HT82K70E-L/HT82K76E-L I/O Type 8-Bit OTP MCU

Technical Document

- Application Note
  - HA0075E MCU Reset and Oscillator Circuits Application Note

Features

- Operating voltage: 1.8V~5.5V
- 43 bidirectional I/O lines
- 4K×16 Program Memory - HT82K70E-L
  8K×16 Program Memory - HT82K76E-L
- 216×8 Data RAM
- One external interrupt input shared with I/O lines
- Two 16-bit programmable Timer/Event Counters with overflow interrupt
- Watchdog Timer function
- Power down and wake-up functions to reduce power consumption
- Crystal and RC oscillator
- 8-level subroutine nesting
- Bit manipulation instruction
- Low Voltage Detector
- Table read instructions
- 63 powerful instructions
- All instructions executed in one or two machine cycles
- Integrated SPI interface (Max. 8Mb/s)
- Some pins with CMOS and NMOS outputs
- 28/48-pin SSOP and 32-pin QFN packages

General Description

The device is an 8-bit high performance, RISC architecture microcontroller devices specifically designed for multiple I/O control product applications. The low voltage operating requirements of these devices opens up new application possibilities.

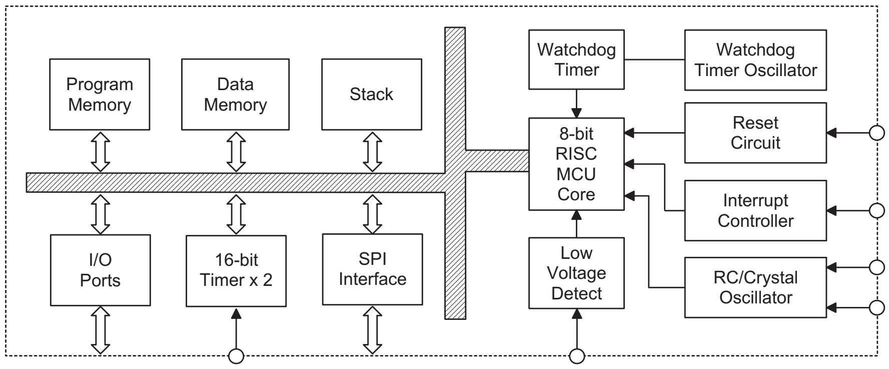
The advantages of low power consumption, I/O flexibility, timer functions, oscillator options, Power-down and wake-up functions, Watchdog timer, motor driving, industrial control, consumer products, subsystem controllers, etc.

Selection Table

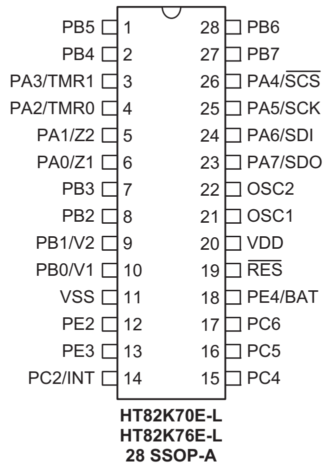
Part No.	Program Memory	Data Memory	I/O	16-bit Timer	LVD for Battery-in	SPI	Stack	Package
HT82K70E-L	4K×16	216×8	43	2	√	√	8	28/48SSOP, 32QFN
HT82K76E-L	8K×16							

Downloaded from Elcodis.com electronic components distributor

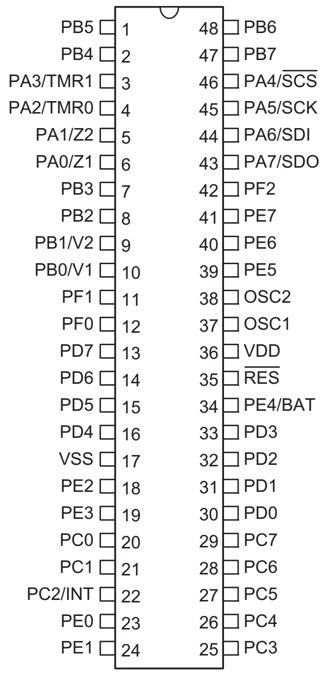
Block Diagram



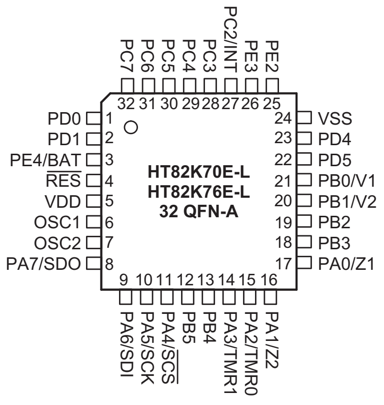
Pin Assignment



HT82K70E-L
HT82K76E-L
28 SSOP-A



HT82K70E-L
HT82K76E-L
48 SSOP-A



Downloaded from Elcodis.com electronic components distributor

Pin Description

Pin Name	I/O	Options	Description
PA0/Z1 PA1/Z2 PA2/TMR0 PA3/$\overline{\text{TMR}}$1 PA4/$\overline{\text{SCS}}$ PA5/SCK PA6/SDI PA7/SDO	I/O	Wake-up Pull-high CMOS or NMOS Schmitt trigger or non-Schmitt trigger	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 input. Configuration options determine if the pins have pull-high resistors. Configuration options determine whether the pins are configured as CMOS or NMOS pins. Configuration options determine whether the pins are configured with Schmitt trigger or non-Schmitt trigger inputs. PA2 is shared with the external timer input pin TMR0. PA3 is shared with the external timer input pin TMR1. PA0 and PA1 are shared with the Z1 and Z2 pins. PA4~ PA7 are pins shared with SPI interface.
PB0/V1 PB1/V2 PB2~PB7	I/O	Wake-up Pull-high	Bidirectional 8-bit input/output port. Each pin, PB0 and PB1 can be configured as wake-up inputs using configuration options. Two configuration options, one for pins PB2 and PB3 and one for pins PB4~PB7, can also setup these pin groups as wake-up inputs. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine if each nibble, PB0~PB3 and PB4~PB7 have pull-high resistors. PB0 and PB1 are shared with the V1 and V2 pins.
PC0~PC1 PC2/INT PC3~PC7	I/O	Wake-up Pull-high	Bidirectional 8-bit input/output port. Each nibble, PC0~PC3 and PC4~PC7, can be configured as wake-up inputs by configuration options. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine if each nibble, PC0~PC3 and PC4~PC7 have pull-high resistors. PC2 is pin shared with the external interrupt input.
PD0~PD7	I/O	Wake-up Pull-high	Bidirectional 8-bit input/output port. Each nibble, PD0~PD3 and PD4~PD7, can be configured as wake-up inputs by configuration options. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine if each nibble, PD0~PD3 and PD4~PD7 have pull-high resistors.
PE0~PE3 PE4/BAT PE5~PE7	I/O	Wake-up Pull-high PE4 IO or BAT	Bidirectional 8-bit input/output port. Each nibble, PE0~PE3 and PE4~PE7, can be configured as wake-up inputs by configuration options. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine if each nibble, PE0~PE3 and PE4~PE7 have pull-high resistors. A configuration option determines if PE4 is an I/O pin or a Battery input pin.
PF0~PF2	I/O	Wake-up Pull-high	Bidirectional 3-bit input/output port. The pins, PF0~PF2 can be configured together to be wake-up inputs using a configuration options. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. A configuration option determine if the pins have pull-high resistors.
OSC1 OSC2	I O	Crystal or RC	OSC1, OSC2 are connected to an external RC network or external crystal,determined by configuration option, for the internal system clock. If the RC system clock option is selected, pin OSC2 can be used to measure the system clock at 1/4 frequency.
$\overline{\text{RES}}$	I	—	Schmitt trigger reset input. Active low
VSS	—	—	Negative power supply, ground
VDD	—	—	Positive power supply

Note: Each pin can be chosen via configuration option to have a wake-up function.

Downloaded from Elcodis.com electronic components distributor

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 (Crystal OSC)	—	f_{SYS}=4MHz	1.8	—	5.5	V
			f_{SYS}=8MHz	3.3	—	5.5	V
I_{DD}	Operating Current (Crystal OSC)	3V	No load, f_{SYS}=6MHz	—	1	2	mA
I_{STB1}	Standby Current	3V	No load, system HALT, WDT Enabled	—	—	20	μA
I_{STB2}	Standby Current	3V	No load, system HALT, WDT Disabled	—	—	5	μA
V_{IL1}	Input Low Voltage for I/O, TMR and INT	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O, TMR and INT	—	—	$0.7V_{DD}$	—	V_{DD}	V
V_{IL2}	Input Low Voltage ($\overline{RES}$)	—	—	0	—	$0.4V_{DD}$	V
V_{IH2}	Input High Voltage ($\overline{RES}$)	—	—	$0.9V_{DD}$	—	V_{DD}	V
I_{OL}	I/O Port Sink Current	3V	$V_{OL}=0.1V_{DD}$	4	—	—	mA
I_{OH}	I/O Port Source Current	3V	$V_{OH}=0.9V_{DD}$	–2.5	–4.5	—	mA
R_{PH}	Pull-high Resistance	3V	—	10	30	50	kΩ

Downloaded from Elcodis.com electronic components distributor

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock (Crystal OSC)	—	1.8V~5.5V	400	—	4000	kHz
			3.3V~5.5V	400	—	8000	kHz
f_{RCSYS}	Watchdog OSC with 6-stage Prescaler Period	3V	—	—	71	—	μs
f_{SPI}	SPI Clock	—	—	$f_{SYS}/64$	—	f_{SYS}	—
t_{WDT}	Watchdog Time-out Period (WDT OSC)	3V	WDTS=1	—	4.57	—	ms
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	ms
$t_{CONFIGURE}$	System Start-up Timer Period	—	—	—	1024	—	t_{RCSYS}
t_{OST}	Oscillation Start-up Timer Period	—	—	—	512	—	t_{SYS}

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

D.C. - A.C. Power-on Reset Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{POR}	Operating Current	1.8V~5.5V	—	—	—	0.7	μA
RSR_{POR}	VDD Rise Rate to Ensure Power-on Reset	—	Without 0.1μF between VDD and VSS	0.05	—	—	V/ms
V_{POR_MAX}	Maximum V_{DD} Start Voltage to Ensure Power-on Reset	—	Ta=25°C, Without 0.1μF between VDD and VSS	0.9	—	1.5	V
t_{POR}	Power-on Reset Low Pulse Width	—	Without 0.1μF between VDD and VSS	2	—	—	μs
		—	With 0.1μF between VDD and VSS	10	—	—	μs

Downloaded from Elcodis.com electronic components distributor

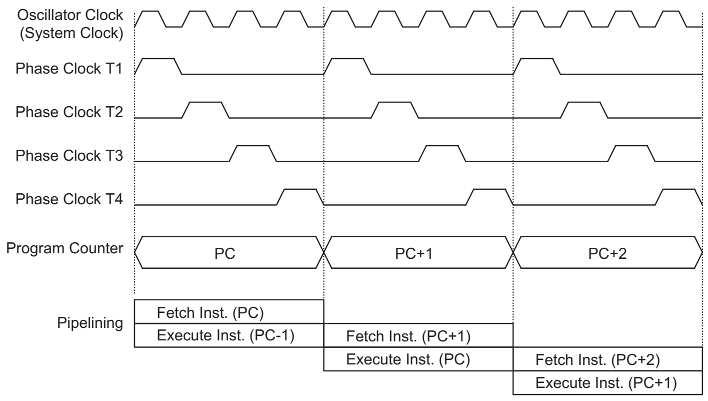
System Architecture

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 control system with maximum reliability and flexibility.

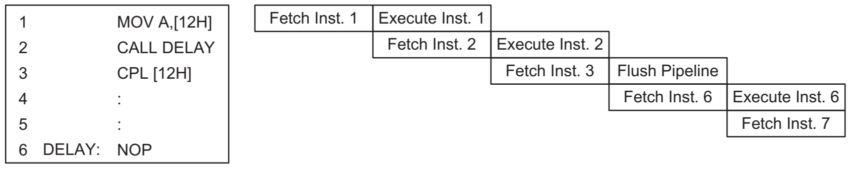
Clocking and Pipelining

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

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



System Clocking and Pipelining



Instruction Fetching

Downloaded from Elcodis.com electronic components distributor

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.

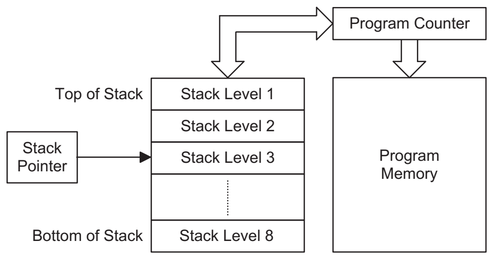
When executing instructions requiring jumps to non-consecutive addresses such as a jump instruction, a subroutine call, interrupt or reset, etc., the microcontroller manages program control by loading the required address into the Program Counter. For conditional skip instructions, once the condition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

Mode	Program Counter Bits												
	b12	**b11**	**b10**	**b9**	**b8**	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
Initial Reset	0	0	0	0	0	0	0	0	0	0	0	0	0
INT 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 Interrupt	0	0	0	0	0	0	0	0	1	0	0	0	0
Skip	Program Counter + 2												
Loading PCL	PC12	PC11	PC10	PC9	PC8	@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: PC12~PC8: Current Program Counter bits @7~@0: PCL bits
#12~#0: Instruction code address bits S12~S0: Stack register bits
For the HT82K70E-L, the Program Counter Bits is 12 bits wide, the b12 column in the table is not applicable
For the HT82K76E-L, the Program Counter Bits is 13 bits wide, i.e. from b12 ~ b0

Downloaded from Elcodis.com electronic components distributor

Arithmetic and Logic Unit – ALU

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

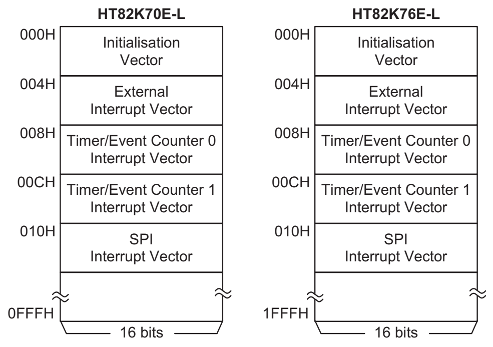
- Arithmetic operations: ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA
- Logic operations: AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA
- Rotation RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC
- Increment and Decrement INCA, INC, DECA, DEC
- Branch decision, JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET,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

Program Memorys

The Program Memory is the location where the user code or program is stored. The device 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. The device is a Mask memory type device and offers the most cost effective solution for high volume products.

Structure

The Program Memory has a capacity of 4K by 16 or 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.


HT82K70E-L
000H
Initialisation Vector
004H
External Interrupt Vector
008H
Timer/Event Counter 0 Interrupt Vector
00CH
Timer/Event Counter 1 Interrupt Vector
010H
SPI Interrupt Vector
0FFFH
16 bits
HT82K76E-L
000H
Initialisation Vector
004H
External Interrupt Vector
008H
Timer/Event Counter 0 Interrupt Vector
00CH
Timer/Event Counter 1 Interrupt Vector
010H
SPI Interrupt Vector
1FFFH
16 bits

Program Memory Structure

Special Vectors

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

- Location 000H
  This vector is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.
- Location 004H
  This vector is used by the external interrupt. If the INT external input pin on the device receives a high to low transition, the program will jump to this location and begin execution, if the interrupt is enabled and the stack is not full.
- Location 008H
  This vector is used by the timer0 counter. If a counter overflow occurs, the program will jump to this location and begin execution if the timer interrupt is enabled and the stack is not full.
- Location 00CH
  This vector is used by the timer1 counter. If a counter overflow occurs, the program will jump to this location and begin execution if the timer interrupt is enabled and the stack is not full.
- Location 010H
  This vector is used by serial interface . When 8-bits of data have been received or transmitted successfully from serial interface, the program will jump to this location and begin execution if the interrupt is enabled and the stack is not full.
- Table location
  Any location in the program memory can be used as look-up tables. There are three method to read the ROM data by two table read instructions: ″TABRDC″ and ″TABRDL″, transfer the contents of the lower-order byte to the specified data memory, and the higher-order byte to TBLH. The three methods are shown as follows:
  - The instructions ″TABRDC [m]″ (the current page, one page=256words), where the table locations is defined by TBLP in the current page. And the configuration option TBHP is disabled (default).
  - The instructions ″TABRDC [m]″, where the table locations is defined by registers TBLP (07H) and TBHP (01FH). And the configuration option TBHP is enabled.
  - The instructions ″TABRDL [m]″, where the table locations is defined by registers TBLP (07H) in the last page (0F00H ~ 0FFFH or 1F00H~1FFFH).

Downloaded from Elcodis.com electronic components distributor

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

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

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

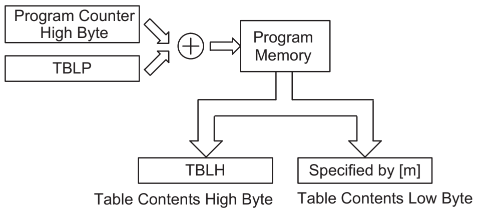


Table Read – TBLP only

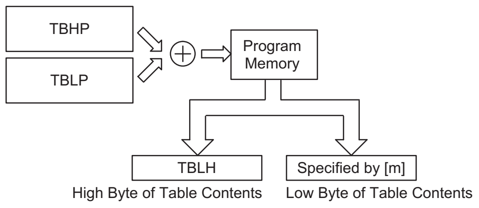


Table Read – TBLP/TBHP

Table Program Example

The following example, for the HT82K76E-L, 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.

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 when Configuration option TBHP is disable
@7~@0: Table Pointer TBLP bits
For the HT82K70E-L, the table address location is 12 bits wide, i.e. from b11 ~ b0
For the HT82K76E-L, the table address location is 13 bits wide, i.e. from b12 ~ b0

Downloaded from Elcodis.com electronic components distributor

• Table Read Program Example

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

mov  tblp,a            ; to the last page 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.

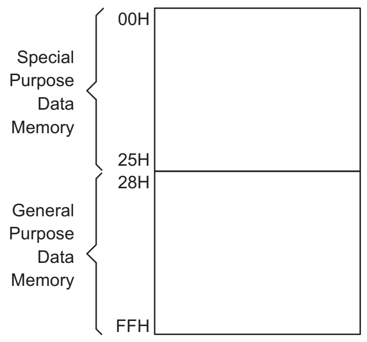
Downloaded from Elcodis.com electronic components distributor

Data Memory

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

Structure

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



Data Memory Structure

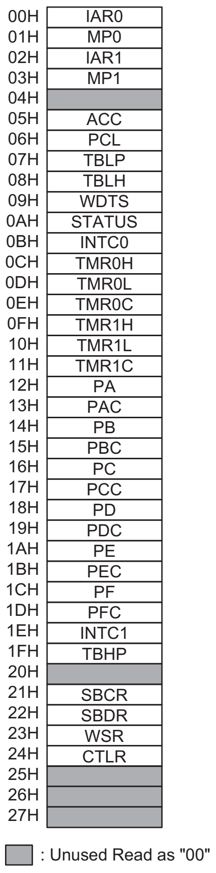
Note: Most of the Data Memory bits can be directly manipulated using the ″SET [m].i″ and ″CLR [m].i″ with the exception of a few dedicated bits. The Data Memory can also be accessed through the memory pointer register MP.

General Purpose Data Memory

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

Special Purpose Data Memory

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



00H	IAR0
01H	MP0
02H	IAR1
03H	MP1
04H	▨
05H	ACC
06H	PCL
07H	TBLP
08H	TBLH
09H	WDTS
0AH	STATUS
0BH	INTC0
0CH	TMR0H
0DH	TMR0L
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	▨
21H	SBCR
22H	SBDR
23H	WSR
24H	CTLR
25H	▨
26H	▨
27H	▨

▨ : Unused Read as "00"

Downloaded from Elcodis.com electronic components distributor

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. 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.

```
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.

Downloaded from Elcodis.com electronic components distributor

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 three special function registers are used to control operation of the look-up table which is stored in the Program Memory. TBLP is the table pointer and indicates the location where the table data is located. Its value must be setup before any table read commands are executed. Its value can be changed, for example using the ″INC″ or ″DEC″ instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location. Once Configuration option TBHP is enabled, the instruction ″TABRDC [m]″ reads the ROM data as defined by TBLP and TBHP value.

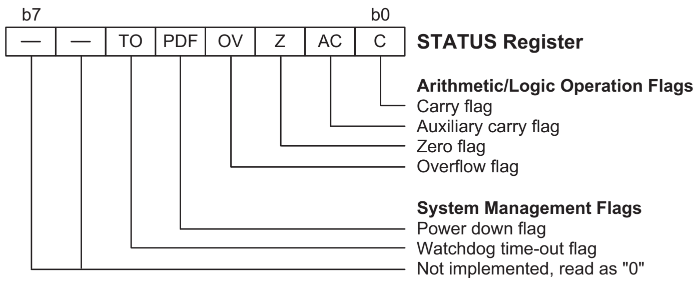
Status Register – STATUS

This 8-bit register contains the zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

With the exception of the TO and PDF flags, bits in the status register can be altered by instructions like most other registers. Any data written into the status register will not change the TO or PDF flag. In addition, operations related to the status register may give different results due to the different instruction operations. The TO flag can be affected only by a system power-up, a WDT time-out or by executing the ″CLR WDT″ or ″HALT″ instruction. The PDF flag is affected only by executing the ″HALT″ or ″CLR WDT″ instruction or during a system power-up.

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

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



Status Register

Downloaded from Elcodis.com electronic components distributor

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

Timer/Event Counter Registers – TMR0H/TMR1H, TMR0L/TMR1L,TMR0C/TMR1C

All devices possess two internal 16-bit count-up timer. An associated register pair known as TMR0L(TMR1L)/TMR0H(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 TMR0C(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~PF2. 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.0~PFC.2, 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 initialization, 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.

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 43 bidirectional input/output lines labeled with port names PA, PB, PC, PD, PE and PF0~PF2.

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

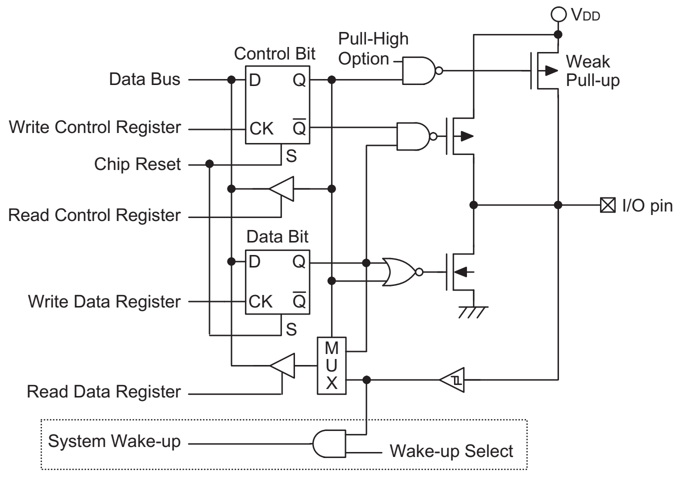
Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, I/O pins, 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. Each pin on all of I/O can be selected individually to have this pull-high resistors feature and each nibble on each of the other ports.

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 or low to high. After a HALT instruction forces the microcontroller into entering the Power Down Mode, the processor will remain idle or in a low-power state until the logic condition of the selected wake-up pin on Port pins changes from high to low or low to high. This function is especially suitable for applications that can be woken up via external switches. Note that each pin on PA, PB, PC, PD, PE and PF0~PF2 can be selected individually to have this wake-up feature.

Downloaded from Elcodis.com electronic components distributor



Generic Input/Output Structure

I/O Port Control Registers

Each I/O port has its own control register PAC, PBC, PCC, PDC, PEC and PFC.0~PFC.2, 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 PC2. 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 INTC0 register must be disabled.
- External Timer 0 Clock Input
  The external timer pin TMR0 is pin-shared with the I/O pin PA2. 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 PA3. 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.
- V1/V2 is for V-axis function
  The V1/V2 pins are pin shared with the PB0/PB1 pins, PB0, PB1 has falling and rising edge wake-up function, if it select can wake-up by configuration option. In HALT Mode if PB0 wake-up the V1-Wakeup [23H.4] will be set, if PB1 wake-up the V2-Wakeup [23H.5] will be set. If user read WSR register by software, the bit will be clear.
- Z1/Z2 is for Z-axis function
  The Z1/Z2 pins are pin shared with the PA0/PA1 pins, PA0, PA1 has falling and rising edge wake-up function, if it select can wake-up by configuration option. In halt mode if PA0 wake-up the Z1-Wakeup [23H.6] will be set, if PA1 wake-up the Z2-Wakeup [23H.7] will be set. If user WSR register by software, the bit will be clear.

Downloaded from Elcodis.com electronic components distributor

Output Pin Slew Rate Control

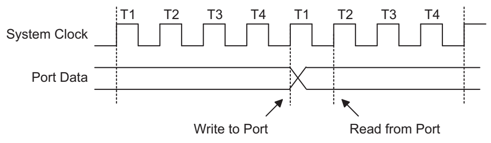
The output pin slew rate can be setup using a configuration option and can be set to be either 0ns, 50ns, 100ns or 200ns.

I/O Pin Structures

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

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, PCC, PDC, PEC and PFC.0~PFC.2 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 PF0~PF2 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 I/O pins has 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 or low to high transition of any of all I/O pins. Single or multiple pins on all I/O 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 16-bit capacities. 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 contain the actual value of the Timer/Event Counter and into which an initial value can be preloaded, and is known as TMR0H/TMR0L, TMR1H/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 sources 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 external timer pin has the name TMR0 or TMR1. 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

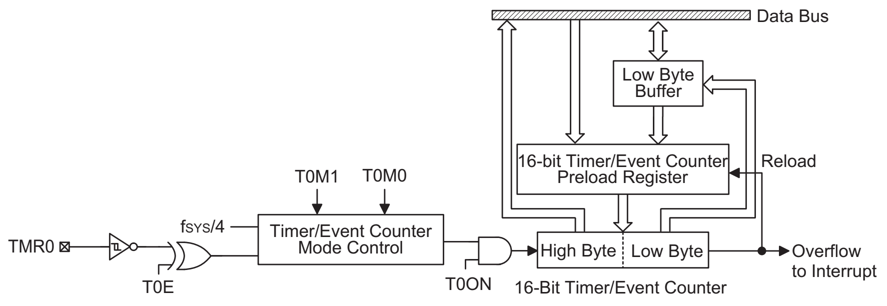
The Timer/Event Counter′s clock can originate from various sources. The instruction clock source or WDTOSC (system clock source divided by 4) is used when the Timer/Event Counter 0 or Timer/Event Counter 1 is in the timer mode or in the pulse width measurement mode. The external clock source is used 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 Registers − TMR0H/TMR0L, TMR1H/TMR1L

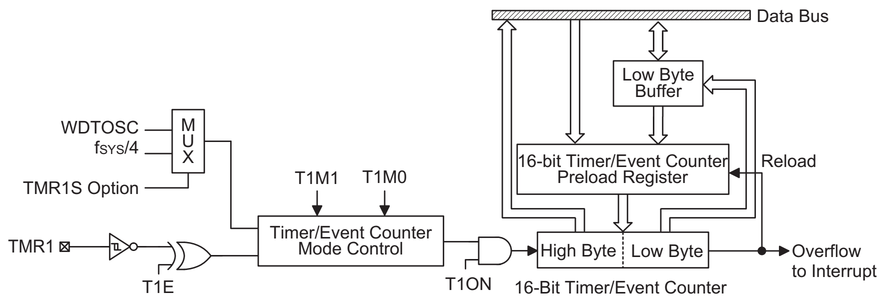
The timer registers are special function registers located in the Special Purpose RAM Data Memory and are the places where the actual timer values are stored. The timer registers are known as TMR0L/ TMR0H, TMR1L /TMR1H. The value in the timer registers increases by one each time an internal clock pulse 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 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 FFFFH, the preload registers must first be cleared to all zeros. It should be noted that after power-on, the preload register

Downloaded from Elcodis.com electronic components distributor


Data Bus
Low Byte Buffer
16-bit Timer/Event Counter Preload Register
Reload
T0M1
T0M0
fSYS/4
TMR0
Timer/Event Counter Mode Control
High Byte
Low Byte
Overflow to Interrupt
T0E
T0ON
16-Bit Timer/Event Counter

16-bit Timer/Event Counter 0 Structure



16-bit Timer/Event Counter 1 Structure

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.

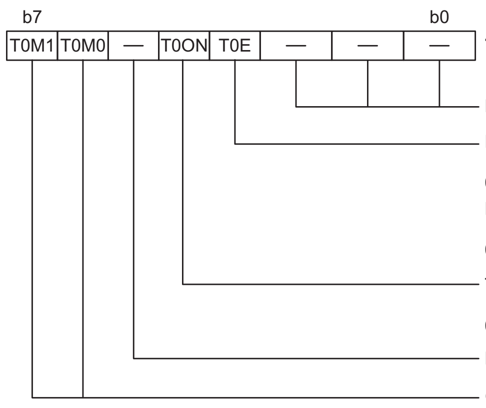
For the 16-bit Timer/Event Counter which has both low byte and high byte timer registers, accessing these registers is carried out in a specific way. It must be note when using instructions to preload data into the low byte timer register, namely TMR0L/TMR1L, the data will only be placed in a low byte buffer and not directly into the low byte timer register. The actual transfer of the data into the low byte timer register is only carried out when a write to its associated high byte timer register, namely TMR0H/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 control register, which has the name TMR0C or TMR1C. It is the Timer Control Register together with its corresponding timer register that control the full operation of the Timer/Event Counter. Before the Timer/Event Counter can be used, it is essential that the Timer Control Register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program initialisation.

To choose which of the three modes the Timer/Event Counter 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, must be set to the required logic levels. The Timer/Event Counter on/off bit, which is bit 4 of the Timer Control Register and known as T0ON or T1ON, provides the basic on/off control of the Timer/Event Counter. Setting the

Downloaded from Elcodis.com electronic components distributor



TMR0C Register

Not implemented, read as "0"

Event Counter Active Edge Select
1: count on falling edge
0: count on rising edge
Pulse Width Measurement Active Edge Select
1: start counting on rising edge, stop on falling edge
0: start counting on falling edge, stop on rising edge

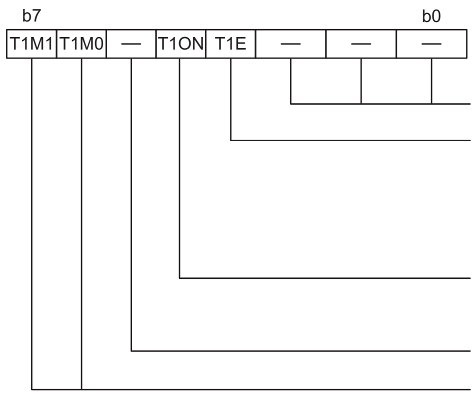
Timer/Event Counter Counting Enable
1: enable
0: disable

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



TMR1C Register

Not implemented, read as "0"

Event Counter Active Edge Select
1: count on falling edge
0: count on rising edge
Pulse Width Measurement Active Edge Select
1: start counting on rising edge, stop on falling edge
0: start counting on falling edge, stop on rising edge

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

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

Downloaded from Elcodis.com electronic components distributor

bit high allows the Timer/Event Counter to run, clearing the bit stops it running. If the Timer/Event Counter 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.

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 is used as the internal clock for the Timer/Event Counter. 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 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

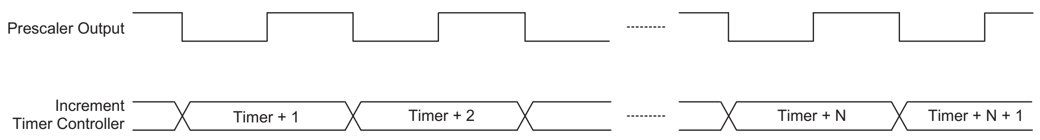
In this mode, the external timer pin, TMR0 or TMR1, is used as the Timer/Event Counter clock source, however it is not divided by the internal prescaler. After the other bits in the Timer Control Register have been setup, the enable bit T0ON or T1ON, which is bit 4 of the Timer Control Register, can 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 an independent pin and not shared with an I/O pin, the only thing to ensure the timer operate as an event counter is to ensure that the Operating Mode Select bits in the Timer Control Register place the Timer/Event Counter in the Event Counting Mode. 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.

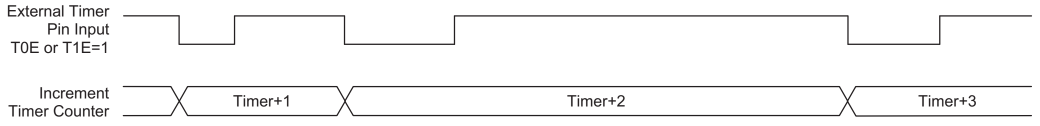
Configuring the Pulse Width Measurement Mode

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 value as shown.

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



Timer Mode Timing Chart



Event Counter Mode Timing Chart

Downloaded from Elcodis.com electronic components distributor

In this mode the internal clock, $f_{SYS}/4$ is used as the internal clock for the 16-bit 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 ensuring that the Timer/Event Counter Interrupt Enable bit in the Interrupt Control Register, INTC0, is reset to zero.

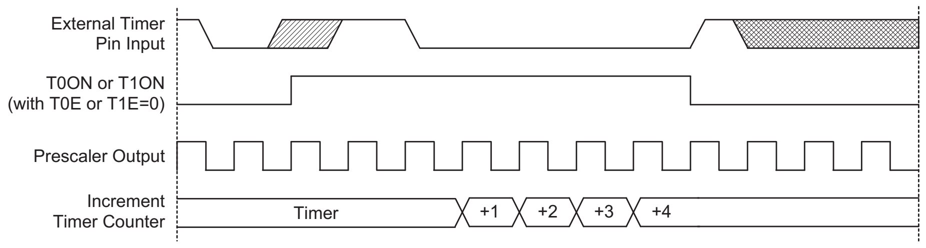
I/O Interfacing

The Timer/Event Counter, when configured to run in the event counter or pulse width measurement mode, requires the use of an external pin for correct operation. 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.

Programming Considerations

When configured to run in the timer mode, an internal timer clock source is used. 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 instruction 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 registers, 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 interrupt 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 register before the timer is switched on; this is because after power-on the initial value of the timer register is 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



Prescaler Output is sampled at every falling edge of T1.

Pulse Width Measure Mode Timing Chart

Downloaded from Elcodis.com electronic components distributor

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 timer interrupt is 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 tobe in the timer mode, which uses the internal system clock as the clock source.

```
org 04h        ; external interrupt vector
reti
org 08h        ; Timer/Event Counter 0 interrupt vector
jmp tmr0int    ; jump here when Timer/Event Counter 0 overflows
org 0ch        ; Timer/Event Counter 1 interrupt vector
jmp tmr1int    ; jump here when Timer/Event Counter 1 overflows
    :
org 20h        ; main program
    :
;internal Timer/Event Counter 0 interrupt routine
tmr0int:
    :
; Timer/Event Counter 0 main program placed here
    :
    reti
    :
;internal Timer/Event Counter 1 interrupt routine
tmr1int:
    :
; Timer/Event Counter 1 main program placed here
    :
    reti
    :
begin:
;setup Timer/Event Counter 0 registers
mov a,0e8h     ; setup low byte preload value for Timer/Event Counter 0
mov tmr0l,a    ; low byte must be setup before high byte
mov a,09bh     ; setup high byte preload value for Timer/Event Counter 0
mov tmr0h,a    ;
mov a,080h     ; setup Timer control register TMR0C
mov tmr0c,a    ; Timer/Event Counter 0 has no prescaler and clock source is fSYS/4

;setup Timer/Event Counter 1 registers
mov a,09bh     ; setup low byte preload value for Timer/Event Counter 1
mov tmr1l,a    ; low byte must be setup before high byte
mov a,0e8h     ; setup high byte preload value for Timer/Event Counter 1
mov tmr1h,a    ;
mov a,080h     ; setup Timer control register TMR1C
mov tmr1c,a    ; Timer/Event Counter 1 has no prescaler and clock source is fSYS/4

; setup interrupt register
mov a,00dh     ; enable master interrupt and timer interrupts
mov intc,a
    :
set tmr0c.4    ; start Timer/Event Counter 0 – note mode bits must be previously setup
set tmr1c.4    ; start Timer/Event Counter 1 – note mode bits must be previously setup
```

Downloaded from Elcodis.com electronic components distributor

Interrupts

Interrupts are an important part of any microcontroller system. When an external interrupt pin transition or two internal function such as a Timer/Event Counter overflow, a transmission or reception of SPI data occurs, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. Each device contains one external interrupts and several internal interrupts functions. The external interrupt is controlled by the action of the external interrupt pins, while the internal interrupts are controlled by the Timer/Event Counter overflow and SPI data transmission or reception.

Interrupt Register

Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by the two interrupt control registers, which are located in the Data Memory. 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

Two Timer/Event Counter overflow, 16-bits of data transmission or reception on either of the one SPI interfaces or an active edge on any of the one external interrupt pins will all generate an interrupt request by setting their corresponding request flag, if their appropriate interrupt enable bit is set. When this happens, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a JMP statement which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI statement, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

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.

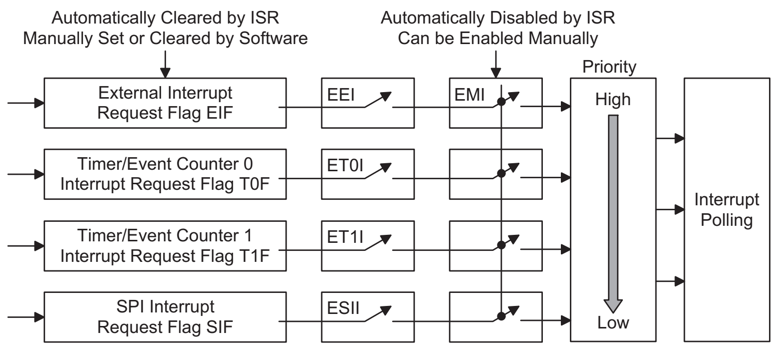
Interrupt Priority

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.

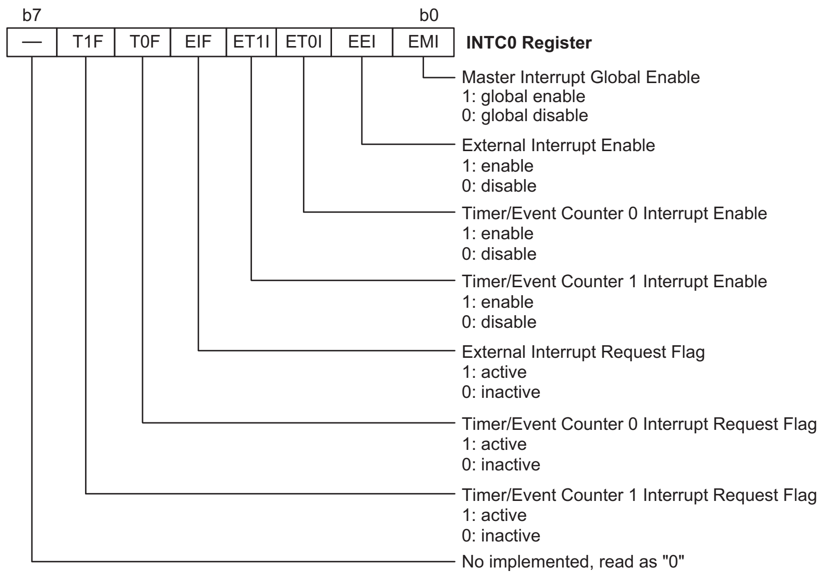
Interrupt Source	Priority	Vector
External Interrupt $\overline{\text{INT}}$	1	0004H
Timer/Event Counter 0 Overflow Interrupt	2	0008H
Timer/Event Counter 1 Overflow Interrupt	3	000CH
SPI Interrupt	4	0010H

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.

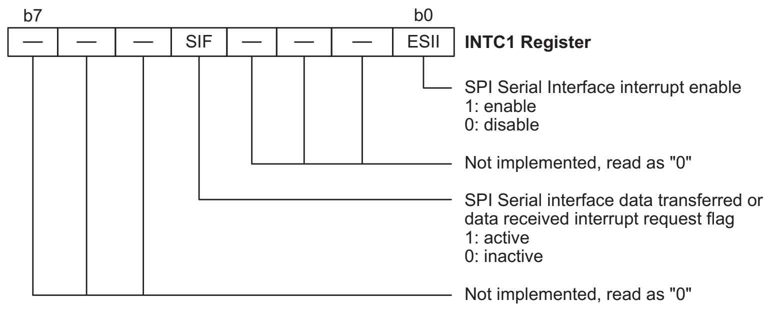
Downloaded from Elcodis.com electronic components distributor


Automatically Cleared by ISR
Manually Set or Cleared by Software
Automatically Disabled by ISR
Can be Enabled Manually
Priority
External Interrupt Request Flag EIF
Timer/Event Counter 0 Interrupt Request Flag T0F
Timer/Event Counter 1 Interrupt Request Flag T1F
SPI Interrupt Request Flag SIF
EEI
ET0I
ET1I
ESII
EMI
High
Low
Interrupt Polling

Interrupt Structure


b7
b0
— T1F T0F EIF ET1I ET0I EEI EMI
INTC0 Register
Master Interrupt Global Enable
1: global enable
0: global disable
External 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
External 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
No implemented, read as "0"

INTC0 Register


b7
b0
— — — SIF — — — ESII
INTC1 Register
SPI Serial Interface interrupt enable
1: enable
0: disable
Not implemented, read as "0"
SPI Serial interface data transferred or data received interrupt request flag
1: active
0: inactive
Not implemented, read as "0"

INTC1 Register

Downloaded from Elcodis.com electronic components distributor

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 high to low transition appears on the interrupt pins. The external interrupt pin is pin-shared with the I/O pins PC2 can only be configured as an external interrupt pin if the corresponding external interrupt enable bits in the interrupt control register INTC0 have been set. The pins must also be setup as inputs by setting the corresponding PCC.2 bits in the port control register. When the interrupt is enabled, the stack is not full and a high to low transition appears on the external interrupt pin, a subroutine call to the external interrupt vector at location 04H will take place. When the interrupt is serviced, the external interrupt request flag, EIF will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor configuration options on these pins will remain valid even if the pins are used as external interrupt inputs.

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 or ET1I, must first be set. An actual Timer/Event Counter interrupt will take place when the Timer/Event Counter interrupt request flag, T0F or 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 or 0CH, will take place. When the interrupt is serviced, the timer interrupt request flag, T0F or T1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

SPI Interrupt

For an SPI Interrupt to occur, the global interrupt enable bit, EMI, and the corresponding SPI interrupt enable bit, ESII, must be first set. The SBEN bit in the SBCR register must also be set. An actual SPI Interrupt will take place when one of the one SPI interrupt request flags, SIF, is set, a situation that will occur when 8-bits of data are transferred or received from either of the SPI interfaces. When the interrupt is enabled, the stack is not full and an SPI interrupt occurs, a subroutine call to the SPI interrupt vector at location 10H, will take place. When the interrupt is serviced, the SPI interrupt request flag, SIF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Programming Considerations

By disabling the interrupt enable bits, a requested interrupt can be prevented from being serviced, however, once an interrupt request flag is set, it will remain in this condition in the interrupt 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.

Downloaded from Elcodis.com electronic components distributor

Reset and Initialisation

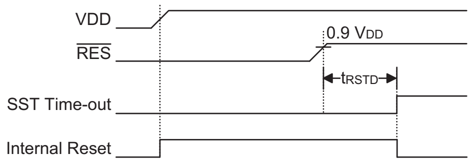
A reset function is a fundamental part of any microcontroller ensuring that the device can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the microcontroller. In this case, internal circuitry will ensure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

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

Reset Functions

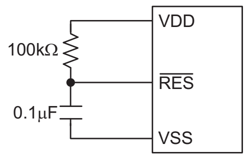
There are five ways in which a microcontroller reset can occur, through events occurring both internally and externally:

- 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.



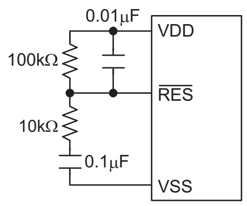
Power-On Reset Timing Chart

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.

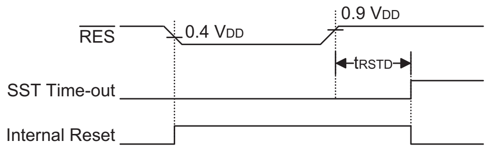


Enhanced Reset Circuit

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

Downloaded from Elcodis.com electronic components distributor

- $\overline{\text{RES}}$ 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.

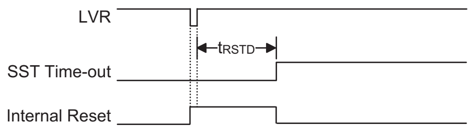


$\overline{\text{RES}}$ **Reset Timing Chart**

- 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″.
- 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.

Reset Initial Conditions

The different types of reset described affect the reset



Low Voltage Reset Timing Chart

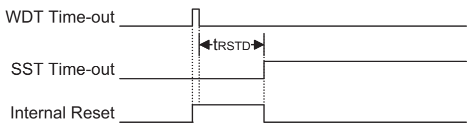
flags in different ways. These flags, known as PDF and TO are located in the status register and are controlled by various microcontroller operations, such as the Power Down function or Watchdog Timer. The reset flags are shown in the table:

TO	PDF	RESET Conditions
0	0	$\overline{\text{RES}}$ reset during power-on
0	1	$\overline{\text{RES}}$ wake-up during Power Down
u	u	$\overline{\text{RES}}$ 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



WDT Time-out Reset during Normal Operation Timing Chart

Downloaded from Elcodis.com electronic components distributor

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)*
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	–xxx xxxx	–uuu uuuu	–uuu uuuu	–uuu uuuu	–uuu uuuu
WDTS	–––– –111	–––– –111	–––– –111	–––– –111	–––– –uuu
STATUS	––00 xxxx	––1u uuuu	––uu uuuu	––01 uuuu	––11 uuuu
INTC0	–000 0000	–000 0000	–000 0000	–000 0000	–uuu uuuu
TMR0H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	00–0 1000	00–0 1000	00–0 1000	00–0 1000	uu–u uuuu
TMR1H	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1L	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	00–0 1000	00–0 1000	00–0 1000	00–0 1000	uu–u uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PE	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PF	–––– –111	–––– –111	–––– –111	–––– –111	uuuu uuuu
PFC	–––– –111	–––– –111	–––– –111	–––– –111	uuuu uuuu
INTC1	–––0 –––0	–––0 –––0	–––0 –––0	–––0 –––0	–––u –––u
TBHP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
SBCR	0110 0000	0110 0000	0110 0000	0110 0000	uuuu uuuu
SBDR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
WSR	0000 0000	uuuu 0000	uuuu 0000	uuuu 0000	uuuu uuuu
CTLR	0000 x000	0000 x000	0000 x000	0000 x000	uuuu xuu0

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

Downloaded from Elcodis.com electronic components distributor

Oscillator

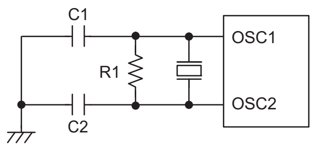
There are two oscillator circuits contained within the device. The first is the system oscillator which utilises an external crystal or RC and the second is the Watchdog timer oscillator which is fully integrated and requires no external components.

System Clock Configurations

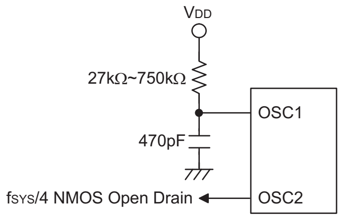
There are two oscillator mode Crystal and RC. For Crystal mode no built-in capacitor between OSC1, OSC2 and GND. The simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation, without requiring external capacitors. However, for some crystal types and frequencies, to ensure oscillation, it may be necessary to add two small value capacitors, C1 and C2. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be connected as shown for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. In most applications, resistor R1 is not required, R1 may be necessary to ensure the oscillator stops running when VDD falls below its operating range.

External RC Oscillator

Using the external system RC oscillator requires that a resistor, with a value between 27kΩ and 750kΩ, is connected between OSC1 and ground, and a capacitor is connected to VDD. The generated system clock divided by 4 will be provided on OSC2 as an output which can be used for external synchronization purposes. Note that as the OSC2 output is an NMOS open-drain type, a pull high resistor should be connected if it to be used to monitor the internal frequency. Although this is a cost effective oscillator configuration, the oscillation frequency can vary with VDD, temperature and process variations and is therefore not suitable for applications where timing is critical or where accurate oscillator frequencies are required.For the value of the external resistor ROSC refer to the Holtek website for typical RC Oscillator vs. Temperature and VDD characteristics graphics. Note that it is the only microcontroller internal circuitry together with the external resistor, that determine the frequency of the oscillator. The external capacitor shown on the diagram does not influence the frequency of oscillation.



Crystal/Ceramic Oscillator



RC Oscillator

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

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 come from the WDT 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.

Downloaded from Elcodis.com electronic components distributor

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.

Wake-up

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

- An external reset
- An external falling edge on any of the I/O pins
- 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.

Configuration options determine which pin or groups of pins can be setup to permit a negative transition on the pin to wake-up the system. When a Port pin wake-up occurs, the program will resume execution at the instruction following the ″HALT″ instruction.

When a PA0/PA1 or PB0/PB1 wake up occurs, bits in the WSR register can be read to know which pin changed first.

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.

Low Voltage Detector – LVD

The Low Voltage Detector internal function provides a means for the user to monitor when the power supply voltage falls below a certain fixed level as specified in the DC characteristics.

Operation

The LVD enable/disable control bit is bit 4 of the CTLR register. Under normal operation, and when the power supply voltage is above the specified VLVD value, specified by the LVD_sel bits in the CTLR register, the Low battery bit will remain at a zero value. If the power supply voltage should fall below this V_{LVD} value then the Low battery bit will change to a high value indicating a low voltage condition. Note that the Low battery bit is a read-only bit. By polling the Low battery bit in the CTLR register, the application program can therefore determine the presence of a low voltage condition.

Downloaded from Elcodis.com electronic components distributor

Watchdog Timer

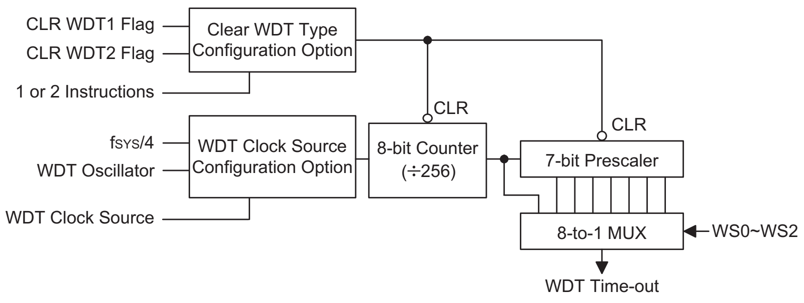
The Watchdog Timer is provided to prevent program malfunctions or sequences from jumping to unknown locations, due to certain uncontrollable external events such as electrical noise. It operates by providing a device reset when the WDT counter overflows. The WDT clock is supplied by its own internal dedicated internal WDT oscillator or by the fSYS/4 clock source. Note that if the WDT enable/disable configuration option has been disabled, then any instruction relating to its operation will result in no operation.

The WDT enable/disable is controlled using a bit in the CTLR register. The WDT clock source and clear instruction type are selected through configuration options. However, it should be noted that the WDT oscillator clock period can vary with VDD, temperature and process variations. Whether the WDT clock source is its own internal WDT oscillator, it is further divided by an internal 6-bit counter and a clearable single bit counter to give longer Watchdog time-outs. As the clear instruction only resets the last stage of the divider chain, for this reason the actual division ratio and corresponding Watchdog Timer time-out can vary by a factor of two.

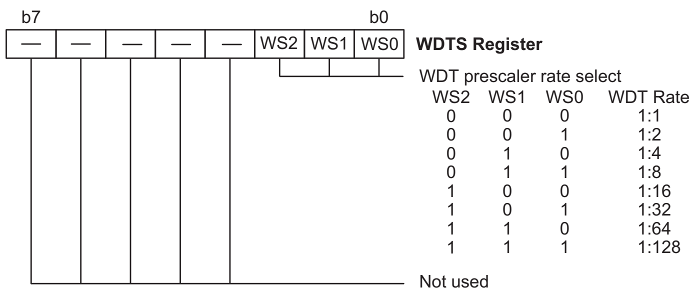
The exact division ratio depends upon the residual value in the Watchdog Timer counter before the clear instruction is executed. It is important to realise that as there are no independent internal registers or configuration options associated with the length of the Watchdog Timer time-out, it is completely dependent upon the frequency the internal WDT oscillator.

Under normal program operation, a WDT time-out will initialise a device reset and set the status bit TO. However, if the system is in the Power Down Mode, when a WDT time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will be reset. Three methods can be adopted to clear the contents of the WDT. The first is an external hardware reset, which means a low level on the $\overline{RES}$ pin, the second is using the watchdog software instructions and the third is via a HALT instruction.

There are two methods of using software instructions to clear the Watchdog Timer, one of which must be chosen by configuration option. The first option is to use the single ″CLR WDT″ instruction while the second is to use the two commands ″CLR WDT1″ and ″CLR WDT2″. For the first option, a simple execution of ″CLR WDT″ will clear the WDT while for the second option, both ″CLR WDT1″ and ″CLR WDT2″ must both be executed to successfully clear the WDT. Note that for this second option, if ″CLR WDT1″ is used to clear the WDT, successive executions of this instruction will have no effect, only the execution of a ″CLR WDT2″ instruction will clear the WDT. Similarly after the ″CLR WDT2″ instruction has been executed, only a successive ″CLR WDT1″ instruction can clear the Watchdog Timer.



Watchdog Timer



WS2	WS1	WS0	WDT Rate
0	0	0	1:1
0	0	1	1:2
0	1	0	1:4
0	1	1	1:8
1	0	0	1:16
1	0	1	1:32
1	1	0	1:64
1	1	1	1:128

Watchdog Timer Register

Downloaded from Elcodis.com electronic components distributor

Bit	Name	Description
7	Z2_WAKEUP	Read only bit. 1: Z2 changed before Z1 0: default
6	Z1_WAKEUP	Read only bit. 1: Z1 changed before Z2 0: default
5	V2_WAKEUP	Read only bit. 1: V2 changed before V1 0: default
4	V1_WAKEUP	Read only bit. 1: V1 changed before V2 0: default
3	SPI_EN	This bit control the shared PIN ($\overline{\text{SCS}}$, SDI, SDO and SCK) is SPI or GPIO mode 1: SPI mode 0: IO mode (default)
2	SPI_CSEN	1: Enable, this bit is used to enable/disable software CSEN function 0: Disable, $\overline{\text{SCS}}$ define as GPIO (default)
1	CKEG	1: SPI first output the data immediately after the SPI is enable. And SPI output the data in the falling edge(polarity=0) or rising edge (polarity=1); SPI read data in the in the rising edge (polarity=0) or falling edge (polarity=1); 0: SPI output the data in the rising edge(polarity=0) or falling edge (polarity=1); SPI read data in the in the falling edge(polarity=0) or rising edge (polarity=1); (default)
0	SPI_ CPOL	1: clock polarity rising 0: clock polarity falling (default)

Note: The Internal Register bit4~bit7 data will clear to zero after F/W read the register.

Wake-up Status Register – WSR

Bit	Name	Description
7 6 5	LVD_sel	To Selected low voltage detector level, Bit 7, 6, 5= 000: 1.0V 001: 1.2V 010: 1.4V 011: 2.0V 100: 2.4V 101: 2.7V 110: 3.0V 111: 3.3V
4	LVD_rd_ctrl	To control the DC/DC to check the LVD voltage 1: enable LVD 0: disable LVD (default)
3	Low battery	Flag for battery low signal (error 5%) 1: battery voltage $\leq$ according LVD_sel voltage 0: battery voltage > according LVD_sel voltage The user should wait at least 20μs after set LVD_rd_ctrl and then read the corresponding voltage Low battery signal
2	WDTEN	To control the Watchdog timer 1: enable 0: disable
1	TMR1S	To selected Timer 1 source 1: WDT OSC 0: $f_{SYS}/4$ Where WDT OSC is selected as TMR1 source. WDT OSC is always enabled.
0	—	Unimplemented, read as ″0″

Control Register – CTLR

Downloaded from Elcodis.com electronic components distributor

SPI Serial Interface

The device include single SPI Serial Interfaces. The SPI interface is a full duplex serial data link, originally designed by Motorola, which allows multiple devices connected to the same SPI bus to communicate with each other. The devices communicate using a master/slave technique where only the single master device can initiate a data transfer. A simple four line signal bus is used for all communication.

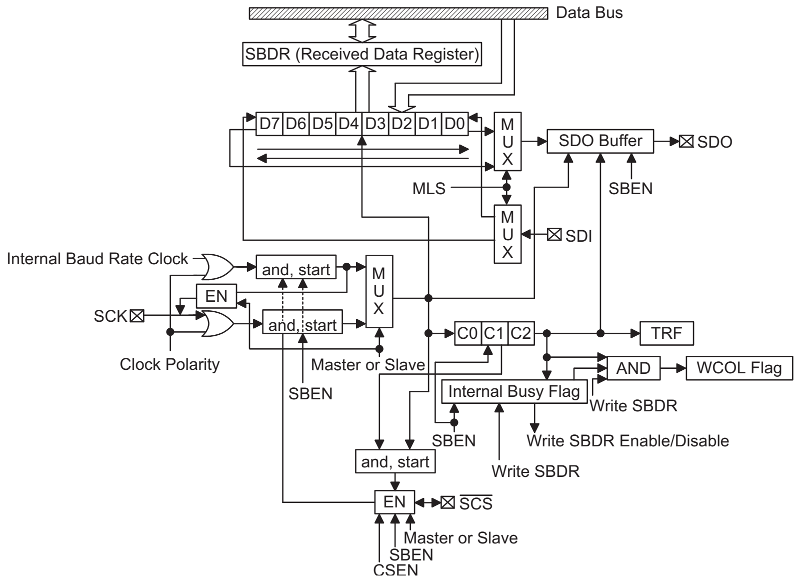
SPI Interface Communication

Four lines are used for SPI communication known as SDI - Serial Data Input, SDO - Serial Data Output, SCK - Serial Clock and $\overline{SCS}$ - Slave Select. Note that the condition of the Slave Select line is conditioned by the CSEN bit in the SBCR control register. If the CSEN bit is high then the $\overline{SCS}$ line is active while if the bit is low then the $\overline{SCS}$ line will be in a floating condition. The following timing diagram depicts the basic timing protocol of the SPI bus.

Registers

There are three registers associated with the SPI Interface. These are the SBCR register which is the control register and the SBDR which is the data register and WSR low nibble byte which is the SPI mode control register. The SBCR register is used to setup the required setup parameters for the SPI bus and also used to store associated operating flags, while the SBDR register is used for data storage.

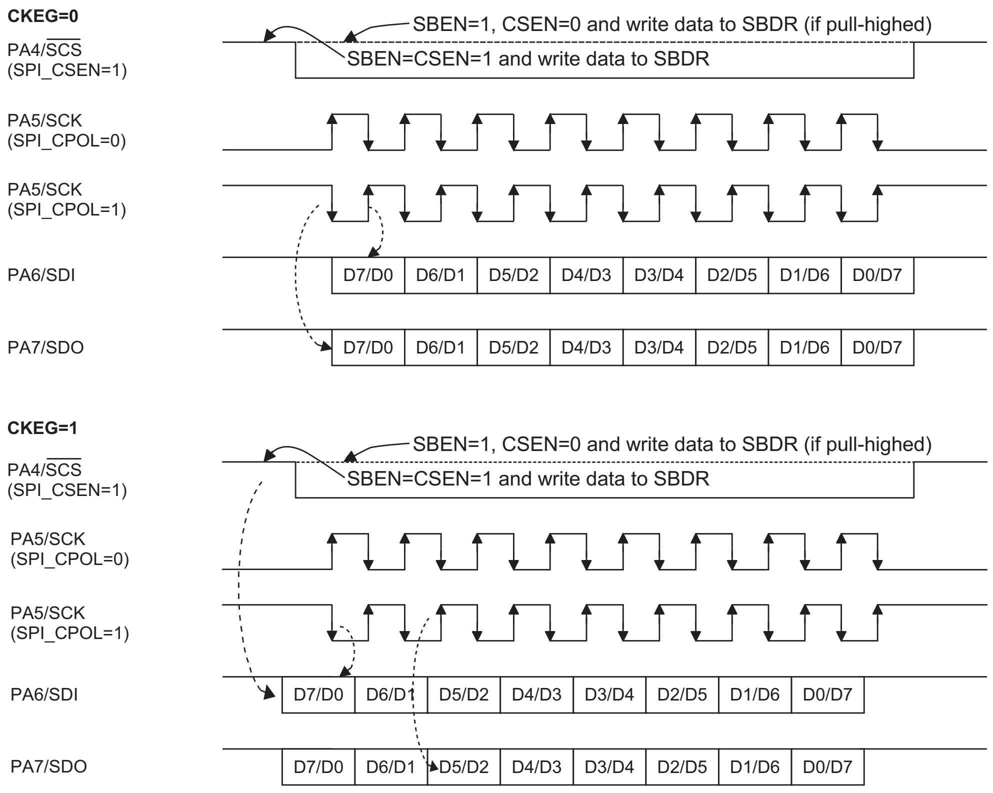
The WSR register low nibble byte is used to select SPI mode, clock polarity edge selection and SPI enable or disable selection.



SPI Block Diagram

Note: WCOL: set by SPI cleared by users
CSEN: enable/disable chip selection function pin
master mode: 1/0 = with/without $\overline{SCS}$ output function
Slave mode: 1/0 = with/without $\overline{SCS}$ input control function
SBEN: enable/disable serial bus (0: initialise all status flags)
when SBEN=0, all status flags should be initialised
when SBEN=1, all SPI related function pins should stay at floating state
TRF: 1 = data transmitted or received, 0= data is transmitting or still not received
CPOL: I/O = clock polarity rising/falling edge: WSR register bit 0
If clock polarity set to rising edge (SPI_CPOL=1), serial clock timing follow SCK, otherwise (SPI_CPOL=0) SCK is the serial clock timing.

Downloaded from Elcodis.com electronic components distributor


CKEG=0
PA4/SCS (SPI_CSEN=1)
SBEN=1, CSEN=0 and write data to SBDR (if pull-highed)
SBEN=CSEN=1 and write data to SBDR
PA5/SCK (SPI_CPOL=0)
PA5/SCK (SPI_CPOL=1)
PA6/SDI
PA7/SDO
D7/D0 D6/D1 D5/D2 D4/D3 D3/D4 D2/D5 D1/D6 D0/D7
CKEG=1
PA4/SCS (SPI_CSEN=1)
SBEN=1, CSEN=0 and write data to SBDR (if pull-highed)
SBEN=CSEN=1 and write data to SBDR
PA5/SCK (SPI_CPOL=0)
PA5/SCK (SPI_CPOL=1)
PA6/SDI
PA7/SDO
D7/D0 D6/D1 D5/D2 D4/D3 D3/D4 D2/D5 D1/D6 D0/D7

SPI Bus Timing

Downloaded from Elcodis.com electronic components distributor

After Power on, the contents of the SBDR register will be in an unknown condition while the SBCR register will default to the condition below:

CKS	M1	M0	SBEN	MLS	CSEN	WCOL	TRF
0	1	1	0	0	0	0	0

Note that data written to the SBDR register will only be written to the TXRX buffer, whereas data read from the SBDR register will actual be read from the register.

Bus Enable/Disable

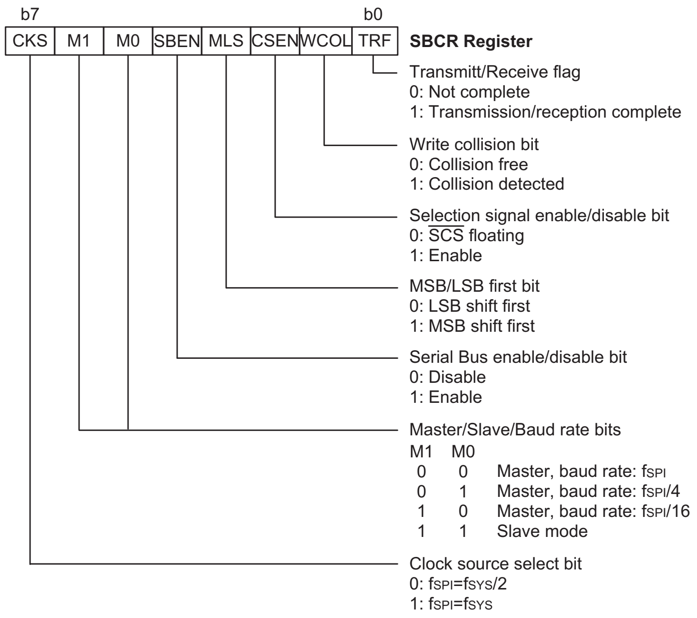
To enable the SPI bus, SBEN should be set high, then SCK, SDI, SDO and $\overline{\text{SCS}}$ lines should all be zero, then wait for data to be written to the SBDR (TXRX buffer) register. For the Master Mode, after data has been written to the SBDR (TXRX buffer) register then transmission or reception will start automatically. When all the data has been transferred the TRF bit should be set. For the Slave Mode, when clock pulses are received on SCK, data in the TXRX buffer will be shifted out or data on SDI will be shifted in.

To Disable the SPI bus SCK, SDI, SDO, $\overline{\text{SCS}}$ should be floating.

Operation

All communication is carried out using the 4-line interface for both Master or Slave Mode. The timing diagram shows the basic operation of the bus.

The CSEN bit in the SBCR register controls the $\overline{\text{SCS}}$ line of the SPI interface. Setting this bit high, will enable the SPI interface by allowing the $\overline{\text{SCS}}$ line to be active, which can then be used to control the SPI interface. If the CSEN bit is low, the $\overline{\text{SCS}}$ line will be in a floating condition and can therefore not be used for control of the SPI interface. The SBEN bit in the SBCR register must also be high which will place the SDI line in a floating condition and the SDO line high. If in the Master Mode the SCK line will be either high or low depending upon the clock polarity configuration option. If in the Slave Mode the SCK line will be in a floating condition. If SBEN is low then the bus will be disabled and $\overline{\text{SCS}}$, SDI, SDO and SCK will all be in a floating condition.



SPI Interface Control Register

Downloaded from Elcodis.com electronic components distributor

In the Master Mode, the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written to the SBDR register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission or reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode:

- Master Mode
  Step 1. Select the clock source using the bit in the SBCR control register
  Step 2. Setup the M0 and M1 bits in the SBCR control register to select the Master Mode and the required Baud rate. Values of 00, 01 or 10 can be selected.
  Step 3. Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this must be same as the Slave device.
  Step 4. Setup the SBEN bit in the SBCR control register to enable the SPI interface.
  Step 5. For write operations: write the data to the SBDR register, which will actually place the data into the TXRX buffer. Then use the SCK and $\overline{\text{SCS}}$ lines to output the data.
  Goto to step 6.For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SBDR register.
  Step 6. Check the WCOL bit, if set high then a collision error has occurred so return to step5. If equal to zero then go to the following step.
  Step 7. Check the bit or wait for an SPI serial bus interrupt.
  Step 8. Read data from the SBDR register
  Step 9. Clear
  Step10. step 5

- Slave Mode
  Step 1. The CKS bit has a don't care value in the slave mode.
  Step 2. Setup the M0 and M1 bits to 11 to select the Slave Mode. The CKS bit is don't care.
  Step 3. Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this must be same as the Master device.
  Step 4. Setup the SBEN bit in the SBCR control register to enable the SPI interface.
  Step 5. For write operations: write data to the SBCR register, which will actually place the data into the TXRX register, then wait for the master clock and $\overline{\text{SCS}}$ signal. After this goto step 6.
  For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SBDR register.
  Step 6. Check the WCOL bit, if set high then a collision error has occurred so return to step5. If equal to zero then go to the following step.
  Step 7. Check the bit or wait for an SPI serial bus interrupt.
  Step 8. Read data from the SBDR register
  Step 9. Clear
  Step10. step 5

SPI Configuration Options and Status Control

Several configuration options exist for the SPI Interface function which must be setup during device programming. One option is to enable the operation of the WCOL, write collision bit, in the SBCR register. Another option exists to select the clock polarity of the SCK line. A configuration option also exists to disable or enable the operation of the CSEN bit in the SBCR register. If the configuration option disables the CSEN bit then this bit cannot be used to affect overall control of the SPI Interface.

SPI include four pins , can share I/O mode status . The status control combine with four bits for WSR and SBCR register. Include SPI_CSEN , SPI_EN for WSR register and CSEN, SBEN for SBCR register.

Control Bit for Register				SPI Share Function Pins Status			
SPI_EN	**SPI_CSEN**	**SBEN**	**CSEN**	**$\overline{\text{SCS}}$**	**SCK**	**SDO**	**SDI**
0	x	x	x	I/O mode	I/O mode	I/O mode	I/O mode
1	0	0	x	I/O mode	SPI mode (Z)	SPI mode (Z)	SPI mode (Z)
1	0	1	x	I/O mode	SPI mode	SPI mode	SPI mode (Z)
1	1	0	x	SPI mode (Z)	SPI mode (Z)	SPI mode (Z)	SPI mode (Z)
1	1	1	0	SPI mode (Z)	SPI mode	SPI mode	SPI mode (Z)
1	1	1	1	SPI mode	SPI mode	SPI mode	SPI mode (Z)

Note: X: don't care
(Z) floating

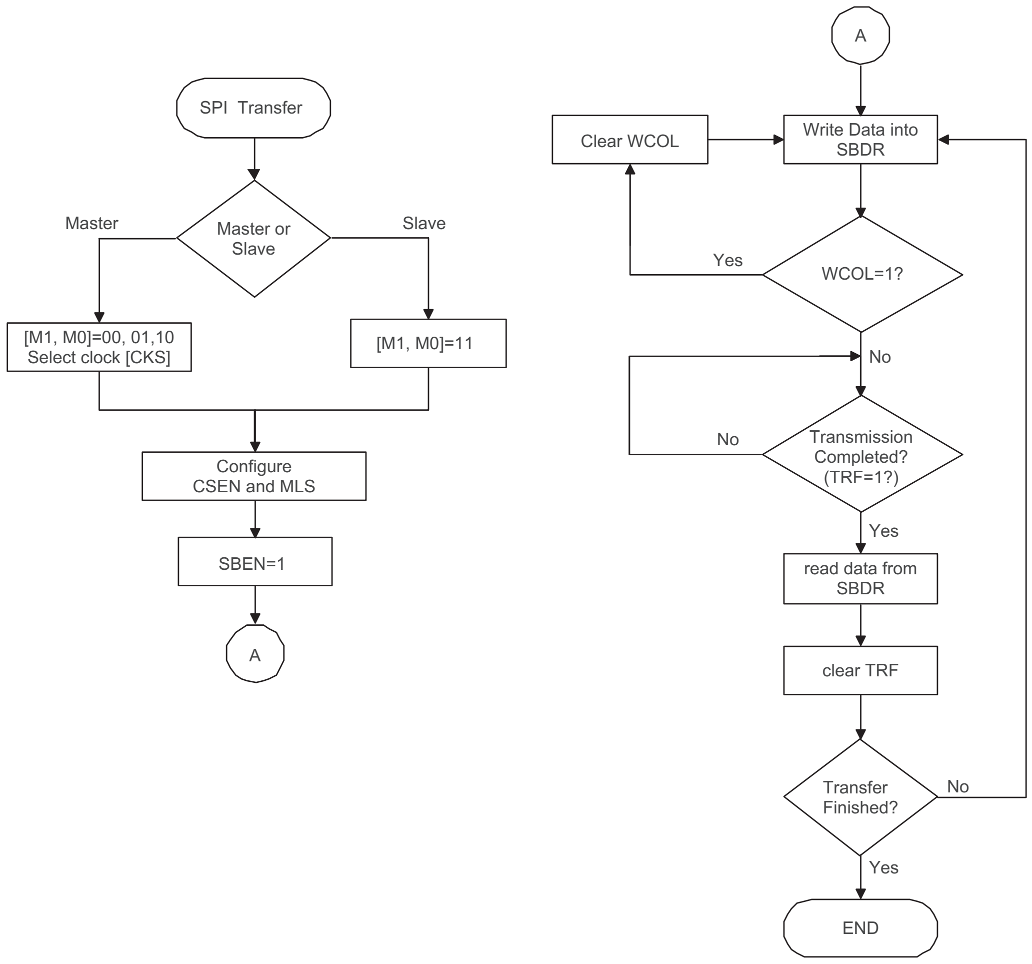
Downloaded from Elcodis.com electronic components distributor

Error Detection

The WCOL bit in the SBCR register is provided to indicate errors during data transfer. The bit is set by the Serial Interface but must be cleared by the application program. This bit indicates a data collision has occurred which happens if a write to the SBDR register takes place during a data transfer operation and will prevent the write operation from continuing. The bit will be set high by the Serial Interface but has to be cleared by the user application program. The overall function of the WCOL bit can be disabled or enabled by a configuration option.

Programming Considerations

When the device is placed into the Power Down Mode note that data reception and transmission will continue. The TRF bit is used to generate an interrupt when the data has been transferred or received.



SPI Transfer Control Flowchart

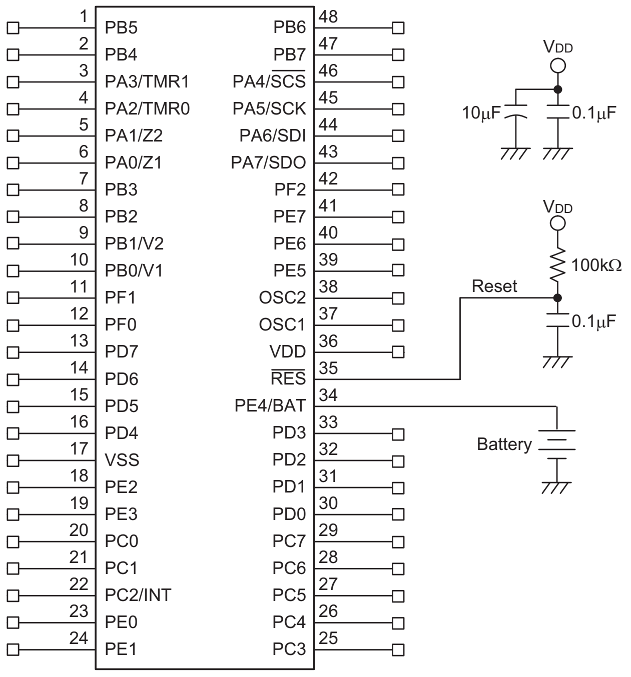
Downloaded from Elcodis.com electronic components distributor

Configuration Options

Item	Options
I/O Options	
1	PA pull-high: enable or disable by bit
2	PB, PC, PD, PE, PF0~PF2 pull-high: enable or disable by nibble
3	PB, PC, PD, PE, PF0~PF2: Schmitt Trigger or Non-Schmitt Trigger by nibble
4	PA wake-up: enable or disable by bit
5	PB0, PB1 wake-up: enable or disable by bit
6	PB2~PB7, PC, PD, PE, PF0~PF2 wake-up: enable or disable by nibble
7	PA input type Schmitt Trigger and Non-Schmitt Trigger by bit.
8	PE4 function option: PE4 as battery LVD input or PE4 as GPIO (default)
9	Output slew rate select: 0ns, 50ns, 100ns or 200ns
10	PA, NMOS or CMOS by bit
11	TBHP: enable or disable
Oscillator Options	
12	OSC type selection: RC or crystal
Watchdog Options	
13	CLRWDT instructions: one or two instructions
14	WDT Clock Source: $f_{sys}/4$ or WDT oscillator

Downloaded from Elcodis.com electronic components distributor

Application Circuits



Downloaded from Elcodis.com electronic components distributor

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.

Downloaded from Elcodis.com electronic components distributor

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

Downloaded from Elcodis.com electronic components distributor

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

Downloaded from Elcodis.com electronic components distributor

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

Downloaded from Elcodis.com electronic components distributor

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

Downloaded from Elcodis.com electronic components distributor

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] ← ACC + 00H or [m] ← ACC + 06H or [m] ← ACC + 60H or [m] ← ACC + 66H
Affected flag(s)	C
DEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	[m] ← [m] – 1
Affected flag(s)	Z
DECA [m]	Decrement Data Memory with result in ACC
Description	Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] – 1
Affected flag(s)	Z
HALT	Enter power down mode
Description	This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared.

Operation	TO ← 0
	PDF ← 1

Affected flag(s)	TO, PDF

Downloaded from Elcodis.com electronic components distributor

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

Downloaded from Elcodis.com electronic components distributor

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

Downloaded from Elcodis.com electronic components distributor

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

Downloaded from Elcodis.com electronic components distributor

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

Downloaded from Elcodis.com electronic components distributor

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

Downloaded from Elcodis.com electronic components distributor

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

Downloaded from Elcodis.com electronic components distributor

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

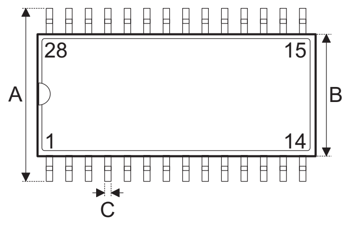
XORM A,[m]	Logical XOR ACC to Data Memory
Description	Data in the specified Data Memory and the Accumulator perform a bitwise logical XOR operation. The result is stored in the Data Memory.
Operation	[m] ← ACC ″XOR″ [m]
Affected flag(s)	Z

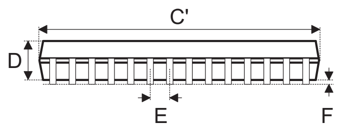
XOR A,x	Logical XOR immediate data to ACC
Description	Data in the Accumulator and the specified immediate data perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ x
Affected flag(s)	Z

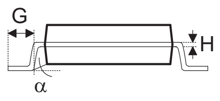
Downloaded from Elcodis.com electronic components distributor

Package Information

28-pin SSOP (150mil) Outline Dimensions



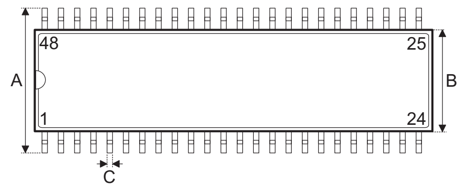


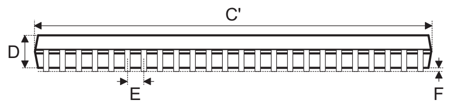


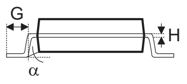
Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	228	—	244
B	150	—	157
C	8	—	12
C′	386	—	394
D	54	—	60
E	—	25	—
F	4	—	10
G	22	—	28
H	7	—	10
α	0°	—	8°

Downloaded from Elcodis.com electronic components distributor

48-pin SSOP (300mil) Outline Dimensions



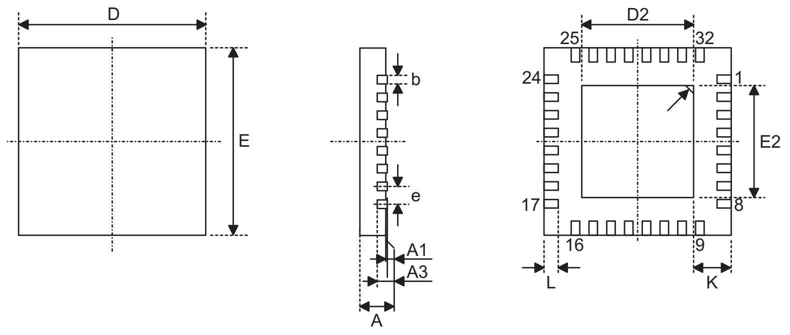




Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	395	—	420
B	291	—	299
C	8	—	12
C′	613	—	637
D	85	—	99
E	—	25	—
F	4	—	10
G	25	—	35
H	4	—	12
α	0°	—	8°

Downloaded from Elcodis.com electronic components distributor

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

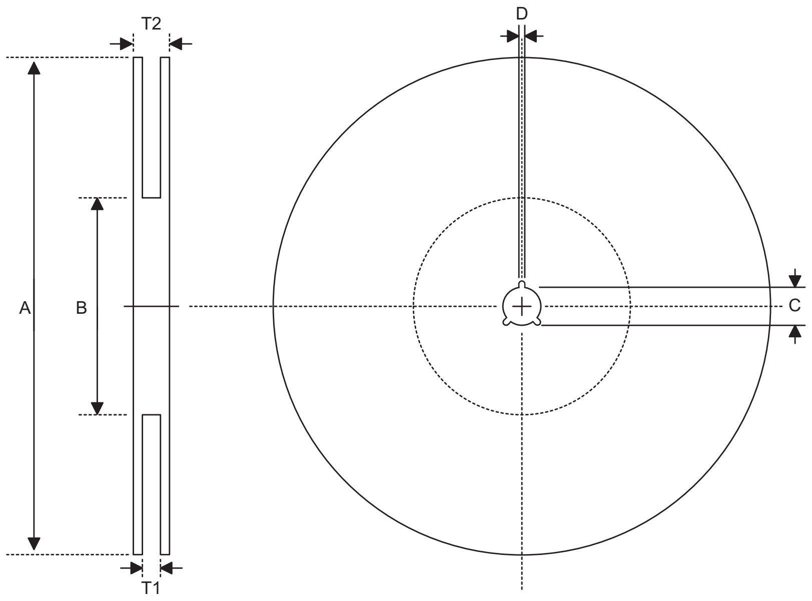


Symbol	Dimensions in mm.		
	Min.	Nom.	Max.
A	0.70	—	0.80
A1	0.00	—	0.05
A3	—	0.20	—
b	0.18	—	0.30
D	—	5.00	—
E	—	5.00	—
e	—	0.50	—
D2	1.25	—	3.25
E2	1.25	—	3.25
L	0.30	—	0.50
K	—	—	—

Downloaded from Elcodis.com electronic components distributor

Product Tape and Reel Specifications

Reel Dimensions



SSOP 28S (150mil)

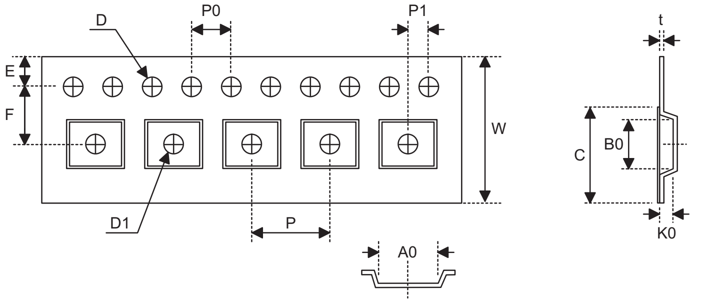
Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0±1.0
B	Reel Inner Diameter	100.0±1.5
C	Spindle Hole Diameter	13.0 $^{+0.5/-0.2}$
D	Key Slit Width	2.0±0.5
T1	Space Between Flange	16.8 $^{+0.3/-0.2}$
T2	Reel Thickness	22.2±0.2

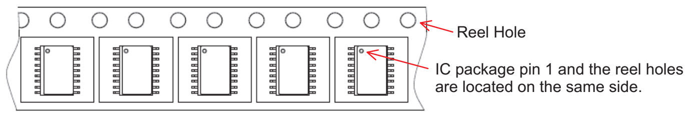
SSOP 48W

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

Downloaded from Elcodis.com electronic components distributor

Carrier Tape Dimensions



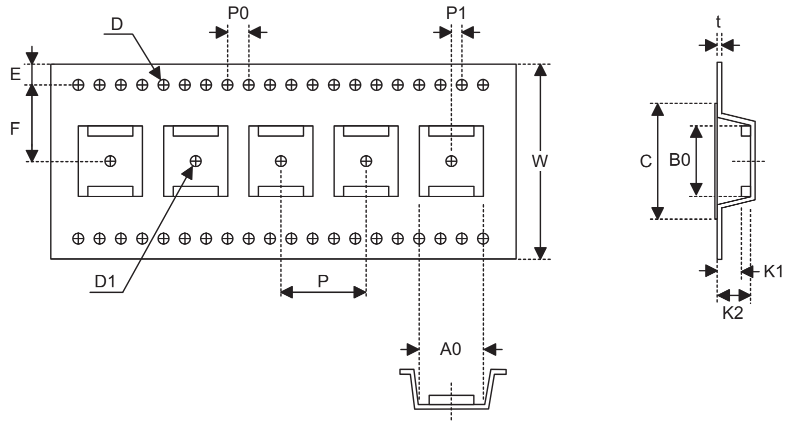

Reel Hole
IC package pin 1 and the reel holes
are located on the same side.

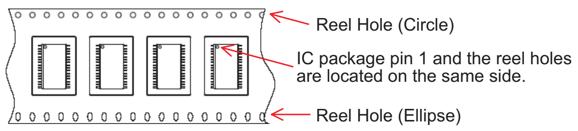
SSOP 28S (150mil)

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

Downloaded from Elcodis.com electronic components distributor

Carrier Tape Dimensions




Reel Hole (Circle)
IC package pin 1 and the reel holes are located on the same side.
Reel Hole (Ellipse)

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

Downloaded from Elcodis.com electronic components distributor

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
Tel: 1-510-252-9880
Fax: 1-510-252-9885
http://www.holtek.com

Copyright © 2009 by HOLTEK SEMICONDUCTOR INC.

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

Downloaded from Elcodis.com electronic components distributor