$ lines will be in a I/O mode and the SPI operating current will be reduced to a minimum value. When the bit is high the SPI interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. Note that when the SIMEN bit changes from low to high the contents of the SPI control registers will be in an unknown condition and should therefore be first initialised by the application program.

- SIM0~SIM2
  These bits setup the overall operating mode of the SIM function. As well as selecting if the I^2C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

SIM0	SIM1	SIM2	SPI Master/Slave Clock Control and I2C Enable
0	0	0	SPI Master, $f_{SYS}/4$
0	0	1	SPI Master, $f_{SYS}/16$
0	1	0	SPI Master, $f_{SYS}/64$
0	1	1	SPI Master, f_{SYS}
1	0	0	SPI Master, $f_{SYS}/2$
1	0	1	SPI Slave
1	1	0	I^2C mode
1	1	0	Not used

SPI Control Register – SIMCTL2

The SIMCTL2 register is also used by the I^2C interface but has the name SIMAR

- TRF
  The TRF bit is the Transmit/Receive Complete flag and is set high automatically when an SPI data transmission is completed, but must be cleared by the application program. It can be used to generate an interrupt.

- WCOL
  The WCOL bit is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMDR register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared by the application program. Note that using the WCOL bit can be disabled or enabled via configuration option.

- CSEN
  The CSEN bit is used as an on/off control for the $\overline{SCS}$ pin. If this bit is low then the $\overline{SCS}$ pin will be disabled and placed into I/O mode. If the bit is high the $\overline{SCS}$ pin will be enabled and used as a select pin. Note that using the CSEN bit can be disabled or enabled via configuration option.

- MLS
  This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

- CKEG and CKPOL
  These two bits are used to setup the way that the clock signal outputs and inputs data on the SPI bus. These two bits must be configured before data transfer is executed otherwise an erroneous clock edge

may be generated. The CKPOL bit determines the base condition of the clock line, if the bit is high then the SCK line will be low when the clock is inactive. When the CKPOL bit is low then the SCK line will be high when the clock is inactive. The CKEG bit determines active clock edge type which depends upon the condition of CKPOL.

CKPOL	CKEG	SCK Clock Signal
0	0	High Base Level Active Rising Edge
0	1	High Base Level Active Falling Edge
1	0	Low Base Level Active Falling Edge
1	1	Low Base Level Active Rising Edge

SPI Communication

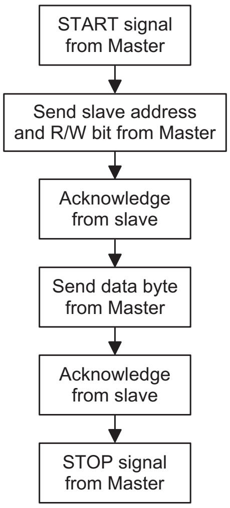
After the SPI interface is enabled by setting the SIMEN bit high, then in the Master Mode, when data is written to the SIMDR register, transmission/reception will begin simultaneously. When the data transfer is complete, the TRF flag will be set automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SIMDR register will be transmitted and any data on the SDI pin will be shifted into the SIMDR register. The master should output an $\overline{SCS}$ signal to enable the slave device before a clock signal is provided and slave data transfers should be enabled/disabled before/after an $\overline{SCS}$ signal is received.

The SPI will continue to function even after a HALT instruction has been executed.

I^{2}C Interface

The I^{2}C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple communication protocol and the ability to accommodate multiple devices on the same bus has made it an extremely popular interface type for many applications.

- I^{2}C Interface Operation
  The I^{2}C serial interface is a two line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs. Note that no chip select line exists, as each device on the I^{2}C bus is identified by a unique address which will be transmitted and received on the I^{2}C bus.
  When two devices communicate with each other on the bidirectional I^{2}C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For these devices, which only operates in slave mode, there are two methods of transferring data on the I^{2}C bus, the slave transmit mode and the slave receive mode.

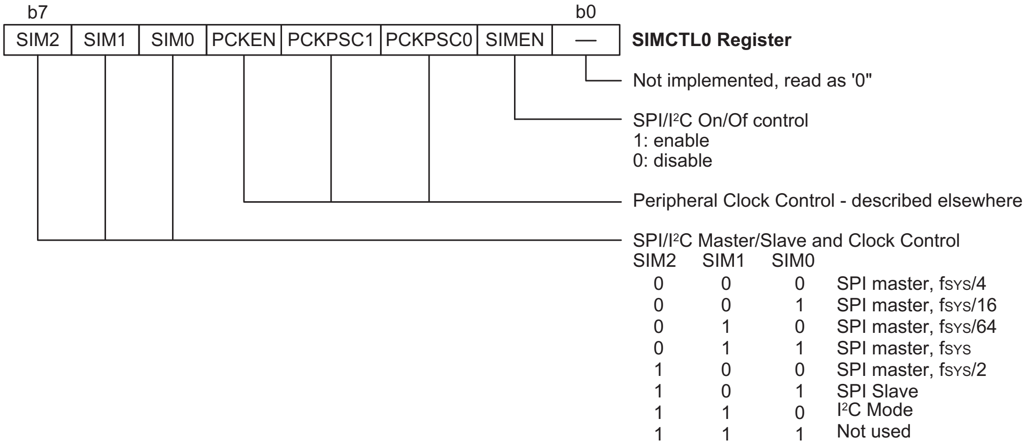


There are several configuration options associated with the I^{2}C interface. One of these is to enable the function which selects the SIM pins rather than normal I/O pins. Note that if the configuration option does not select the SIM function then the SIMEN bit in the SIMCTL0 register will have no effect. A configuration option exists to allow a clock other than the system clock to drive the I^{2}C interface. Another configuration option determines the debounce time of the I^{2}C interface. This uses the internal clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 1 or 2 system clocks.

SIM	Function
SIM function	SIM interface enable or disable
I^{2}C debounce	No debounce, 1 system clock; 2 system clocks

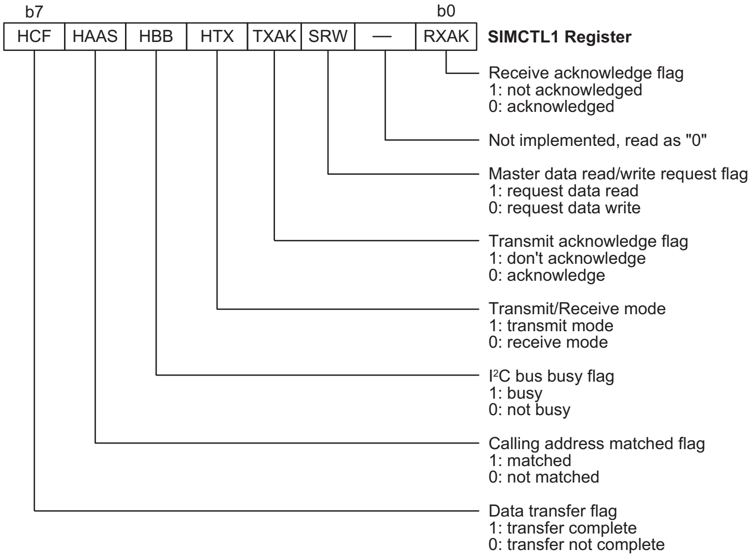
I^{2}C Interface Configuration Options

- I^{2}C Registers
  There are three control registers associated with the I^{2}C bus, SIMCTL0, SIMCTL1 and SIMAR and one data register, SIMDR. The SIMDR register, which is shown in the above SPI section, is used to store the data being transmitted and received on the I^{2}C bus. Before the microcontroller writes data to the I^{2}C bus, the actual data to be transmitted must be placed in the SIMDR register. After the data is received from the I^{2}C bus, the microcontroller can read it from the SIMDR register. Any transmission or reception of data from the I^{2}C bus must be made via the SIMDR register.

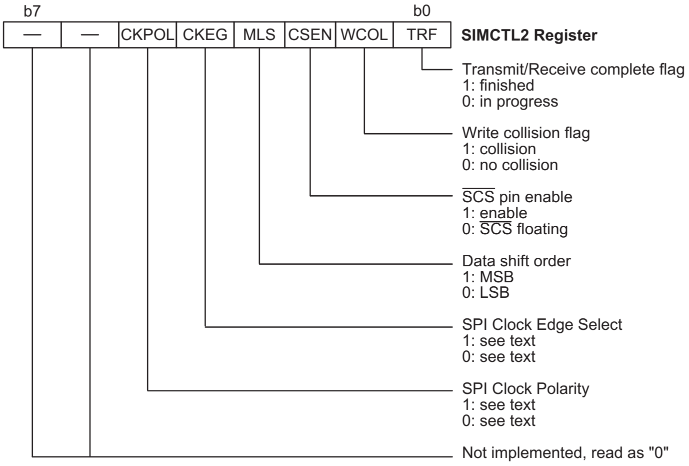


SIM2	SIM1	SIM0	
0	0	0	SPI master, $f_{SYS}/4$
0	0	1	SPI master, $f_{SYS}/16$
0	1	0	SPI master, $f_{SYS}/64$
0	1	1	SPI master, f_{SYS}
1	0	0	SPI master, $f_{SYS}/2$
1	0	1	SPI Slave
1	1	0	I²C Mode
1	1	1	Not used

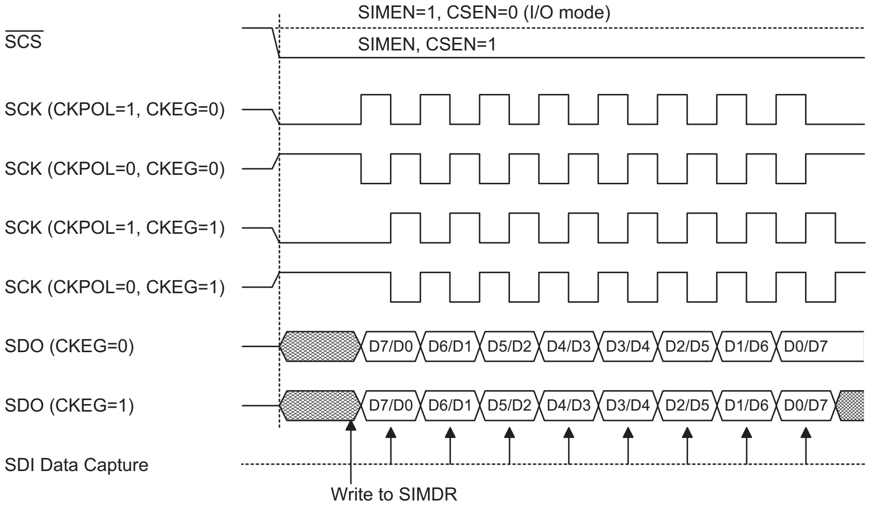
SPI/I²C Control Register – SIMCTL0



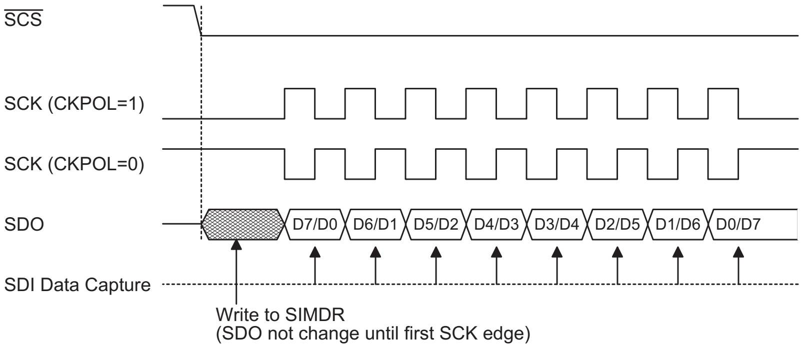
I²C Control Register – SIMCTL1



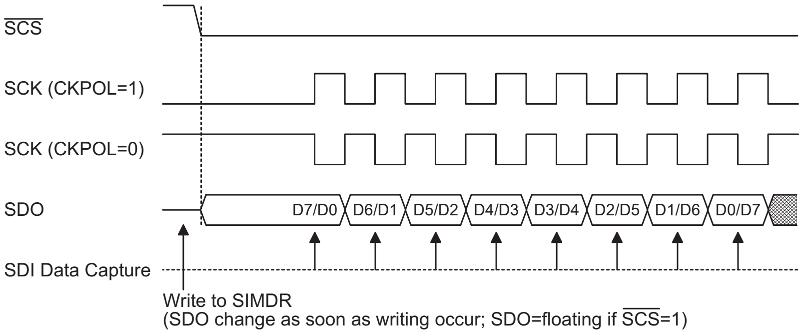
SPI Control Register – SIMCTL2



SPI Master Mode Timing

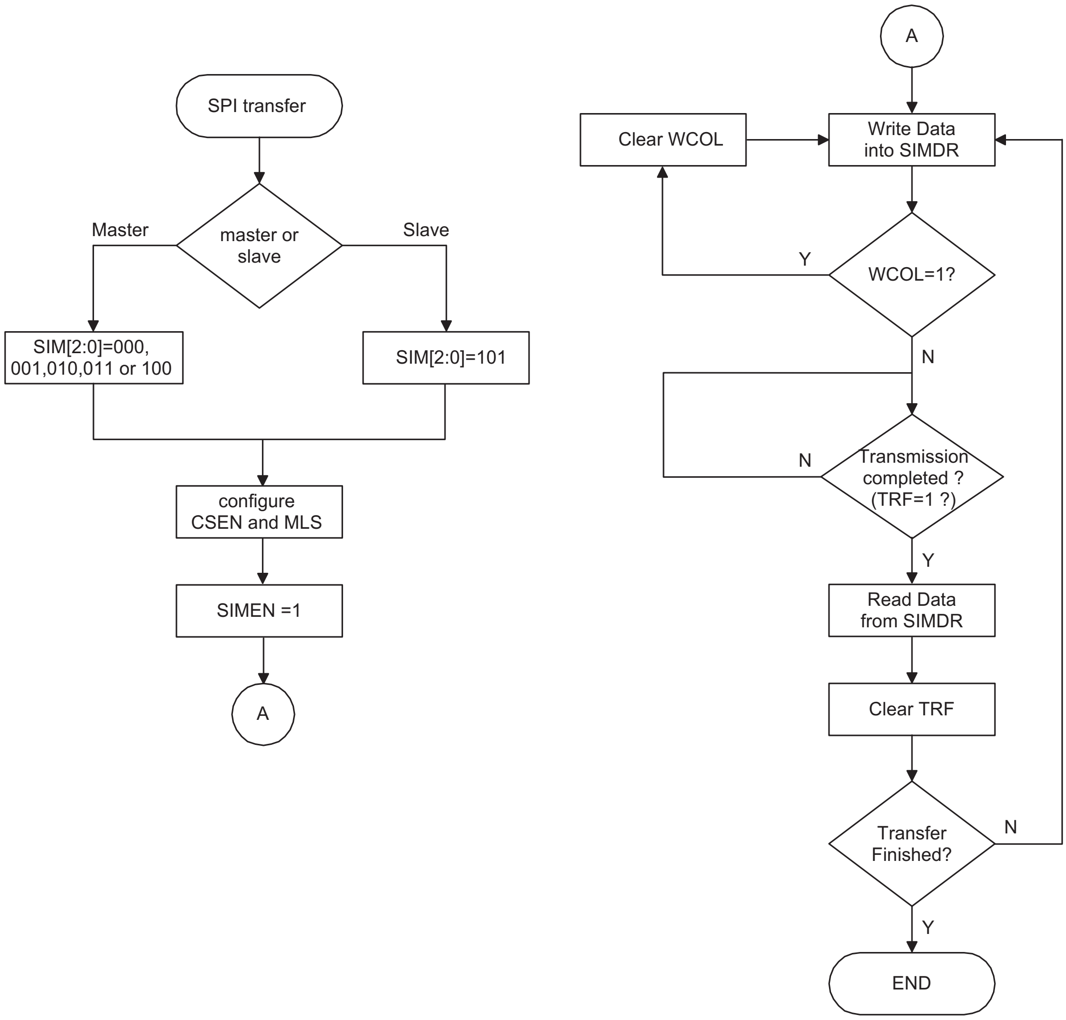


SPI Slave Mode Timing (CKEG=0)

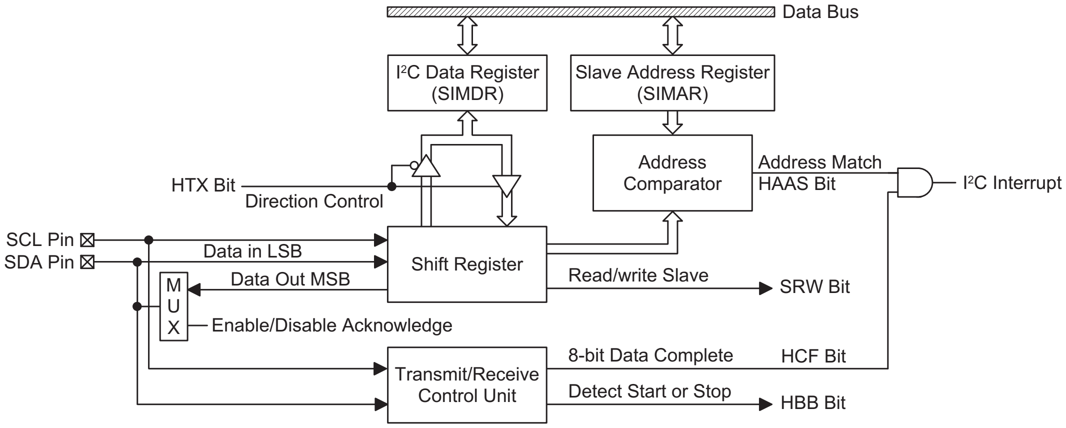


Note: For SPI slave mode, if SIMEN=1 and CSEN=0, SPI is always enabled and ignore the SCS level.

SPI Slave Mode Timing (CKEG=1)



SPI Transfer Control Flowchart



I^{2}C Block Diagram

- SIMEN
  The SIMEN bit is the overall on/off control for the I^{2}C interface. When the SIMEN bit is cleared to zero to disable the I^{2}C interface, the SDA and SCL lines will be in a I/O mode and the I^{2}C operating current will be reduced to a minimum value. In this condition the pins can be used as SEG functions. When the bit is high the I^{2}C interface is enabled. The SIMconfiguration option must have first enabled the SIMinterface for this bit to be effective. Note that when the SIMEN bit changes from low to high the contents of the I^{2}C control registers will be in an unknown condition and should therefore be first initialised by the application program
- SIM0~SIM2
  These bits setup the overall operating mode of the SIM function. To select the I^{2}C function, bits SIM2~SIM0 should be set to the value 110.
- RXAK
  The RXAK flag is the receive acknowledge flag. When the RXAKbit has been reset to zero it means that a correct acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When in the transmit mode, the transmitter checks the RXAK bit to determine if the receiver wishes to receive the next byte. The transmitter will therefore continue sending out data until the RXAK bit is set high. When this occurs, the transmitter will release the SDA line to allow the master to send a STOP signal to release the bus.
- SRW
  The SRW bit is the Slave Read/Write bit. This bit determines whether the master device wishes to transmit or receive data from the I^{2}C bus. When the transmitted address and slave address match, that is when the HAAS bit is set high, the device will check the SRW bit to determine whether it should be in transmit mode or receive mode. If the SRW bit is high, the master is requesting to read data from the bus, so the device should be in transmit mode. When the SRW bit is zero, the master will write data to the bus, therefore the device should be in receive mode to read this data.
- TXAK
  The TXAK flag is the transmit acknowledge flag. After the receipt of 8-bits of data, this bit will be transmitted to the bus on the 9th clock. To continue receiving more data, this bit has to be reset to zero before further data is received.
- HTX
  The HTX flag is the transmit/receive mode bit. This flag should be set high to set the transmit mode and low for the receive mode.
- HBB
  The HBB flag is the I^{2}C busy flag. This flag will be high when the I^{2}C bus is busy which will occur when a START signal is detected. The flag will be reset to zero when the bus is free which will occur when a STOP signal is detected.
- HASS
  The HASS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.
- HCF
  The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

I^{2}C Control Register – SIMAR

The SIMARregister is also used by the SPI interface but has the name SIMCTL2.

The SIMAR register is the location where the 7-bit slave address of the microcontroller is stored. Bits 1~7 of the SIMAR register define the microcontroller slave address. Bit 0 is not defined. When a master device, which is connected to the I^{2}C bus, sends out an address, which matches the slave address in the SIMAR register, the microcontroller slave device will be selected.

I^{2}C Bus Communication

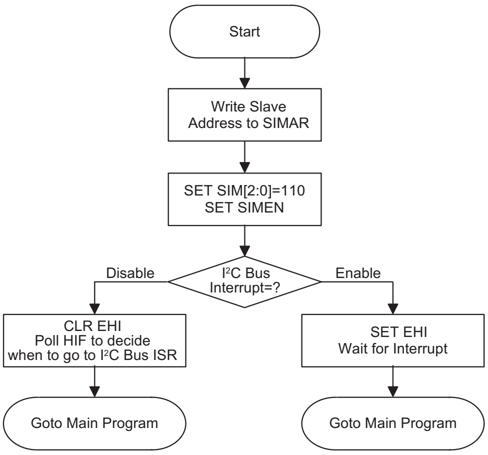
Communication on the I^{2}C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I^{2}C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the microcontroller matches that of the transmitted address, the HAAS bit in the SIMCTL1 register will be set and an I^{2}C interrupt will be generated. After entering the interrupt service routine, the microcontroller slave device must first check the condition of the HAAS bit to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the microcontroller to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I^{2}C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

Step 1

Write the slave address of the microcontroller to the I^{2}C bus address register SIMAR.

Step 2

Set the SIMEN bit in the SIMCTL0 register to ″1″ to enable the I^{2}C bus.



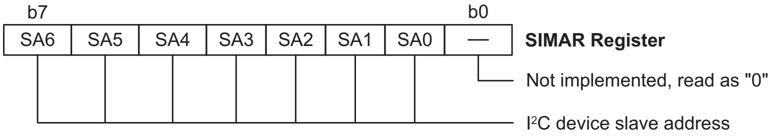
I^{2}C Bus Initialisation Flow Chart

Step 3

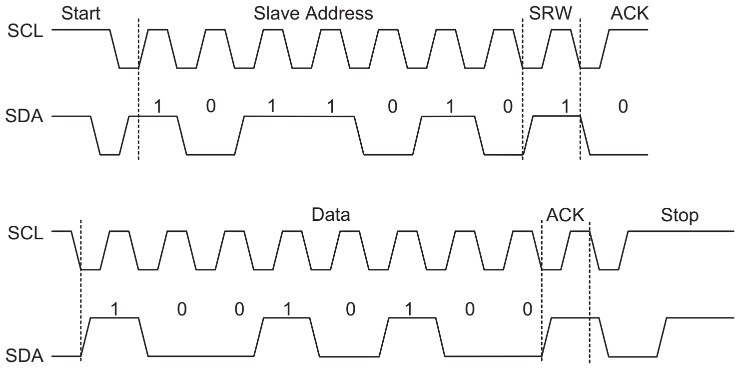
Set the ESIM bit of the interrupt control register to enable the I^{2}C bus interrupt.

- Start Signal
  The START signal can only be generated by the master device connected to the I^{2}C bus and not by the microcontroller, which is only a slave device. This START signal will be detected by all devices connected to the I^{2}C bus. When detected, this indicates that the I^{2}C bus is busy and therefore the HBB bit will be set. A START condition occurs when a high to low transition on the SDA line takes place when the SCL line remains high.
- Slave Address
  The transmission of a START signal by the master will be detected by all devices on the I^{2}C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal I^{2}C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SRW bit of the SIMCTL1 register. The device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The microcontroller slave device will also set the status flag HAAS when the addresses match.

  As an I^{2}C bus interrupt can come from two sources, when the program enters the interrupt subroutine, the HAAS bit should be examined to see whether the interrupt source has come from a matching slave address or from the completion of a data byte transfer. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMDR register, or in the receive mode where it must implement a dummy read from the SIMDR register to release the SCL line.
- SRW Bit
  The SRW bit in the SIMCTL1 register defines whether the microcontroller slave device wishes to read data from the I^{2}C bus or write data to the I^{2}C bus. The microcontroller should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW bit is set to ″1″ then this indicates that the master wishes to read data from the I^{2}C bus, therefore the microcontroller slave device must be setup to send data to the I^{2}C bus as a transmitter. If the SRW bit is ″0″ then this indicates that the master wishes to send data to the I^{2}C bus, therefore the microcontroller slave device must be setup to read data from the I^{2}C bus as a receiver.
- Acknowledge Bit
  After the master has transmitted a calling address, any slave device on the I^{2}C bus, whose own internal address matches the calling address, must generate an acknowledge signal. This acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS bit is high, the addresses have matched and the microcontroller slave device must check the SRW bit to determine if it is to be a transmitter or a receiver. If the SRW bit is high, the microcontroller slave device should be setup to be a transmitter so the HTX bit in the SIMCTL1 register should be set to ″1″ if the SRW bit is low then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMCTL1 register should be set to ″0″.



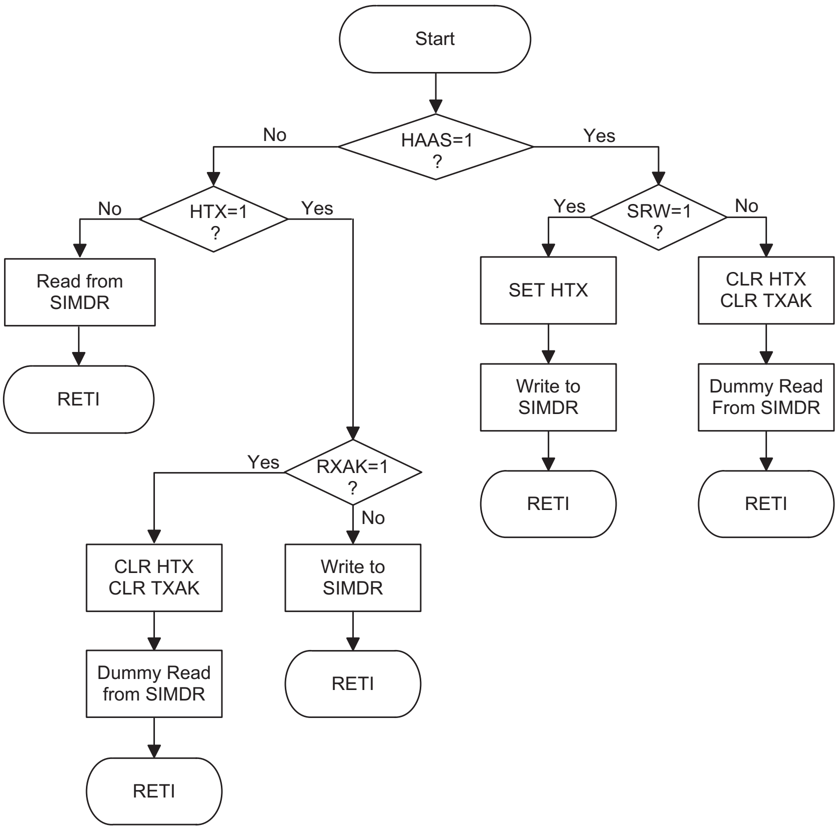
I^{2}C Slave Address Register – SIMAR



S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver 1 bit)
P=Stop (1 bit)

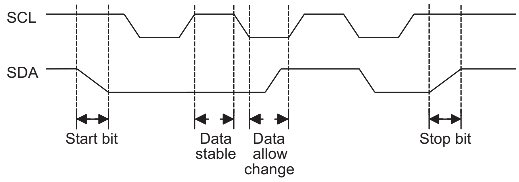
S	SA	SR	M	D	A	D	A	--------	S	SA	SR	M	D	A	D	A	--------	P

I^{2}C Communication Timing Diagram



I^{2}C Bus ISR Flow Chart

- Data Byte
  The transmitted data is 8-bits wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8-bits of data, the receiver must transmit an acknowledge signal, level ″0″, before it can receive the next data byte. If the transmitter does not receive an acknowledge bit signal from the receiver, then it will release the SDA line and the master will send out a STOP signal to release control of the I^{2}C bus. The corresponding data will be stored in the SIMDR register. If setup as a transmitter, the microcontroller slave device must first write the data to be transmitted into the SIMDR register. If setup as a receiver, the microcontroller slave device must read the transmitted data from the SIMDR register.



Data Timing Diagram

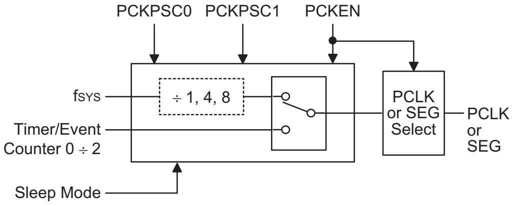
- Receive Acknowledge Bit
  When the receiver wishes to continue to receive the next data byte, it must generate an acknowledge bit, known as TXAK, on the 9th clock. The microcontroller slave device, which is setup as a transmitter will check the RXAK bit in the SIMCTL1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.

Peripheral Clock Output

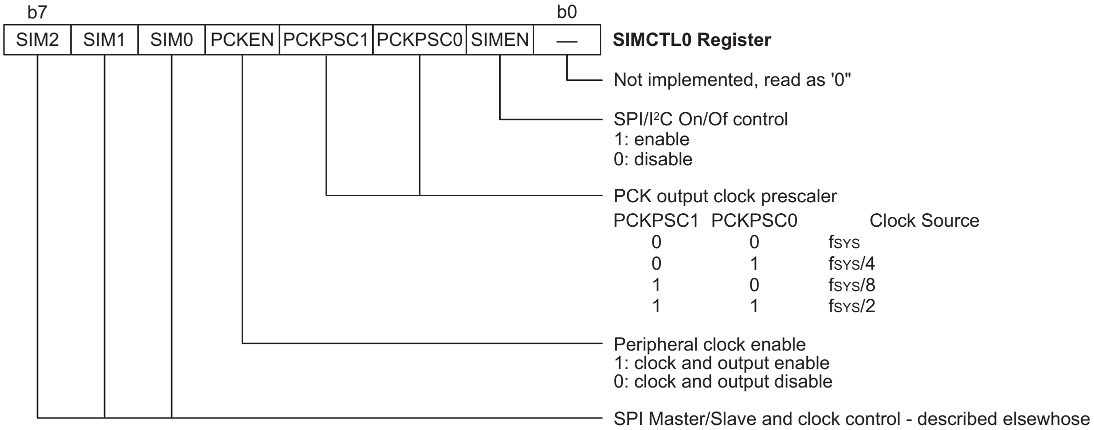
The Peripheral Clock Output allows the device to supply external hardware with a clock signal synchronised to the microcontroller clock.

Peripheral Clock Operation

As the peripheral clock output pin, PCK, is shared with I/O line, the required pin function is chosen via PCKEN in the SIMCTL0 register. The clock source for the Peripheral Clock Output originates from the system clock or a divided ration of the system clock. The PCKEN bit in the SIMCTRL0 register is the overall on/off control, setting PCKEN bit high enables the Peripheral Clock, clearing the bit to zero disables it. The required division ratio of the system clock is selected using the PCKP1 and PCKP0 bits in the same register. If the device is powered down, this will disable the Peripheral Clock output.



Peripheral Clock Block Diagram



b7							b0
SIM2	SIM1	SIM0	PCKEN	PCKPSC1	PCKPSC0	SIMEN	—

SIMCTL0 Register

Not implemented, read as '0"

SPI/I^{2}C On/Of control
1: enable
0: disable

PCK output clock prescaler

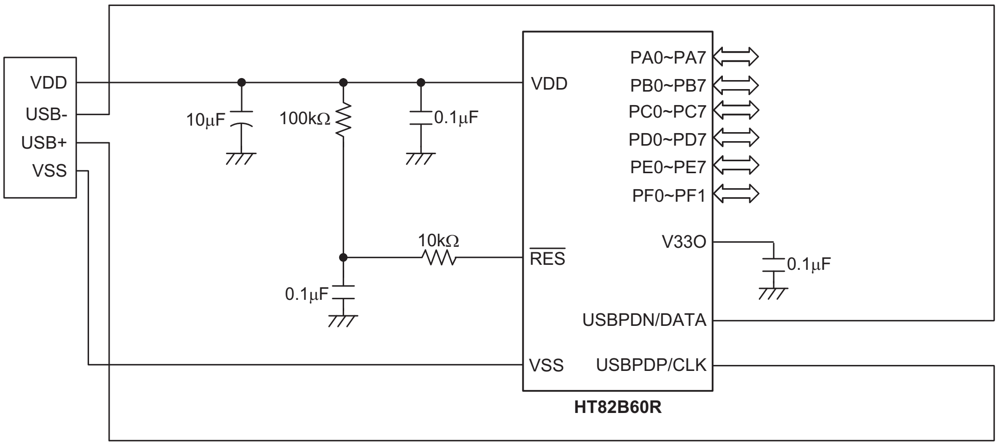
PCKPSC1	PCKPSC0	Clock Source
0	0	f_{SYS}
0	1	$f_{SYS}/4$
1	0	$f_{SYS}/8$
1	1	$f_{SYS}/2$

Peripheral clock enable
1: clock and output enable
0: clock and output disable

SPI Master/Slave and clock control - described elsewhose

Peripheral Clock Output Control – SIMCTL0

Application Circuits



Note: The resistance and capacitance for the reset circuit should be designed in such a way as to ensure that the VDD is stable and remains within a valid operating voltage range before bringing $\overline{\text{RES}}$ high.

Components with * are used for EMC issue.

Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

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

Mnemonic	Description	Cycles	Flag Affected
Rotate			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	1^{Note}	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	1^{Note}	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	1^{Note}	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	1^{Note}	C
Data Move			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1^{Note}	None
MOV A,x	Move immediate data to ACC	1	None
Bit Operation			
CLR [m].i	Clear bit of Data Memory	1^{Note}	None
SET [m].i	Set bit of Data Memory	1^{Note}	None
Branch			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1^{Note}	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1^{note}	None
SZ [m].i	Skip if bit i of Data Memory is zero	1^{Note}	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1^{Note}	None
SIZ [m]	Skip if increment Data Memory is zero	1^{Note}	None
SDZ [m]	Skip if decrement Data Memory is zero	1^{Note}	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1^{Note}	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1^{Note}	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
Table Read			
TABRDC [m](4)	Read ROM code (locate by TBLP and TBHP) to data memory and TBLH	2^{Note}	None
TABRDC [m](5)	Read ROM code (current page) to data memory and TBLH	2^{Note}	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2^{Note}	None
Miscellaneous			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1^{Note}	None
SET [m]	Set Data Memory	1^{Note}	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer	1	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1^{Note}	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

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

Instruction Definition

ADC A,[m]	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C

ADCM A,[m]	Add ACC to Data Memory with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m] + C
Affected flag(s)	OV, Z, AC, C

ADD A,[m]	Add Data Memory to ACC
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + [m]
Affected flag(s)	OV, Z, AC, C

ADD A,x	Add immediate data to ACC
Description	The contents of the Accumulator and the specified immediate data are added. The result is stored in the Accumulator.
Operation	ACC ← ACC + x
Affected flag(s)	OV, Z, AC, C

ADDM A,[m]	Add ACC to Data Memory
Description	The contents of the specified Data Memory and the Accumulator are added. The result is stored in the specified Data Memory.
Operation	[m] ← ACC + [m]
Affected flag(s)	OV, Z, AC, C

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

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

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

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

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

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

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

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

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

CPL [m]	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	$[m] \leftarrow \overline{[m]}$
Affected flag(s)	Z

CPLA [m]	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1′s complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow \overline{[m]}$
Affected flag(s)	Z

DAA [m]	Decimal-Adjust ACC for addition with result in Data Memory
Description	Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.
Operation	$[m] \leftarrow ACC + 00H$ or $[m] \leftarrow ACC + 06H$ or $[m] \leftarrow ACC + 60H$ or $[m] \leftarrow ACC + 66H$
Affected flag(s)	C

DEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	$[m] \leftarrow [m] - 1$
Affected flag(s)	Z

DECA [m]	Decrement Data Memory with result in ACC
Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	$ACC \leftarrow [m] - 1$
Affected flag(s)	Z

HALT	Enter power down mode
Description	This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.
Operation	$TO \leftarrow 0$ $PDF \leftarrow 1$
Affected flag(s)	TO, PDF

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [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 and the carry flag are rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC.i ← [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	If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	Skip if [m] = 0
Affected flag(s)	None

SZA [m]	Skip if Data Memory is 0 with data movement to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	ACC ← [m] Skip if [m] = 0
Affected flag(s)	None

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

TABRDC [m]	Read table (current page) to TBLH and Data Memory
Description	The low byte of the program code (current page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None

TABRDC [m]	Move the ROM code (locate by TBLP and TBHP) to TBLH and data memory (ROM code TBHP is enabled)
Description	The low byte of ROM code addressed by the table pointers (TBLP and TBHP) is moved to the specified data memory and the high byte transferred to TBLH directly.
Operation	[m] ← 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

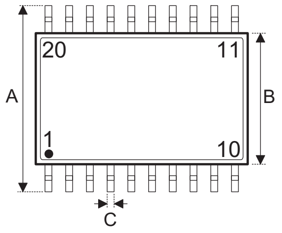
XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

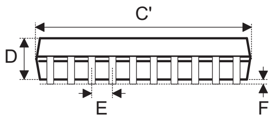
XORM A,[m]	Logical XOR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″XOR″ [m]
Affected flag(s)	Z

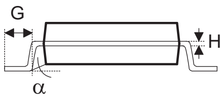
XOR A,x	Logical XOR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ x
Affected flag(s)	Z

Package Information

20-pin SSOP (150mil) Outline Dimensions



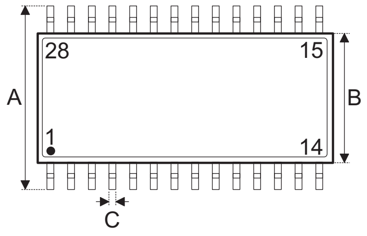


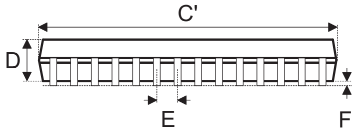


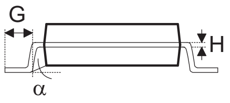
Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.158
C	0.008	—	0.012
C′	0.335	—	0.347
D	0.049	—	0.065
E	—	0.025	—
F	0.004	—	0.010
G	0.015	—	0.050
H	0.007	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.79	—	6.20
B	3.81	—	4.01
C	0.20	—	0.30
C′	8.51	—	8.81
D	1.24	—	1.65
E	—	0.64	—
F	0.10	—	0.25
G	0.38	—	1.27
H	0.18	—	0.25
α	0°	—	8°

28-pin SSOP (150mil) Outline Dimensions



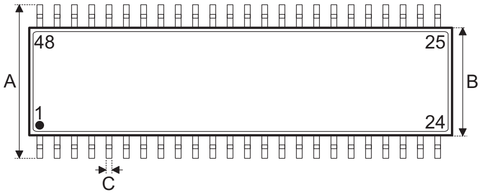


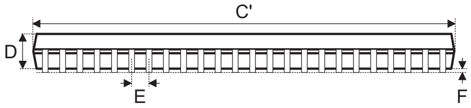


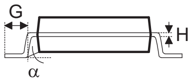
Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.228	—	0.244
B	0.150	—	0.157
C	0.008	—	0.012
C′	0.386	—	0.394
D	0.054	—	0.060
E	—	0.025	—
F	0.004	—	0.010
G	0.022	—	0.028
H	0.007	—	0.010
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	5.79	—	6.20
B	3.81	—	3.99
C	0.20	—	0.30
C′	9.80	—	10.01
D	1.37	—	1.52
E	—	0.64	—
F	0.10	—	0.25
G	0.56	—	0.71
H	0.18	—	0.25
α	0°	—	8°

48-pin SSOP (300mil) Outline Dimensions



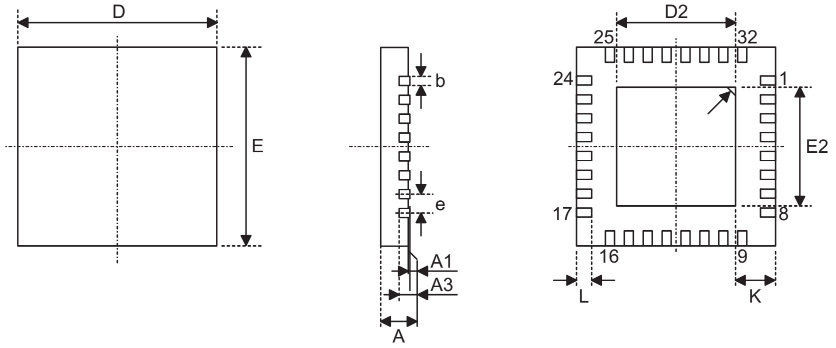




Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.395	—	0.420
B	0.291	—	0.299
C	0.008	—	0.012
C′	0.613	—	0.637
D	0.085	—	0.099
E	—	0.025	—
F	0.004	—	0.010
G	0.025	—	0.035
H	0.004	—	0.012
α	0°	—	8°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	10.03	—	10.67
B	7.39	—	7.59
C	0.20	—	0.30
C′	15.57	—	16.18
D	2.16	—	2.51
E	—	0.64	—
F	0.10	—	0.25
G	0.64	—	0.89
H	0.10	—	0.30
α	0°	—	8°

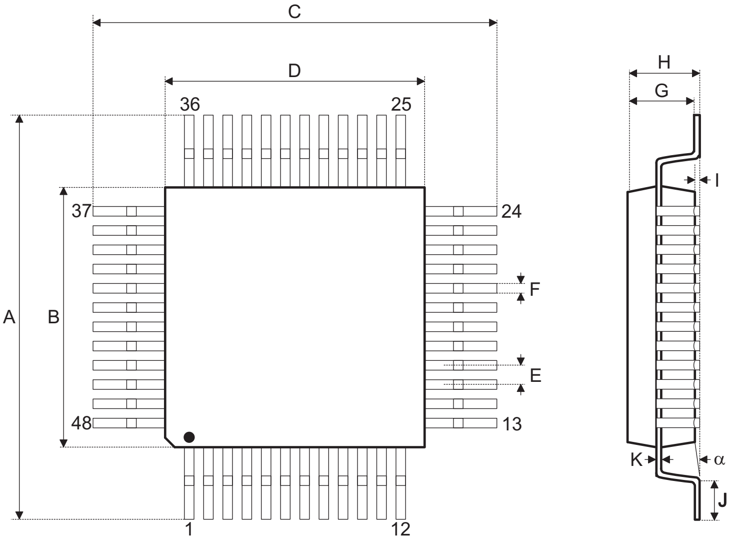
SAW Type 32-pin (5mm×5mm) QFN Outline Dimensions



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.028	—	0.031
A1	0.000	—	0.002
A3	—	0.008	—
b	0.007	—	0.012
D	—	0.197	—
E	—	0.197	—
e	—	0.020	—
D2	0.049	—	0.128
E2	0.049	—	0.128
L	0.012	—	0.020
K	—	—	—

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	0.70	—	0.80
A1	0.00	—	0.05
A3	—	0.20	—
b	0.18	—	0.30
D	—	5.00	—
E	—	5.00	—
e	—	0.50	—
D2	1.25	—	3.25
E2	1.25	—	3.25
L	0.30	—	0.50
K	—	—	—

48-pin LQFP (7mm×7mm) Outline Dimensions

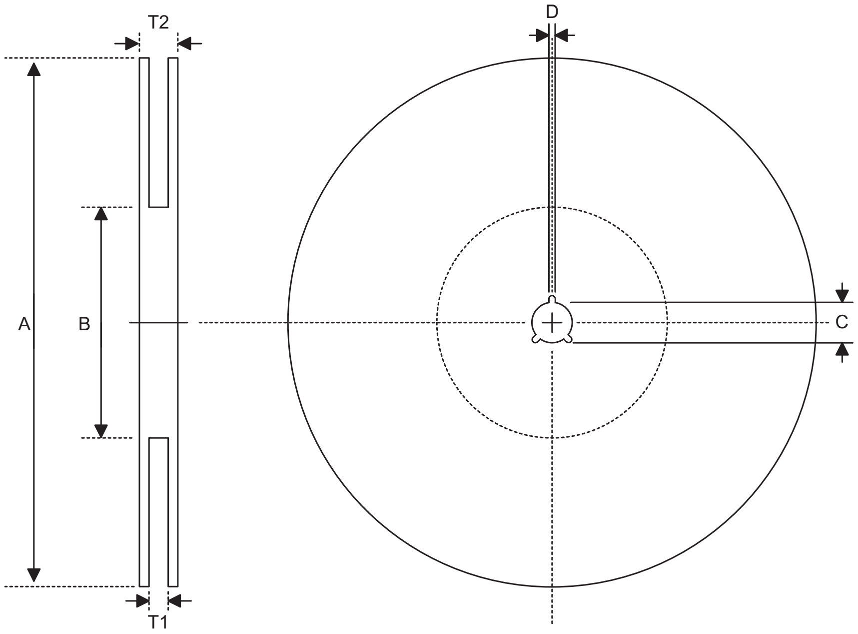


Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.350	—	0.358
B	0.272	—	0.280
C	0.350	—	0.358
D	0.272	—	0.280
E	—	0.020	—
F	—	0.008	—
G	0.053	—	0.057
H	—	—	0.063
I	—	0.004	—
J	0.018	—	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	8.90	—	9.10
B	6.90	—	7.10
C	8.90	—	9.10
D	6.90	—	7.10
E	—	0.50	—
F	—	0.20	—
G	1.35	—	1.45
H	—	—	1.60
I	—	0.10	—
J	0.45	—	0.75
K	0.10	—	0.20
α	0°	—	7°

Product Tape and Reel Specifications

Reel Dimensions



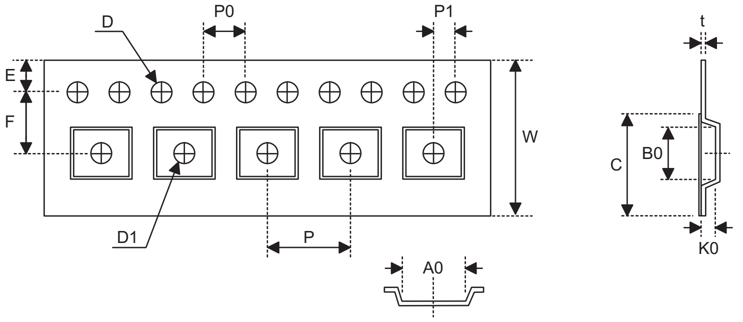
SSOP 20S (150mil), SSOP 28S (150mil)

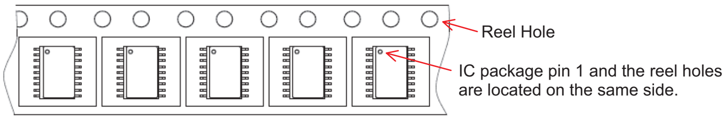
Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	$13.0^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	$16.8^{+0.3/-0.2}$
T2	Reel Thickness	22.2±0.2

SSOP 48W

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±0.1
C	Spindle Hole Diameter	$13.0^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	$32.2^{+0.3/-0.2}$
T2	Reel Thickness	38.2±0.2

Carrier Tape Dimensions



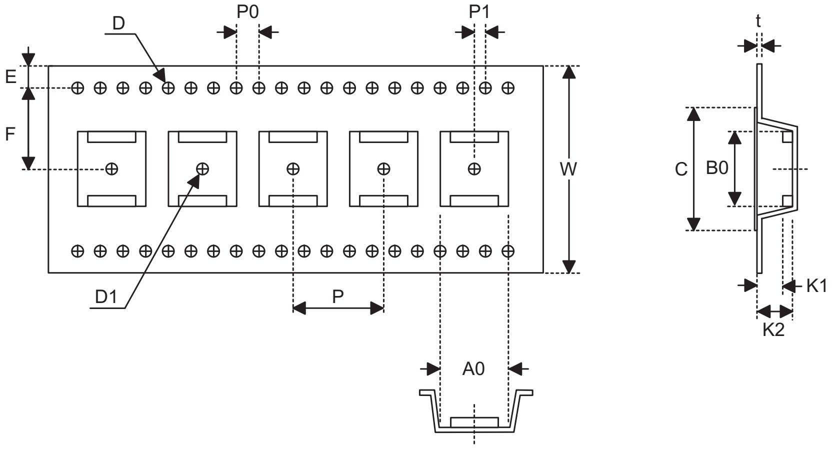


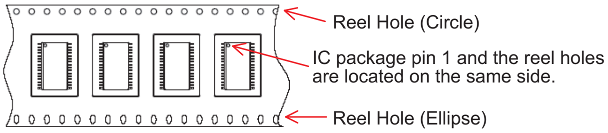
SSOP 20S (150mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	16.0 $^{+0.3/-0.1}$
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	7.5±0.1
D	Perforation Diameter	1.5 $^{+0.1/-0.0}$
D1	Cavity Hole Diameter	1.50 $^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	6.5±0.1
B0	Cavity Width	9.0±0.1
K0	Cavity Depth	2.3±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	13.3±0.1

SSOP 28S (150mil)

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





SSOP 48W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	32.0±0.3
P	Cavity Pitch	16.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	14.2±0.1
D	Perforation Diameter	2 Min.
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	12.0±0.1
B0	Cavity Width	16.2±0.1
K1	Cavity Depth	2.4±0.1
K2	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.35±0.05
C	Cover Tape Width	25.5±0.1

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

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

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

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