HT45R37

TinyPower™ C/R-F Type 8-Bit OTP MCU

Technical Document

- Tools Information
- FAQs
- Application Note
  - HA0075E MCU Reset and Oscillator Circuits Application Note

Features

- Operating voltage:
  f_{SYS}=32768Hz: 2.2V~5.5V
  f_{SYS}=4MHz: 2.2V~5.5V
  f_{SYS}=8MHz: 3.0V~5.5V
  f_{SYS}=12MHz: 4.5V~5.5V
- OTP Program Memory: 4K×15
- RAM Data Memory: 160×8
- 27 bidirectional I/O lines
- Three external interrupt input shared with I/O lines
- Two 8-bit programmable Timer/Event Counter with overflow interrupt and 7-stage prescaler
- External C/R to F converter
- 16-Channel Capacitor/Resistor sensor inputs
- Integrated Crystal, IRC, ERC and RTC oscillator
- Fully integrated internal RC oscillator available with three frequencies: 4MHz, 8MHz or 12MHz
- Watchdog Timer function
- LIRC or RTC oscillator function for watchdog timer
- PFD and Buzzer for audio frequency generation
- Dual Serial Interfaces: SPI and I^2C
- 4 operating modes: normal, slow, idle and sleep
- 6-level subroutine nesting
- 6-channel 12-bit resolution A/D converter
- 2-channel 12-bit PWM output shared with I/O line
- Low voltage reset function - 2.1V, 3.15V, 4.2V
- Low voltage detect function - 2.2V, 3.3V, 4.4V
- Bit manipulation instruction
- 15-Bit table read instructions
- 63 powerful instructions
- Up to 0.33μs instruction cycle with 12MHz system clock at V_{DD}=5V
- All instructions executed in one or two machine cycles
- Idle/Sleep mode and wake-up functions to reduce power consumption
- Time-Base and RTC interrupt
- 20-pin DIP/SOP, 24/28-pin SKDIP/SOP/SSOP and 32-pin QFN packages

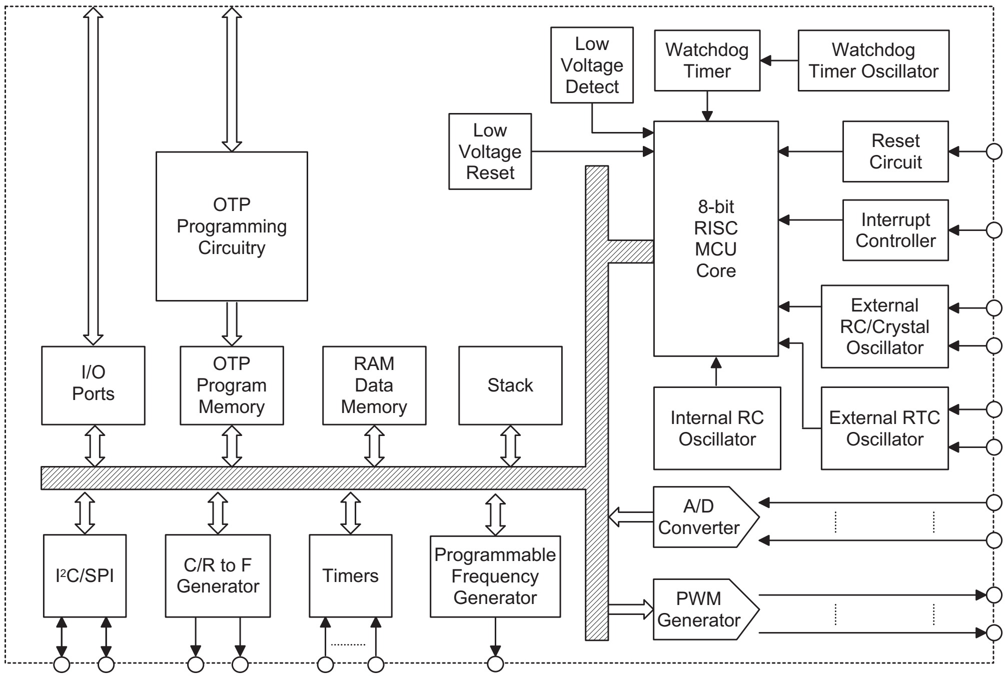
General Description

The HT45R37 is a TinyPower™ C/R to F Type 8-bit high performance RISC architecture microcontroller. As a C/R to F type of MCU, the device can interface to 16 external C or R type sensors and convert their values to a frequency value for processing. In addition its internal A/D converter allows the device to interface directly to analog signals and its integrated dual channel Pulse Width Modulators allows control of external motors, LEDs etc.

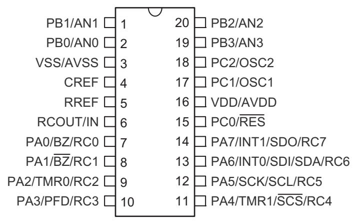
With their fully integrated SPI and I^2C functions, designers are provided with a means of easy communication with external peripheral hardware. The benefits of integrated A/D, C/R to F converter, and PWM functions, in addition to low power consumption, high performance, I/O flexibility and low-cost, provides the device with the versatility for a wide range of products in the home appliance and industrial application areas. Some of these products could include electronic metering, environmental monitoring, handheld instruments, electronically controlled tools, motor driving in addition to many others.

The unique Holtek TinyPower technology also gives the devices extremely low current consumption characteristics, an extremely important consideration in the present trend for low power battery powered applications. The usual Holtek MCU features such as power down and wake-up functions, oscillator options, programmable frequency divider, etc. combine to ensure user applications require a minimum of external components.

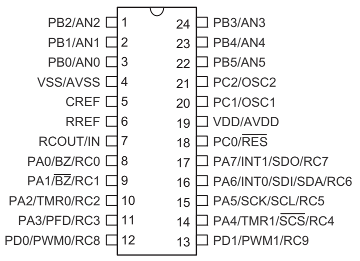
Block Diagram



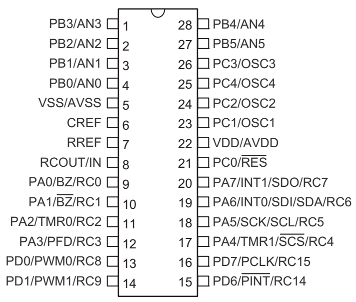
Pin Assignment



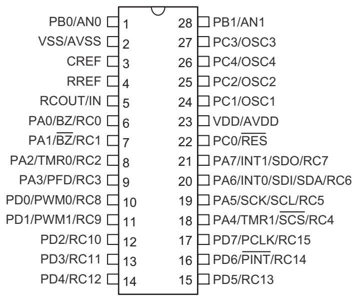
HT45R37
20 DIP-A/SOP-A



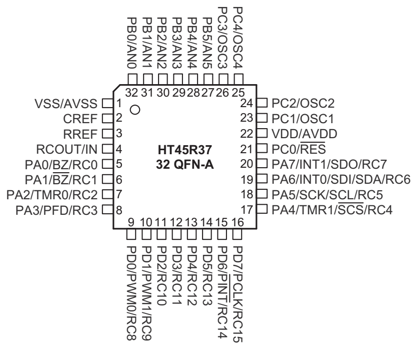
HT45R37
24 SKDIP-A/SOP-A



HT45R37-A
28 SKDIP-A/SOP-A/SSOP-A



HT45R37-B
28 SKDIP-B/SOP-B/SSOP-B



Pin Description

Pin Name	I/O	Configuration Option	Description
PA0/BZ/RC0 PA1/$\overline{\text{BZ}}$/RC1 PA2/TMR0/RC2 PA3/PFD/RC3 PA4/TMR1/$\overline{\text{SCS}}$/RC4 PA5/SCK/SCL/RC5 PA6/INT0/SDI/SDA/RC6 PA7/INT1/SDO/RC7	I/O SIM	BZ/$\overline{\text{BZ}}$ PFD RC0~RC7	Bidirectional 8-bit input/output port. Each individual bit on this port can be configured as a wake-up input using the PAWU register. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. A pull high resistor can be connected to each pin using the PAPU register. Pins PA0, PA1, PA3, PA2, PA4, PA6 and PA7 are shared with BZ, $\overline{\text{BZ}}$, PFD, TMR0, TMR1, INT0 and INT1 respectively. PA7 is also pin-shared with the SPI bus data output line, SDO. PA6 is also pin-shared with the SPI bus data line, SDI and the I^{2}C Bus data line SDA. PA5 is also pin-shared with the SPI bus clock line, SCK, and the I^{2}C Bus clock line SCL. PA4 is also pin-shared with the SPI bus select line, $\overline{\text{SCS}}$. The C/R to F converter inputs RC0~RC7 are shared with pins Pins PA0~PA7 respectively.
PB0/AN0~PB5/AN5	I/O	—	Bidirectional 6-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. A pull high resistor can be connected to each pin using the PBPU register. PB is pin-shared with the A/D input pins. The A/D inputs are selected via software instructions. Once selected as an A/D input, the I/O function and pull-high resistor selections are disabled automatically.
PC0/$\overline{\text{RES}}$	I/O	PC0 or $\overline{\text{RES}}$	Bidirectional line I/O. Software instructions determine if the pin is an NMOS output or Schmitt Trigger input. A configuration option determines if the pin is to be used as a $\overline{\text{RES}}$ pin or as an I/O pin. This pin does not have an internal pull-high function.
PD0/PWM0/RC8 PD1/PWM1/RC9 PD2/RC10~PD5/RC13 PD6/$\overline{\text{PINT}}$/RC14 PD7/PCLK/RC15	I/O	RC8~RC15	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt trigger input. A pull high resistor can be connected to each pin using the PDPU register. The PWM outputs, PWM0 and PWM1, are pin shared with pins PD0 and PD1, the function of which is chosen using the PWM registers. PD6 is pin-shared with the Peripheral Interrupt line, $\overline{\text{PINT}}$. PD7 is pin-shared with the Peripheral Clock line, PCLK. The C/R to F converter inputs RC8~RC15 are shared with pins Pins PD0~PD7 respectively.
RCOUT/IN	I	—	Capacitor or resistor connection pin to RC OSC/ Oscillation input.
RREF	I	—	Reference resistor connection pin
CREF	I	—	Reference capacitor connection pin
PC1/OSC1 PC2/OSC2	I/O	1.Int. RC OSC 2.Crystal OSC 3.Ext. RC OSC	Bidirectional 2-line I/O. Software instructions determine if the pins are CMOS outputs or Schmitt Trigger inputs. A pull high resistor can be connected to each pin using the PCPU register. Configuration options determine if the pins are to be used as oscillator pins or I/O pins. Configuration options also determine which oscillator mode is selected. The three oscillator modes are: 1. Internal RC OSC: both pins configured as I/Os. 2. External crystal OSC: both pins configured as OSC1/OSC2. 3. External RC OSC+PC2: PC1 is configured as OSC1 pin, PC2 configured as an I/O. If the internal RC OSC is selected, the frequency will be fixed at either 4MHz, 8MHz or 12MHz, dependent upon which configuration option is chosen.

Pin Name	I/O	Configuration Option	Description
PC3/OSC3 PC4/OSC4	I/O	RTC OSC	Bidirectional 2-line I/O. Software instructions determine if the pins are CMOS outputs or Schmitt Trigger inputs. A pull high resistor can be connected to each pin using the PCPU register. Configuration options determine if the pins are to be used as oscillator pins or I/O pins. If configuration options select oscillator pins, the pins are connected to a 32768Hz crystal oscillator.
VDD/AVDD	—	—	Positive power supply/analog positive power supply.
VSS/AVSS	—	—	Negative power supply, ground/analog negative power supply, ground

Note: The Pin Description table represents the largest package available, therefore some of the pins and functions may not be available on smaller package types.

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	80mA
Total Power Dissipation	500mW
Storage Temperature	–50°C to 125°C
Operating Temperature	–40°C to 85°C
I_{OH} Total	–80mA

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=4MHz	2.2	—	5.5	V
			f_{SYS}=8MHz	3.0	—	5.5	V
			f_{SYS}=12MHz	4.5	—	5.5	V
AV_{DD}	A/D Operating Voltage	—	—	2.7	—	5.5	V
I_{DD1}	Operating Current (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=f_M=1MHz	—	170	250	μA
		5V		—	380	700	μA
I_{DD2}	Operating Current (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=f_M=2MHz	—	240	360	μA
		5V		—	490	800	μA
I_{DD3}	Operating Current (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=f_M=4MHz (note 4)	—	440	660	μA
		5V		—	900	1350	μA
I_{DD4}	Operating Current (EC Mode, Filter On)	3V	No load, f_{SYS}=f_M=4MHz	—	380	570	μA
		5V		—	720	1080	μA
I_{DD5}	Operating Current (Crystal OSC, RC OSC)	5V	No load, f_{SYS}=f_M=8MHz	—	1.8	2.7	mA
I_{DD6}	Operating Current (Crystal OSC, RC OSC)	5V	No load, f_{SYS}=f_M=12MHz	—	2.6	4.0	mA
I_{DD7}	Operating Current (Slow Mode, f_M=4MHz) (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=f_{SLOW}=500kHz	—	150	220	μA
		5V		—	340	510	μA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{DD8}	Operating Current (Slow Mode, f_M=4MHz) (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=f_{SLOW}=1MHz	—	180	270	μA
		5V		—	400	600	μA
I_{DD9}	Operating Current (Slow Mode, f_M=4MHz) (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=f_{SLOW}=2MHz	—	270	400	μA
		5V		—	560	840	μA
I_{DD10}	Operating Current (Slow Mode, f_M=8MHz) (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=f_{SLOW}=1MHz	—	240	360	μA
		5V		—	540	810	μA
I_{DD11}	Operating Current (Slow Mode, f_M=8MHz) (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=f_{SLOW}=2MHz	—	320	480	μA
		5V		—	680	1020	μA
I_{DD12}	Operating Current (Slow Mode, f_M=8MHz) (Crystal OSC, RC OSC)	3V	No load, f_{SYS}=f_{SLOW}=4MHz	—	500	750	μA
		5V		—	1000	1500	μA
I_{DD13}	Operating Current (f_{SYS}=32768Hz (note 1) or 32K_INT internal RC OSC)	3V	No load, WDT off	—	8	16	μA
		5V		—	15	30	μA
I_{STB1}	Standby Current (Sleep) (f_{SYS}, f_{SUB}, f_S, f_{WDT}=off)	3V	No load, system HALT, WD off	—	—	1	μA
		5V		—	—	2	μA
I_{STB2}	Standby Current (Sleep) (f_{SYS}, f_{WDT}=f_{SUB}=32768Hz (note 1) or 32K_INT RC OSC)	3V	No load, system HALT, WDT on	—	2	4	μA
		5V		—	4	6	μA
I_{STB3}	Standby Current (Idle) (f_{SYS}, f_{WDT}=off; f_S=f_{SUB}=32768Hz (note 1) or 32K_INT RC OSC)	3V	No load, system HALT, WDT off	—	2	4	μA
		5V		—	4	6	μA
I_{STB4}	Standby Current (Idle) (f_{SYS}=on, f_{SYS}=f_M=4MHz, f_{WDT}=off, f_S (note 3)= f_{SUB}=32768Hz (note 1) or 32K_INT RC OSC)	3V	No load, system HALT, WDT off, SPI or I^2C on, PCLK on, PCLK=f_{SYS}/8	—	150	250	μA
		5V		—	350	550	μA
V_{IL1}	Input Low Voltage for I/O Ports, TMR and INT	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, 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
V_{LVR}	Low Voltage Reset Voltage	—	Configuration option: 2.1V	1.98	2.1	2.22	V
		—	Configuration option: 3.15V	2.98	3.15	3.32	V
		—	Configuration option: 4.2V	3.98	4.2	4.42	V
V_{LVD}	Low Voltage Detector Voltage	—	Configuration option: 2.2V	2.08	2.2	2.32	V
		—	Configuration option: 3.3V	3.12	3.3	3.50	V
		—	Configuration option: 4.4V	4.12	4.4	4.70	V
I_{OL1}	I/O Port Sink Current - Except PC0	3V	V_{OL}=$0.1V_{DD}$	6	12	—	mA
		5V		10	25	—	mA

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
I_{OH1}	I/O Port Source Current	3V	V_{OH}=0.9V_{DD}	–2	–4	—	mA
		5V		–5	–8	—	mA
I_{OL2}	PC0 Sink Current	3V	V_{OL}=0.1V_{DD}	0.8	1.5	—	mA
		5V		2	4	—	mA
R_{PH}	Pull-high Resistance for I/O Ports	3V	—	20	60	100	kΩ
		5V		10	30	50	kΩ
V_{AD}	A/D Input Voltage	—	—	0	—	AV_{DD}	V
V_{REF}	A/D Input Reference Voltage Range	—	—	—	AV_{DD}	—	V
DNL	A/C Differential Non-Linearity	—	AV_{DD}=5V, V_{REF}=A_{VDD}, t_{AD}=0.5μs	–2	—	2	LSB
INL	ADC Integral Non-Linearity	—	AV_{DD}=5V, V_{REF}=A_{VDD}, t_{AD}=0.5μs	–4	—	4	LSB
I_{ADC}	Additional Power Consumption if A/D Converter is Used	3V	—	—	0.5	1	mA
		5V		—	1.5	3	mA

Note: 1. 32768Hz is in slow start mode (RTCC.4=1) for the D.C. current measurement.

2. f_S is the internal clock for Buzzer, RTC, Time base and WDT.

3. Both Timer/Event Counters are off. Timer filter is disabled for all test conditions.

4. All peripherals are in OFF condition if not mentioned at I_{DD}, I_{STB} tests.

A.C. Characteristics

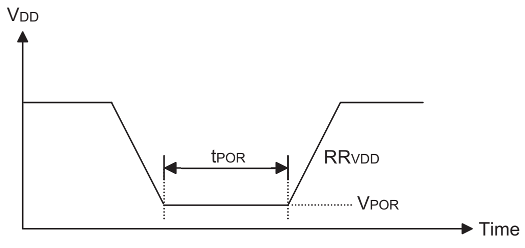
Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS1}	System Clock (Crystal OSC, ERC OSC)	—	2.2V~5.5V	400	—	4000	kHz
			3.0V~5.5V	400	—	8000	kHz
			4.5V~5.5V	400	—	12000	kHz
f_{SYS2}	System Clock (HIRC OSC)	—	5V	–2%	4000	+2%	kHz
			5V	–2%	8000	+2%	kHz
			5V	–2%	12000	+2%	kHz
f_{SYS3}	System Clock (RTC Crystal OSC)	—	2.2V~5.5V	—	32768	—	Hz
$f_{4MRCOSC}$	4MHz External RC OSC	5V	External R=150kΩ	–2%	4000	+2%	kHz
f_{RTCOSC}	RTC Frequency	—	—	—	32768	—	Hz
f_{TIMER}	Timer I/P Frequency (TMR0/TMR1)	—	2.2V~5.5V	0	—	4000	kHz
			3.0V~5.5V	0	—	8000	kHz
			4.5V~5.5V	0	—	12000	kHz
f_{RC32K}	32K RC Period (LIRC)	—	2.2V~5.5V, After Trim	28.1	31.25	34.4	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{LVR}	Low Voltage Reset Time	—	—	0.1	0.4	0.6	ms
t_{SST1}	System Start-up Timer Period	—	Power-on	—	1024	—	t_{SYS}*
t_{SST2}	System Start-up Timer Period for XTAL or RTC oscillator	—	Wake-up from Power Down Mode	—	1024	—	t_{SYS}*
t_{SST3}	System Start-up Timer Period for RC or External Clock	—	Wake-up from Power Down Mode	—	1	2	t_{SYS}*
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{AD}	A/D Clock Period	—	—	0.5	—	—	μs
t_{ADC}	A/D Conversion Time	—	—	—	16	—	t_{AD}
RESTD	Reset Delay Time	—	—	—	100	—	ms

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

Power-on Reset Characteristics

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



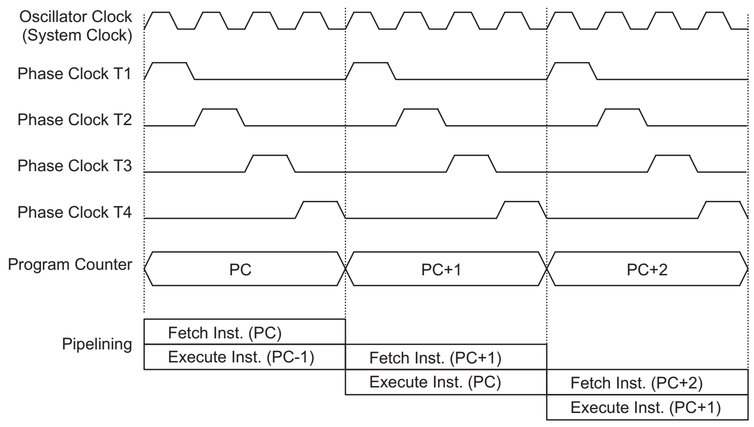
System Architecture

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

Clocking and Pipelining

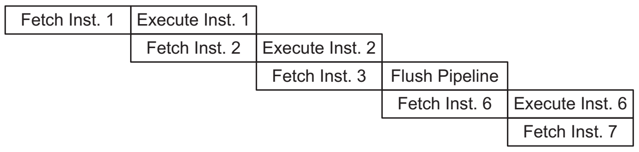
The main system clock, derived from either a Crystal/Resonator or RC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry 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

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



Instruction Fetching

Program Counter

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

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 writable 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.

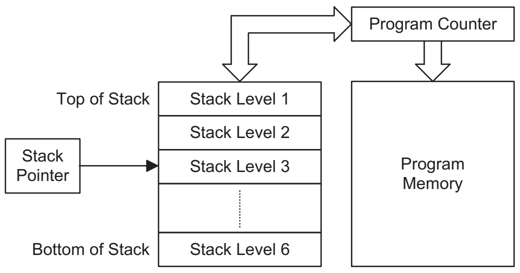
Mode	**Program Counter Bits**											
	b11	**b10**	**b9**	**b8**	**b7**	**b6**	**b5**	**b4**	**b3**	**b2**	**b1**	**b0**
Initial Reset	0	0	0	0	0	0	0	0	0	0	0	0
External Interrupt 0	0	0	0	0	0	0	0	0	0	1	0	0
External Interrupt 1	0	0	0	0	0	0	0	0	1	0	0	0
Timer/Event Counter 0 Overflow	0	0	0	0	0	0	0	0	1	1	0	0
Timer/Event Counter 1 Overflow	0	0	0	0	0	0	0	1	0	0	0	0
SPI/I^{2}C Interrupt	0	0	0	0	0	0	0	1	0	1	0	0
C/R to F Converter Interrupt	0	0	0	0	0	0	0	1	0	1	0	0
Time Base Interrupt	0	0	0	0	0	0	0	1	1	0	0	0
RTC Interrupt	0	0	0	0	0	0	0	1	1	0	0	0
A/D Converter Interrupt	0	0	0	0	0	0	0	1	1	0	0	0
External Peripheral Interrupt	0	0	0	0	0	0	0	1	1	0	0	0
Skip	Program Counter + 2											
Loading PCL	PC11	PC10	PC9	PC8	@7	@6	@5	@4	@3	@2	@1	@0
Jump, Call Branch	#11	#10	#9	#8	#7	#6	#5	#4	#3	#2	#1	#0
Return from Subroutine	S11	S10	S9	S8	S7	S6	S5	S4	S3	S2	S1	S0

Program Counter

Note: PC11~PC8: Current Program Counter bits
#11~#0: Instruction code address bits
@7~@0: PCL bits
S11~S0: Stack register bits

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack has multiple levels depending upon the device and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, SP, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.



If the stack is full and an enabled interrupt takes place, the interrupt request flag will be recorded but the acknowledge signal will be inhibited. When the Stack Pointer is decremented, by RET or RETI, the interrupt will be serviced. This feature prevents stack overflow allowing the programmer to use the structure more easily. However, when the stack is full, a CALL subroutine instruction can still be executed which will result in a stack overflow. Precautions should be taken to avoid such cases which might cause unpredictable program branching.

Arithmetic and Logic Unit – ALU

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

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

Program Memory

The Program Memory is the location where the user code or program is stored. For these device the Program Memory is an OTP type, which means it can be programmed only one time. By using the appropriate programming tools, this OTP memory device offer users the flexibility to conveniently debug and develop their applications while also offering a means of field programming.

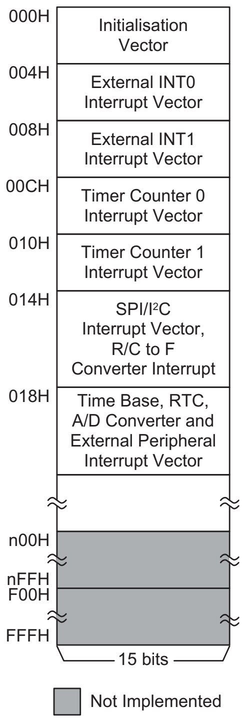
Structure

The Program Memory has a capacity of 4K by 15 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 a separate table pointer register.

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 0. If the external interrupt pin receives an active edge, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full.
- Location 008H
  This vector is used by the external interrupt 1. If the external interrupt pin receives an active edge, the program will jump to this location and begin execution if the external interrupt is enabled and the stack is not full.
- Location 00CH
  This internal vector is used by the Timer/Event Counter 0. If a Timer/Event Counter 0 overflow occurs, the program will jump to this location and begin execution if the timer/event counter interrupt is enabled and the stack is not full.



000H
Initialisation Vector
004H
External INT0 Interrupt Vector
008H
External INT1 Interrupt Vector
00CH
Timer Counter 0 Interrupt Vector
010H
Timer Counter 1 Interrupt Vector
014H
SPI/I²C Interrupt Vector, R/C to F Converter Interrupt
018H
Time Base, RTC, A/D Converter and External Peripheral Interrupt Vector
n00H
nFFH
F00H
FFFH
15 bits
Not Implemented

Program Memory Structure

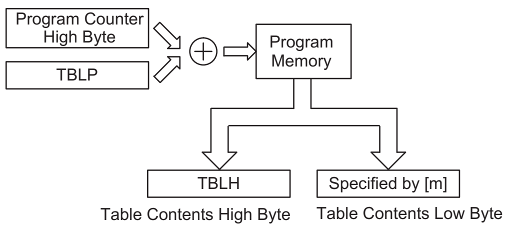
- Location 010H
  This internal vector is used by the Timer/Event Counter 1. If a Timer/Event Counter 1 overflow occurs, the program will jump to this location and begin execution if the timer/event counter interrupt is enabled and the stack is not full.
- Location 014H
  This internal vector is used by the Multi-function Interrupt 0. The Multi-function Interrupt 0 vector is shared by the SPI/I^{2}C and the C/R to F converter interrupt. When an interrupt signal is generated, the program will jump to this location and begin execution if the interrupt is enabled and the stack is not full.
- Location 018H
  This internal vector is used by the Multi-function Interrupt 1. The Multi-function Interrupt 1 vector is shared by the Real Time Clock interrupt, Time Base interrupt, A/D converter interrupt and External Peripheral interrupt. When a interrupt signal is generated, the program will jump to this location and begin execution if the interrupt is enabled and the stack is not full.

Look-up Table

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

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

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



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

Table Location

Note: PC11~PC8: Current Program Counter bits @7~@0: Table Pointer TBLP bits

Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the last page which is stored there using the ORG statement. The value at this ORG statement is ″F00H″ which refers to the start address of the last page within the 4K Program Memory of the HT45R37. 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 ″F06H″ 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.

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 table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of the 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.

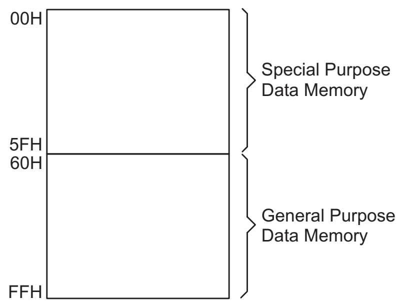
```
Tempreg1        db   ?       ; temporary register #1
tempreg2        db   ?       ; temporary register #2
     :
     :

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 ″F06H″ 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 ″F05H″ transferred to
                             ; tempreg2 and TBLH
                             ; in this example the data ″1AH″ is transferred to
                             ; tempreg1 and data ″0FH″ to register tempreg2
     :
     :

org  F00h                    ; sets initial address of last page

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

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 Data Memory is subdivided into two sections, the Special Purpose Data Memory and the General Purpose Data Memory. The start address of the Data Memory is the address ″00H″. The last Data Memory address is ″FFH″.



Data Memory Structure

Note: Most of the Data Memory bits can be directly manipulated using the ″SET [m].i″ and ″CLR [m].i″ with the exception of a few dedicated bits. The Data Memory can also be accessed through the memory pointer registers MP0 and MP1.

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 Data 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 read and write type but some are protected and are read 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	RTCC
0AH	STATUS
0BH	INTC0
0CH	▨
0DH	TMR0
0EH	TMR0C
0FH	▨
10H	▨
11H	▨
12H	PA
13H	PAC
14H	PB
15H	PBC
16H	PC
17H	PCC
18H	PD
19H	PDC
1AH	PWM0L
1BH	PWM0H
1CH	PWM1L
1DH	PWM1H
1EH	INTC1
1FH	MFIC0
20H	MFIC1
21H	ASCR0
22H	ASCR1
23H	ASCR2
24H	ADRL
25H	ADRH
26H	ADCR
27H	ACSR
28H	CLKMOD
29H	PAWU
2AH	PAPU
2BH	PBPU
2CH	PCPU
2DH	PDPU
2EH	INTEDGE
2FH	MISC
30H	TMRAH
31H	TMRAL
32H	RCOCCR
33H	TMRBH
34H	TMRBL
35H	RCOCR
36H	SIMCTL0
37H	SIMCTL1
38H	SIMDR
39H	SIMAR/SIMCTL2
3AH	TMR1
3BH	TMR1C
3CH	▨
3DH	▨
3EH	▨
3FH	▨
40H~5HF	🟨

▨ : Unused read as "00"

🟨 : Unused read as unknown

Special Purpose Data Memory

Special Function Registers

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

Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal register space, do not actually physically exist as normal registers. The method of indirect addressing for 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 Pointers – MP0, MP1

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

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

```
data .section  ′data′
adres1   db ?
adres2   db ?
Adres3   db ?
adres4   db ?
block    db ?
code .section at 0 ′code′
org   00h

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

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

continue:
```

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

Accumulator – ACC

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

Program Counter Low Register – PCL

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

Look-up Table Registers – TBLP, TBLH

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

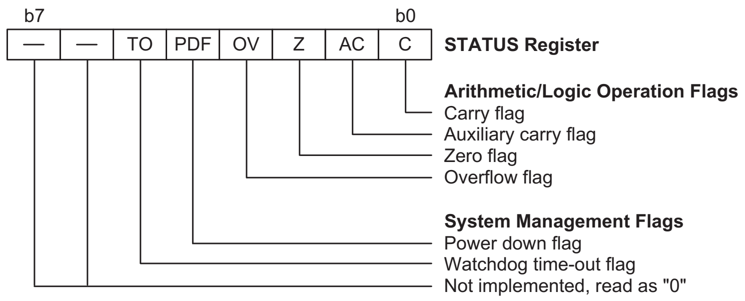
Status Register – STATUS

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

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

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

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



Status Register

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

Interrupt Control Registers

These 8-bit registers, known as the INTC0, INTC1, MFIC0, MFIC1 and INTEDGE registers, control the operation of the external and internal Timer/Event Counter interrupt, Time Base interrupt, Real Time Clock interrupt, A/D converter interrupt, C/R to F converter interrupt and SPI/I^{2}C interrupt. By setting various bits within these registers using standard bit manipulation instructions, the enable/disable function of each interrupt can be independently controlled. A master interrupt bit within this register, the EMI bit, acts like a global enable/disable and is used to set all of the interrupt enable bits on or off. This bit is cleared when an interrupt routine is entered to disable further interrupt and is set by executing the ″RETI″ instruction.

Timer/Event Counter Registers

The devices contains two 8-bit Timer/Event Counters. The registers, TMR0 and TMR1 are the locations where the timer values are located. These registers can also be preloaded with fixed data to allow different time intervals to be setup. The 8-bit Timer/Event Counters have an associated control register, TMR0C and TMR1C, which contain the setup information for these timers, 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 their associated control registers play a prominent role. All I/O ports have a designated register correspondingly labeled as PA, PB, PC and PD. 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 and PDC, 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.

Pulse Width Modulator Registers

The devices contain two Pulse Width Modulator outputs each with their own related independent control register pair, known as PWM0L/PWM0H and PWM1L/ PWM1H. The 12-bit contents of each register pair, which defines the duty cycle value for the modulation cycle of the Pulse Width Modulator, along with an enable bit are contained in these register pairs.

A/D Converter Registers – ADRL, ADRH, ADCR, ACSR

The device contains a multiple channel 12-bit A/D converter. The correct operation of the A/D requires the use of two data registers and two control registers. The two data registers, a high byte data register known as ADRH, and a low byte data register known as ADRL, are the register locations where the digital value is placed after the completion of an analog to digital conversion cycle. Functions such as the A/D enable/disable, A/D channel selection and A/D clock frequency are determined using the two control registers, ADCR and ACSR.

C/R to F Converter Registers

The device contains a 16-channel C/R to F converter. The correct operation of the C/R to F converter requires the use of two 16-bit counters and five control registers. The two 16-bit counters, a high byte register known as TMRAH/TMRBH, and a low byte register known as TMRAL/TMRBL. The channel selection of the C/R to F converter is setup via the ASCR0~ASCR2 control registers. The configuration of the C/R to F converter is setup via the RCOCCR or RCOCR control registers.

Serial Interface Registers

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

Port A Wake-up Register – PAWU

All pins on Port A have a wake-up function enable a low going edge on these pins to wake-up the device when it is in a power down mode. The pins on Port A that are used to have a wake-up function are selected using this resister.

Pull-High Resistors – PAPU, PBPU, PCPU, PDPU

All I/O pins on Ports PA, PB, PC and PD, if setup as inputs, can be connected to an internal pull-high resistor. The pins which require a pull-high resistor to be connected are selected using these registers.

Register – CLKMOD

The device operates using a dual clock system whose mode is controlled using this register. The register controls functions such as the clock source, the idle mode enable and the division ratio for the slow clock.

Miscellaneous Register – MISC

The miscellaneous register is used to control two functions. The four lower bits are used for the Watchdog Timer control, while the highest four bits are used to select open drain outputs for pins PA0~PA3.

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 selections for all ports and wake-up selections on certain pins, the user is provided with an I/O structure to meet the needs of a wide range of application possibilities.

The device provides multiple bidirectional input/output lines labeled with port names PA, PB, PC and PD. These I/O ports are mapped to the Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction ″MOV A,[m]″, where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using registers PAPU, PBPU, PCPU and PDPU and are implemented using weak PMOS transistors.

Port A Wake-up

The HALT instruction forces the microcontroller into a Power Down condition which preserves power, a feature that is important for battery and other low-power applications. Various methods exist to wake-up the microcontroller, one of which is to change the logic condition on one of the Port A pins from high to low. After a HALT instruction forces the microcontroller into entering a Power Down condition, the processor will remain in a low-power state until the logic condition of the selected wake-up pin on Port A changes from high to low. This function is especially suitable for applications that can be woken up via external switches. Each pin on Port A can be selected individually to have this wake-up feature using the PAWU register.

Port A Open Drain Function

All I/O pins in the device have CMOS structures, however Port A pins PA0~PA3 can also be setup as open drain structures. This is implemented using the ODE0~ ODE3 bits in the MISC register.

I/O Port Control Registers

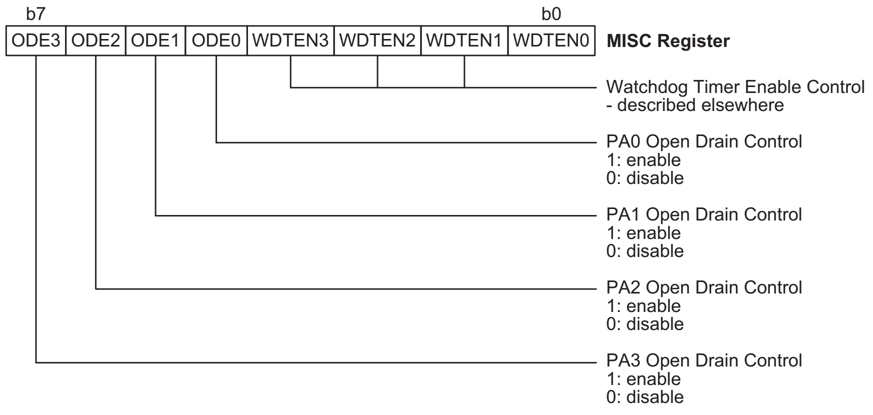
Each I/O port has its own control register known as PAC, PBC, PCC and PDC, to control the input/output configuration. With this control register, each CMOS output or input with or without pull-high resistor structures can be reconfigured dynamically under software control. Each pin of the I/O ports is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

Pin-shared Functions

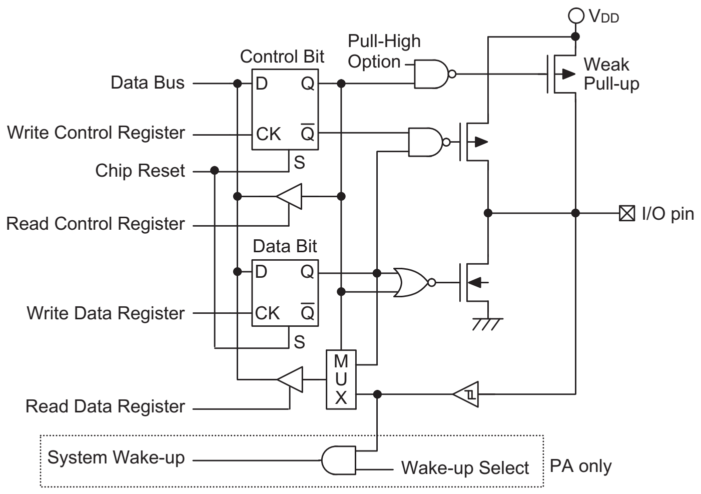
The flexibility of the microcontroller range is greatly enhanced by the use of pins that have more than one function. Limited numbers of pins can force serious design constraints on designers but by supplying pins with multi-functions, many of these difficulties can be overcome. For some pins, the chosen 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 Inputs
  The external interrupt pins INT0 and INT1 are pin-shared with I/O pins. For applications not requiring external interrupt inputs, 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.

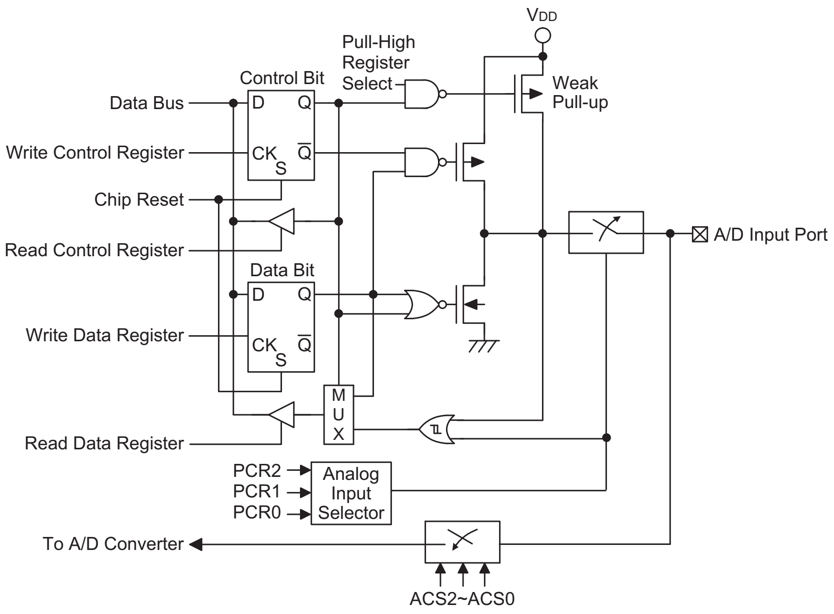
- External Timer Clock Input
  The external timer pins TMR0 and TMR1 are pin-shared with I/O pins. To configure them to operate as timer inputs, the corresponding control bits in the timer control register must be correctly set and the pin must also be setup as an input. Note that the original I/O function will remain even if the pin is setup to be used as an external timer input.


b7
b0
ODE3
ODE2
ODE1
ODE0
WDTEN3
WDTEN2
WDTEN1
WDTEN0
MISC Register
Watchdog Timer Enable Control - described elsewhere
PA0 Open Drain Control 1: enable 0: disable
PA1 Open Drain Control 1: enable 0: disable
PA2 Open Drain Control 1: enable 0: disable
PA3 Open Drain Control 1: enable 0: disable

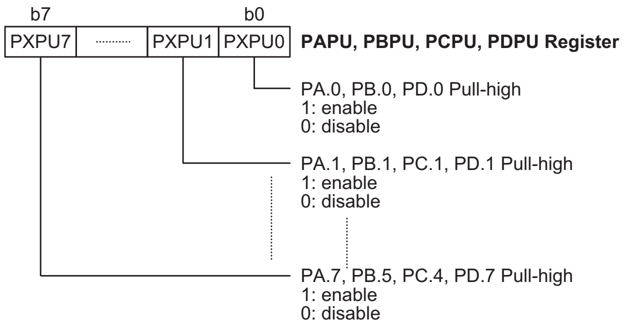
PA0~PA3 Open Drain Control – MISC



Generic Input/Output Structure



A/D Input/Output Structure


b7
b0
PXPU7
PXPU1
PXPU0
PAPU, PBPU, PCPU, PDPU Register
PA.0, PB.0, PD.0 Pull-high
1: enable
0: disable
PA.1, PB.1, PC.1, PD.1 Pull-high
1: enable
0: disable
PA.7, PB.5, PC.4, PD.7 Pull-high
1: enable
0: disable

Pull-High Resistor Register – PAPU, PBPU, PCPU, PDPU

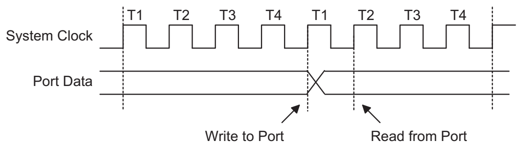
- PFD Output
  The device contains a PFD function whose single output is pin-shared with I/O pin PA3. The output function of this pin is chosen via a configuration option and remains fixed after the device is programmed. Note that the corresponding bit of the port control register, PAC.3, must setup the pin as an output to enable the PFD output. If the PAC port control register has setup the pin as an input, then the pin will function as a normal logic input with the usual pull-high selection, even if the PFD configuration option has been selected.
- PWM Outputs
  The device contains two PWM outputs shared with pins PD0 and PD1. The PWM output functions are chosen via registers. Note that the corresponding bit of the port control register, PDC, must setup the pin as an output to enable the PWM output. If the PDC port control register has setup the pin as an input, then the pin will function as a normal logic input with the usual pull-high selection, even if the PWM registers have enabled the PWM function.
- A/D Inputs
  The device contains a multi-channel A/D converter inputs. All of these analog inputs are pin-shared with I/O pins on Port B. If these pins are to be used as A/D inputs and not as normal I/O pins then the corresponding bits in the A/D Converter Control Register, ADCR, must be properly set. There are no configuration options associated with the A/D function. If used as I/O pins, then full pull-high resistor register remain, however if used as A/D inputs then any pull-high resistor selections associated with these pins will be automatically disconnected.

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



Read/Write Timing

Port A 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 transition of any of the Port A pins. Single or multiple pins on Port A 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. The device contains two 8-bit count-up timers. 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. The provision of a prescaler to the clock circuitry of the 8-bit Timer/Event Counter also gives added range to this timer.

There are two types of registers related to the Timer/Event Counters. The first are the registers that contain the actual value of the Timer/Event Counter and into which an initial value can be preloaded. Reading from these registers 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. The Timer/Event Counters can have the their clock configured to come from an internal clock source. In addition, their clock source can also be configured to come from an external timer pin.

Configuring the Timer/Event Counter Input Clock Source

The internal timer′s clock can originate from various sources. The system clock source is used when the Timer/Event Counter is in the timer mode or in the pulse width measurement mode. This internal clock source is f_{SYS} which is also divided by a prescaler, the division ratio of which is conditioned by the Timer Control Register, TMRnC, bits TnPSC0~ TnPSC2.

An external clock source is used when the timer is in the event counting mode, the clock source being provided on an external timer pin TMR0 or TMR1 depending upon which timer is used. Depending upon the condition of the TnE bit, each high to low, or low to high transition on the external timer pin will increment the counter by one.

Timer Registers – TMR0, TMR1

The timer registers are special function registers located in the Special Purpose Data Memory and is the place where the actual timer value is stored. These registers are known as TMR0 or TMR1. The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reset with the initial preload register value and continue counting.

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

Timer Control Registers – TMR0C, TMR1C

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

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

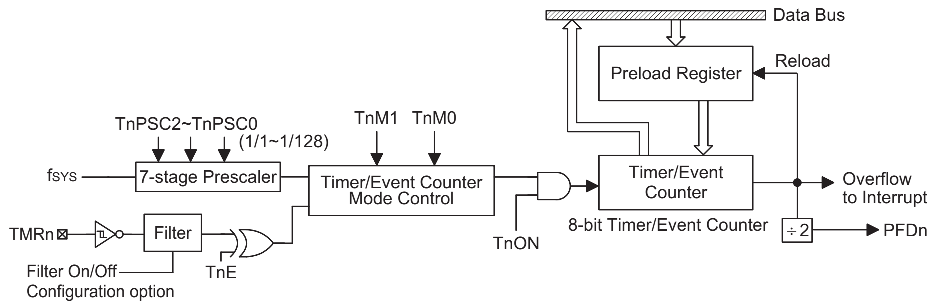
To choose which of the three modes the timer is to operate in, either in the timer mode, the event counting mode or the pulse width measurement mode, bits 7 and 6 of the corresponding Timer Control Register, which are known as the bit pair TnM1/TnM0, must be set to the required logic levels. The timer-on bit, which is bit 4 of the Timer Control Register and known as TnON, depending upon which timer is used, provides the basic on/off control of the respective timer. Setting the bit high allows the counter to run, clearing the bit stops the counter. For timers that have prescalers, bits 0~2 of the Timer Control Register determine the division ratio of the input clock prescaler. The prescaler bit settings have no effect if an external clock source is used. If the timer is in the event count or pulse width measurement mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as TnE.

Configuring the Timer Mode

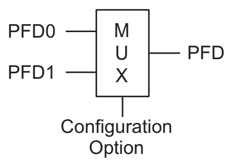
In this mode, the Timer/Event Counter can be utilised to measure fixed time intervals, providing an internal interrupt signal each time the Timer/Event Counter overflows. To operate in this mode, the Operating Mode Select bit pair, TnM1/TnM0, in the Timer Control Register must be set to the correct value as shown.

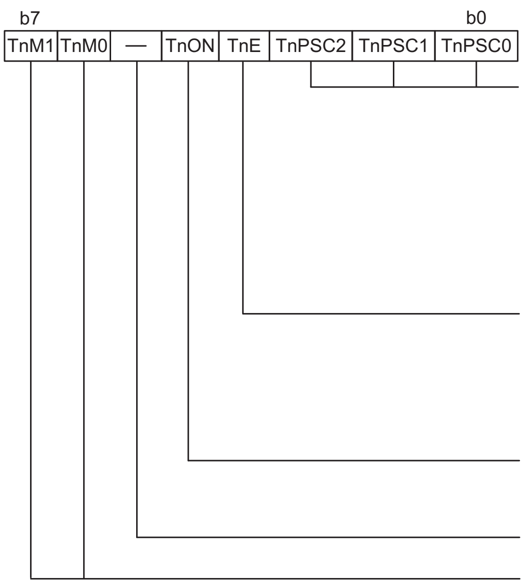
	Bit7	Bit6
Control Register Operating Mode Select Bits for the Timer Mode	1	0

In this mode the internal clock, f_{SYS} , is used as the internal clock for the Timer/Event Counter. However, the clock source, f_{SYS}, for the 8-bit timer is further divided by



8-bit Timer/Event Counter Structure (n=0, 1)





TMRnC Register (n=0, 1)

Timer prescaler rate select

TnPSC2	TnPSC1	TnPSC0	Timer Rate
0	0	0	1:1
0	0	1	1:2
0	1	0	1:4
0	1	1	1:8
1	0	0	1:16
1	0	1	1:32
1	1	0	1:64
1	1	1	1:128

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

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

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

Not implemented, read as "0"

Operating mode select

TnM1	TnM0	
0	0	no mode available
0	1	event counter mode
1	0	timer mode
1	1	pulse width measurement mode

Timer/Event Counter Control Register – TMRnC

a prescaler, the value of which is determined by the Prescaler Rate Select bits TnPSC2~TnPSC0, which are bits 2~0 in the Timer Control Register. After the other bits in the Timer Control Register have been setup, the enable bit TnON or TnON, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. Each time an internal clock cycle occurs, the Timer/Event Counter increments by one. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the corresponding Interrupt Control Register, 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, TnM1/TnM0, 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, 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 TnON, 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, TnE, 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 corresponding Interrupt Control Register, is reset to zero.

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

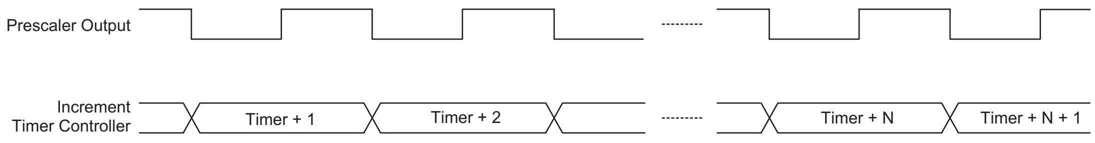
Configuring the Pulse Width Measurement Mode

In this mode, the Timer/Event Counter can be utilised to measure the width of external pulses applied to the external timer pin. To operate in this mode, the Operating Mode Select bit pair, TnM1/TnM0, in the Timer Control Register must be set to the correct value as shown.

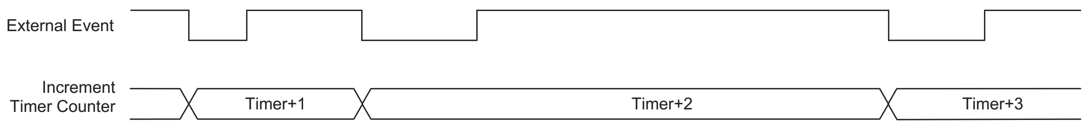
Control Register Operating Mode Select Bits for the Pulse Width Measurement Mode	Bit7	Bit6
	1	1

In this mode the internal clock, f_{SYS}, is used as the internal clock for the Timer/Event Counter. However, the clock source, f_{SYS}, for the 8-bit timer is further divided by a prescaler, the value of which is determined by the Prescaler Rate Select bits TnPSC2~TnPSC0, which are bits 2~0 in the Timer Control Register. After the other bits in the Timer Control Register have been setup, the enable bit TnON, 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.

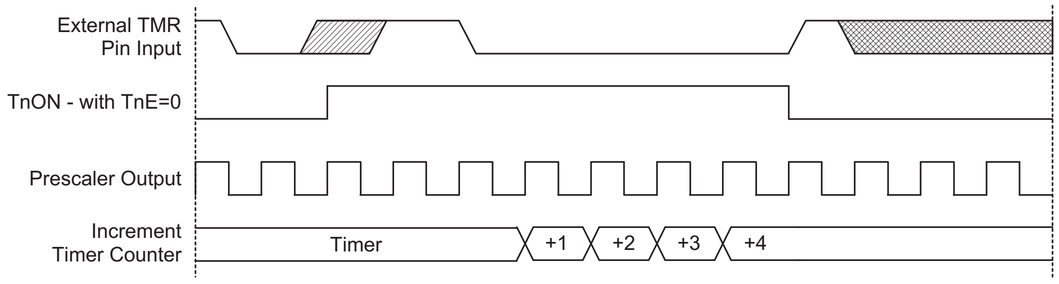
If the Active Edge Select bit TnE, which is bit 3 of the Timer Control Register, is low, once a high to low transition has been received on the external timer pin, the Timer/Event Counter will start counting until the external timer pin returns to its original high level. At this point the enable bit will be automatically reset to zero and the



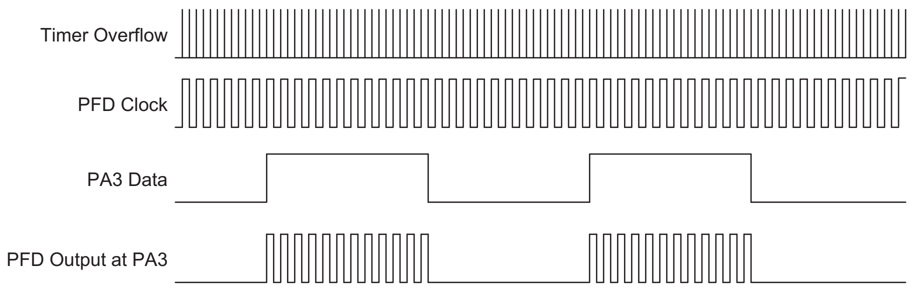
Timer Mode Timing Chart



Event Counter Mode Timing Chart



Pulse Width Measure Mode Timing Chart



PFD Output Control

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 corresponding Interrupt Control Register, is reset to zero.

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

Programmable Frequency Divider – PFD

The Programmable Frequency Divider provides a means of producing a variable frequency output suitable for applications requiring a precise frequency generator.

The PFD output is pin-shared with the I/O pin PA3. The PFD function is selected via configuration option, however, if not selected, the pin can operate as a normal I/O pin.

The clock source for the PFD circuit can originate from either Timer/Event Counter 0 or Timer/Event Counter 1 overflow signal selected via configuration option. The output frequency is controlled by loading the required values into the timer registers and prescaler registers to give the required division ratio. The timer will begin to count-up from this preload register value until full, at which point an overflow signal is generated, causing the PFD output to change state. The timer will then be automatically reloaded with the preload register value and continue counting-up.

For the PFD output to function, it is essential that the corresponding bit of the Port A control register PAC bit 3 is setup as an output. If setup as an input the PFD output will not function, however, the pin can still be used as a normal input pin. The PFD output will only be activated if bit PA3 is set to ″1″. This output data bit is used as the on/off control bit for the PFD output. Note that the PFD output will be low if the PA3 output data bit is cleared to ″0″.

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

Prescaler

Bits TnPSC0~TnPSC2 of the control register can be used to define the pre-scaling stages of the internal clock source of the Timer/Event Counter. The Timer/Event Counter overflow signal can be used to generate signals for the PFD and Timer Interrupt.

I/O Interfacing

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

Timer/Event Counter Pins Internal Filter

The external Timer/Event Counter pins are connected to an internal filter to reduce the possibility of unwanted event counting events or inaccurate pulse width measurements due to adverse noise or spikes on the external Timer/Event Counter input signal. As this internal filter circuit will consume a limited amount of power, a configuration option is provided to switch off the filter function, an option which may be beneficial in power sensitive applications, but in which the integrity of the input signal is high. Care must be taken when using the filter on/off configuration option as it will be applied not only to both external Timer/Event Counter pins but also to the external interrupt input pins. Individual Timer/Event Counter or external interrupt pins cannot be selected to have a filter on/off function.

Programming Considerations

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

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

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

Timer Program Example

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

```
org 04h             ; external interrupt vector
reti
org 0Ch             ; Timer/Event Counter 0 interrupt vector
jmp tmrint          ; jump here when the Timer/Event Counter 0 overflows
:
org 20h             ; main program
;internal Timer/Event Counter 0 interrupt routine
tmrint:
:
; Timer/Event Counter 0 main program placed here
:
reti
:
:

begin:
;setup Timer 0 registers
mov a,09bh          ; setup Timer 0 preload value
mov tmr0,a;
mov a,081h          ; setup Timer 0 control register
mov tmr0c,a         ; timer mode and prescaler set to /2
; setup interrupt register
mov a,009h          ; enable master interrupt and timer interrupt
mov int0c,a
set tmr0c.4         ; start Timer/Event Counter 0 - note mode bits must be previously setup
```

C/R to F Converter

The C/R to F Converter function within the device enables external resistance and capacitance to be converted into a frequency. With this function the device has a way of measuring external capacitance and resistance values and can therefore be used in applications such as touch switches.

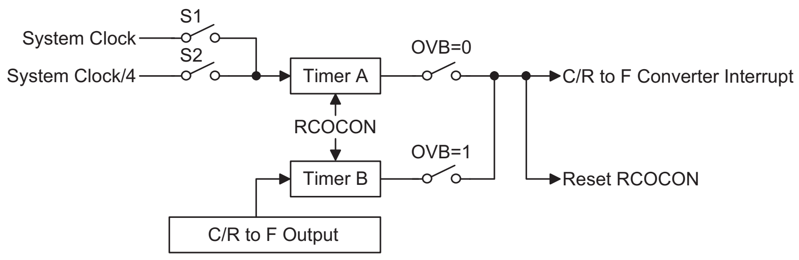
C/R to F Operation

The C/R to F function is implemented using an external RC oscillator. A single external reference resistor and external reference capacitor are required to be connected as shown. These components and the internal inverter circuits form an oscillator circuit whose frequency is dependent upon the value of the external capacitance and resistance. By using two internal 16-bit programmable count-up timers, known as Timer A and Timer B, the converted frequency value can be measured.

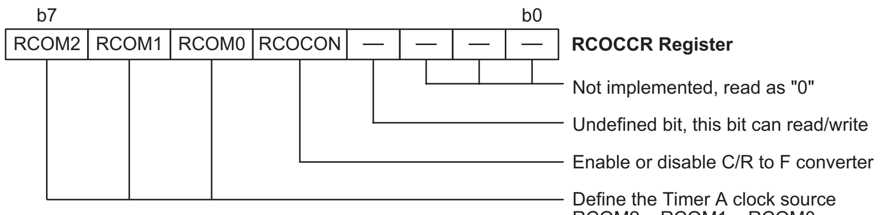
The value of the two internal 16-bit programmable count-up counters, known as Timer A and Timer B, are stored within two pairs of registers, TMRAL/TMRAH and TMRBL/TMRBH registers. Two other registers, RCOCCR and RCOCR control the overall operation of the C/R to F converter.

The Timer A clock source comes from the system clock or from the system clock/4, determined by the RCOM bits in the RCOCCR register. This clock source determines the value in the TMRAL/TMRAH registers. The Timer B clock source comes from the external RC oscillator circuit and therefore determines the value in the TMRBL/TMRBL registers. It is a combination of a fixed frequency clock source driving Timer A and a varying frequency clock source driving Timer B that enables external resistance and capacitance values to be measured.

The OVB bit, in the RCOCR register, decides whether Timer A or Timer B overflows. When this happens, the RCOCF bit will be set and an external RC oscillation converter interrupt occurs. When the C/R to F converter causes Timer A or Timer B to overflow, the RCOCON bit in the RCOCCR register will be reset to zero and the counter will stop counting. Writing initial values to the TMRAL/TMRAH and TMRBL/TMRBH registers places a start value into Timer A and Timer B. Note that writing to the low byte registers, TMRAL and TMRBL, only writes the data into a low byte buffer. However writing to the high byte registers, TMRAH and TMRBH, will write both the high byte values and the low byte buffer values directly into the Timer A and Timer B simultaneously.

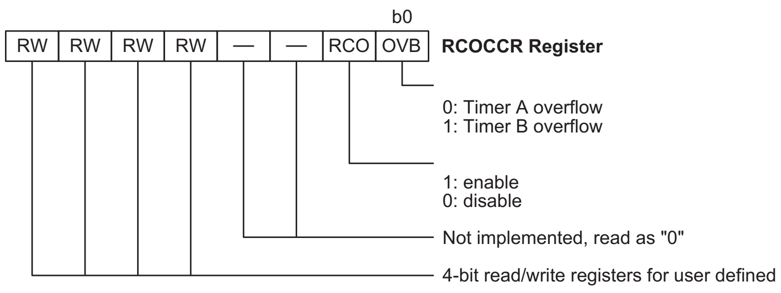


C/R to F Converter


b7
b0
RCOM2
RCOM1
RCOM0
RCOCON
—
—
—
—
RCOCCR Register
Not implemented, read as "0"
Undefined bit, this bit can read/write
Enable or disable C/R to F converter
Define the Timer A clock source

RCOM2	RCOM1	RCOM0	
0	0	0	system clock
0	0	1	system clock/4
0	1	0	unused
0	1	1	unused
1	0	0	unused
1	0	1	unused
1	1	0	unused
1	1	1	unused

RCOCCR Register



RCOCR Register

The values in Timer A and Timer B is changed by writing to the high byte registers, TMRAH and TMRBH, but writing to the low byte registers TMRAL and TMRBL will keep the values in Timer A and Timer B unchanged. Reading registers TMRAH and TMRBH will also latch the TMRAL TMRBL values into the low byte buffer to avoid false timing problems. Reading from registers TMRAL and TMRBL returns the contents of the low byte buffer only. Therefore, the low byte of Timer A and Timer B cannot be read directly. TMRAH and TMRBH must be read first to ensure that the low byte contents of Timer A and Timer B are latched into the buffer.

The external resistor and capacitor, together with internal inverters, form an oscillation circuit which is the clock source for Timer B and therefore the input to registers TMRBL and TMRBH. The RCOM0, RCOM1 and RCOM2 bits of RCOCCR define the clock source of Timer A.

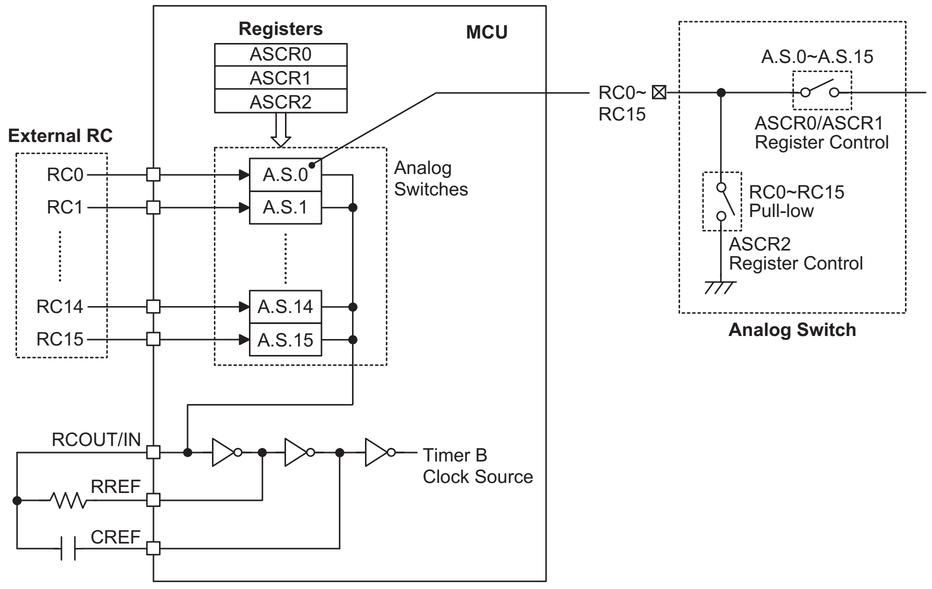
If the RCOCON bit in the RCOCCR register is set high, Timer A and Timer B will start counting until either Timer A or Timer B overflows. The relevant Timer will then generate an interrupt request flag which is the RCOCF bit in the INTC1 register. Timer A and Timer B will stop counting and will also reset the RCOCON bit to zero at the same time. If the RCOCON bit is set high, then the TMRAL/TMRAH and TMRBL/TMRBH register cannot be read or written to.

C/R to F Converter Analog Switches

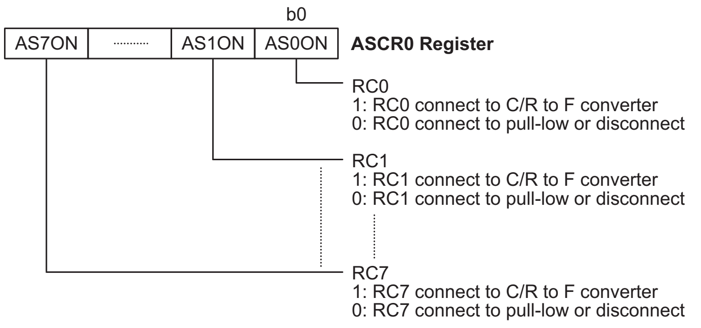
The device contains only one internal C/R to F converter function, however it can be connected to any of the 16 external channels using its internal analog switch function. This enables applications such as multi-channel or matrix touch switch applications to be implemented. The bits in the ASCR0~ASCR1 registers select which of the 16-channels is to be connected to the internal C/R to F converter. The bits in the ASCR2 register selects what happens to the channel when the channel is inactive. When a channel is inactive it can be selected to be pulled low to ground or not using bits in the ASCR2 register. As there are only 8-bits to control 16-channels the function is selected in channel pairs. There are configuration options which must first be selected to choose which pins are to be used as inputs to the C/R to F converter.

If the configuration options have selected PA0~PA7 to be normal I/O pins, then the corresponding bit 0~bit7 bits in the ASCR0 register will have no function and will be read as zero. Similarly if the configuration options have selected PD0~PD7 to be normal I/O pins, then the corresponding bit 0~bit7 bits in the ASCR1 register will have no function and will be read as zero.

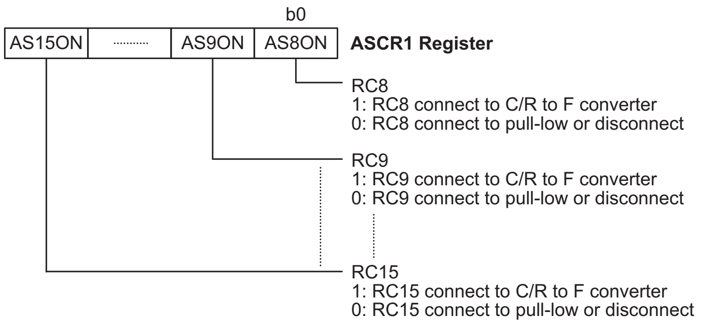
If the configuration options have selected PA0~PA7 to be normal I/O pins, then the corresponding bits in the ASCR2 register, bit 0 ~ bit3, must be cleared to zero to disable the RC0/RC1, RC2/RC3, RC4/RC5 and RC6/RC7 pull-low resistors. Similarly if the configuration options have selected PD0~PD7 to be normal I/O pins, then the corresponding bits in the ASCR2 register, bit 4~bit7, must be cleared to zero to disable the RC8/RC9, RC10/RC11, RC12/RC13 or RC14/RC15 pull-low resistors.



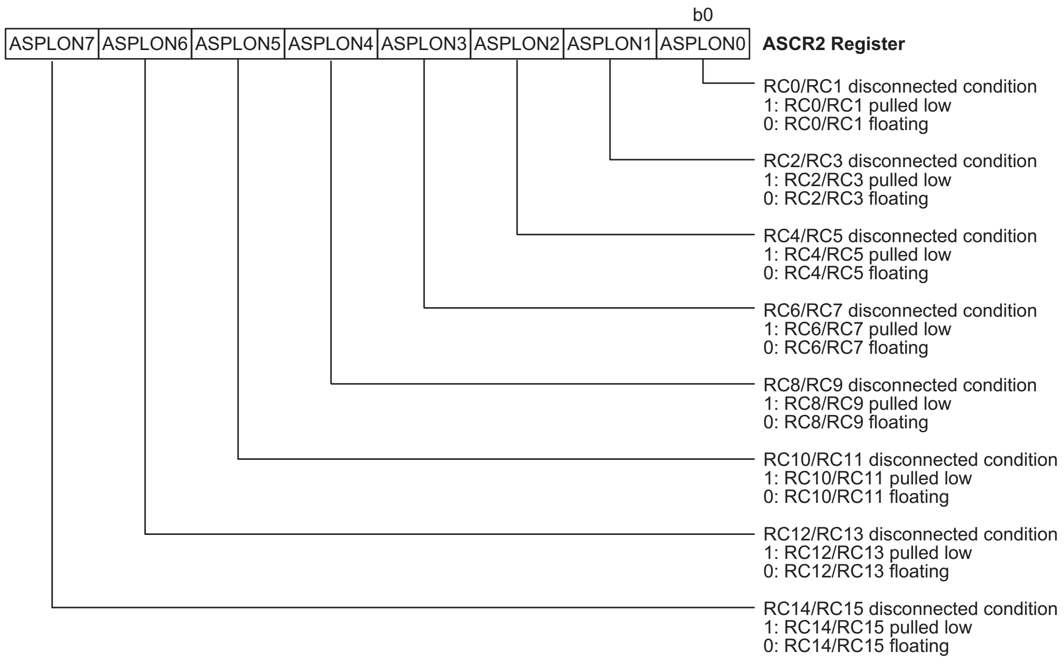
Analog Switches and C/R to F Oscillator


b0
AS7ON
AS1ON
AS0ON
ASCR0 Register
RC0
1: RC0 connect to C/R to F converter
0: RC0 connect to pull-low or disconnect
RC1
1: RC1 connect to C/R to F converter
0: RC1 connect to pull-low or disconnect
RC7
1: RC7 connect to C/R to F converter
0: RC7 connect to pull-low or disconnect

ASCR0 Register


b0
AS15ON
AS9ON
AS8ON
ASCR1 Register
RC8
1: RC8 connect to C/R to F converter
0: RC8 connect to pull-low or disconnect
RC9
1: RC9 connect to C/R to F converter
0: RC9 connect to pull-low or disconnect
RC15
1: RC15 connect to C/R to F converter
0: RC15 connect to pull-low or disconnect

ASCR1 Register



ASCR2 Register

Programming Example

The following example shows the principles of how the C/R to F converter function is programmed:

External RC oscillation converter mode example program - Timer A overflow:

```
clr RCOCCR
mov a, 00000010b          ; Enable External RC oscillation mode and set Timer A overflow
mov RCOCR,a
clr mfic0.1               ; Clear External RC Oscillation Converter interrupt request flag
clr mfic0.5
mov a, low (65536-1000)   ; Give timer A initial value
mov tmral, a              ; Timer A count 1000 time and then overflow
mov a, high (65536-1000)
mov tmrah, a
mov a, 00h                ; Give timer B initial value
mov tmrbl, a
mov a, 00h
mov tmrbh, a
mov a, 00110000b          ; Timer A clock source=fSYS/4 and timer on
mov RCOCCR, a
p10:
clr wdt
snz mfic0.5               ; Polling External RC Oscillation Converter interrupt request flag
jmp p10
clr mfic0.5               ; Clear External RC Oscillation Converter interrupt request flag
                          ; Program continue
```

Pulse Width Modulator

The device contains twos Pulse Width Modulation, PWM, outputs. Useful for such applications such as motor speed control, the PWM function provides an output with a fixed frequency, but with a duty cycle that can be varied by setting particular values into the corresponding PWM register pair.

Channel	PWM Mode	Output Pin	Register Names
1	8+4	PD0	PWM0L~ PWM0H
2	8+4	PD1	PWM1L~ PWM1H

PWM Overview

A register pair, located in the Data Memory is assigned to each Pulse Width Modulator output and are known as the PWM registers. It is in each register pair that the 12-bit value, which represents the overall duty cycle of one modulation cycle of the output waveform, should be placed. The PWM registers also contain the enable/disable control bit for the PWM outputs. To increase the PWM modulation frequency, each modulation cycle is modulated into sixteen individual modulation sub-sections, known as the 8+4 mode. Note that it is only necessary to write the required modulation value into the corresponding PWM register as the subdivision of the waveform into its sub-modulation cycles is implemented automatically within the microcontroller hardware. The PWM clock source is the system clock f_{SYS}.

This method of dividing the original modulation cycle into a further 16 sub-cycles enables the generation of higher PWM frequencies, which allow a wider range of applications to be served. As long as the periods of the generated PWM pulses are less than the time constants of the load, the PWM output will be suitable as such long time constant loads will average out the pulses of the PWM output. The difference between what is known as the PWM cycle frequency and the PWM modulation frequency should be understood. As the PWM clock is the system clock, f_{SYS}, and as the PWM value is 12-bits wide, the overall PWM cycle frequency is $f_{SYS}/4096$. However, when in the 8+4 mode of operation, the PWM modulation frequency will be $f_{SYS}/256$.

PWM Modulation Frequency	PWM Cycle Frequency	PWM Cycle Duty
$f_{SYS}/256$	$f_{SYS}/4096$	(PWM register value)/4096

8+4 PWM Mode Modulation

Each full PWM cycle, as it is 12-bits wide, has 4096 clock periods. However, in the 8+4 PWM mode, each PWM cycle is subdivided into sixteen individual sub-cycles known as modulation cycle 0 ~ modulation cycle 15, denoted as ″i″ in the table. Each one of these sixteen sub-cycles contains 256 clock cycles. In this mode, a modulation frequency increase of sixteen is achieved. The 12-bit PWM register value, which represents the overall duty cycle of the PWM waveform, is divided into two groups. The first group which consists of bit4~bit11 is denoted here as the DC value. The second group which consists of bit0~bit3 is known as the AC value. In the 8+4 PWM mode, the duty cycle value of each of the two modulation sub-cycles is shown in the following table.

Parameter	AC (0~15)	DC (Duty Cycle)
Modulation cycle i (i=0~15)	$i<AC$	$\frac{DC+1}{256}$
	$i\geq AC$	$\frac{DC}{256}$

8+4 Mode Modulation Cycle Values

The accompanying diagram illustrates the waveforms associated with the 8+4 mode of PWM operation. It is important to note how the single PWM cycle is subdivided into 16 individual modulation cycles, numbered 0~15 and how the AC value is related to the PWM value.

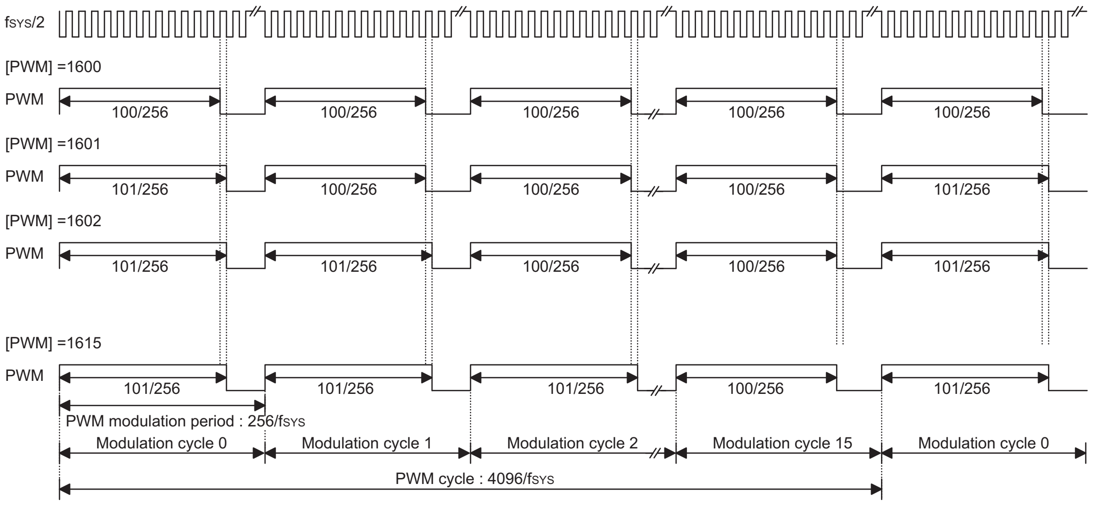
PWM Output Control

The two outputs, PWM0 and PWM1, are shared with pins PD0 and PD1. To operate as a PWM output and not as an I/O pin, bit 0 of the relevant PWM low byte register bit must be set high. A zero must also be written to the corresponding bit in the PDC port control register, to ensure that the PWM0 output pin is setup as an output. After these two initial steps have been carried out, and of course after the required PWM 12-bit value has been written into the PWM register pair register, setting the corresponding bit in the PD data register high will enable the PWM data to appear on the pin. Writing a zero to the bit will disable the PWM output function and force the output low. In this way, the Port D data output register bits, can also be used as an on/off control for the PWM function. Note that if the enable bit in the PWM register is set high to enable the PWM function, but if the corresponding bit in the PDC control register is high to configure the pin as an input, then the pin can still function as a normal input line, with pull-high resistor selections.

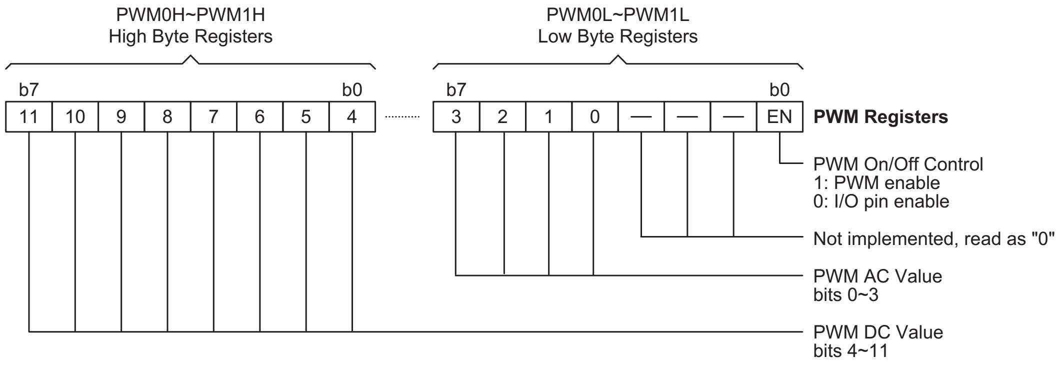
PWM Programming Example

The following sample program shows how the PWM output is setup and controlled.

```
mov a,64h        ; setup PWM0 value to 1600 decimal which is 640H
mov pwm0h,a      ; setup PWM0H register value
clr pwm0l        ; setup PWM0L register value
clr pdc.0        ; setup pin PD0 as an output
set pwm0en       ; set the PWM0 enable bit
set pd.0         ; Enable the PWM0 output
  :   :
  :   :
clr pd.0         ; PWM0 output disabled – PD0 will remain low
```



8+4 PWM Mode



PWM Register Pairs

Analog to Digital Converter

The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the microcontroller, the need for external components is reduced significantly with the corresponding follow-on benefits of lower costs and reduced component space requirements.

A/D Overview

The device contains an 6-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into either a 12-bit digital value.

Input Channels	Conversion Bits	Input Pins
6	12	PB0~PB5

The accompanying block diagram shows the overall internal structure of the A/D converter, together with its associated registers.

A/D Converter Data Registers – ADRL, ADRH

The device, which has an internal 12-bit A/D converter, requires two data registers, a high byte register, known as ADRH, and a low byte register, known as ADRL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. Only the high byte register, ADRH, utilises its full 8-bit contents. The low byte register utilises only 4 bit of its 8-bit contents as it contains only the lowest bits of the 12-bit converted value.

In the following table, D0~D11 is the A/D conversion data result bits.

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADRL	D3	D2	D1	D0	—	—	—	—
ADRH	D11	D10	D9	D8	D7	D6	D5	D4

A/D Data Registers

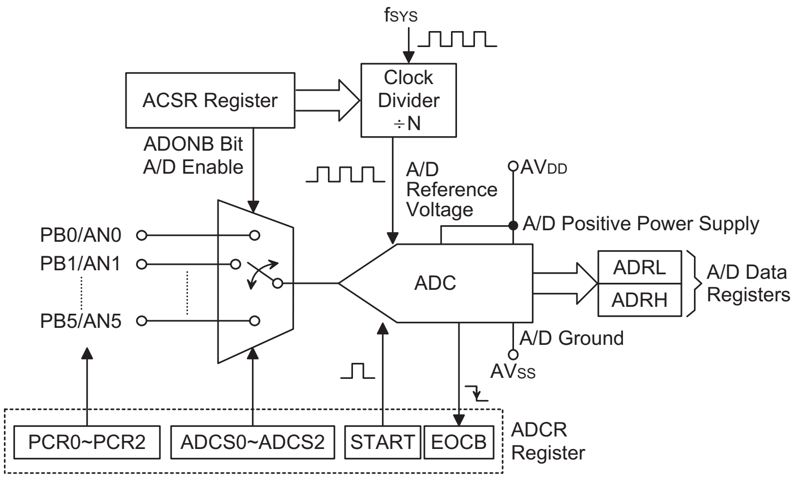
A/D Converter Control Registers – ADCR, ACSR

To control the function and operation of the A/D converter, two control registers known as ADCR and ACSR are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, which pins are used as analog inputs and which are used as normal I/Os, the A/D clock source as well as controlling the start function and monitoring the A/D converter end of conversion status.

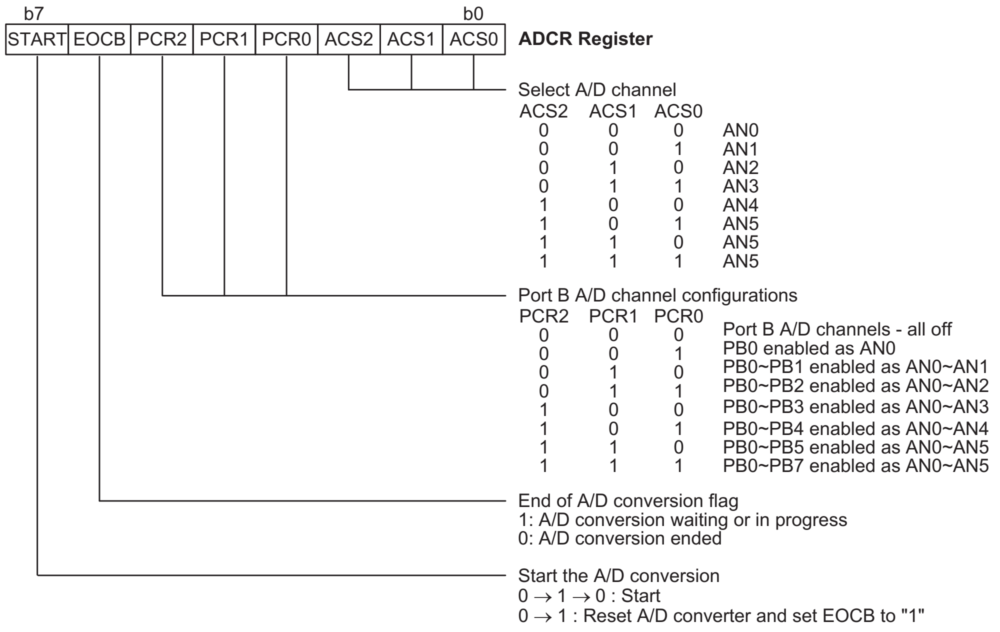
The ACS2~ACS0 bits in the ADCR register define the channel number. As the device contains only one actual analog to digital converter circuit, each of the individual 6 analog inputs must be routed to the converter. It is the function of the ACS2~ACS0 bits in the ADCR register to determine which analog channel is actually connected to the internal A/D converter.

The ADCR control register also contains the PCR2~PCR0 bits which determine which pins on Port B are used as analog inputs for the A/D converter and which pins are to be used as normal I/O pins. Note that if the PCR2~PCR0 bits are all set to zero, then all the Port B pins will be setup as normal I/Os and the internal A/D converter circuitry will be powered off to reduce the power consumption.

The START bit in the register is used to start and reset the A/D converter. When the microcontroller sets this bit from low to high and then low again, an analog to digital



A/D Converter Structure



Select A/D channel

ACS2	ACS1	ACS0	
0	0	0	AN0
0	0	1	AN1
0	1	0	AN2
0	1	1	AN3
1	0	0	AN4
1	0	1	AN5
1	1	0	AN5
1	1	1	AN5

Port B A/D channel configurations

PCR2	PCR1	PCR0	
0	0	0	Port B A/D channels - all off
0	0	1	PB0 enabled as AN0
0	1	0	PB0~PB1 enabled as AN0~AN1
0	1	1	PB0~PB2 enabled as AN0~AN2
1	0	0	PB0~PB3 enabled as AN0~AN3
1	0	1	PB0~PB4 enabled as AN0~AN4
1	1	0	PB0~PB5 enabled as AN0~AN5
1	1	1	PB0~PB7 enabled as AN0~AN5

End of A/D conversion flag
1: A/D conversion waiting or in progress
0: A/D conversion ended

Start the A/D conversion
$0 \rightarrow 1 \rightarrow 0$: Start
$0 \rightarrow 1$: Reset A/D converter and set EOCB to "1"

A/D Converter Control Register − ADCR

conversion cycle will be initiated. When the START bit is brought from low to high but not low again, the EOCB bit in the ADCR register will be set high and the analog to digital converter will be reset. It is the START bit that is used to control the overall on/off operation of the internal analog to digital converter.

The EOCB bit in the ADCR register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically cleared to zero by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can be used to poll the EOCB bit in the ADCR register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

The clock source for the A/D converter, which originates from the system clock f_{SYS}, is first divided by a division ratio, the value of which is determined by the ADCS2, ADCS1 and ADCS0 bits in the ACSR register.

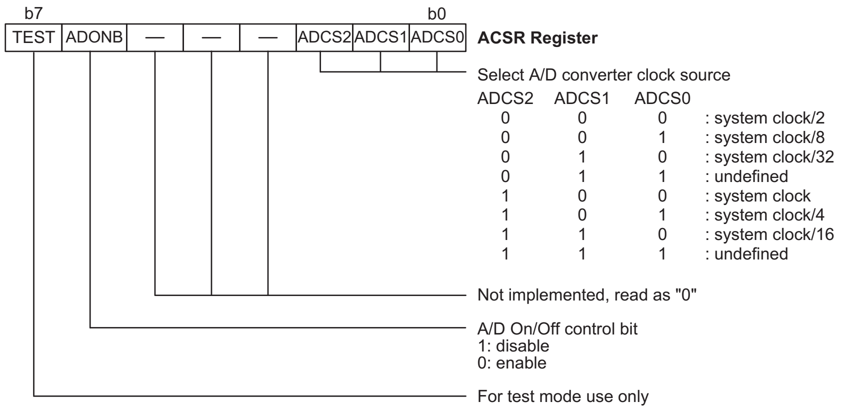
Controlling the on/off function of the A/D converter circuitry is implemented using the ADONB bit in the ACSR register and the value of the PCR bits in the ADCR register. Both the ADONB bit must cleared to zero and the value of the PCR bits must have a non-zero value for the A/D converter to be enabled.

PCR	ADONB	A/D
0	x	Off
> 0	0	On
> 0	1	Off

Although the A/D clock source is determined by the system clock f_{SYS}, and by bits ADCS2, ADCS1 and ADCS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the minimum value of permissible A/D clock period, t_{AD}, is 0.5μs, care must be taken for system clock speeds in excess of 4MHz. For system clock speeds in excess of 4MHz, the ADCS2, ADCS1 and ADCS0 bits should not be set to ″000″. Doing so will give A/D clock periods that are less than the minimum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where, depending upon the device, special care must be taken, as the values may be less than the specified minimum A/D Clock Period.

f_{SYS}	A/D Clock Period (t_{AD})			
	ADCS2, ADCS1, ADCS0=000 (f_{SYS}/2)	ADCS2, ADCS1, ADCS0=001 (f_{SYS}/8)	ADCS2, ADCS1, ADCS0=010 (f_{SYS}/32)	ADCS2, ADCS1, ADCS0=011
1MHz	2μs	8μs	32μs	Undefined
2MHz	1μs	4μs	16μs	Undefined
4MHz	500ns*	2μs	8μs	Undefined
8MHz	250ns*	1μs	4μs	Undefined
12MHz	167ns*	667ns*	2.67μs	Undefined

A/D Clock Period Examples



ADCS2	ADCS1	ADCS0	
0	0	0	: system clock/2
0	0	1	: system clock/8
0	1	0	: system clock/32
0	1	1	: undefined
1	0	0	: system clock
1	0	1	: system clock/4
1	1	0	: system clock/16
1	1	1	: undefined

A/D Converter Control Register – ACSR

A/D Input Pins

All of the A/D analog input pins are pin-shared with the I/O pins on Port B. Bits PCR2~PCR0 in the ADCR register, determine whether the input pins are setup as normal Port B input/output pins or whether they are setup as analog inputs. In this way, pins can be changed under program control to change their function from normal I/O operation to analog inputs and vice versa. Pull-high resistors, which are setup through register programming, apply to the input pins only when they are used as normal I/O pins, if setup as A/D inputs the pull-high resistors will be automatically disconnected. Note that it is not necessary to first setup the A/D pin as an input in the PBC port control register to enable the A/D input as when the PCR2~PCR0 bits enable an A/D input, the status of the port control register will be overridden. The A/D converter has its own power supply pins AVDD and AVSS pin. The analog input values must not be allowed to exceed the value of AVDD.

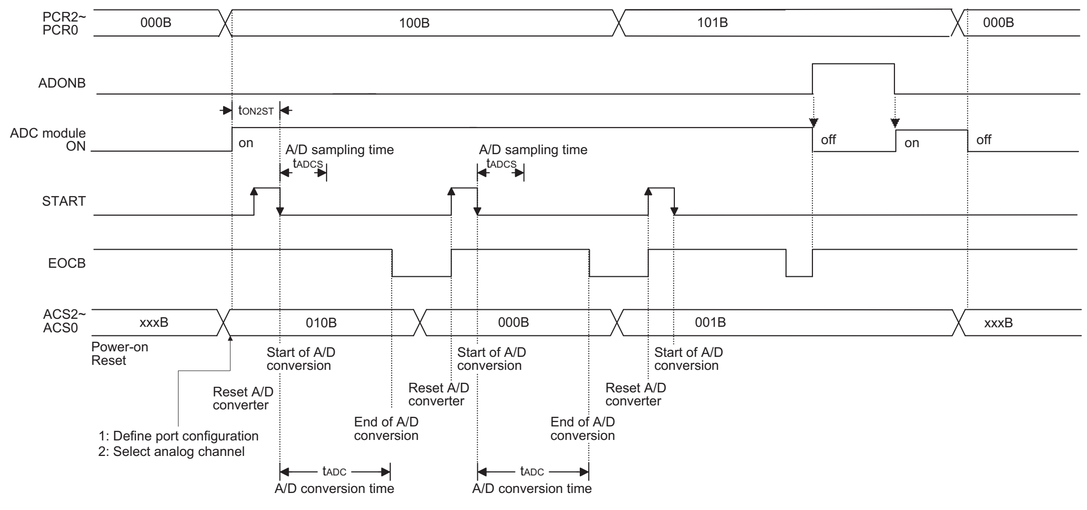
Initialising the A/D Converter

The internal A/D converter must be initialised in a special way. Each time the Port B A/D channel selection bits are modified by the program, the A/D converter must be re-initialised. If the A/D converter is not initialised after the channel selection bits are changed, the EOCB flag may have an undefined value, which may produce a false end of conversion signal. To initialise the A/D converter after the channel selection bits have changed, then, within a time frame of one to ten instruction cycles, the START bit in the ADCR register must first be set high and then immediately cleared to zero. This will ensure that the EOCB flag is correctly set to a high condition.

Summary of A/D Conversion Steps

The following summarises the individual steps that should be executed in order to implement an A/D conversion process.

- Step 1
  Select the required A/D conversion clock by correctly programming bits ADCS2, ADCS1 and ADCS0 in the register.
- Step 2
  Enable the A/D by clearing the in the ACSR register to zero.
- Step 3
  Select which channel is to be connected to the internal A/D converter by correctly programming the ACS2~ACS0 bits which are also contained in the register.
- Step 4
  Select which pins on Port B are to be used as A/D inputs and configure them as A/D input pins by correctly programming the PCR2~PCR0 bits in the ADCR register. Note that this step can be combined with Step 2 into a single ADCR register programming operation.



Note: A/D clock must be fsys, $f_{SYS}/2$, $f_{SYS}/4$, $f_{SYS}/8$, $f_{SYS}/16$ or $f_{SYS}/32$
$t_{ADCS}=4t_{AD}$
$t_{ADC}= t_{ADCS}+n*t_{AD}$; n=bit count of ADC resolution

A/D Conversion Timing

- Step 5
  If the interrupts are to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, in the INTC0 interrupt control register must be set to ″1″, the multi-function interrupt enable bit, EMFI1, in the INTC1 register and the A/D converter interrupt bit, EADI, in the MFIC1 register must also be set to ″1″.

- Step 6
  The analog to digital conversion process can now be initialised by setting the START bit in the ADCR register from ″0″ to ″1″ and then to ″0″ again. Note that this bit should have been originally set to ″0″.

- Step 7
  To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data registers ADRL and ADRH can be read to obtain the conversion value. As an alternative method if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

Note: When checking for the end of the conversion process, if the method of polling the EOCB bit in the ADCR register is used, the interrupt enable step above can be omitted.

The accompanying diagram shows graphically the various stages involved in an analog to digital conversion process and its associated timing.

The setting up and operation of the A/D converter function is fully under the control of the application program as there are no configuration options associated with the A/D converter. After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions. The time taken for the A/D conversion is $16t_{AD}$ where t_{AD} is equal to the A/D clock period.

Programming Considerations

When programming, special attention must be given to the A/D channel selection bits in the register. If these bits are all cleared to zero no external pins will be selected for use as A/D input pins allowing the pins to be used as normal I/O pins. When this happens the power supplied to the internal A/D circuitry will be reduced resulting in a reduction of supply current. This ability to reduce power by turning off the internal A/D function by clearing the A/D channel selection bits may be an important consideration in battery powered applications. The ADONB bit in the ACSR register can also be used to power down the A/D function.

Another important programming consideration is that when the A/D channel selection bits change value, the A/D converter must be re-initialised. This is achieved by pulsing the START bit in the ADCR register immediately after the channel selection bits have changed state. The exception to this is where the channel selection bits are all cleared, in which case the A/D converter is not required to be re-initialised.

A/D Programming Example

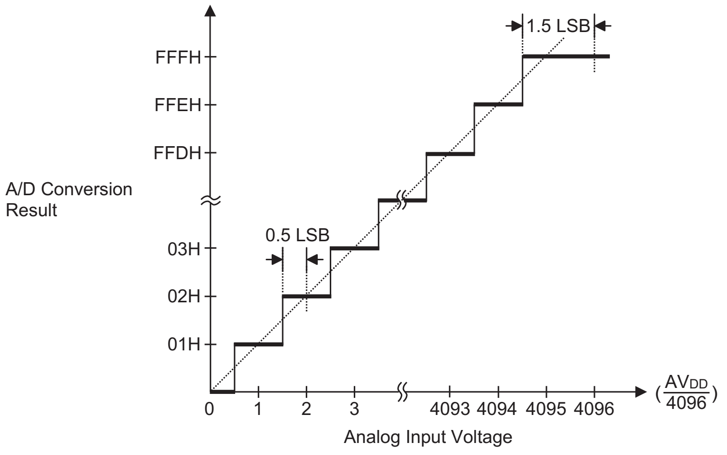
The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the EOCB bit in the ADCR register is used to detect when the conversion cycle is complete, whereas in the second example, the A/D interrupt is used to determine when the conversion is complete.

Example: using an EOCB polling method to detect the end of conversion

```
    clr  EADI                 ; disable ADC interrupt
    mov  a,00000001B
    mov  ACSR,a               ; select fSYS/8 as A/D clock and turn on ADONB bit
    mov  a,00100000B          ; setup ADCR register to configure Port PB0~PB3
                              ; as A/D inputs
    mov  ADCR,a               ; and select AN0 to be connected to the A/D converter
         :
         :                    ; As the Port B channel bits have changed the
                              ; following START
                              ; signal (0-1-0) must be issued
                              ; instruction cycles
         :
Start_conversion:
    clr  START
    set  START                ; reset A/D
    clr  START                ; start A/D
Polling_EOC:
    sz   EOCB                 ; poll the ADCR register EOCB bit to detect end
                              ; of A/D conversion
    jmp  polling_EOC          ; continue polling
    mov  a,ADRL               ; read low byte conversion result value
    mov  adrl_buffer,a        ; save result to user defined register
    mov  a,ADRH               ; read high byte conversion result value
    mov  adrh_buffer,a        ; save result to user defined register
         :
    jmp  start_conversion     ; start next A/D conversion
```

Example: using the interrupt method to detect the end of conversion

```
    clr  EADI                 ; disable ADC interrupt
    mov  a,00000001B
    mov  ACSR,a               ; select fSYS/8 as A/D clock and turn on ADONB bit

    mov  a,00100000B          ; setup ADCR register to configure Port PB0~PB3
                              ; as A/D inputs
    mov  ADCR,a               ; and select AN0 to be connected to the A/D
         :
                              ; As the Port B channel bits have changed the
                              ; following START signal(0-1-0) must be issued
                              ;
         :
Start_conversion:
    clr  START
    set  START                ; reset A/D
    clr  START                ; start A/D
    clr  ADF                  ; clear ADC interrupt request flag
    set  EADI                 ; enable ADC interrupt
    set  EMFI1                ; enable multi-function 1 interrupt
    set  EMI                  ; enable global interrupt
         :
         :
         :
; ADC interrupt service routine
ADC_:
    mov  acc_stack,a          ; save ACC to user defined memory
    a,STATUS
    mov  status_stack,a       ; save STATUS to user defined memory
         :
         :
    mov  a,ADRL               ; read low byte conversion result value
    mov  adrl_buffer,a        ; save result to user defined register
    mov  a,ADRH               ; read high byte conversion result value
    mov  adrh_buffer,a        ; save result to user defined register
         :
         :
EXIT__ISR:
    mov  a,status_stack
    mov  STATUS,a             ; restore STATUS from user defined memory
    mov  a,acc_stack          ; restore ACC from user defined memory
    clr  ADF                  ; clear ADC interrupt flag
    reti
```



Ideal A/D Transfer Function

A/D Transfer Function

As the device contain a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the AV_{DD} voltage, this gives a single bit analog input value of $AV_{DD}/4096$. The diagram show the ideal transfer function between the analog input value and the digitised output value for the A/D converter.

Note that to reduce the quantisation error, a 0.5 LSB offset is added to the A/D Converter input. Except for the digitised zero value, the subsequent digitised values will change at a point 0.5 LSB below where they would change without the offset, and the last full scale digitised value will change at a point 1.5 LSB below the AV_{DD} level.

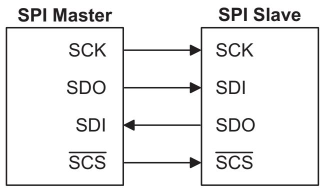
Serial Interface Function

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

SPI Interface

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

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



SPI Master/Slave Connection

SPI Interface Operation

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

The SPI function in this device offers the following features:

- Full duplex synchronous data transfer
- Both Master and Slave modes

- LSB first or MSB first data transmission modes
- Transmission complete flag
- Rising or falling active clock edge
- WCOL and CSEN bit enabled or disable select

The status of the SPI interface pins is determined by a number of factors such as whether the device is in the master or slave mode and upon the condition of certain control bits such as CSEN, SIMEN and $\overline{SCS}$. In the table I, Z represents an input floating condition.

There are several configuration options associated with the SPI interface. One of these is to enable the SIM function which selects the SIM pins rather than normal I/O pins. Note that if the configuration option does not select the SIM function then the SIMEN bit in the SIMCTL0 register will have no effect. Another two SIM configuration options determine if the CSEN and WCOL bits are to be used.

Configuration Option	Function
SIM Function	SIM interface enable/disable
SPI CSEN bit	Enable/Disable
SPI WCOL bit	Enable/Disable

SPI Interface Configuration Options

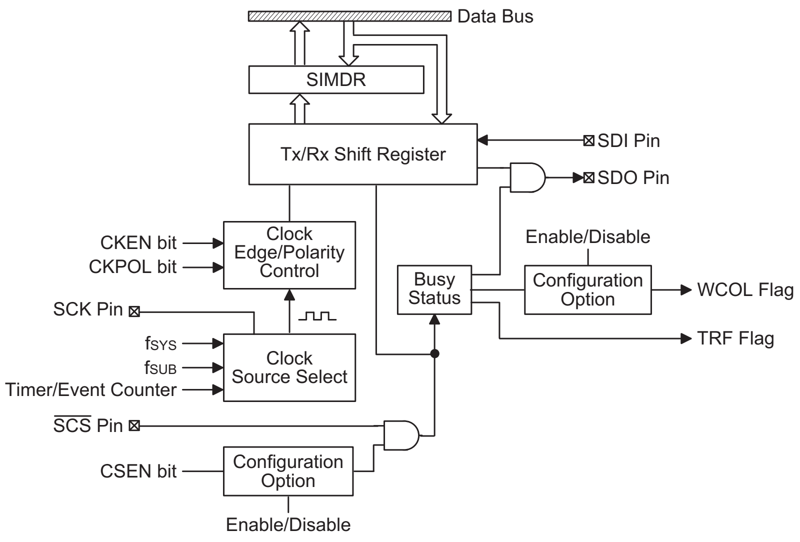
SPI Registers

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

Pin	Master/Salve SIMEN=0	Master – SIMEN=1		Slave – SIMEN=1		
		CSEN=0	CSEN=1	CSEN=0	CSEN=1 $\overline{SCS}$=0	CSEN=1 $\overline{SCS}$=1
$\overline{SCS}$	Z	Z	L	Z	I, Z	I, Z
SDO	Z	O	O	O	O	Z
SDI	Z	I, Z	I, Z	I, Z	I, Z	Z
SCK	Z	H: CKPOL=0 L: CKPOL=1	H: CKPOL=0 L: CKPOL=1	I, Z	I, Z	Z

Note: ″Z″ floating, ″H″ output high, ″L″ output low, ″I″ Input, ″O″output level, ″I,Z″ input floating (no pull-high)

SPI Interface Pin Status



SPI Block Diagram

SIMCTL0 Register

b7							b0
SIM2	SIM1	SIM0	PCKEN	PCKPSC1	PCKPSC0	SIMEN	—

- — : Not implemented, read as '0"
- SIMEN: SPI/I²C On/Of control
  1: enable
  0: disable
- PCKEN, PCKPSC1, PCKPSC0: Peripheral Clock Control - described elsewhere
- SIM2, SIM1, SIM0: SPI/I²C Master/Slave and Clock Control

SIM2	SIM1	SIM0	
0	0	0	master, $f_{SYS}/4$
0	0	1	master, $f_{SYS}/16$
0	1	0	master, $f_{SYS}/64$
0	1	1	master, f_{SUB}
1	0	0	master, Timer/Event Counter 0 output/2
1	0	1	Slave
1	1	0	I²C mode
1	1	1	Not used

SPI/I²C Control Register – SIMCTL0

SIMCTL1 Register

b7							b0
HCF	HAAS	HBB	HTX	TXAK	SRW	—	RXAK

- RXAK: Receive acknowledge flag
  1: not acknowledged
  0: acknowledged
- — : Not implemented, read as "0"
- SRW: Master data read/write request flag
  1: request data read
  0: request data write
- TXAK: Transmit acknowledge flag
  1: don't acknowledge
  0: acknowledge
- HTX: Transmit/Receive mode
  1: transmit mode
  0: receive mode
- HBB: I²C bus busy flag
  1: busy
  0: not busy
- HAAS: Calling address matched flag
  1: matched
  0: not matched
- HCF: Data transfer flag
  1: transfer complete
  0: transfer not complete

I²C Control Register – SIMCTL1

SIMCTL2 Register

b7							b0
—	—	CKPOL	CKEG	MLS	CSEN	WCOL	TRF

- TRF: Transmit/Receive complete flag
  1: finished
  0: in progress
- WCOL: Write collision flag
  1: collision
  0: no collision
- CSEN: $\overline{SCS}$ pin enable
  1: enable
  0: $\overline{SCS}$ floating
- MLS: Data shift order
  1: MSB
  0: LSB
- CKEG: SPI Clock Edge Select
  1: see text
  0: see text
- CKPOL: SPI Clock Polarity
  1: see text
  0: see text
- — —: Not implemented, read as "0"

SPI Control Register – SIMCTL2

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

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

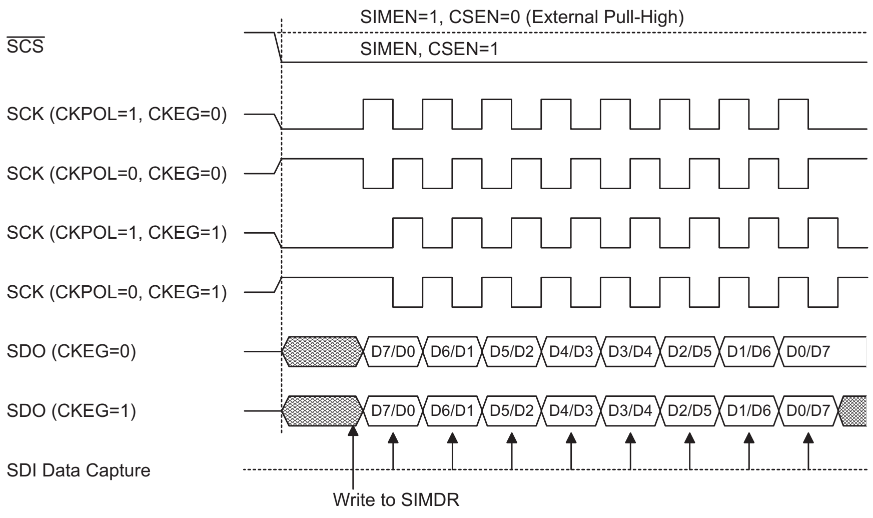
There are also two control registers for the SPI interface, SIMCTL0 and SIMCTL2. Note that the SIMCTL2 register also has the name SIMAR which is used by the I^2C function. The SIMCTL1 register is not used by the SPI function, only by the I^2C function. Register SIMCTL0 is used to control the enable/disable function and to set the data transmission clock frequency. Although not connected with the SPI function, the SIMCTL0 register is also used to control the Peripheral Clock prescaler. Register SIMCTL2 is used for other control functions such as LSB/MSB selection, write collision flag etc. The SIMIDLE bit in the CLKMOD register is used to select if the SIMcontinues running when the device is in the IDLE mode. Setting the bit high allows the SIM to maintain operation when the device is in the Idle mode. Clearing the bit to zero disables any SIM operations when in the Idle mode.

The following gives further explanation of each SIMCTL1 register bit:

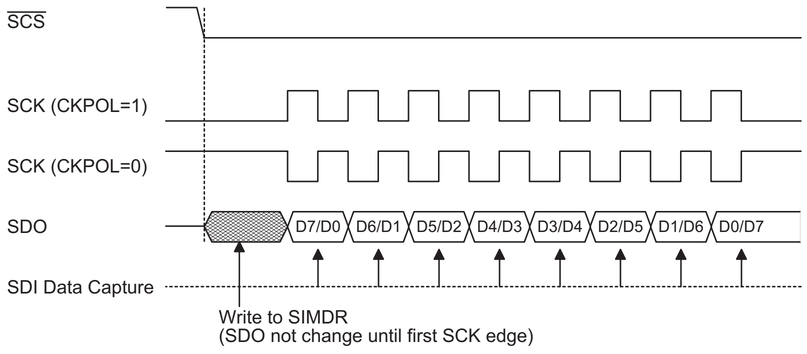
- SIMEN
  The bit is the overall on/off control for the SPI interface. When the SIMEN bit is cleared to zero to disable the SPI interface, the SDI, SDO, SCK and $\overline{SCS}$ lines will be in a floating condition and the SPI operating current will be reduced to a minimum value. When the bit is high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

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

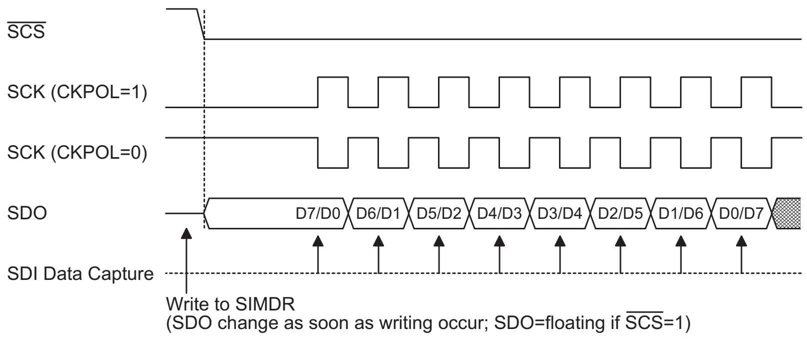
SIM0	SIM1	SIM2	SPI Master/Slave Clock Control and I2C Enable
0	0	0	SPI Master, $f_{SYS}/4$
0	0	1	SPI Master, $f_{SYS}/16$
0	1	0	SPI Master, $f_{SYS}/64$
0	1	1	SPI Master, f_{SUB}
1	0	0	SPI Master Timer/Event Counter 0 output/2
1	0	1	SPI Slave
1	1	0	I^2C mode
1	1	0	Not used



SPI Master Mode Timing

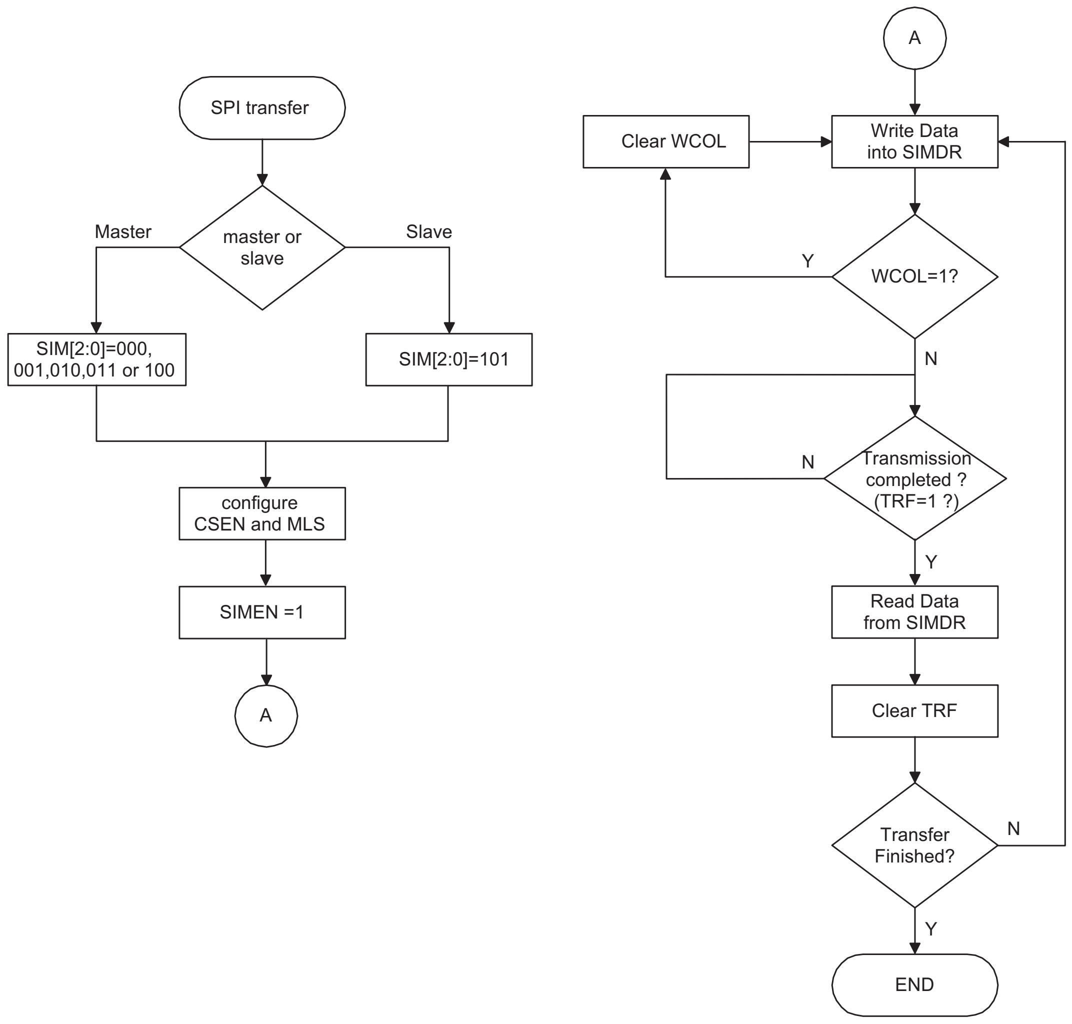

SCS
SCK (CKPOL=1)
SCK (CKPOL=0)
SDO
SDI Data Capture
D7/D0 D6/D1 D5/D2 D4/D3 D3/D4 D2/D5 D1/D6 D0/D7
Write to SIMDR
(SDO not change until first SCK edge)

SPI Slave Mode Timing (CKEG=0)


SCS
SCK (CKPOL=1)
SCK (CKPOL=0)
SDO
SDI Data Capture
D7/D0 D6/D1 D5/D2 D4/D3 D3/D4 D2/D5 D1/D6 D0/D7
Write to SIMDR
(SDO change as soon as writing occur; SDO=floating if SCS=1)

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

SPI Slave Mode Timing (CKEG=1)



SPI Transfer Control Flowchart

SPI Control Register – SIMCTL2

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

- TRF
  The TRF bit is the Transmit/Receive Complete flag and is set high automatically when an SPI data transmission is completed, but must be cleared by the application program. It can be used to generate an interrupt.
- WCOL
  The WCOL bit is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMDR register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared by the application program. Note that using the WCOL bit can be disabled or enabled via configuration option.
- CSEN
  The CSEN bit is used as an on/off control for the $\overline{SCS}$ pin. If this bit is low then the $\overline{SCS}$ pin will be disabled and placed into a floating condition. If the bit is high the $\overline{SCS}$ pin will be enabled and used as a select pin. Note that using the CSEN bit can be disabled or enabled via configuration option.
- MLS
  This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.
- CKEG and CKPOL
  These two bits are used to setup the way that the clock signal outputs and inputs data on the SPI bus. These two bits must be configured before data transfer is executed otherwise an erroneous clock edge may be generated. The CKPOL bit determines the base condition of the clock line, if the bit is high then the SCK line will be low when the clock is inactive. When the CKPOL bit is low then the SCK line will be high when the clock is inactive. The CKEG bit determines active clock edge type which depends upon the condition of CKPOL.

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

SPI Communication

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

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

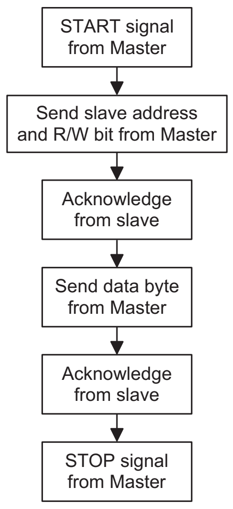
I^{2}C Interface

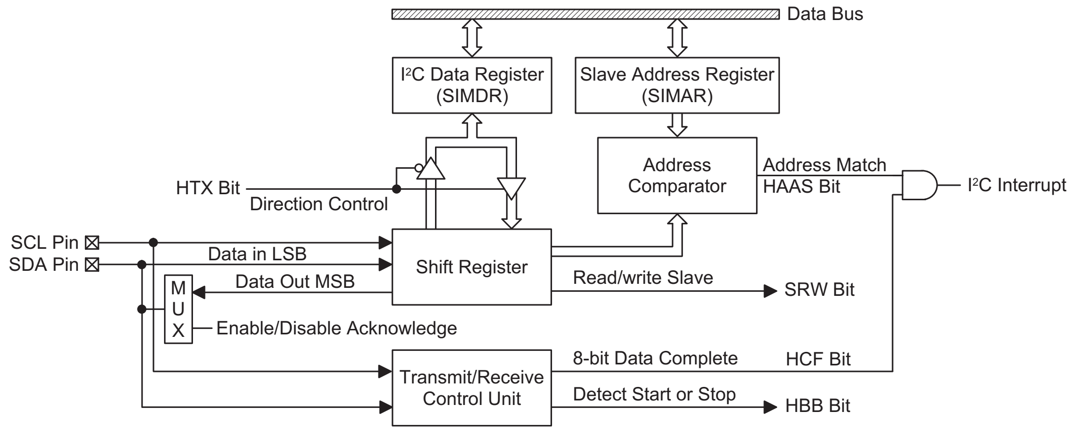
The I^{2}C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple communication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

I^{2}C Interface Operation

The I^{2}C serial interface is a two line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs. Note that no chip select line exists, as each device on the I^{2}C bus is identified by a unique address which will be transmitted and received on the I^{2}C bus.

When two devices communicate with each other on the bidirectional I^{2}C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For these devices, which only operates in slave mode, there are two





I²C Block Diagram

methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode.

There are several configuration options associated with the I²C interface. One of these is to enable the function which selects the SIM pins rather than normal I/O pins. Note that if the configuration option does not select the SIM function then the SIMEN bit in the SIMCTL0 register will have no effect. A configuration option exists to allow a clock other than the system clock to drive the I²C interface. Another configuration option determines the debounce time of the I²C interface. This uses the internal clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 1 or 2 system clocks.

SIM	Function
SIM function	SIM Interface enable or disable
I²C debounce	No debounce, 1 system clock; 2 system clocks

I²C Interface Configuration Options

I²C Registers

There are three control registers associated with the I²C bus, SIMCTL0, SIMCTL1 and SIMAR and one data register, SIMDR. The SIMDR register, which is shown in the above SPI section, is used to store the data being transmitted and received on the I²C bus. Before the microcontroller writes data to the I²C bus, the actual data to be transmitted must be placed in the SIMDR register. After the data is received from the I²C bus, the microcontroller can read it from the SIMDR register. Any transmission or reception of data from the I²C bus must be made via the SIMDR register.

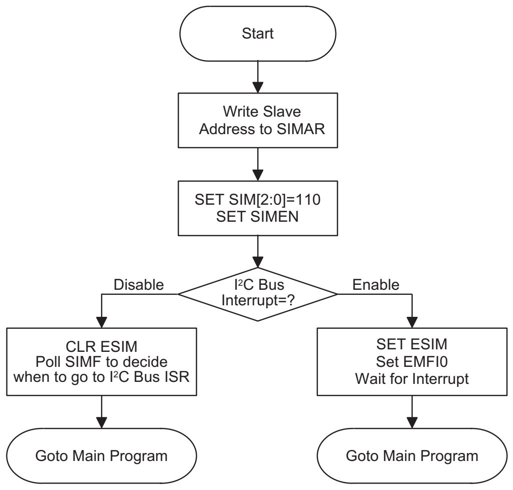
Note that the SIMAR register also has the name SIMCTL2 which is used by the SPI function. Bits SIMIDLE, SIMEN and bits SIM0~SIM2 in register SIMCTL0 are used by the I²C interface. The SIMCTL0 register is shown in the above SPI section.

- SIMIDLE
  The SIMIDLE bit is used to select if the I²C interface continues running when the device is in the IDLE mode. Setting the bit high allows the I²C interface to maintain operation when the device is in the Idle mode. Clearing the bit to zero disables any I²C operations when in the Idle mode.
  This SPI/I²C idle mode control bit is located at CLKMOD register bit4.

- SIMEN
  The SIMEN bit is the overall on/off control for the I²C interface. When the SIMEN bit is cleared to zero to disable the I²C interface, the SDA and SCL lines will be in a floating condition and the I²C operating current will be reduced to a minimum value. When the bit is high the I²C interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. Note that when the SIMEN bit changes from low to high the contents of the I²C control registers will be in an unknown condition and should therefore be first initialised by the application program

- SIM0~SIM2
  These bits setup the overall operating mode of the SIM function. To select the I²C function, bits SIM2~SIM0 should be set to the value 110.

- RXAK
  The RXAK flag is the receive acknowledge flag. When the RXAK bit has been reset to zero it means that a correct acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When in the transmit mode, the transmitter checks the RXAK bit to determine if the receiver wishes to receive the next byte. The transmitter will therefore continue sending out data until the RXAK bit is set high. When this occurs, the device will release the SDA line to allow the master to send a STOP signal to release the bus.



I^{2}C Bus Initialisation Flow Chart

- SRW
  The SRW bit is the Slave Read/Write bit. This bit determines whether the master device wishes to transmit or receive data from the I^{2}C bus. When the transmitted address and slave address match, that is when the HAAS bit is set high, the device will check the SRW bit to determine whether it should be in transmit mode or receive mode. If the SRW bit is high, the master is requesting to read data from the bus, so the device should be in transmit mode. When the SRW bit is zero, the master will write data to the bus, therefore the device should be in receive mode to read this data.
- TXAK
  The TXAK flag is the transmit acknowledge flag. After the receipt of 8-bits of data, this bit will be transmitted to the bus on the 9th clock. To continue receiving more data, this bit has to be reset to zero before further data is received.
- HTX
  The HTX flag is the transmit/receive mode bit. This flag should be set high to set the transmit mode and low for the receive mode.
- HBB
  The HBB flag is the I^{2}C busy flag. This flag will be high when the I^{2}C bus is busy which will occur when a START signal is detected. The flag will be reset to zero when the bus is free which will occur when a STOP signal is detected.
- HASS
  The HASS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.
- HCF
  The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

I^{2}C Control Register – SIMAR

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

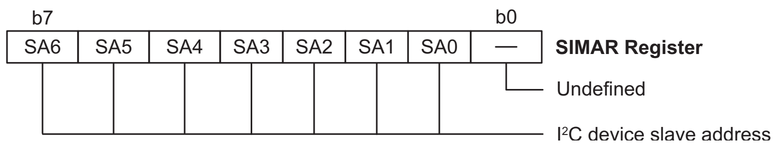
The SIMAR register is the location where the 7-bit slave address of the microcontroller is stored. Bits 1~7 of the SIMAR register define the microcontroller slave address. Bit 0 is not defined. When a master device, which is connected to the I^{2}C bus, sends out an address, which matches the slave address in the SIMAR register, the microcontroller slave device will be selected. Note that the SIMAR register is the same register as SIMCTL2 which is used by the SPI interface.

I^{2}C Bus Communication

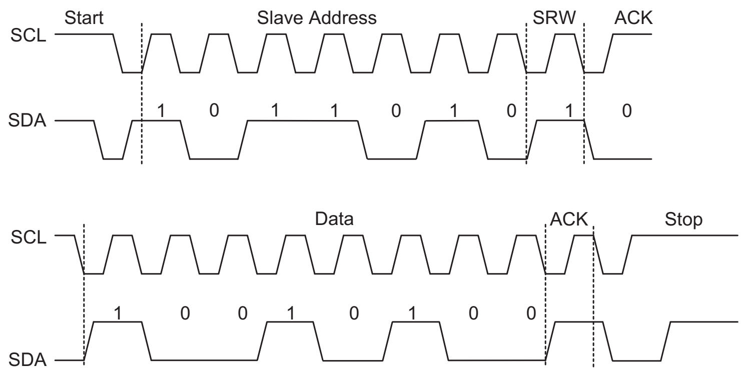
Communication on the I^{2}C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I^{2}C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of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

Step 1

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



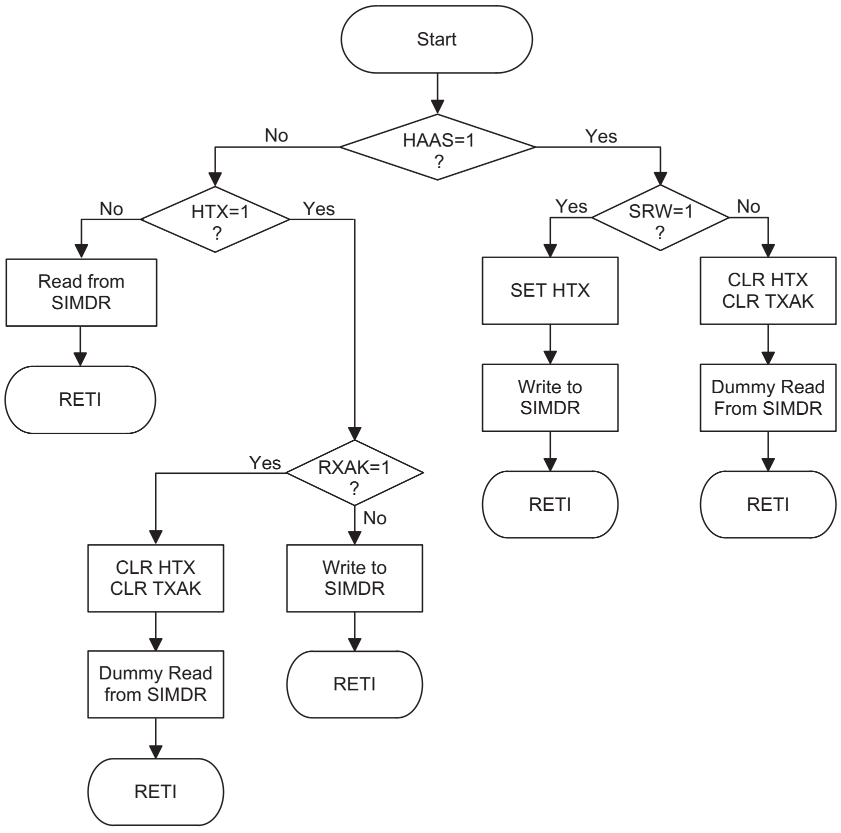
I^{2}C Slave Address Register – SIMAR



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

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

I^{2}C Communication Timing Diagram



I^{2}C Bus ISR Flow Chart

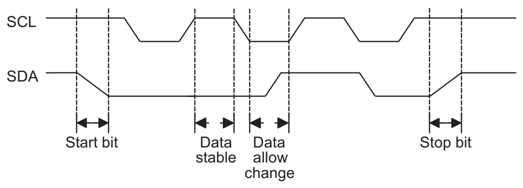
Step 2

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

Step 3

Set the ESIM and EMFI0 bits of the interrupt control register to enable the I^{2}C bus interrupt.

- Start Signal
  The START signal can only be generated by the master device connected to the I^{2}C bus and not by the microcontroller, which is only a slave device. This START signal will be detected by all devices connected to the I^{2}C bus. When detected, this indicates that the I^{2}C bus is busy and therefore the HBB bit will be set. A START condition occurs when a high to low transition on the SDA line takes place when the SCL line remains high.
- Slave Address
  The transmission of a START signal by the master will be detected by all devices on the I^{2}C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal I^{2}C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the SRW bit of the SIMCTL1 register. The device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The microcontroller slave device will also set the status flag HAAS when the addresses match. As an I^{2}C bus interrupt can come from two sources, when the program enters the interrupt subroutine, the HAAS bit should be examined to see whether the interrupt source has come from a matching slave address or from the completion of a data byte transfer. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMDR register, or in the receive mode where it must implement a dummy read from the SIMDR register to release the SCL line.
- SRW Bit
  The SRW bit in the SIMCTL1 register defines whether the microcontroller slave device wishes to read data from the I^{2}C bus or write data to the I^{2}C bus. The microcontroller should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW bit is set to ″1″ then this indicates that the master wishes to read data from the I^{2}C bus, therefore the microcontroller slave device must be setup to send data to the I^{2}C bus as a transmitter. If the SRW bit is ″0″ then this indicates that the master wishes to send data to the I^{2}C bus, therefore the microcontroller slave device must be setup to read data from the I^{2}C bus as a receiver.
- Acknowledge Bit
  After the master has transmitted a calling address, any slave device on the I^{2}C bus, whose own internal address matches the calling address, must generate an acknowledge signal. This acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS bit is high, the addresses have matched and the microcontroller slave device must check the SRW bit to determine if it is to be a transmitter or a receiver. If the SRW bit is high, the microcontroller slave device should be setup to be a transmitter so the HTX bit in the SIMCTL1 register should be set to ″1″ if the SRW bit is low then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMCTL1 register should be set to ″0″.
- Data Byte
  The transmitted data is 8-bits wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8-bits of data, the receiver must transmit an acknowledge signal, level ″0″, before it can receive the next data byte. If the transmitter does not receive an acknowledge bit signal from the receiver, then it will release the SDA line and the master will send out a STOP signal to release control of the I^{2}C bus. The corresponding data will be stored in the SIMDR register. If setup as a transmitter, the microcontroller slave device must first write the data to be transmitted into the SIMDR register. If setup as a receiver, the microcontroller slave device must read the transmitted data from the SIMDR register.



Data Timing Diagram

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

Buzzer

Operating in a similar way to the Programmable Frequency Divider, the Buzzer function provides a means of producing a variable frequency output, suitable for applications such as Piezo-buzzer driving or other external circuits that require a precise frequency generator.

Buzzer Operation

The BZ and $\overline{BZ}$ pins form a complementary pair, and are pin-shared with I/O pins, PA0 and PA1. A configuration option is used to select from one of three buzzer options. The first option is for both pins PA0 and PA1 to be used as normal I/Os, the second option is for both pins to be configured as BZ and $\overline{BZ}$ buzzer pins, the third option selects only the PA0 pin to be used as a BZ buzzer pin with the PA1 pin retaining its normal I/O pin function. Note that the $\overline{BZ}$ pin is the inverse of the BZ pin which together generate a differential output which can supply more power to connected interfaces such as buzzers.

The buzzer is driven by the internal clock source, , which then passes through a divider, the division ratio of which is selected by configuration options to provide a range of buzzer frequencies from $f_S/2^2$ to $f_S/2^9$. The clock source that generates f_S, which in turn controls the buzzer frequency, can originate from three different sources, the 32768Hz oscillator, the 32K_INT oscillator or the System oscillator/4, the choice of which is determined by the f_S clock source configuration option. Note that the buzzer frequency is controlled by configuration options, which select both the source clock for the internal clock f_S and the internal division ratio. There are no internal registers associated with the buzzer frequency.

If the configuration options have selected both pins PA0 and PA1 to function as a BZ and $\overline{BZ}$ complementary pair of buzzer outputs, then for correct buzzer operation it is essential that both pins must be setup as outputs by setting bits PAC0 and PAC1 of the PAC port control register to zero. The PA0 data bit in the PA data register must also be set high to enable the buzzer outputs, if set low, both pins PA0 and PA1 will remain low. In this way the single bit PA0 of the PA register can be used as an on/off control for both the BZ and $\overline{BZ}$ buzzer pin outputs. Note that the PA1 data bit in the PA register has no control over the $\overline{BZ}$ buzzer pin PA1.

PA0/PA1 Pin Function Control

PAC Register PAC0	PAC Register PAC1	PA Data Register PA0	PA Data Register PA1	Output Function

0	0	1	x	PA0=BZ
				PA1=$\overline{BZ}$

0	0	0	x	PA0="0"
				PA1="0"

0	1	1	x	PA0=BZ
				PA1=input line

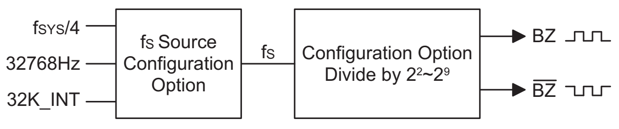
0	1	0	x	PA0="0"
				PA1=input line

1	0	x	D	PA0=input line
				PA1=D

1	1	x	x	PA0=input line PA0=input line

"x" stands for don't care

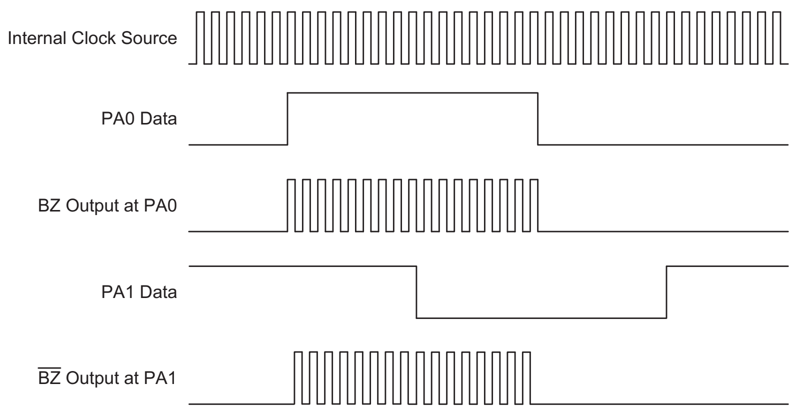
"D" stands for Data "0" or "1"


$f_{SYS}/4$
32768Hz
32K_INT
f_S Source Configuration Option
f_S
Configuration Option Divide by 2^2~2^9
BZ
$\overline{BZ}$

Buzzer Function

If configuration options have selected that only the PA0 pin is to function as a BZ buzzer pin, then the PA1 pin can be used as a normal I/O pin. For the PA0 pin to function as a BZ buzzer pin, PA0 must be setup as an output by setting bit PAC0 of the PAC port control register to zero. The PA0 data bit in the PA data register must also be set high to enable the buzzer output, if set low pin PA0 will remain low. In this way the PA0 bit can be used as an on/off control for the BZ buzzer pin PA0. If the PAC0 bit of the PAC port control register is set high, then pin PA0 can still be used as an input even though the configuration option has configured it as a BZ buzzer output.

Note that no matter what configuration option is chosen for the buzzer, if the port control register has setup the pin to function as an input, then this will override the configuration option selection and force the pin to always behave as an input pin. This arrangement enables the pin to be used as both a buzzer pin and as an input pin, so regardless of the configuration option chosen; the actual function of the pin can be changed dynamically by the application program by programming the appropriate port control register bit.



Buzzer Output Pin Control

Note: The above drawing shows the situation where both pins PA0 and PA1 are selected by configuration option to be BZ and $\overline{BZ}$ buzzer pin outputs. The Port Control Register of both pins must have already been setup as output. The data setup on pin PA1 has no effect on the buzzer outputs.

Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer/Event Counter or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupts functions. The external interrupts are controlled by the action of the external INT0, INT1 and $\overline{\text{PINT}}$ pins, while the internal interrupts are controlled by functions such as the Timer/Event Counter overflows, the Time Base interrupt, the RTC interrupt, the SPI/I^{2}C interrupt, C/R to F converter interrupt and the A/D converter interrupt etc.

Interrupt Registers

Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by the INTC0, INTC1, MFIC0, and MFIC1 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

A Timer/Event Counter overflow, Time Base, RTC overflow, SPI/I^{2}C data transfer complete, C/R to F converter interrupt, an end of A/D conversion or the external interrupt line being triggered are some of the events which will generate an interrupt request by setting their corresponding request flag. When this happens and if their appropriate interrupt enable bit is set, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a JMP statement which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI statement, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

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

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

Interrupt Priority

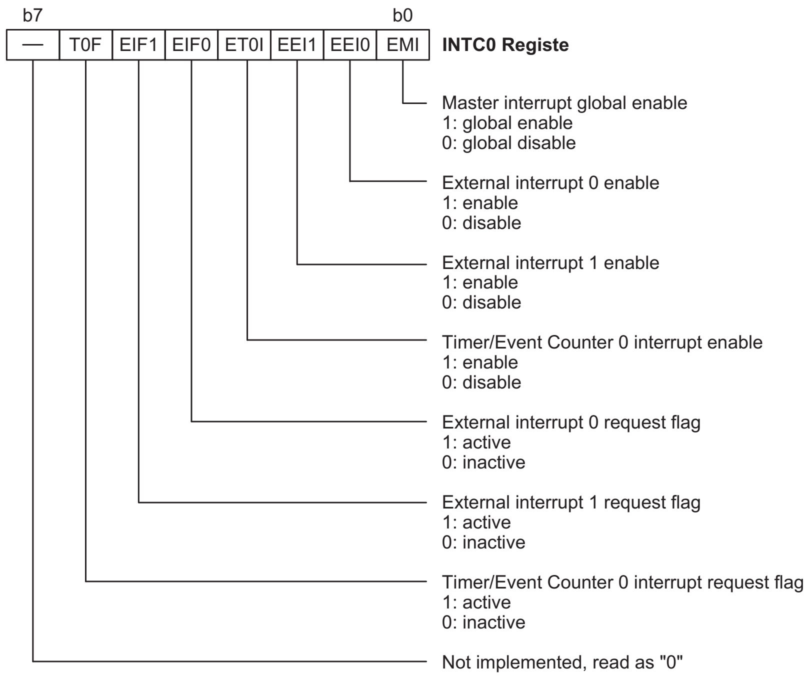
Interrupts, occurring in the interval between the rising edges of two consecutive T2 pulses, will be serviced on the latter of the two T2 pulses, if the corresponding interrupts are enabled. In case of simultaneous requests, the following table shows the priority that is applied.

Interrupt Source	Priority	Vector
External Interrupt 0	1	04H
External Interrupt 1	2	08H
Timer/Event Counter 0 Overflow	3	0CH
Timer/Event Counter 1 Overflow	4	10H
Multi Function 0 Interrupt	5	14H
Multi Function 1 Interrupt	6	18H

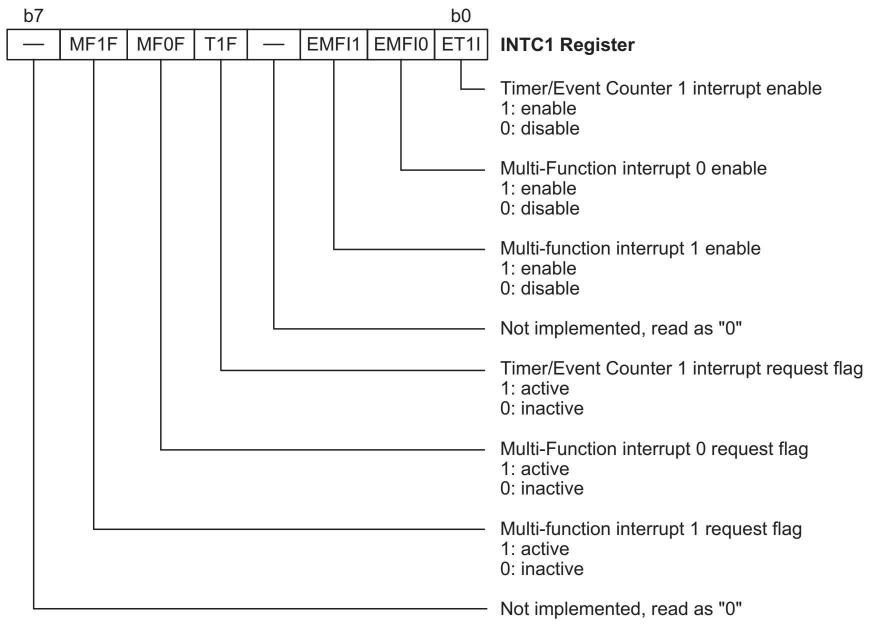
The SPI/I^{2}C interrupt, C/R to F converter interrupt share the same vector which is Multi Function 0 Interrupt vector at location 14H. The A/D converter interrupt, Real Time clock interrupt, Time Base interrupt and External Peripheral interrupt share the same vector which is the Multi Function 1 Interrupt vector at location 18H. Each interrupt has its own interrupt flag but share the global MF0F or MF1F Multi Function interrupt flag. The MF0F and MF1F flags will be cleared by hardware once the Multi-function interrupt is serviced, however the individual interrupts that have triggered the Multi-function interrupt need to be cleared by the application program

External Interrupt

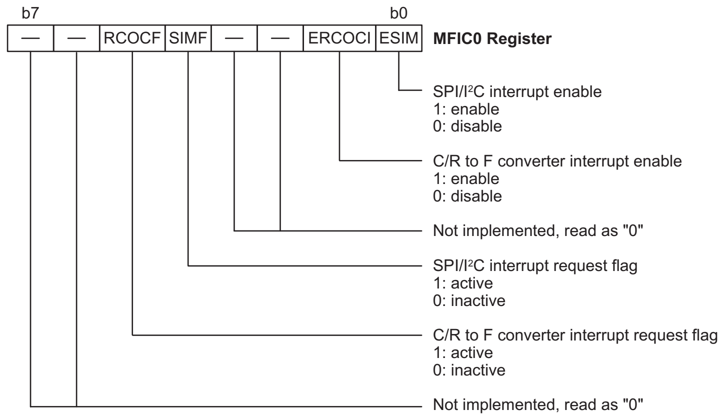
For an external interrupt to occur, the global interrupt enable bit, EMI, and external interrupt enable bits, EEI0 and EEI1, must first be set. Additionally the correct interrupt edge type must be selected using the INTEDGE register to enable the external interrupt function and to choose the trigger edge type. An actual external interrupt will take place when the external interrupt request flag, EIF0 or EIF1, is set, a situation that will occur when a transition, whose type is chosen by the edge select bit, appears on the INT0 or INT1 pin. The external interrupt pins are pin-shared with the I/O pins PA6 and PA7 and can only be configured as external interrupt pins if their corresponding external interrupt enable bit in the INTC0 register has been set.


b7
b0
—
T0F
EIF1
EIF0
ET0I
EEI1
EEI0
EMI
INTC0 Registe
Master interrupt global enable
1: global enable
0: global disable
External interrupt 0 enable
1: enable
0: disable
External interrupt 1 enable
1: enable
0: disable
Timer/Event Counter 0 interrupt enable
1: enable
0: disable
External interrupt 0 request flag
1: active
0: inactive
External interrupt 1 request flag
1: active
0: inactive
Timer/Event Counter 0 interrupt request flag
1: active
0: inactive
Not implemented, read as "0"

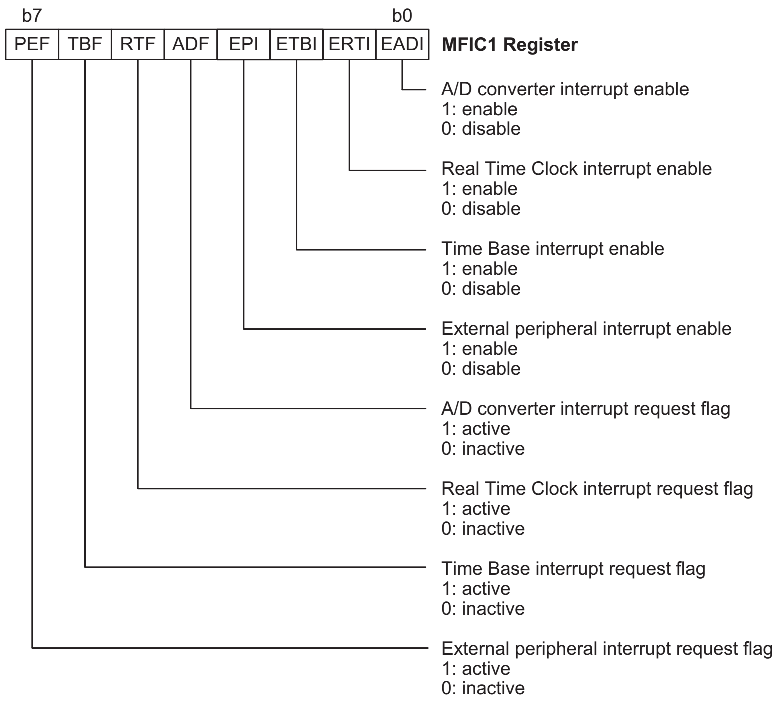
Interrupt Control Register INTC0


b7
b0
—
MF1F
MF0F
T1F
—
EMFI1
EMFI0
ET1I
INTC1 Register
Timer/Event Counter 1 interrupt enable
1: enable
0: disable
Multi-Function interrupt 0 enable
1: enable
0: disable
Multi-function interrupt 1 enable
1: enable
0: disable
Not implemented, read as "0"
Timer/Event Counter 1 interrupt request flag
1: active
0: inactive
Multi-Function interrupt 0 request flag
1: active
0: inactive
Multi-function interrupt 1 request flag
1: active
0: inactive
Not implemented, read as "0"

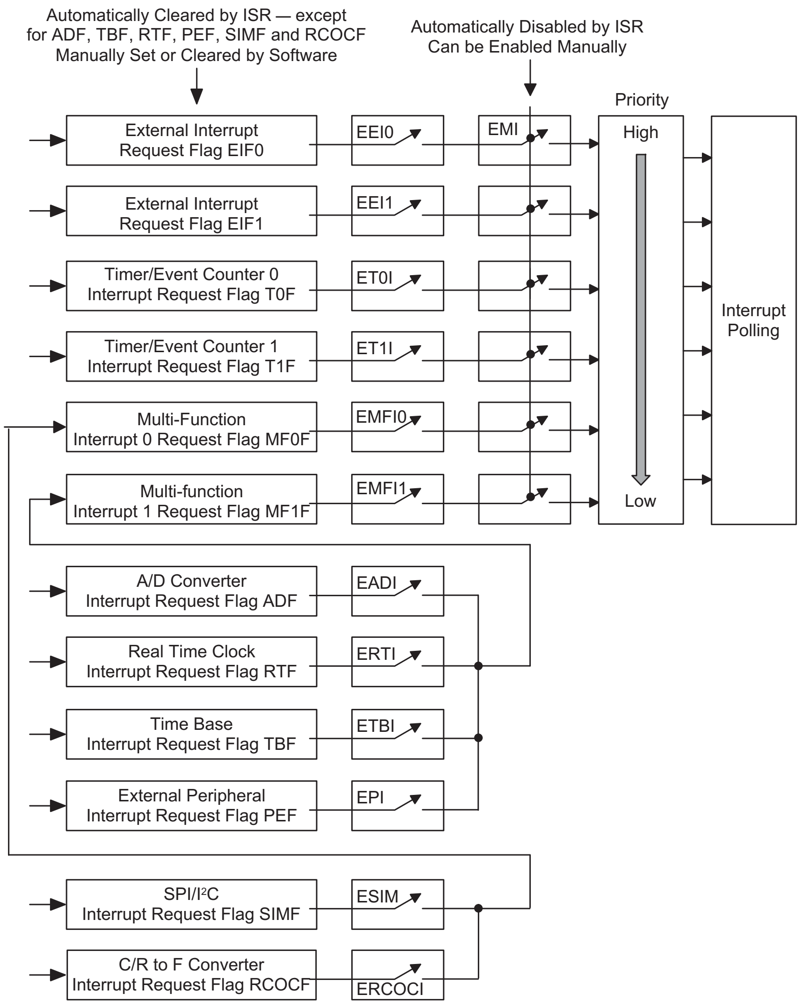
Interrupt Control Register INTC1


b7
b0
—
—
RCOCF
SIMF
—
—
ERCOCI
ESIM
MFIC0 Register
SPI/I²C interrupt enable
1: enable
0: disable
C/R to F converter interrupt enable
1: enable
0: disable
Not implemented, read as "0"
SPI/I²C interrupt request flag
1: active
0: inactive
C/R to F converter interrupt request flag
1: active
0: inactive
Not implemented, read as "0"

Interrupt Control Register – MFIC0


b7
b0
PEF
TBF
RTF
ADF
EPI
ETBI
ERTI
EADI
MFIC1 Register
A/D converter interrupt enable
1: enable
0: disable
Real Time Clock interrupt enable
1: enable
0: disable
Time Base interrupt enable
1: enable
0: disable
External peripheral interrupt enable
1: enable
0: disable
A/D converter interrupt request flag
1: active
0: inactive
Real Time Clock interrupt request flag
1: active
0: inactive
Time Base interrupt request flag
1: active
0: inactive
External peripheral interrupt request flag
1: active
0: inactive

Interrupt Control Register – MFIC1


Automatically Cleared by ISR — except for ADF, TBF, RTF, PEF, SIMF and RCOCF Manually Set or Cleared by Software
Automatically Disabled by ISR Can be Enabled Manually
External Interrupt Request Flag EIF0
External Interrupt Request Flag EIF1
Timer/Event Counter 0 Interrupt Request Flag T0F
Timer/Event Counter 1 Interrupt Request Flag T1F
Multi-Function Interrupt 0 Request Flag MF0F
Multi-function Interrupt 1 Request Flag MF1F
A/D Converter Interrupt Request Flag ADF
Real Time Clock Interrupt Request Flag RTF
Time Base Interrupt Request Flag TBF
External Peripheral Interrupt Request Flag PEF
SPI/I²C Interrupt Request Flag SIMF
C/R to F Converter Interrupt Request Flag RCOCF
EEI0
EEI1
ET0I
ET1I
EMFI0
EMFI1
EADI
ERTI
ETBI
EPI
ESIM
ERCOCI
EMI
Priority
High
Low
Interrupt Polling

Interrupt Structure

The pin must also be setup as an input by setting the corresponding PAC.6 and PAC.7 bits in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector at location 04H or 08H, will take place. When the interrupt is serviced, the external interrupt request flags, EIF0 or EIF1, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on this pin will remain valid even if the pin is used as an external interrupt input.

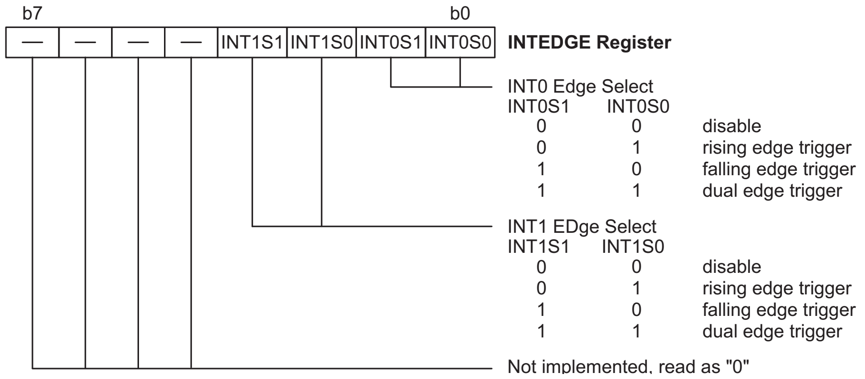
The INTEDGE register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising and falling edge types can be chosen along with an option to allow both edge types to trigger an external interrupt. Note that the INTEDGE register can also be used to disable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

The external interrupt pins are connected to an internal filter to reduce the possibility of unwanted external interrupts due to adverse noise or spikes on the external interrupt input signal. As this internal filter circuit will consume a limited amount of power, a configuration option is provided to switch off the filter function, an option which may be beneficial in power sensitive applications, but in which the integrity of the input signal is high. Care must be taken when using the filter on/off configuration option as it will be applied not only to both the external interrupt pins but also to the Timer/Event Counter external input pins. Individual external interrupt or Timer/Event Counter pins cannot be selected to have a filter on/off function.

External Peripheral Interrupt

The External Peripheral Interrupt operates in a similar way to the external interrupt and is contained within Multi-function Interrupt 1.

For an external peripheral interrupt to occur, the global interrupt enable bit, EMI, external peripheral interrupt enable bit, EPI, and Multi-function Interrupt 1 enable bit,



Interrupt Active Edge Register – INTEDGE

EMF1I, must first be set. An actual external peripheral interrupt will take place when the external interrupt request flag, PEF, is set, a situation that will occur when a negative transition, appears on the $\overline{\text{PINT}}$ pin. When the interrupt is enabled, the stack is not full and a negative transition type appears on the external peripheral interrupt pin, a subroutine call to the Multi-function Interrupt 1 vector at location18H, will take place. When the external peripheral interrupt is serviced, the EMI bit will be cleared to disable other interrupts, however only the MF1F interrupt request flag will be reset. As the PEF flag will not be automatically reset, it has to be cleared by the application program.

The pin must also be setup as an input by setting high the corresponding PDC.6 bit in the port control register. Note that any pull-high resistor software configuration on this pin will remain valid even if the pin is used as an external interrupt input.

Timer/Event Counter Interrupt

For a Timer/Event Counter 0 or Timer/Event Counter 1 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 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 0CH or 10C, 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.

A/D Interrupt

The A/D Interrupt is contained within the Multi-function Interrupt 1.

For an A/D Interrupt to be generated, the global interrupt enable bit, EMI, A/D Interrupt enable bit, EADI, and Multi-function Interrupt 1 enable bit, EMF1I, must first be set. An actual A/D Interrupt will take place when the A/D Interrupt request flag, ADF, is set, a situation that will occur when the A/D conversion process has finished. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the Multi-function Interrupt 1 vector at location18H, will take place. When the A/D Interrupt is serviced, the EMI bit will be cleared to disable other interrupts, however only the MF1F interrupt request flag will be reset. As the ADF flag will not be automatically reset, it has to be cleared by the application program.

Multi-function Interrupts

Additional interrupts known as the Multi-function interrupts are provided. Unlike the other interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the A/D Converter interrupt, Time Base interrupt, Real Time Clock interrupt, External Peripheral interrupt, SIM Interface Interrupt and the C/R to F interrupt.

For a Multi-function interrupt to occur, the global interrupt enable bit, EMI, and the Multi-function interrupt enable bit, EMF0I or EMF1I, must first be set. An actual Multi-function interrupt will take place when the Multi-function interrupt request flag, MF0F, or MF1F is set. This will occur when either a Time Base overflow, a Real Time Clock overflow, an A/D conversion completion, an External Peripheral Interrupt, C/R to F converter counters, Timer A or Timer B overflow or SIM data transfer or I2C address match occurs. When the interrupt is enabled and the stack is not full, and either one of the interrupts contained within the Multi-function interrupts occurs, a subroutine call to one of the Multi-function interrupt vector at location 014H or 018H will take place. When the interrupt is serviced, the Multi-Function request flag, MF0F or MF1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. However, it must be noted that the request flags from the original source of the Multi-function interrupt, namely the Time-Base interrupt, Real Time Clock interrupt, A/D Converter interrupt, External Peripheral interrupt, SIM Interface or C/R to F converter Interrupt will not be automatically reset and must be manually reset by the application program.

SPI/I^{2}C Interface Interrupt

The SPI/I^{2}C interface Interrupt is contained within the Multi-function Interrupt 0.

For an /I^{2}C interrupt to occur, the global interrupt enable bit, EMI, the corresponding interrupt enable bit, ESIM and Multi-function Interrupt 0 enable bit, EMF0I, must be first set. An actual SPI/I^{2}C interrupt will take place when the SPI/I^{2}C reset function interface request flag, SIMF, is set, a situation that will occur when a byte of data has been transmitted or received by the SPI/I^{2}C interface or when an I^{2}C address match occurs. When the interrupt is enabled, the stack is not full and a byte of data has been transmitted or received by the SPI/I^{2}C interface or an I^{2}C address match occurs, a subroutine call to the Multi-function Interrupt 0 vector at location 14H, will take place. When the interrupt is serviced, the Multi-function Interrupt 0 request flag, MF0F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. As the SIMF flag will not be automatically reset it has to be cleared by the application program.

C/R to F Converter Interrupt

The C/R to converter Interrupt is contained within the Multi-function Interrupt 0.
For a C/R to F converter interrupt to be generated, the global interrupt enable bit, EMI, the corresponding interrupt enable bit, ERCOCI and Multi-function Interrupt 0 enable bit, EMF0I, must be first set. An actual C/R to F converter interrupt will take place when the C/R to F converter interrupt request flag, RCOCF, is set, a situation that will occur when one of the C/R to F converter counters, Timer A or Timer B, overflows. When the interrupt is enabled, the stack is not full and a C/R to F converter counter overflow occurs, a subroutine call to the Multi-function Interrupt 0 vector at location 14H, will take place. When the interrupt is serviced, the Multi-function Interrupt 0 request flag, MF0F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. As the RCOCF flag will not be automatically reset it has to be cleared by the application program.

Real Time Clock Interrupt

The Real Time Clock Interrupt is contained within the Multi-function Interrupt 1.

For a Real Time Clock interrupt to be generated, the global interrupt enable bit, EMI , Real Time Clock interrupt enable bit, ERTI, and Multi-function Interrupt 1 enable bit, EMF1I, must first be set. An actual Real Time Clock interrupt will take place when the Real Time Clock request flag, RTF, is set, a situation that will occur when the Real Time Clock overflows. When the interrupt is enabled, the stack is not full and the Real Time Clock overflows, a subroutine call to the Multi-function Interrupt 1 vector at location18H, will take place. When the Real Time Clock interrupt is serviced, the EMI bit will be cleared to disable other interrupts, however only the MF1F interrupt request flag will be reset. As the RTF flag will not be automatically reset, it has to be cleared by the application program.

Similar in operation to the Time Base interrupt, the purpose of the RTC interrupt is also to provide an interrupt signal at fixed time periods. The RTC interrupt clock source originates from the internal clock source f_S. This f_S input clock first passes through a divider, the division ratio of which is selected by programming the appropriate bits in the RTCC register to obtain longer RTC interrupt periods whose value ranges from $2^8/f_S$~$2^{15}/f_S$. The clock source that generates f_S, which in turn controls the RTC interrupt period, can originate from three different sources, the 32768Hz oscillator, 32K_INT oscillator or the System oscillator/4, the choice of which is determine by the f_S clock source configuration option.

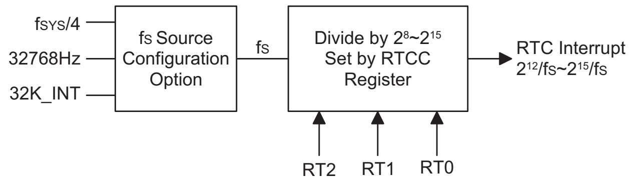
Note that the RTC interrupt period is controlled by both configuration options and an internal register RTCC. A configuration option selects the source clock for the internal clock f_S, and the RTCC register bits RT2, RT1 and RT0 select the division ratio. Note that the actual division ratio can be programmed from 2^8 to 2^{15}.

Essentially operating as a programmable timer, when the Real Time Clock overflows it will set a Real Time Clock interrupt flag which will in turn generate an Interrupt request via the Multi-function Interrupt 1 vector.

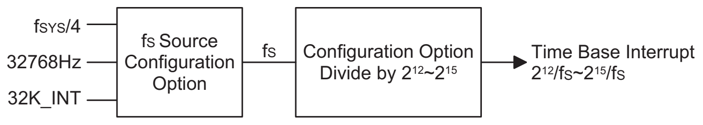
Time Base Interrupt

The Time Base Interrupt is contained within the Multi-function Interrupt 1.

For a Time Base Interrupt to be generated, the global interrupt enable bit, EMI,Time Base Interrupt enable bit, ETBI, and Multi-function Interrupt enable 1 bit, EMF1I, must first be set. An actual Time Base Interrupt will take place when the Time Base Interrupt request flag, TBF, is set, a situation that will occur when the Time Base over-


fSYS/4
32768Hz
32K_INT
fS Source Configuration Option
fS
Divide by 2^8~2^15 Set by RTCC Register
RTC Interrupt 2^12/fs~2^15/fs
RT2
RT1
RT0

RTC Interrupt


fSYS/4
32768Hz
32K_INT
fS Source Configuration Option
fS
Configuration Option Divide by 2^{12}~2^{15}
Time Base Interrupt 2^{12}/fS~2^{15}/fS

Time Base Interrupt

flows. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to the Multi-function Interrupt 1 vector at location18H, will take place. When the Time Base Interrupt is serviced, the EMI bit will be cleared to disable other interrupts, however only the MF1F interrupt request flag will be reset. As the TBF flag will not be automatically reset, it has to be cleared by the application program.

The purpose of the Time Base function is to provide an interrupt signal at fixed time periods. The Time Base interrupt clock source originates from the Time Base interrupt clock source originates from the internal clock source f_S. This f_S input clock first passes through a divider, the division ratio of which is selected by configuration options to provide longer Time Base interrupt periods. The Time Base interrupt time-out period ranges from $2^{12}/f_S$~$2^{15}/f_S$. The clock source that generates f_S, which in turn controls the Time Base interrupt period, can originate from three different sources, the 32768Hz oscillator, the 32K_INT internal oscillator or the System oscillator/4, the choice of which is determine by the f_S clock source configuration option.

Essentially operating as a programmable timer, when the Time Base overflows it will set a Time Base interrupt flag which will in turn generate an Interrupt request via the Multi-function Interrupt 1 vector.

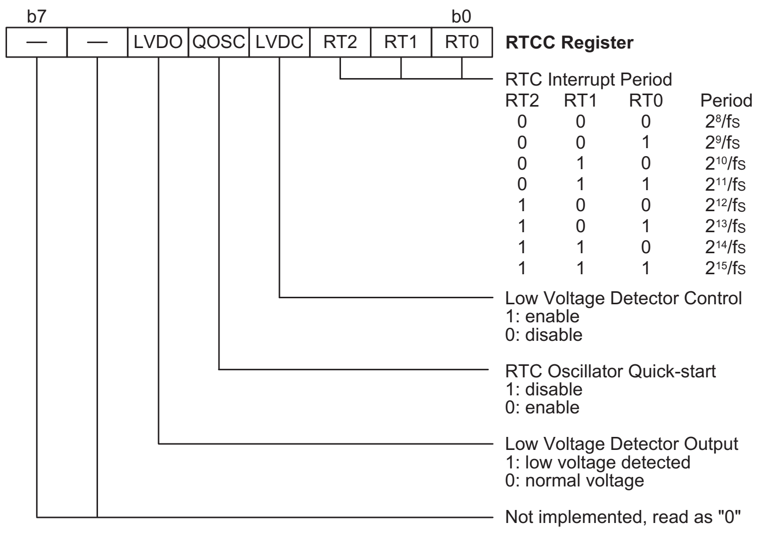
Programming Considerations

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 INTC0, INTC1, MFIC0 and MFIC1 registers until the corresponding interrupt is serviced or until the request flag is cleared by the application program. Note that if a specific interrupt uses a Multi-function Interrupt vector then its interrupt request flag will not be automatically reset when the program enters the interrupt service routine. Only the Multi-function interrupt request flag will be automatically reset.

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 status or other registers are altered by the interrupt service program, which may corrupt the desired control sequence, then the contents should be saved in advance.



RT2	RT1	RT0	Period
0	0	0	2^8/fs
0	0	1	2^9/fs
0	1	0	2^{10}/fs
0	1	1	2^{11}/fs
1	0	0	2^{12}/fs
1	0	1	2^{13}/fs
1	1	0	2^{14}/fs
1	1	1	2^{15}/fs

Real Time Clock Control Register – RTCC

Reset and Initialisation

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

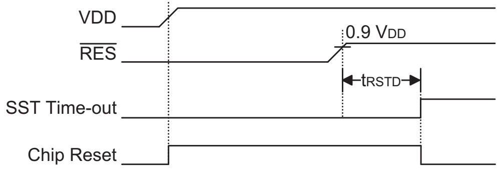
In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{\text{RES}}$ line is forcefully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

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

Reset Functions

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

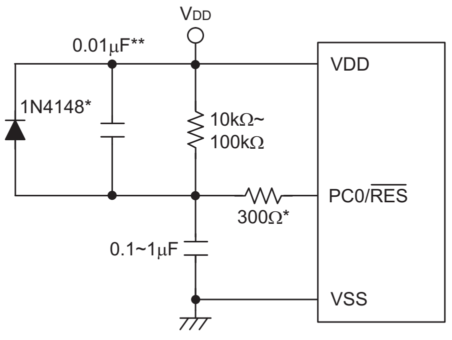
- Power-on Reset
  The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. All the I/O port and port control registers will power up in a high condition ensuring that all pins will be first set to inputs.
  Although the microcontroller has an internal RC reset function, if the VDD power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the $\overline{\text{RES}}$ pin, whose additional time delay will ensure that the $\overline{\text{RES}}$ pin remains low for an extended period to allow the power supply to stabilise. As the $\overline{\text{RES}}$ pin is shared with an I/O pin its function must be selected using a configuration option. 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.

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



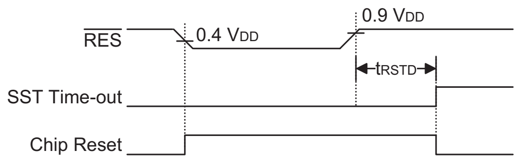
Note: "*" It is recommended that this component is added for added ESD protection

"**" It is recommended that this component is added in environments where power line noise is significant

External $\overline{\text{RES}}$ Circuit

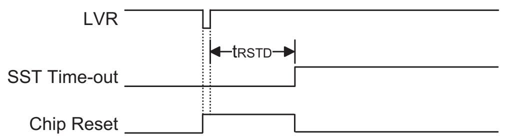
More information regarding external reset circuits is located in Application Note HA0075E on the Holtek website.

• $\overline{\text{RES}}$ Pin Reset
This type of reset occurs when the microcontroller is already running and the $\overline{\text{RES}}$ pin is forcefully pulled low by external hardware such as an external switch. In this case as in the case of other reset, the Program Counter will reset to zero and program execution initiated from this point. The $\overline{\text{RES}}$ pin is shared with an I/O pin so its function must be first selected using a configuration option.



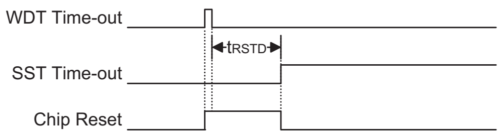
$\overline{\text{RES}}$ Reset Timing Chart

• Low Voltage Reset – LVR
The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device, which is selected via a configuration option. If the supply voltage of the device drops to within a range of 0.9V~V_{LVR} such as might occur when changing the battery, the LVR will automatically reset the device internally. The LVR includes the following specifications: For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~V_{LVR} must exist for greater than the value t_{LVR} specified in the A.C. characteristics. If the low voltage state does not exceed 1ms, the LVR will ignore it and will not perform a reset function. One of a range of specified voltage values for V_{LVR} can be selected using configuration options. The V_{LVR} value will be selected as a pair in conjunction with a Low Voltage Detect value.



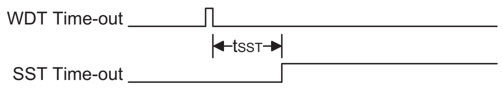
Low Voltage Reset Timing Chart

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



WDT Time-out Reset during Normal Operation Timing Chart

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



WDT Time-out Reset during Power Down

Reset Initial Conditions

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

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

Note: ″u″ stands for unchanged

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

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

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects each of the microcontroller internal registers. Note that where more than one package type exists the table will reflect the situation for the larger package type.

Register	Reset (Power-on)	$\overline{RES}$ Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	–xxx xxxx	–uuu uuuu	–uuu uuuu	–uuu uuuu
RTCC	––00 0111	––00 0111	––00 0111	––uu uuuu
STATUS	––00 xxxx	––uu uuuu	––1u uuuu	––11 uuuu
INTC0	–000 0000	–000 0000	–000 0000	–uuu uuuu
TMR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR0C	00–0 1000	00–0 1000	00–0 1000	uu–u uuuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	––11 1111	––11 1111	––11 1111	––uu uuuu
PBC	––11 1111	––11 1111	––11 1111	––uu uuuu
PC	–––1 1111	–––1 1111	–––1 1111	–––u uuuu
PCC	–––1 1111	–––1 1111	–––1 1111	–––u uuuu
PD	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PWM0L	0000 –––0	0000 –––0	0000 –––0	uuuu –––u
PWM0H	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWM1L	0000 –––0	0000 –––0	0000 –––0	uuuu –––u
PWM1H	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC1	–000 –000	–000 –000	–000 –000	–uuu –uuu
MFIC0	––00 ––00	––00 ––00	––00 ––00	––uu ––uu
MFIC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
ASCR0	1111 1111	1111 1111	1111 1111	uuuu uuuu
ASCR1	1111 1111	1111 1111	1111 1111	uuuu uuuu
ASCR2	1111 1111	1111 1111	1111 1111	uuuu uuuu
ADRL	xxxx ––––	xxxx ––––	xxxx ––––	uuuu ––––
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	10–– –000	10–– –000	10–– –000	uu–– –uuu
CLKMOD	0000 0x11	0000 0x11	0000 0x11	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	Reset (Power-on)	$\overline{RES}$ Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (HALT)
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBPU	– – 00 0000	– – 00 0000	– – 00 0000	– – uu uuuu
PCPU	– – – 0 000 –	– – – 0 000 –	– – – 0 000 –	– – – u uuu –
PDPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTEDGE	– – – – 0000	– – – – 0000	– – – – 0000	– – – – uuuu
MISC	0000 1010	0000 1010	0000 1010	uuuu uuuu
TMRAH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRAL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RCOCCR	0000 1 – – –	0000 1 – – –	0000 1 – – –	uuuu u – – –
TMRBH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMRBL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RCOCR	1xxx – – 00	1xxx – – 00	1xxx – – 00	uuuu – – uu
SIMCTL0	1110 000 –	1110 000 –	1110 000 –	uuuu uuu –
SIMCTL1	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMDR	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMAR/SIMCTL2	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
TMR1C	00 – 0 1000	00 – 0 1000	00 – 0 1000	uu – u uuuu

Note: ″u″ stands for unchanged
″x″ stands for unknown
″–″ stands for unimplemented

Oscillator

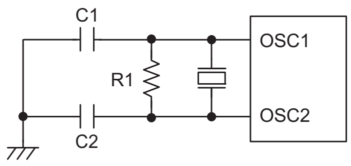
Various oscillator options offer the user a wide range of functions according to their various application requirements. Five types of system clocks can be selected while various clock source options for the Watchdog Timer are provided for maximum flexibility. All oscillator options are selected through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

System Clock Configurations

There are six methods of generating the system clock, three high oscillators, two low oscillators and an externally supplied clock. The three high oscillators are the external crystal/ceramic oscillator, internal RC oscillator and the external RC network. The two low oscillators are the fully integrated 32K_INT oscillator and the external 32768Hz oscillator. Selecting whether the low or high oscillator is used as the system oscillator is implemented using the HLCLK bit in the CLKMOD register. The source clock for the high and low oscillators is chosen via configuration options. The frequency of the slow oscillator is also determined using the SLOWC0~SLOWC2 bits in the CLKMOD register.

System Crystal/Ceramic Oscillator

After selecting the external crystal configuration option, the simple connection of a crystal across OSC1 and OSC2, is normally all that is required to 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



Crystal/Ceramic Oscillator

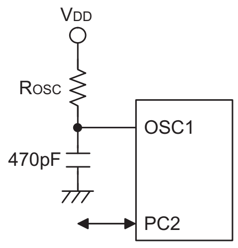
C2. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be connected as shown for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. In most applications, resistor R1 is not required, however for those applications where the LVR function is not used, R1 may be necessary to ensure the oscillator stops running when VDD falls below its operating range. The internal oscillator circuit contains a filter circuit to reduce the possibility of erratic operation due to noise on the oscillator pins. An additional configuration option must be setup to configure the device according to whether the oscillator frequency is high, defined as equal to or above 1MHz, or low, which is defined as below 1 MHz.

More information regarding oscillator applications is located on the Holtek website.

External System RC Oscillator

After selecting the correct configuration option, using the external system RC oscillator requires that a resistor, with a value between 47kΩ and 1.5MΩ, is connected between OSC1 and VDD, and a 470pF capacitor is connected to ground. Although this is a cost effective oscillator configuration, the oscillation frequency can vary with VDD, temperature and process variations and is therefore not suitable for applications where timing is critical or where accurate oscillator frequencies are required. For the value of the external resistor R_{OSC} refer to the Appendix section for typical RC Oscillator vs. Temperature and VDD characteristics graphics.

Note that an internal capacitor together with the external resistor, R_{OSC}, are the components which determine the frequency of the oscillator. The external capacitor shown on the diagram does not influence the frequency of oscillation. Note that if this external system RC oscillation option is selected, as it requires OSC1 external pin for its operation, the PC2/OSC2 pin is free for use as normal I/O pin.The internal oscillator circuit contains a filter circuit to reduce the possibility of erratic operation due to noise on the oscillator pins.



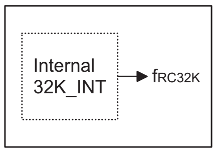
RC Oscillator

Internal RC Oscillator

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of either 4MHz, 8MHz or 12MHz, the choice of which is indicated by the configuration options. Note that if this internal system clock option is selected, as it requires no external pins for its operation, the OSC1 and OSC2 pins are free for use as normal I/O pins. Refer to the Appendix section for more information on the actual internal oscillator frequency vs. Temperature and VDD characteristics graphics.

Internal 32K_INT Oscillator

When the device enters the Sleep, Slow or Idle Mode, its high frequency clock is switched off to reduce activity and to conserve power. However, in many microcontroller applications it may be necessary to keep some internal functions operational even when the microcontroller is in the Power-down mode. To do this, the device has a 32K_INT oscillator, which is a fully integrated free running RC oscillator with a typical period of 31.2μs at 5V, requiring no external components. It is selected via configuration option.



Internal 32K_INT Oscillator

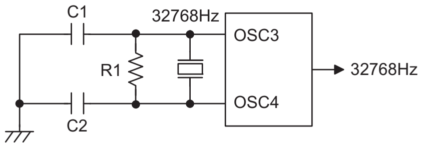
External 32768Hz Oscillator

With a function similar to the internal 32K-INT 32KHz oscillator, that is to keep some device functions operational during power down, this device also has an external 32768Hz oscillator. This clock source has a fixed frequency of 32768Hz and requires a 32768Hz crystal to be connected between pins OSC3 and OSC4.

The external resistor and capacitor components connected to the 32768Hz crystal are not necessary to provide oscillation. For applications where precise frequencies are essential, these components may be required to provide frequency compensation due to different crystal manufacturing tolerances.

A configuration option selects whether the external 32768Hz oscillator or the internal 32K_INT oscillator is selected. Selecting low frequency oscillators for use as a system oscillator is implmented using bits in the CLKMOD register.

During power-up there is a time delay associated with the 32768Hz oscillator waiting for it to start-up. To minimise this time delay, bit 4 of the RTCC register, known as the QOSC bit, is provided to have a quick start-up function. During a power-up condition, this bit will be cleared to zero which will initiate the 32768Hz oscillator quick start-up function. However, as there is additional power consumption associated with this quick start-up function, to reduce power consumption after start-up takes place, it is recommended that the application program should set the QOSC bit high for about 2 seconds after power-on. It should be noted that, no matter what condition the QOSC bit is set to, the 32768Hz oscillator will always function normally, only there is more power consumption associated with the quick start-up function.



External 32768Hz Oscillator

Note that if this external 32768Hz oscillation option is not selected, the PC3/OSC3 and PC4/OSC4 pins are free for use as normal I/O pins.

External Oscillator

The system clock can also be supplied by an externally supplied clock giving users a method of synchronising their external hardware to the microcontroller operation. This is selected using a configuration option and supplying the clock on pin OSC1. Note that if this external system clock option is selected, as it requires OSC1 external pin for its operation, the PC2/OSC2 pin is free for use as normal I/O pin. The internal oscillator circuit contains a filter circuit to reduce the possibility of erratic operation due to noise on the oscillator pin.

System Operating Modes

The devices have the ability to operate in several different modes. This range of operating modes, known as Normal Mode, Slow Mode, Idle Mode and Sleep Mode, allow the devices to run using a wide range of different slow and fast clock sources. The devices also possess the ability to dynamically switch between different clocks and operating modes. With this choice of operating functions, and a HALT instruction, users are provided with a high degree of flexibility to ensure they obtain optimal performance from the device, in terms of operating speed and power requirements.

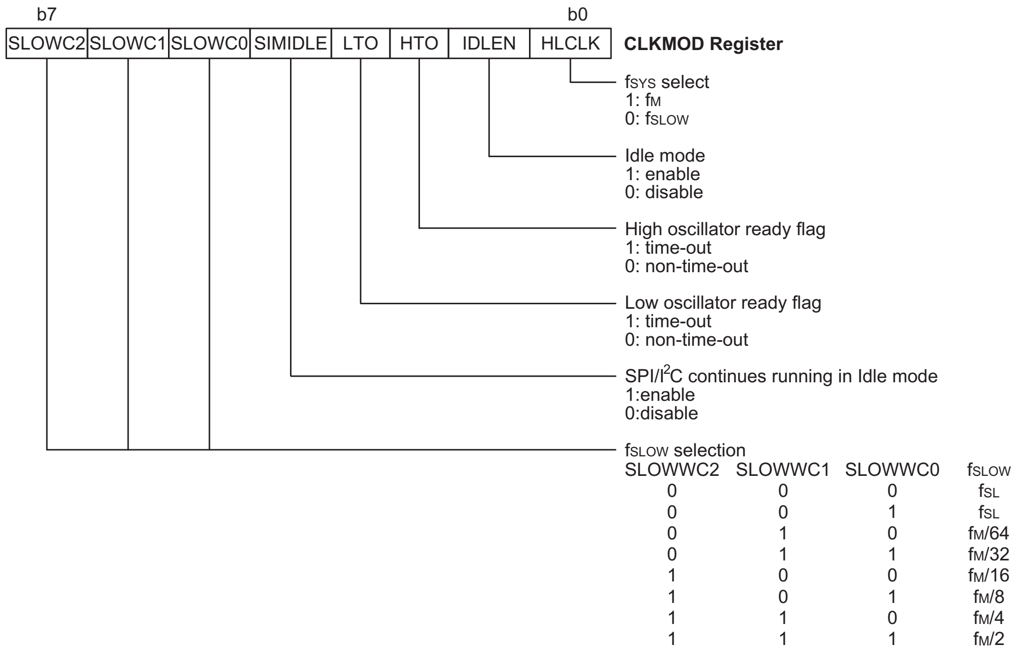
Operating Mode Clock Sources

In discussing the system clocks for the devices, they can be seen as having a dual clock mode. These dual clocks are what are known as a High Oscillator and the other as a Low Oscillator. The High and Low Oscillator are the system clock sources and can be selected dynamically using the HLCLK bit in the CLKMOD register.

The High Oscillator has the internal name f_M whose source is selected using a configuration option from a choice of either an external crystal/resonator, external RC oscillator, internal RC oscillator or external clock source.

The Low Oscillator clock source, has the internal name f_{SL}, whose source is also selected by configuration option from a choice of either an external 32768Hz oscillator or the internal 32K_INT oscillator. This internal f_{SL}, f_M clock, is further modified by the SLOWC0~SLOWC2 bits in the CLKMOD register to provide the low frequency clock source f_{SLOW}.

An additional sub internal clock, with the internal name f_{SUB}, is a 32kHz clock source which can be sourced from either the internal 32K_INT oscillator or an external 32768Hz crystal, selected by configuration option. To-



SLOWWC2	SLOWWC1	SLOWWC0	f_{SLOW}
0	0	0	f_{SL}
0	0	1	f_{SL}
0	1	0	$f_M/64$
0	1	1	$f_M/32$
1	0	0	$f_M/16$
1	0	1	$f_M/8$
1	1	0	$f_M/4$
1	1	1	$f_M/2$

Clock Control Register − CLKMOD

gether with $f_{SYS}/4$, it is used as a clock source for certain internal functions such as the Watchdog Timer, Buzzer, RTC Interrupt and Time Base Interrupt. The internal clock f_S, is simply a choice of either f_{SUB} or $f_{SYS}/4$, using a configuration option.

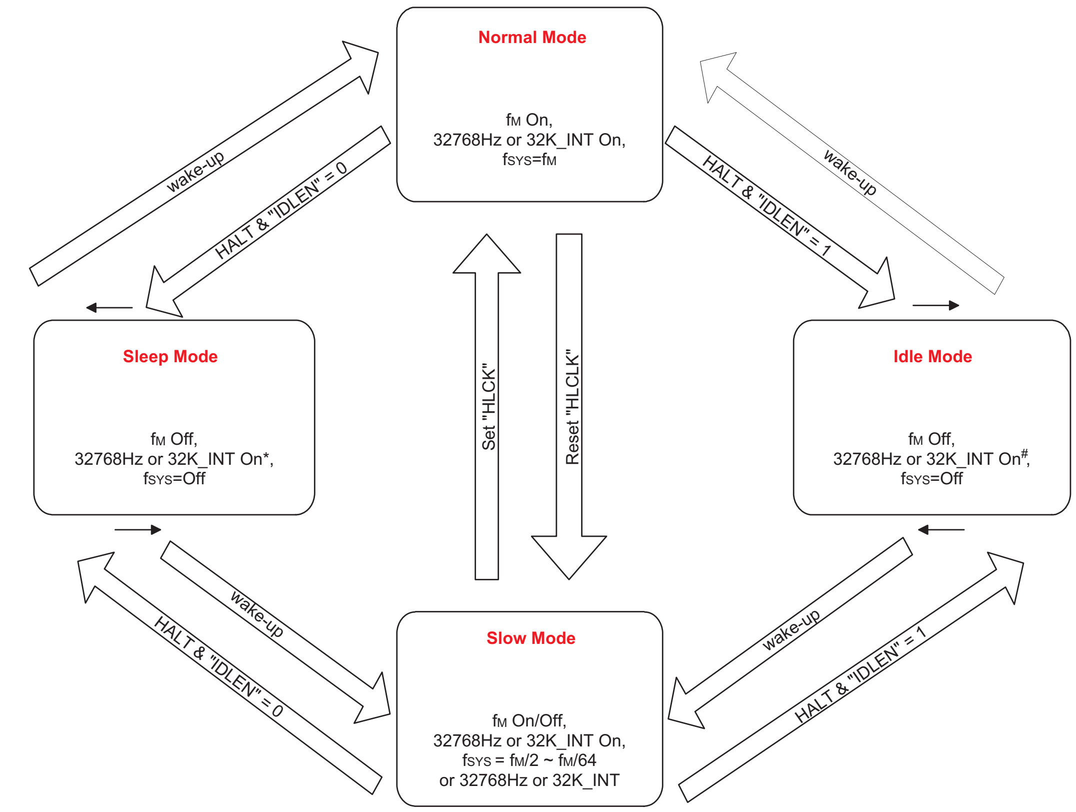
Operating Modes

After the correct clock source configuration selections are made, overall operation of the chosen clock is achieved using the CLKMOD register. A combination of the HLCLK and IDLEN bits in the CLKMOD register and use of the HALT instruction determine in which mode the device will be run. The devices can operate in the following Modes.

- **Normal mode**
  f_M on, f_{SLOW} on, $f_{SYS}=f_M$, CPU on, f_S on, f_{WDT} on/off depending upon the WDT configuration option and WDT control register.
- **Slow mode0**
  f_M off, f_{SLOW}=32K_INT oscillator or the 32768Hz oscillator, $f_{SYS}=f_{SLOW}$, CPU on, f_S on, f_{WDT} on/off depending upon the WDT configuration option and WDT control register.
- **Slow mode1**
  f_M on, $f_{SLOW}=f_M/2\sim f_M/64$, $f_{SYS}=f_{SLOW}$, CPU on, f_S on, f_{WDT} on/off depending upon the WDT configuration option and WDT control register.
- **Idle mode**
  f_M, f_{SLOW}, f_{SYS} off, CPU off; f_{SUB} on, f_S on/off by selecting f_{SUB} or $f_{SYS}/4$, f_{WDT} on/off depending upon the WDT configuration option and WDT control register.
- **Sleep mode**
  f_M, f_{SLOW}, f_{SYS}, f_S, CPU off; f_{SUB}, f_{WDT} on/off depending upon the WDT configuration option and WDT control register.

Switching Between Modes

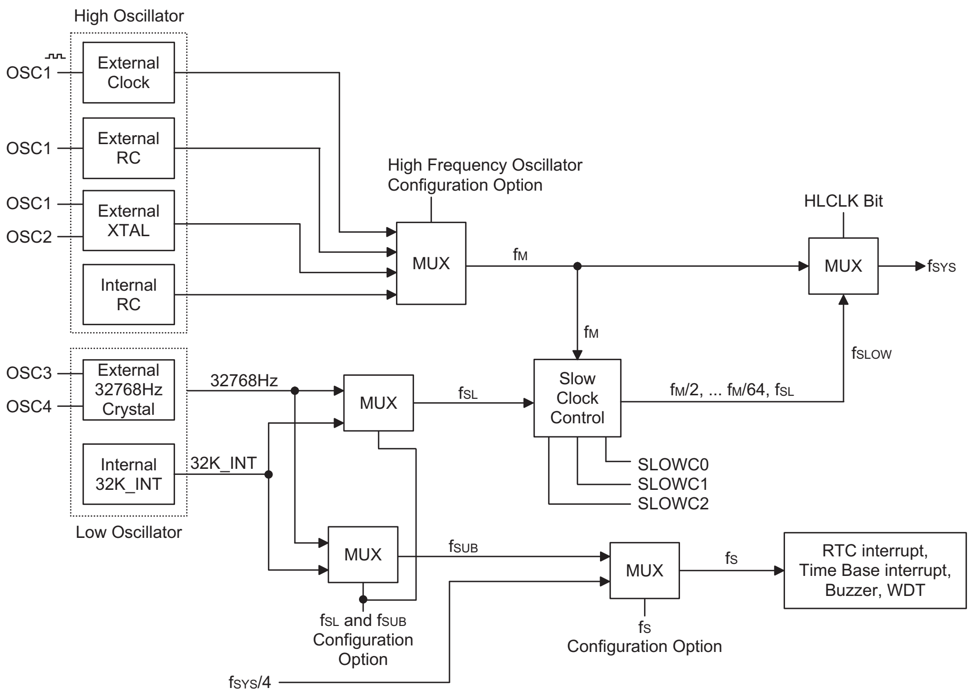
The device switches between the different operating modes using a combination of the HALT instruction and the IDLEN bit in the CLKMOD register. Switching to one of the lower power modes enables the normal operating current to be reduced to a lower operating level or to a very low standby current level, a feature which is very important in low power battery applications.



"*" Depends the WDT enable/disable condition.
if WDT is enabled, f_{SUB} = 32K_INT, then 32K_INT On

"#" Either the 32768Hz or 32K_INT must be ON.
If f_{SUB}=32K_INT, then 32K_INT is ON.

Dual Clock Mode Operation



Dual Clock Mode Structure

Power Down Mode and Wake-up

Executing the HALT Instruction

If the device is running in the Normal Mode and the HALT instruction is executed then the system clock will stop to conserve power. Depending upon the condition of the IDLEN bit in the CLKMOD register, the system will enter either the Sleep or Idle Mode. In these Modes the system clock will stop running to conserve power, however as either the 32K_INT or the external 32KHz oscillator may continue to operate, certain internal functions may remain operational.

When the HALT instruction is executed in the Normal Mode, the following will occur:

- The system oscillator will stop running.
- The system will enter either the Sleep or Idle Mode.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting if the WDT clock source is selected to come from the 32K_INT or external 32KHz oscillator. The WDT will stop if its clock source originates from the system clock.
- The I/O ports will maintain their present condition.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

Standby Current Considerations

As the main reason for entering the Sleep or Idle Mode is to keep the current consumption of the MCU to as low a value as possible, perhaps only in the order of several micro-amps, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimized. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to devices which have different package types, as there may be unbonbed pins, which must either be setup as outputs or if setup as inputs must have pull-high resistors connected. Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. Also note that additional standby current will also be required if the configuration options have enabled the Watchdog Timer internal oscillator.

Wake-up

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

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

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

Each pin on Port A can be setup, using the PAWU register, to permit a negative transition on the pin to wake-up the system. When a Port A pin wake-up occurs, the program will resume execution at the instruction following the ″HALT″ instruction.

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

Fast Wake-Up

To minimise power consumption the device can enter the SLEEP or IDLE Mode, where the clock source to the device will be stopped. However when the device is woken up again, it can take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume. To ensure the device is up and running as fast as possible a Fast Wake-Up function is provided, which allows f_{SUB}, namely either the RTC or LIRC oscillator, to act as a temporary clock to first drive the system until the original system oscillator has stabilised. As the clock source for the Fast Wake-Up function is the Watchdog Timer clock, the Watchdog Timer must be enabled for this function to operate. If the Watchdog Timer is not enabled then the Fast Start-up function cannot be used. The Fast Wake-Up enable/disable function is controlled using the configuration option.

If the Crystal oscillator is selected as the NORMAL Mode system clock, and if the Fast Wake-Up function is enabled, then it will take one to two t_{SUB} clock cycles of the LIRC or RTC oscillator for the system to wake-up. The system will then initially run under the f_{SUB} clock source until 1024 Crystal clock cycles have elapsed, at which point the HTO flag will switch high and the system will switch over to operating from the Crystal oscillator.

If the ERC or HIRC oscillators or LIRC oscillator is used as the system oscillator then it will take 1~2 clock cycles of the ERC, HIRC or LIRC to wake up the system from the SLEEP or IDLE Mode.

Note that if the Watchdog Timer is disabled, which means that the RTC and LIRC are all both off, then there will be no Fast Wake-Up function available when the device wakes-up from the SLEEP Mode.

Low Voltage Detector – LVD

The Low Voltage Detect 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.

LVD Operation

The LVD function must be first enabled via a configuration option after which bits 3 and 5 of the RTCC register are used to control the overall function of the LVD. Bit 3 is the enable/disable control bit and is known as LVDC, when set low the overall function of the LVD will be disabled. Bit 5 is the LVD detector output bit and is known as LVDO. Under normal operation, and when the power supply voltage is above the specified VLVD value in the DC characteristic section, the LVDO bit will remain at a zero value. If the power supply voltage should fall below this VLVD value then the LVDO bit will change to a high value indicating a low voltage condition. Note that the LVDO bit is a read-only bit. By polling the LVDO bit in the RTCC register, the application program can therefore determine the presence of a low voltage condition.

After power-on, or after a reset, the LVD will be switched off by clearing the LVDC bit in the RTCC register to zero. Note that if the LVD is enabled there will be some power consumption associated with its internal circuitry, however, by clearing the LVDC bit to zero the power can be minimised. It is important not to confuse the LVD with the LVR function. In the LVR function an automatic reset will be generated by the microcontroller, whereas in the LVD function only the LVDO bit will be affected with no influence on other microcontroller functions.

There are a range of voltage values, selected using a configuration option, which can be chosen to activate the LVD.

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 Watchdog Timer counter overflows.

Watchdog Timer Operation

The Watchdog Timer clock source is provided by the internal clock, f_S, which is in turn supplied by one of two sources selected by configuration option: f_{SUB} or $f_{SYS}/4$. Note that if the Watchdog Timer configuration option has been disabled, then any instruction relating to its operation will result in no operation.

Most of the Watchdog Timer options, such as enable/disable, Watchdog Timer clock source and clear instruction type are selected using configuration options. In addition to a configuration option to enable the Watchdog Timer, there are four bits, WDTEN3~ WDTEN0, in the MISC register to offer an additional enable control of the Watchdog Timer. These bits must be set to a specific value of 1010 to disable the Watchdog Timer. Any other values for these bits will keep the Watchdog Timer enabled. After power on these bits will have the disabled value of 1010.

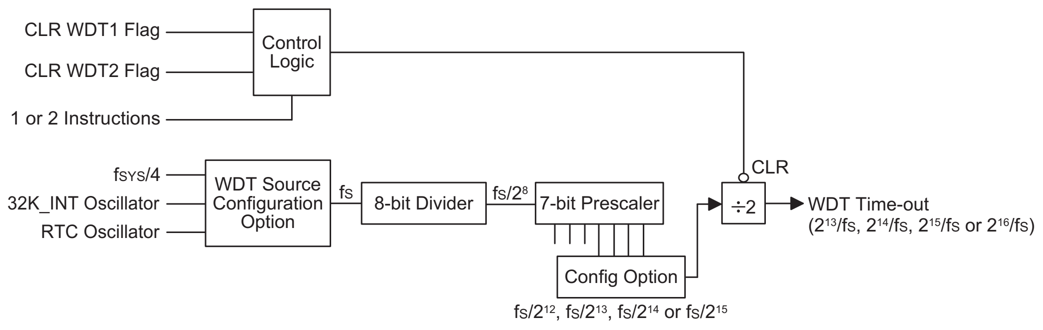
One of the WDT clock sources is the internal f_{SUB}, which can be sourced from either the 32K_INT internal oscillator or the 32768Hz oscillator. The 32K_INT internal oscillator has an approximate period of 31.2μs at a supply voltage of 5V. However, it should be noted that this specified internal clock period can vary with VDD, temperature and process variations. The 32768Hz oscillator is supplied by an external 32768Hz crystal. The other Watchdog Timer clock source option is the $f_{SYS}/4$ clock. Whether the Watchdog Timer clock source is its own internal 32K_INT, the 32768Hz oscillator or $f_{SYS}/4$, it is divided by 2^{13}~2^{16}, using configuration option to obtain the required Watchdog Timer time-out period. The max time out period is when the 2^{16} option is selected. This time-out period may vary with temperature, VDD and process variations. 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.

If the $f_{SYS}/4$ clock is used as the Watchdog Timer clock source, it should be noted that when the system enters the Power Down Mode, then the instruction clock is stopped and the Watchdog Timer will lose its protecting purposes. For systems that operate in noisy environments, using the 32K_INT RC oscillator is strongly recommended.

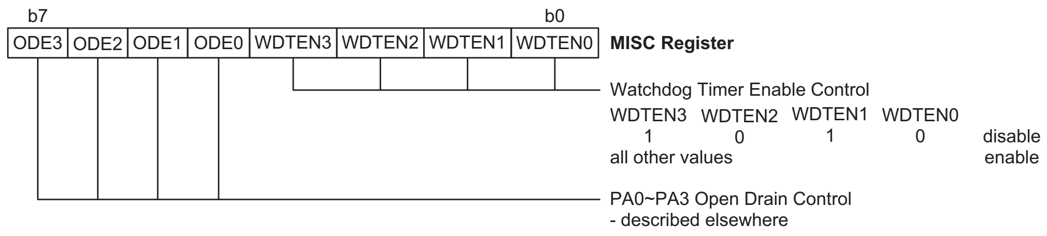
Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the Power Down Mode, when a Watchdog Timer time-out occurs, the TO bit in the status register will be set and only the Program Counter and Stack Pointer will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

Clearing the Watchdog Timer

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



Watchdog Timer



Watchdog Timer Software Control – MISC

Configuration Options

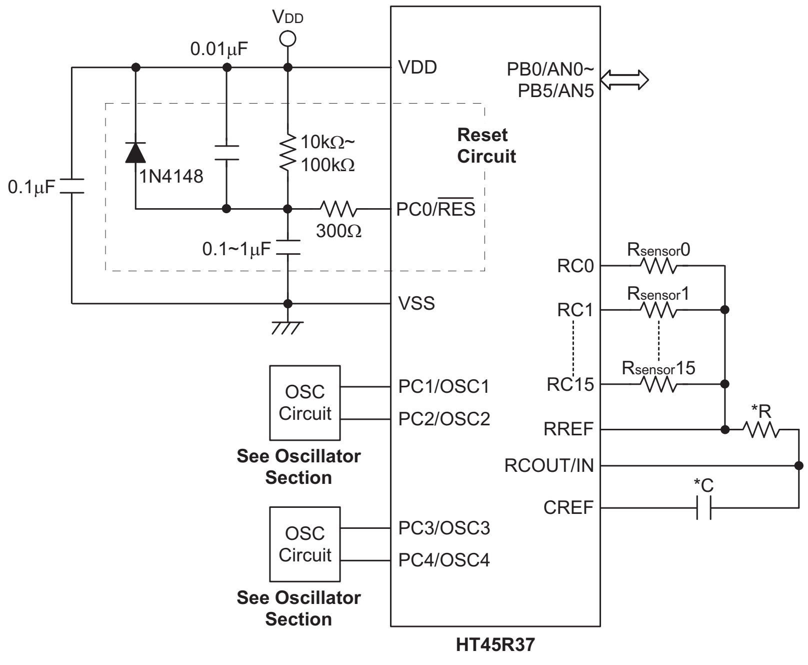
Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. As these options are programmed into the device using the hardware programming tools, once they are selected they cannot be changed later as the application software has no control over the configuration options. All options must be defined for proper system function, the details of which are shown in the table.

No.	Options
Oscillator Options	
1	High Frequency Oscillator type selection 1. External Crystal Oscillator 2. External RC Oscillator 3. Internal RC Oscillator 4. Externally supplied clock
2	f_{SL} and f_{SUB} clock selection (Low frequency oscillator type selection) 1. 32768Hz External Oscillator 2. 32K_INT Internal Oscillator
3	Internal RC Oscillator frequency select: 4MHz, 8MHz or 12MHz
4	f_S clock selection: f_{SUB} or $f_{SYS}/4$
5	XTAL mode selection: 455kHz or 1M~12MHz
6	Fast wake-up from HALT mode (Only for external crystal oscillator): enable/disable
Reset Option	
7	Rest pin Function: PC0 or Reset Pin
PFD Options	
8	PA3: normal I/O or PFD output
9	PFD clock selection: Timer/Event Counter 0 or Timer/Event Counter 1
Buzzer Options	
10	PA0/PA1: normal I/O or BZ/$\overline{BZ}$ or PA0=BZ and PA1 as normal I/O
11	Buzzer frequency: $f_S/2^2$, $f_S/2^3$, $f_S/2^4$, $f_S/2^5$, $f_S/2^6$, $f_S/2^7$, $f_S/2^8$, $f_S/2^9$
Time Base Option	
12	Time base time-out period: $2^{12}/f_S$, $2^{13}/f_S$, $2^{14}/f_S$, $2^{15}/f_S$
Watchdog Options	
13	Watchdog Timer function: enable or disable
14	CLRWDT instructions: 1 or 2 instructions
15	WDT time-out period: $2^{12}/f_S$~$2^{13}/f_S$, $2^{13}/f_S$~$2^{14}/f_S$, $2^{14}/f_S$~$2^{15}/f_S$, $2^{15}/f_S$~$2^{16}/f_S$
LVD/LVR Options	
16	LVD function: enable or disable
17	LVR function: enable or disable
18	LVR/LVD voltage: 2.1V/2.2V or 3.15V/3.3V or 4.2V/4.4V
SPI Options	
19	SIM interface enable/disable
20	SPI_WCOL: enable/disable
21	SPI_CSEN: enable/disable, used to enable/disable (1/0) software CSEN function
I^{2}C Option	
22	I^{2}C debounce Time: no debounce, 1 system clock debounce, 2 system clock debounce

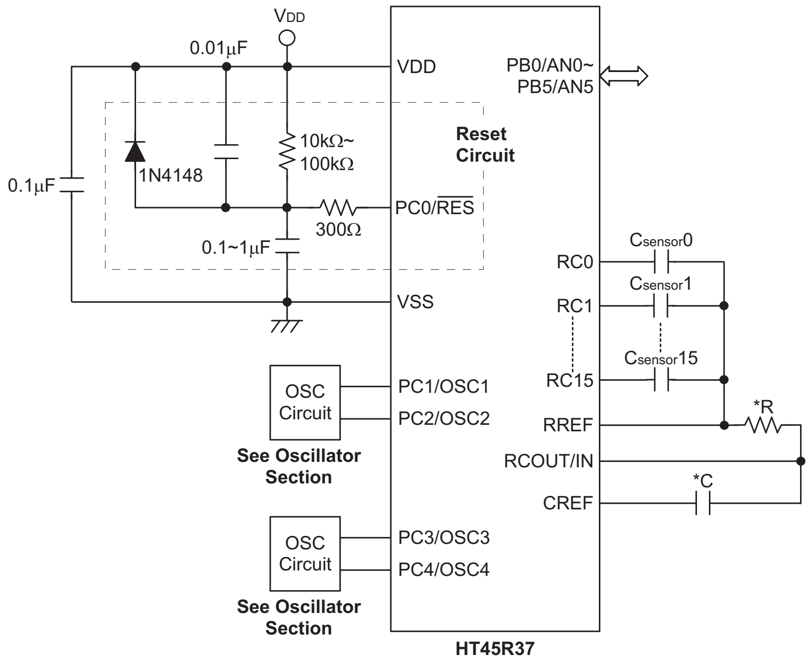
No.	Options
Timer/Event Counter and External Interrupt Pins Filter Option	
23	Interrupt and Timer/Event Counter input pins internal filter On/Off control – applies to all pins
C/R to F Converter Option	
24	I/O pins or C/R to F converter inputs

Application Circuits

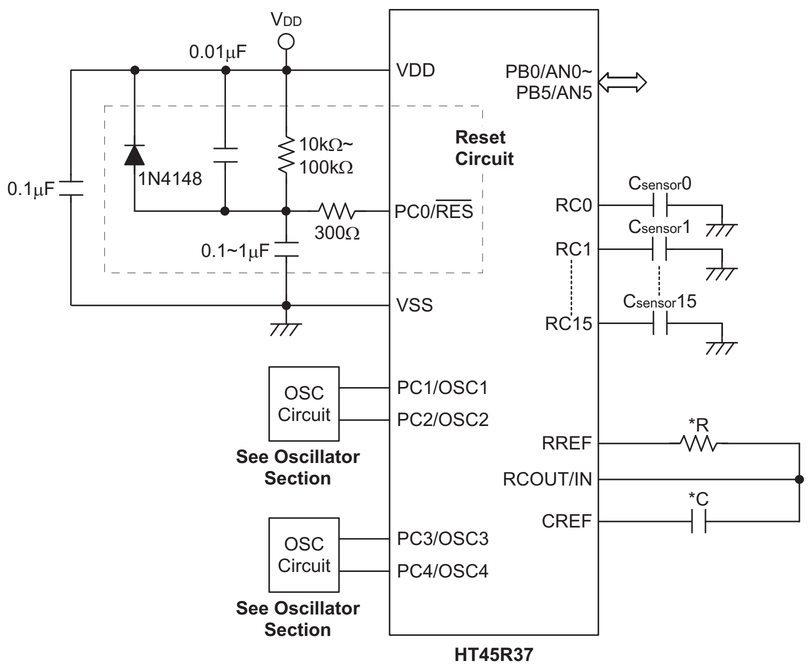
C/R to F Application Circuit 1



C/R to F Application Circuit 2



C/R to F Application Circuit 3



Note: 1. The ″*R″ resistance and ″*C″ capacitance should be consideration for the frequency of RC OSC.

2. $R_{sensor}0$ ~ $R_{sensor}15$ are the resistance sensors.

3. $C_{sensor}0$ ~ $C_{sensor}15$ are the capacitance sensors.

Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table conventions:

x: Bits immediate data
m: Data Memory address
A: Accumulator
i: 0~7 number of bits
addr: Program memory address

Mnemonic	Description	Cycles	Flag Affected
Arithmetic			
ADD A,[m]	Add Data Memory to ACC	1	Z, C, AC, OV
ADDM A,[m]	Add ACC to Data Memory	1Note	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	1Note	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	1Note	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	1Note	Z, C, AC, OV
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	1Note	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	1Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	1Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	1Note	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	1Note	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	1Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	1Note	Z

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

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

Instruction Definition

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

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None
CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None
CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None
CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.

Operation	WDT cleared
	TO ← 0
	PDF ← 0

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

Operation	WDT cleared
	TO ← 0
	PDF ← 0

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

Operation	WDT cleared
	TO ← 0
	PDF ← 0

Affected flag(s)	TO, PDF

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

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

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

DEC [m]	Decrement Data Memory
Description	Data in the specified Data Memory is decremented by 1.
Operation	[m] ← [m] − 1
Affected flag(s)	Z

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

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

Operation	TO ← 0
	PDF ← 1

Affected flag(s)	TO, PDF

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

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

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

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

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

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

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

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

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

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

Affected flag(s)	None

RLC [m]	Rotate Data Memory left through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0.
Operation	[m].(i+1) ← [m].i; (i = 0~6) [m].0 ← C C ← [m].7
Affected flag(s)	C
RLCA [m]	Rotate Data Memory left through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s)	C
RR [m]	Rotate Data Memory right
Description	The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i = 0~6) [m].7 ← [m].0
Affected flag(s)	None
RRA [m]	Rotate Data Memory right with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s)	None
RRC [m]	Rotate Data Memory right through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.
Operation	[m].i ← [m].(i+1); (i = 0~6) [m].7 ← C C ← [m].0
Affected flag(s)	C
RRCA [m]	Rotate Data Memory right through Carry with result in ACC
Description	Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.

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

Affected flag(s)	C

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

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

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

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

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

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

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

SWAP [m]	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	[m].3~[m].0 ↔ [m].7 ~ [m].4
Affected flag(s)	None

SWAPA [m]	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC.3 ~ ACC.0 ← [m].7 ~ [m].4 ACC.7 ~ ACC.4 ← [m].3 ~ [m].0
Affected flag(s)	None

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

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

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

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

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

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

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

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

Package Information

20-pin DIP (300mil) Outline Dimensions

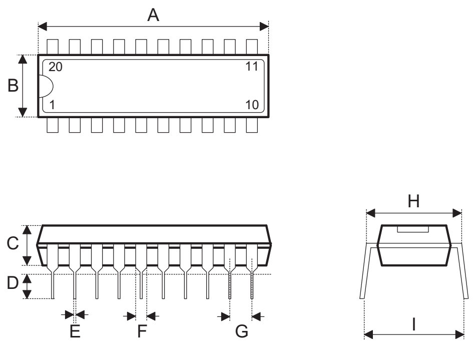


Fig1. Full Lead Packages

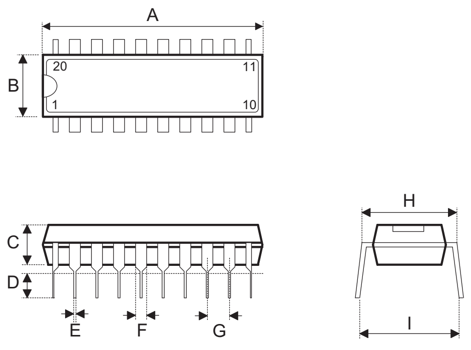


Fig2. 1/2 Lead Packages

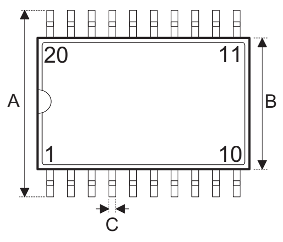
• MS-001d (see fig1)

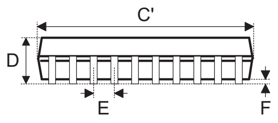
Symbol	Dimensions in mil		
	Min.	**Nom.**	**Max.**
A	980	—	1060
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

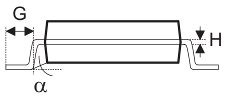
• MO-095a (see fig2)

Symbol	Dimensions in mil		
	Min.	**Nom.**	**Max.**
A	945	—	985
B	275	—	295
C	120	—	150
D	110	—	150
E	14	—	22
F	45	—	60
G	—	100	—
H	300	—	325
I	—	—	430

20-pin SOP (300mil) Outline Dimensions







• MS-013

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	393	—	419
B	256	—	300
C	12	—	20
C′	496	—	512
D	—	—	104
E	—	50	—
F	4	—	12
G	16	—	50
H	8	—	13
α	0°	—	8°

24-pin SKDIP (300mil) Outline Dimensions

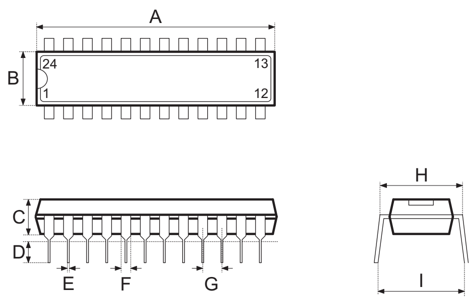


Fig1. Full Lead Packages

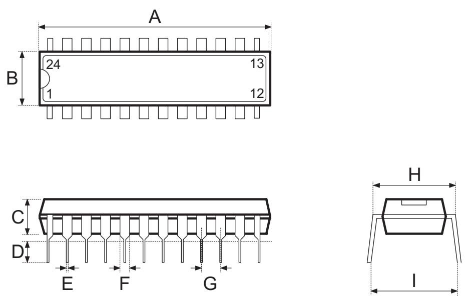


Fig2. 1/2 Lead Packages

- MS-001d (see fig1)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1230	—	1280
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

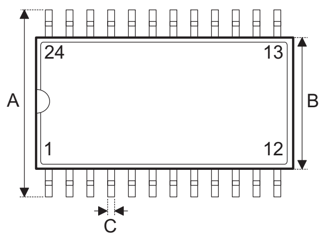
- MS-001d (see fig2)

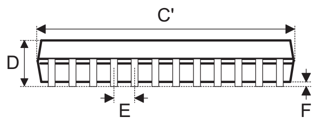
Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1160	—	1195
B	240	—	280
C	115	—	195
D	115	—	150
E	14	—	22
F	45	—	70
G	—	100	—
H	300	—	325
I	—	—	430

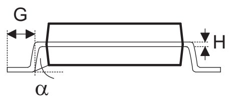
• MO-095a (see fig2)

Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1145	—	1185
B	275	—	295
C	120	—	150
D	110	—	150
E	14	—	22
F	45	—	60
G	—	100	—
H	300	—	325
I	—	—	430

24-pin SOP (300mil) Outline Dimensions



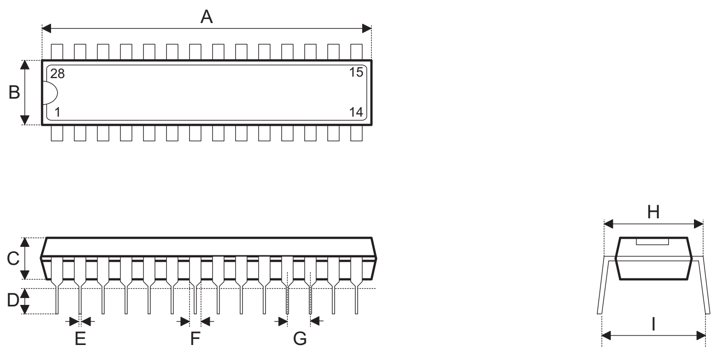




• MS-013

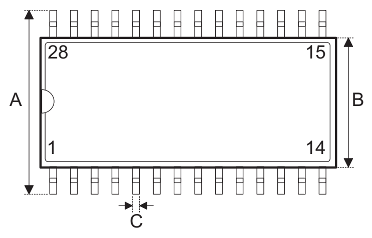
Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	393	—	419
B	256	—	300
C	12	—	20
C′	598	—	613
D	—	—	104
E	—	50	—
F	4	—	12
G	16	—	50
H	8	—	13
α	0°	—	8°

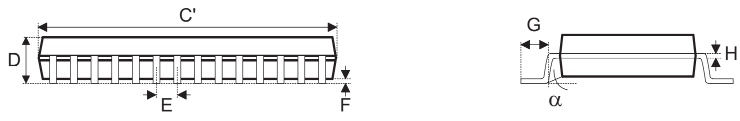
28-pin SKDIP (300mil) Outline Dimensions



Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	1375	—	1395
B	278	—	298
C	125	—	135
D	125	—	145
E	16	—	20
F	50	—	70
G	—	100	—
H	295	—	315
I	—	—	375

28-pin SOP (300mil) Outline Dimensions

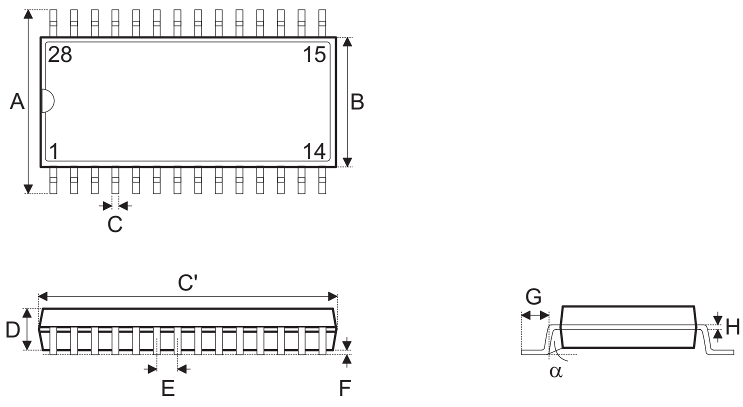




• MS-013

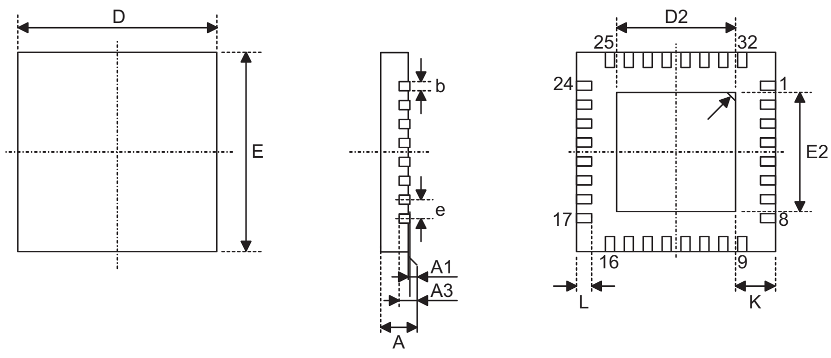
Symbol	Dimensions in mil		
	Min.	Nom.	Max.
A	393	—	419
B	256	—	300
C	12	—	20
C′	697	—	713
D	—	—	104
E	—	50	—
F	4	—	12
G	16	—	50
H	8	—	13
α	0°	—	8°

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°

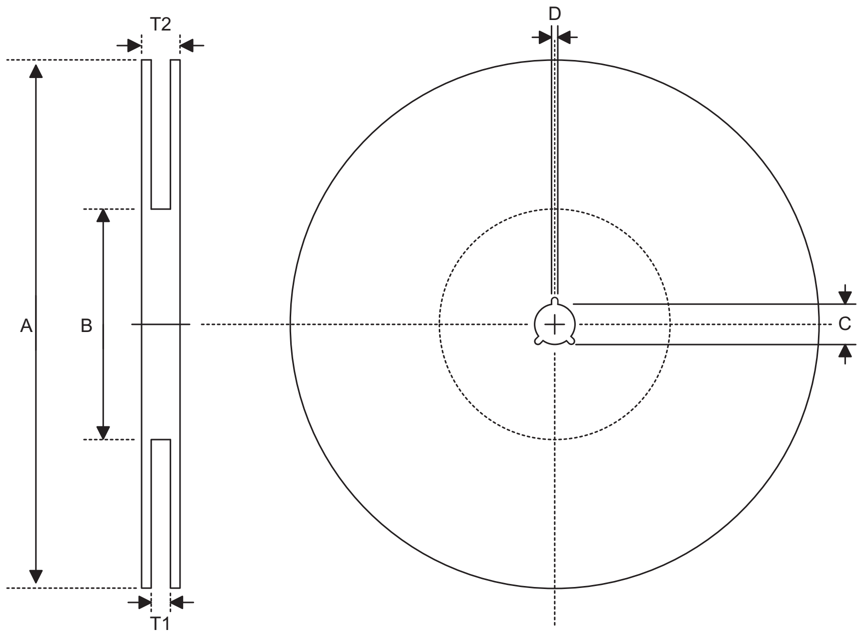
SAW Type 32-pin (5mm×5mm) QFN Outline Dimensions



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	—	—	—

Product Tape and Reel Specifications

Reel Dimensions



SOP 20W

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	$24.8^{+0.3/-0.2}$
T2	Reel Thickness	30.2±0.2

SOP 24W

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	$24.8^{+0.3/-0.2}$
T2	Reel Thickness	30.2±0.2

SOP 28W (300mil)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0$\pm$1.0
B	Reel Inner Diameter	100.0$\pm$1.5
C	Spindle Hole Diameter	$13.0^{+0.5/-0.2}$
D	Key Slit Width	2.0$\pm$0.5
T1	Space Between Flange	$24.8^{+0.3/-0.2}$
T2	Reel Thickness	30.2$\pm$0.2

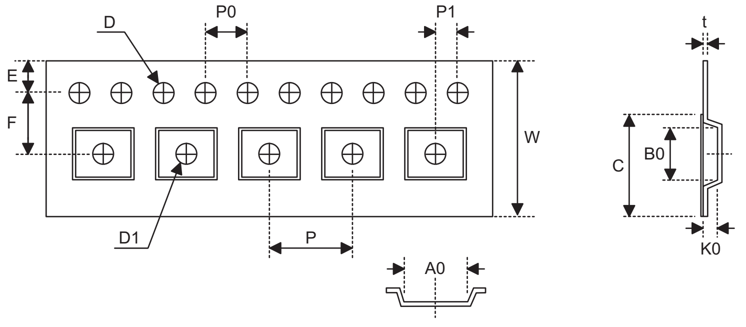
SSOP 28S (150mil)

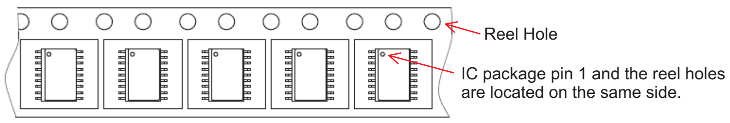
Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0$\pm$1.0
B	Reel Inner Diameter	100.0$\pm$1.5
C	Spindle Hole Diameter	$13.0^{+0.5/-0.2}$
D	Key Slit Width	2.0$\pm$0.5
T1	Space Between Flange	$16.8^{+0.3/-0.2}$
T2	Reel Thickness	22.2$\pm$0.2

SAW Type QFN 32-pin (5mm$\times$5mm)

Symbol	Description	Dimensions in mm
A	Reel Outer Diameter	330.0$\pm$1.0
B	Reel Inner Diameter	100.0$\pm$0.1
C	Spindle Hole Diameter	$13.0^{+0.5/-0.2}$
D	Key Slit Width	2.0$\pm$0.5
T1	Space Between Flange	$12.5^{+0.3/-0.2}$
T2	Reel Thickness	—

Carrier Tape Dimensions




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

SOP 20W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	$24.0^{+0.3/-0.1}$
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	$1.5^{+0.1/-0.0}$
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.8±0.1
B0	Cavity Width	13.3±0.1
K0	Cavity Depth	3.2±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	21.3±0.1

SOP 24W

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0±0.3
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.1
F	Cavity to Perforation (Width Direction)	11.5±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	10.9±0.1
B0	Cavity Width	15.9±0.1
K0	Cavity Depth	3.1±0.1
t	Carrier Tape Thickness	0.35±0.05
C	Cover Tape Width	21.3±0.1

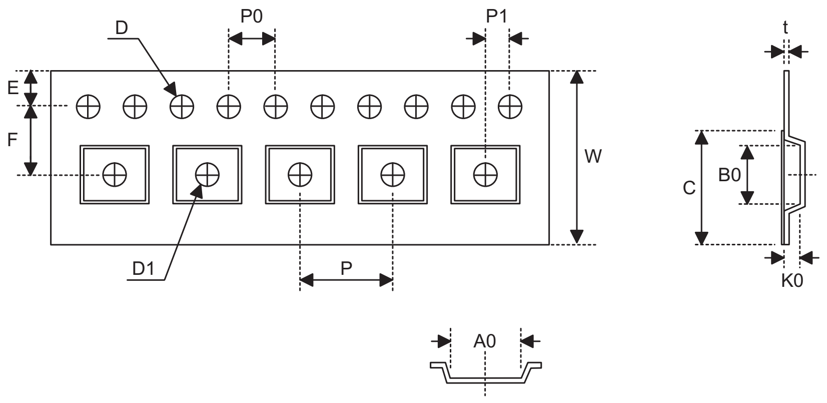
SOP 28W (300mil)

Symbol	Description	Dimensions in mm
W	Carrier Tape Width	24.0±0.3
P	Cavity Pitch	12.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	11.5±0.1
D	Perforation Diameter	$1.5^{+0.1/-0.0}$
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.0±0.1
A0	Cavity Length	10.85±0.10
B0	Cavity Width	18.34±0.10
K0	Cavity Depth	2.97±0.10
t	Carrier Tape Thickness	0.35±0.01
C	Cover Tape Width	21.3±0.1

SSOP 28S (150mil)

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

SAW Type QFN 32-pin (5mm×5mm)



Symbol	Description	Dimensions in mm
W	Carrier Tape Width	12.0±0.3
P	Cavity Pitch	8.0±0.1
E	Perforation Position	1.75±0.10
F	Cavity to Perforation (Width Direction)	5.50±0.05
D	Perforation Diameter	$1.5^{+0.1/-0.0}$
D1	Cavity Hole Diameter	$1.50^{+0.25/-0.00}$
P0	Perforation Pitch	4.0±0.1
P1	Cavity to Perforation (Length Direction)	2.00±0.05
A0	Cavity Length	5.25±0.10
B0	Cavity Width	5.25±0.10
K0	Cavity Depth	1.1±0.1
t	Carrier Tape Thickness	0.30±0.05
C	Cover Tape Width	9.3±0.1

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

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

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

Holtek Semiconductor Inc. (Shenzhen Sales Office)
5F, Unit A, Productivity Building, 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 Inc. (Beijing Sales Office)
Suite 1721, Jinyu Tower, A129 West Xuan Wu Men Street, Xicheng District, Beijing, China 100031
Tel: 86-10-6641-0030, 86-10-6641-7751, 86-10-6641-7752
Fax: 86-10-6641-0125

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