Brushless DC Motor Flash Type 8-Bit MCU

Technical Document

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

Features

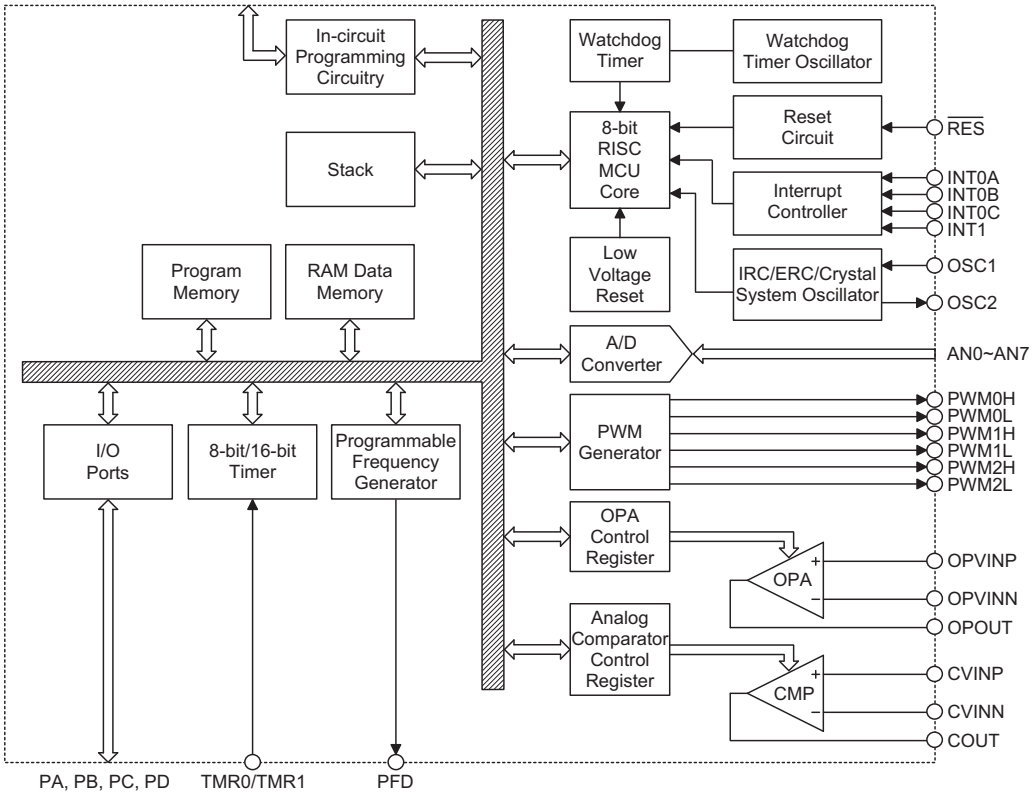
- Operating voltage:
  f_{SYS}= 0.4~20MHz: 4.5V~5.5V
- 26 bidirectional I/O lines
- External interrupt inputs shared with 4 I/O lines
- 8-bit programmable Timer/Event Counter with overflow interrupt and 7-stage prescaler
- 16-bit programmable Timer/Event Counter with overflow interrupt and 7-stage prescaler
- 4096×15 Flash Program Memory
- 192×8 Data Memory RAM
- On-chip crystal, internal RC and external RC oscillator
- Fully integrated internal RC oscillator with three fixed frequencies: 12MHz, 16MHz or 20MHz
- Watchdog Timer function
- PFD for audio frequency generation
- Power down and wake-up functions to reduce power consumption
- Up to 0.2μs instruction cycle with 20MHz system clock at V_{DD}=5V
- 8-level subroutine nesting
- 8 channel 12-bit resolution A/D converter
- 3 pairs of 10-bit PWM with complementary outputs shared with six I/O lines and with 3 PWM duty control registers
- Bit manipulation instruction
- Table read instructions
- 63 powerful instructions
- All instructions in one or two machine cycles
- Low voltage reset function
- Low voltage detect function
- Integrated operational Amplifier
- Integrated Analog Comparator with interrupt function
- 28-pin SOP package

General Description

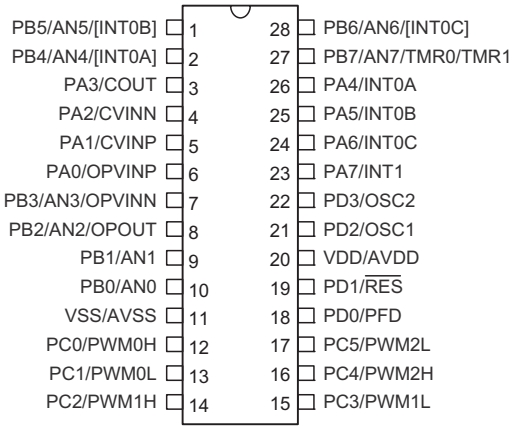
The HT45FM03B is 8-bit, high performance, RISC architecture microcontroller device which includes a host of fully integrated special features specifically designed for brushless DC motor applications.

The advantages of low power consumption, I/O flexibility, programmable frequency divider, timer functions, oscillator options, multi-channel A/D Converter, Pulse Width Modulation function, power-down and wake-up functions, although especially directed at brushless DC motor applications, the enhanced versatility of this device also makes it applicable for use in a wide range of A/D application possibilities such as sensor signal processing, motor driving, industrial control, consumer products, subsystem controllers, etc.

Block Diagram



Pin Assignment



HT45FM03B
28 SOP-A

Pad Description

Pad Name	I/O	Option	Description
PA0/OPVINP PA1/CVINP PA2/CVINN PA3/COUT PA4/INT0A PA5/INT0B PA6/INT0C PA7/INT1	I/O	Pull-high Wake-up INT0A INT0B INT0C	Bidirectional 8-bit input/output port. Each pin can be configured as wake-up input by configuration option. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine if the pins have pull-high resistors. Pins PA4~PA6 are pin-shared with INT0A, INT0B and INT0C, the function being selected via configuration options. PA7 is pin-shared with the external interrupt pin INT1. PA1, PA2 and PA3 are pin-shared with comparator pins CVINP, CVINN and COUT. PA0 is shared with OPVINP.
PB0/AN0 PB1/AN1 PB2/AN2/OPOUT PB3/AN3/OPVINN PB4/AN4/INT0A PB5/AN5/INT0B PB6/AN6/INT0C PB7/AN7/TMR0/TMR1	I/O	Pull-high INT0A INT0B INT0C	Bidirectional 8-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine if the pins have pull-high resistors. PB is pin-shared with the A/D inputs. Pins PB4~PB6 are also pin-shared with INT0A, INT0B and INT0C, the function being selected via configuration options. Pins PB2 and PB3 are also pin pin-shared with operational amplifier pins OPOUT and OPVINN. Pin PB7 is also pin-shared with timer input pins TMR0 and TMR1.
PC0/PWM0H PC1/PWM0L PC2/PWM1H PC3/PWM1L PC4/PWM2H PC5/PWM2L	I/O	Pull-high	Bidirectional 6-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Configuration options determine if the pins have pull-high resistors. PC is pin shared with the Pulse Width Modulation complimentary output pairs, PWM0H~PWM2H and PWM0L~PWM2L.
PD0/PFD	I/O	Pull-high PFD	Bidirectional 1-line I/O. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. A configuration option determines if pin PD0 has a pull-high resistor. Pin PD0 is shared with the PFD output.
PD1/$\overline{\text{RES}}$	I/O	PD1 or $\overline{\text{RES}}$	Bidirectional 1-bit input/output port. Software instructions determine if the pin is a CMOS output or Schmitt Trigger input. Pin PD1 does not have a pull-high option. Pin PD1 is pin-shared with the reset input pin $\overline{\text{RES}}$. $\overline{\text{RES}}$ is a Schmitt Trigger reset input. Active low.
PD2/OSC1 PD3/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. Pin PD2~PD3 do not have pull-high options. 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+PD3: PD2 is configured as OSC1 pin, PD3 configured as an I/O. If the internal RC OSC is selected, the frequency will be fixed at either 12MHz, 16MHz or 20MHz, dependent upon which configuration option is chosen.
VSS	—	—	Negative power supply, ground
AVSS	—	—	Ground connection for A/D converter. The VSS and AVSS are the same pin at 28 pin package.
VDD	—	—	Positive power supply
AVDD	—	—	Power supply connection for A/D converter. The VDD and AVDD are the same pin at 28 pin package.

Absolute Maximum Ratings

Supply Voltage ..... V_{SS}–0.3V to V_{SS}+6.0V
Input Voltage ..... V_{SS}–0.3V to V_{DD}+0.3V
I_{OL} Total 150mA
Total Power Dissipation 500mW

Storage Temperature ..... –50°C to 125°C
Operating Temperature ..... –40°C to 125°C
I_{OH} Total ..... –100mA

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

D.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
V_{DD}	Operating Voltage	—	f_{SYS}=0.4~20MHz	4.5	—	5.5	V
I_{DD1}	Operating Current (Crystal OSC, ERC OSC, IRC OSC)	5V	No load, f_{SYS}=12MHz ADC disable	—	3.5	5.5	mA
I_{DD2}	Operating Current (Crystal OSC, ERC OSC, IRC OSC)	5V	No load, f_{SYS}=16MHz ADC disable	—	4.5	7.0	mA
I_{DD3}	Operating Current (Crystal OSC, ERC OSC, IRC OSC)	5V	No load, f_{SYS}=20MHz ADC disable	—	5.5	8.5	mA
I_{STB1}	Standby Current (WDT Enabled)	5V	No load, system HALT	—	—	10	μA
I_{STB2}	Standby Current (WDT Disabled)	5V	No load, system HALT	—	—	2	μA
V_{IL1}	Input Low Voltage for I/O Ports, TMR0, TMR1, INT0A, INT0B, INT0C and INT1	—	—	0	—	$0.3V_{DD}$	V
V_{IH1}	Input High Voltage for I/O Ports, TMR0, TMR1, INT0A, INT0B, INT0C and INT1	—	—	$0.7V_{DD}$	—	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	—	LVR enable, 4.2V option	–5%	4.2	+5%	V
V_{LVD}	Low Voltage Detector Voltage	—	LVDEN = 1, V_{LVD} = 4.4V	–5%	4.4	+5%	V
I_{OL1}	I/O Port Sink Current	5V	V_{OL}=0.1V_{DD}	10	20	—	mA
I_{OH1}	I/O Port Source Current	5V	V_{OH}=0.9V_{DD}	–5	–10	—	mA
R_{PH}	Pull-high Resistance	5V	—	10	30	50	kΩ

A.C. Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
f_{SYS}	System Clock	—	4.5V~5.5V	400	—	20000	kHz
f_{HIRC}	System Clock (HIRC)	5V	Ta=25°C	–2%	12	+2%	MHz
		5V	Ta=25°C	–2%	16	+2%	MHz
		5V	Ta=25°C	–2%	20	+2%	MHz
		4.5V~5.5V	Ta= –20°C~125°C	–8%	12	+4%	MHz
			Ta= –20°C~125°C	–7%	16	+5%	MHz
			Ta= –20°C~125°C	–9%	20	+4%	MHz
			Ta= –40°C~125°C	–9%	12	+4%	MHz
			Ta= –40°C~125°C	–8%	16	+5%	MHz
			Ta= –40°C~125°C	–12%	20	+4%	MHz
f_{ERC}	System Clock (ERC)	5V	Ta=25°C, R=120kΩ *	–2%	12	+2%	MHz
		4.5V~5.5V	Ta= –40°C~125°C, R=120kΩ *	–5%	12	+4%	MHz
f_{TIMER}	Timer I/P Frequency (TMR0/TMR1)	—	—	0	—	4000	kHz
t_{WDTOSC}	Watchdog Oscillator Period	5V	—	32	65	130	μs
t_{RES}	External Reset Low Pulse Width	—	—	1	—	—	μs
t_{SST}	System Start-up Timer Period	—	Wake-up from HALT	—	1024	—	t_{SYS}
t_{INT}	Interrupt Pulse Width	—	—	1	—	—	μs
t_{LVR}	Low Voltage Width to Reset	—	—	0.25	1	2	ms

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

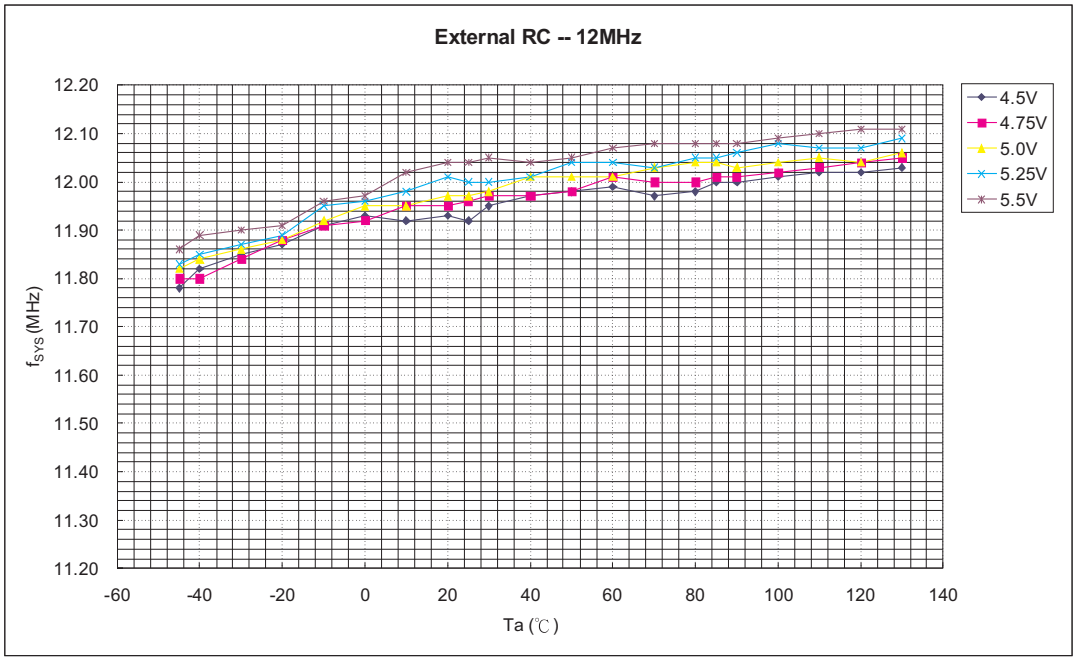
2. * For f_{ERC}, as the resistor tolerance will influence the frequency a precision resistor is recommended.

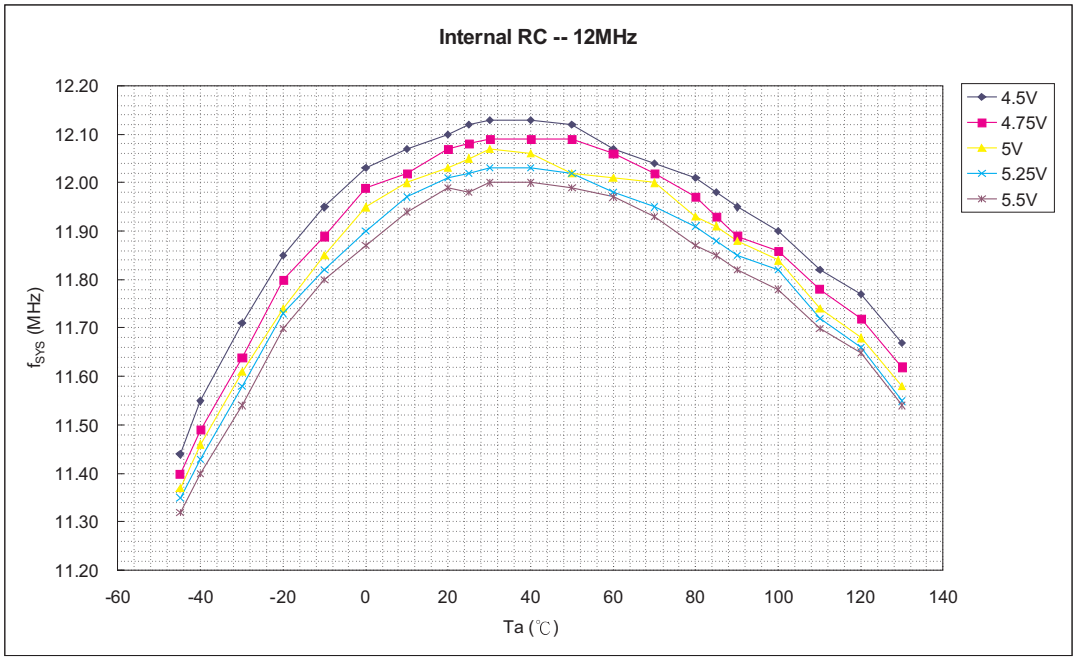
3. To maintain the accuracy of the internal HIRC oscillator frequency, a 0.1μF decoupling capacitor should be connected between VDD and VSS and located as close to the device as possible.

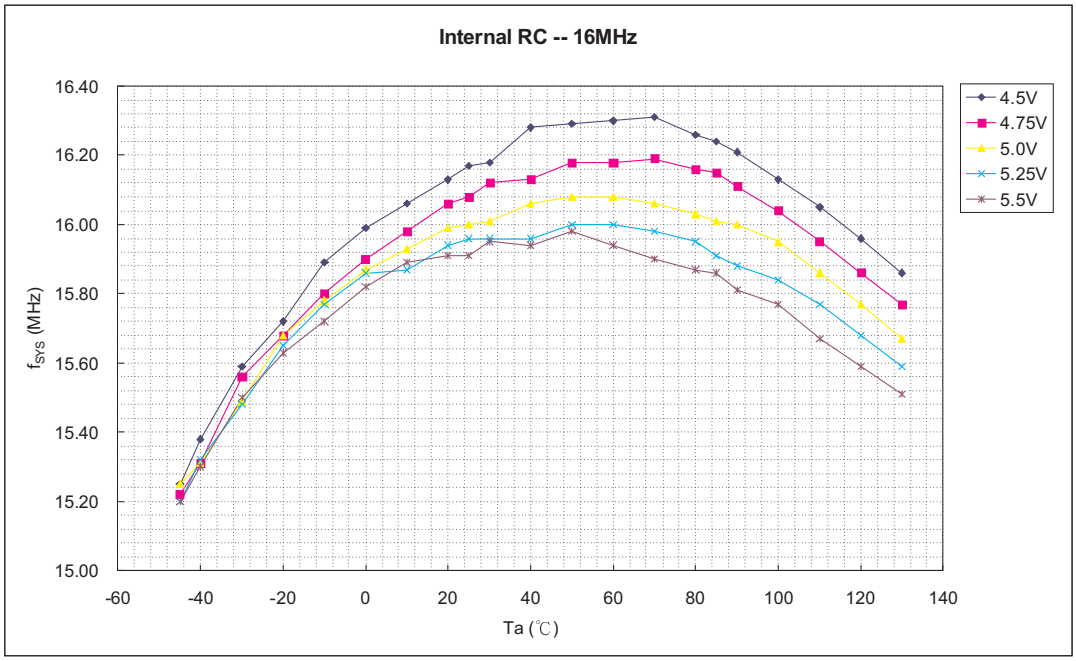
Oscillator Temperature/Frequency Characteristics

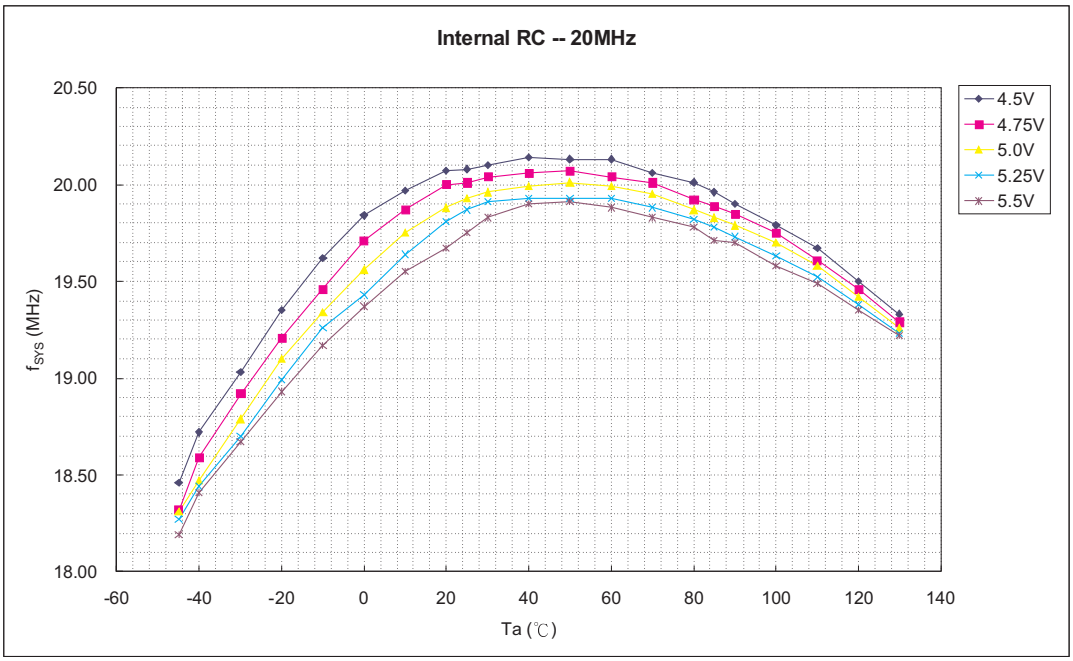
The following characteristic graphics depicts typical oscillator behavior. The data presented here is a statistical summary of data gathered on units from different lots over a period of time. This is for information only and the figures were not tested during manufacturing.

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






Internal RC -- 16MHz
16.40
16.20
16.00
15.80
15.60
15.40
15.20
15.00
fSYS (MHz)
-60
-40
-20
0
20
40
60
80
100
120
140
Ta (℃)
4.5V
4.75V
5.0V
5.25V
5.5V


Internal RC -- 20MHz
20.50
20.00
19.50
19.00
18.50
18.00
fSYS (MHz)
-60
-40
-20
0
20
40
60
80
100
120
140
Ta (℃)
4.5V
4.75V
5.0V
5.25V
5.5V

A/D Converter Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
AVDD	A/D Converter Operating Voltage	—	—	4.5	—	VDD	V
VAD	AD Input Voltage	—	—	0	—	VREF	V
VREF	A/D Converter Input Reference Voltage Range	—	—	—	AVDD	—	V
DNL	Differential Non-linearity	—	t_{AD}= 0.5μs	—	±1	±2	LSB
INL	Integral Non-linearity	—	t_{AD}= 0.5μs	—	±2	±4	LSB
I_{ADC}	Additional Power Consumption if A/D Converter is Used	5V	No load, t_{AD}= 0.5μs	—	1.5	3.0	mA
t_{AD}	A/D Converter Clock Period	—	—	0.5	—	100	μs
t_{ADC}	A/D Conversion Time (Note)	—	12 bit ADC	—	16	—	t_{AD}

Note: ADC conversion time (t_{ADC}) is include ADC sample time $4t_{AD}$.

OP Amplifier Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
D.C. Electrical Characteristic							
V_{DD}	Operating Voltage	—	—	4.5	—	5.5	V
V_{OS}	Input Offset Voltage	5V	By calibration	–5	—	5	mV
V_{CM}	Common Mode Voltage Range	—	—	V_{SS}	—	V_{DD}–1.4	V
PSRR	Power Supply Rejection Ratio	—	—	60	—	—	dB
CMRR	Common Mode Rejection Ratio	—	V_{DD}=5V V_{CM}=0~V_{DD}–1.4V	60	—	—	dB
A.C. Electrical Characteristic							
A_{OL}	Open Loop Gain	—	—	60	80	—	dB
SR	Slew Rate+, Rate–	—	No load	—	1	—	V/μs
GBW	Gain Band Width	—	RL=1MΩ, CL=100pF	—	—	100	kHz

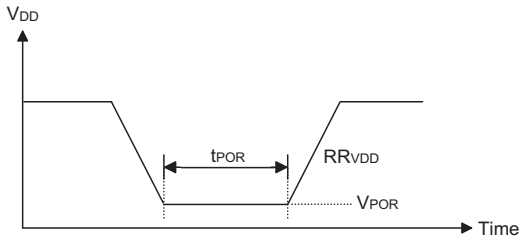
Analog Comparator Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V_{DD}	Conditions				
VDD	Analog Comparator Operating Voltage	—	—	4.5	—	5.5	V
VOS	Analog Comparator Input Offset Voltage	5V	By calibration	–5	—	5	mV
VCM	Analog Comparator Common Mode Voltage Range	—	—	0	—	V_{DD}–1.4	V
t_{PD}	Analog Comparator Response Time	—	—	—	—	2	μs
VHYS	Analog Comparator Hysteresis Width	5V	Analog Comparator Hysteresis enable	—	40	—	mV

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



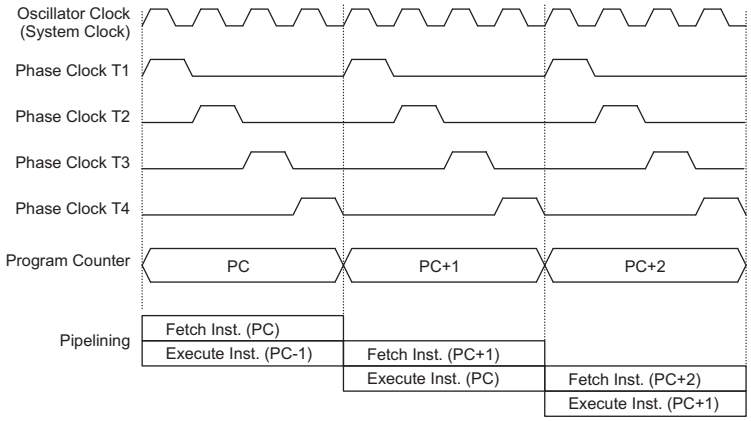
System Architecture

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.

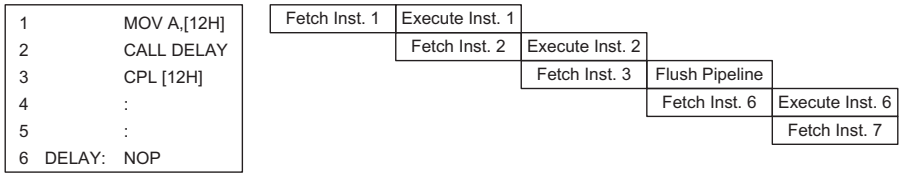
Clocking and Pipelining

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



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 by user.

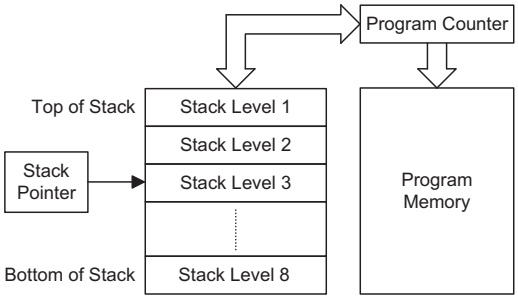
When executing instructions requiring jumps to non-consecutive addresses such as a jump instruction, a subroutine call, interrupt or reset, etc., the microcontroller manages program control by loading the required address into the Program Counter. For conditional skip instructions, once the condition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

The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and 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.

Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack has 8 levels and is neither part of the data nor part of the program space, and can neither be read from nor writeable. The activated level is indexed by the Stack Pointer, SP, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.



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

Mode	Program Counter Bits											
	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
Analog Comparator Interrupt	0	0	0	0	0	0	0	0	0	1	0	0
External Interrupt 0	0	0	0	0	0	0	0	0	1	0	0	0
Multi-function Interrupt	0	0	0	0	0	0	0	0	1	1	0	0
PWM Interrupt	0	0	0	0	0	0	0	1	0	0	0	0
Timer/Event Counter 0 Overflow	0	0	0	0	0	0	0	1	0	1	0	0
Timer/Event Counter 1 Overflow	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

Arithmetic and Logic Unit – ALU

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

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

Flash Program Memory

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

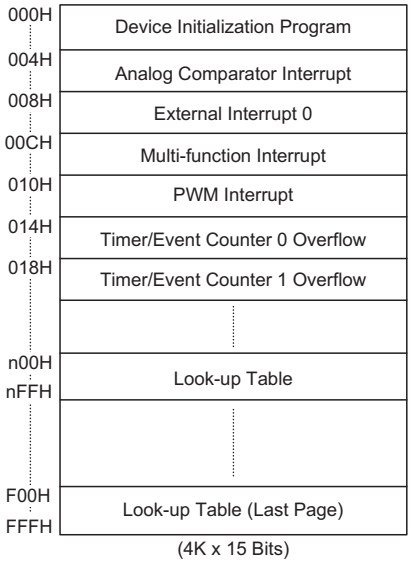
Structure

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 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 Analog Comparator interrupt. If an Analog Comparator interrupt, resulting from a falling edge on the Analog Comparator output occurs, the program will jump to this location and begin execution if the Analog Comparator interrupt 0 is enabled and the stack is not full.
- Location 008H
  This vector is used by the external interrupt 0. If the INT0A, INT0B or INT0C external interrupt pins on the device receives an edge transition, the program will jump to this location and begin execution if the external interrupt 0 is enabled and the stack is not full.
- Location 00CH
  This vector is used by the multi-function interrupt. If the external interrupt 1 pin on the device receives an edge transition, or when an A/D conversion cycle is complete, the program will jump to this location and begin execution if the multi-function interrupt is enabled and the stack is not full. The external interrupt 1 active edge transition type, whether high to low, low to high or both, is specified in the Configuration Options.
- Location 010H
  This vector is used by the PWM interrupt. If a PWM interrupt, resulting from a PWMxH is from inactive to active, the program will jump to this location and begin execution if the PWM interrupt is enabled and the stack is not full.
- Location 014H
  This internal vector is used by the Timer/Event Counter 0. If the counter overflow occurs, the program will jump to this location and begin execution if the timer/event counter 0 interrupt is enabled and the stack is not full.
- Location 018H
  This internal vector is used by the Timer/Event Counter 1. If the counter overflow occurs, the program will jump to this location and begin execution if the timer/event counter 1 interrupt is enabled and the stack is not full.



Program Memory Structure

Look-up Table

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

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

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

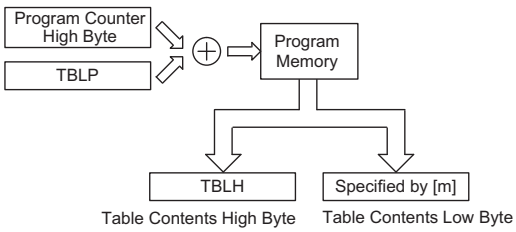


Table Program Example

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 device. The table pointer is setup here to have an initial value of ″06H″. This will ensure that the first data read from the data table will be at the Program Memory address ″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 the table read instructions. If using the table read instructions, the Interrupt Service Routines may change the value of TBLH and subsequently cause errors if used again by the main routine. As a rule it is recommended that simultaneous use of the table read instructions should be avoided. However, in situations where simultaneous use cannot be avoided, the interrupts should be disabled prior to the execution of any main routine table-read instructions. Note that all table related instructions require two instruction cycles to complete their operation.

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

```
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 tempreg1
                            ; 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
                            ; the value "0FH" will be transferred to the high byte
                            ; register TBLH
            :
            :

org         0F00h           ; sets initial address of last page

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

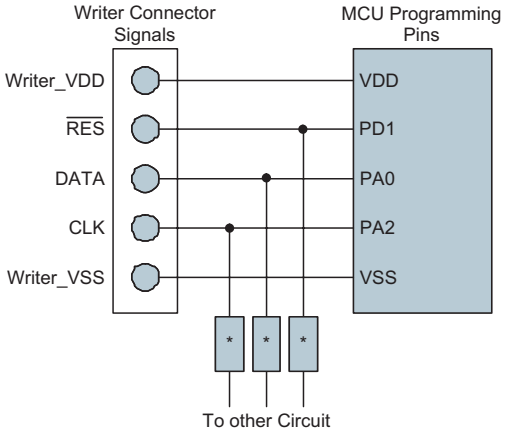
In Circuit Programming

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

As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 5-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or upgrading the program at a later stage. This enables product manufacturers to easily keep their manufactured products supplied with the latest program releases without removal and re-insertion of the device.

The Program Memory can be programmed serially in-circuit using this 5-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Two additional lines are required for the power supply and one line for the reset. The technical details regarding the in-circuit programming of the devices are beyond the scope of this document and will be supplied in supplementary literature.

During the programming process the $\overline{\text{RES}}$ pin will be held low by the programmer disabling the normal operation of the microcontroller and taking control of the PA0 and PA2 I/O pins for data and clock programming purposes. The user must there take care to ensure that no other outputs are connected to these two pins.



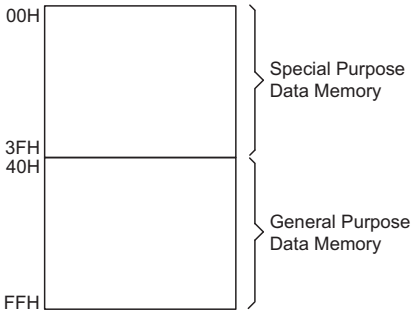
Note: * may be resistor or capacitor. The resistance of * must be greater than 1kΩ or the capacitance of * must be less than 1nF.

Data Memory

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

Structure

The two sections of Data Memory, the Special Purpose and General Purpose Data Memory are located at consecutive locations. All are implemented in RAM and are 8 bits wide. The start address of the Data Memory is the address ″00H″.

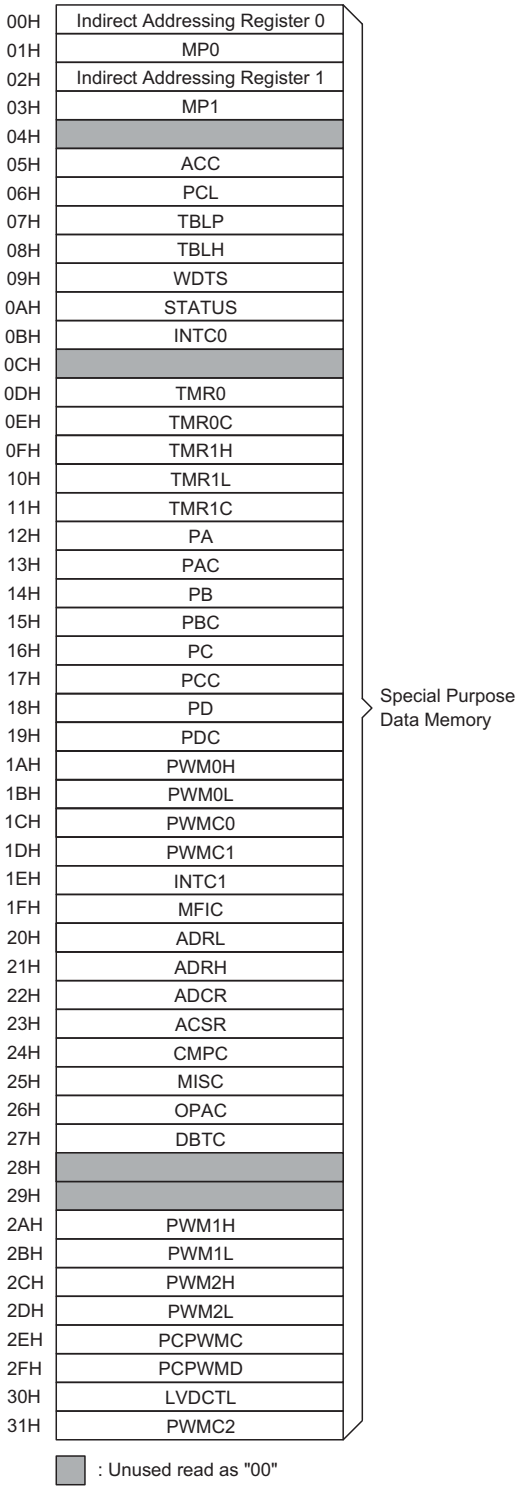


Data Memory Structure

Note: Most of the RAM Data Memory bits can be directly manipulated using the ″SET [m].i″ and ″CLR [m].i″ instructions with the exception of a few dedicated bits. The RAM 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 RAM memory that is known as General Purpose Data Memory. This area of Data Memory is fully accessible by the user program for both read and write operations. By using the ″SET [m].i″ and ″CLR [m].i″ instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.



Special Purpose RAM 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″.

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 Register – IAR0, IAR1

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

Memory Pointer – MP0, MP1

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

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.

Watchdog Timer Register – WDTS

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

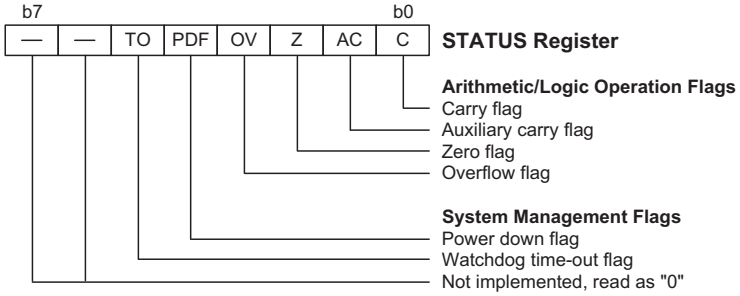
Status Register – STATUS

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

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

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.



Status Register

- **Z** is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- **OV** is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
- **PDF** is cleared by a system power-up or executing the ″CLR WDT″ instruction. PDF is set by executing the ″HALT″ instruction.
- **TO** is cleared by a system power-up or executing the ″CLR WDT″ or ″HALT″ instruction. TO is set by a WDT time-out.

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

Interrupt Control Register – INTC0, INTC1, MFIC

These 8-bit registers, known as INTC0, INTC1 and MFIC register, control the operation of both external and internal interrupts. By setting various bits within this register using standard bit manipulation instructions, the enable/disable function of each interrupt can be independently controlled. A master interrupt bit within this register, the EMI bit, acts like a global enable/disable and is used to set all of the interrupt enable bits on or off. This bit is cleared when an interrupt routine is entered to disable further interrupt and is set by executing the ″RETI″ instruction.

Timer/Event Counter Registers

The device contains an 8-bit Timer/Event Counter and a 16-bit Timer/Event Counter. For the 8-bit Timer/Event Counter an associated register known as TMR0 is the location where the timer's 8-bit value is located. An associated control register, known as TMR0C, contains the setup information for this timer. For the 16-bit Timer/Event Counter two associated register known as TMR1L and TRM1H are the locations where the timer's 16-bit values are located. An associated control register, known as TMR1C, contains the setup information for this timer.

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 – PWM0H, PWM0L, PWM1H, PWM1L, PWM2H, PWM2L, PWMC0, PWMC1, PWMC2, PCPWMC, PCPWMD

The device contains a 3-channel 10-bit Pulse Width Modulator function. Each PWM channel has its own complementary pair of output and their own related register pair, PWM0L/PWM0H, PWM1L/PWM1H, and PWM2L/PWM2H as well as PWMC0, PWMC1, PWMC2, PCPWMC and PCPWMD. The PWMxH and PWMxL (x=0~2) register defines the duty cycle value for the modulation cycle of the Pulse Width Modulator PWM0 or PWM1 or PWM2.

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

The device contains an 8-channel 12-bit A/D converter. The correct operation of the A/D requires the use of two data registers, a control register and a clock source register. 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. The channel selection and configuration of the A/D converter is setup via the control register ADCR while the A/D clock frequency is defined by the clock source register, ACSR.

Analog Comparator Control Register – CMPC

This register is used to provide control over the internal Analog Comparator function.

Miscellaneous Control Register – MISC

This register is used to provide control over the internal Analog Comparator and PWM functions.

Operation Amplifier Control Register – OPAC

This register is used to provide control over the internal Operation Amplifier function.

Analog Comparator Interrupt Debounce Time Control Register – DBTC

This register is used to provide control over the internal Analog Comparator Interrupt debounce time, PWMxH and PWMxL output control, INT0A, INT0B and INT0C pin-shared output disable control and PWMxH/PWMxL full active/inactive control.

Input/Output Ports

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

The device provides 26 bidirectional input/output lines labeled with port names PA, PB, PC and PD. These I/O ports are mapped to the RAM Data Memory with specific addresses as shown in the Special Purpose Data Memory table. All of these I/O ports can be used for input and output operations. For input operation, these ports are non-latching, which means the inputs must be ready at the T2 rising edge of instruction "MOV A,[m]", where m denotes the port address. For output operation, all the data is latched and remains unchanged until the output latch is rewritten.

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 selectable via configuration options and are implemented using a weak PMOS transistor.

Port A Wake-up

The HALT instruction forces the microcontroller into a Power Down condition which preserve 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 idle or in a low-power state until the logic condition of the selected wake-up pin on Port A changes from high to low. This function is especially suitable for applications that can be woken up via external switches. Note that each pin on Port A can be selected individually to have this wake-up feature.

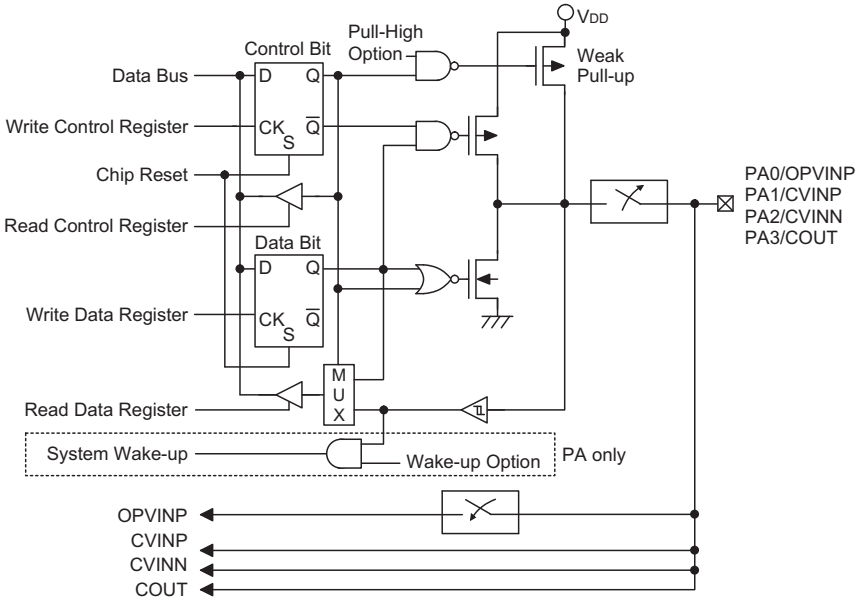
I/O Port Control Registers

Each I/O port has its own control register 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.

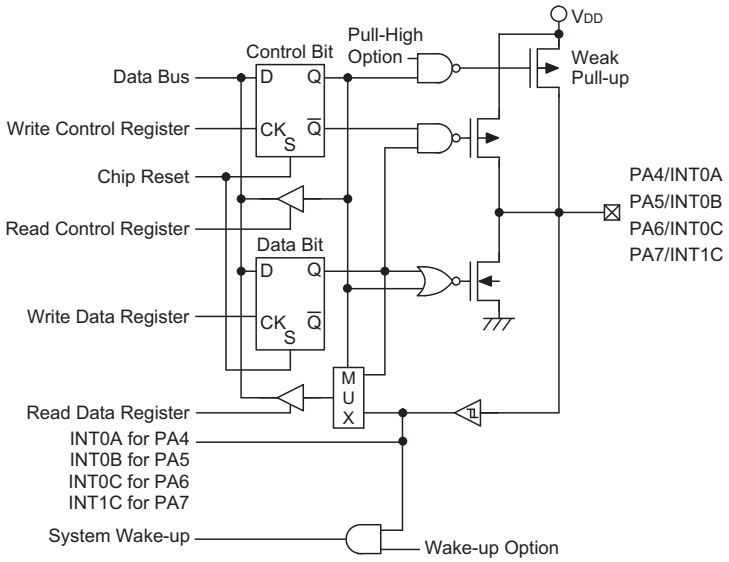
Pin-shared Functions

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.

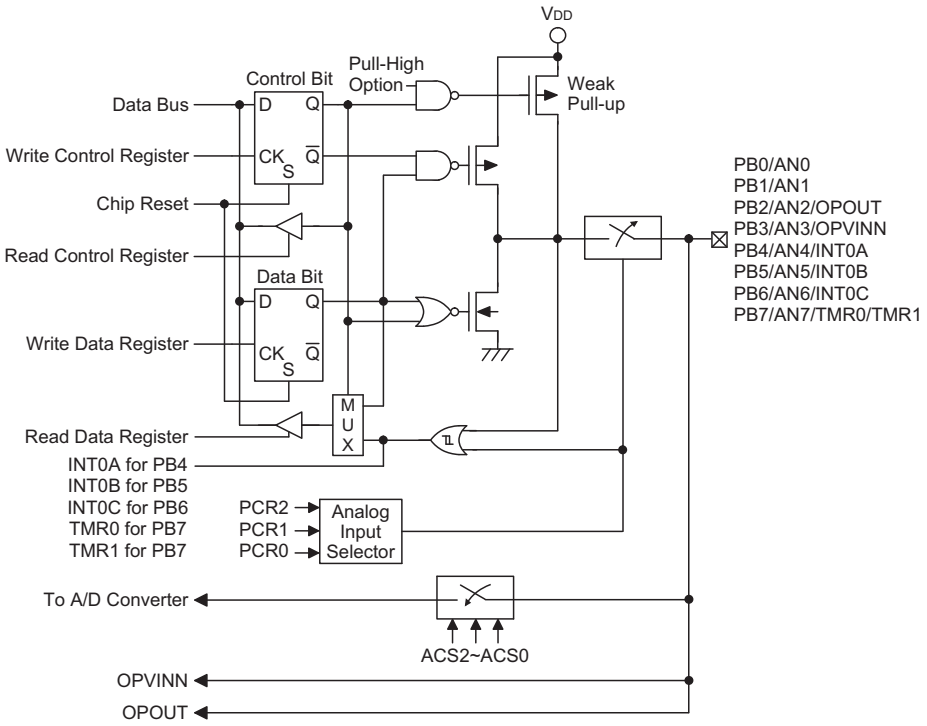
- External Interrupt 0 Input
  The external interrupt pins INT0A, INT0B and INT0C are pin-shared with the I/O pins PA4~PA6 or PB4~PB6. The function is chosen using configuration options. For applications not requiring these external interrupt inputs, the pin-shared external interrupt pins can be used as normal I/O pins, however to do this, the external interrupt 0 enable bits in the INTC0 register must be disabled. To configure them to operate as external interrupt inputs, the corresponding bits in the interrupt control register must be correctly set and the pins must be setup as inputs. Note that the original I/O function will remain even if these pins are setup to be used as external interrupt 0 inputs. The INT0A, INT0B and INT0C pins can be selected as input line only by software. If the HSIC is 1, the INT0A, INT0B and INT0C pin shared I/O output function are disabled and these I/O can be input only and without pull-high resistor.
- External Interrupt 1 Input
  The external interrupt pin INT1 is pin-shared with the I/O pin PA7. For applications not requiring an INT1 input, the pin can be used as a normal I/O pin, however to do this, the external interrupt 1 enable bits in the INTC0 register must be disabled. To configure it to operate as an external interrupt 1 input, the corresponding bits in the interrupt control register must be correctly set and the pin must 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 interrupt.



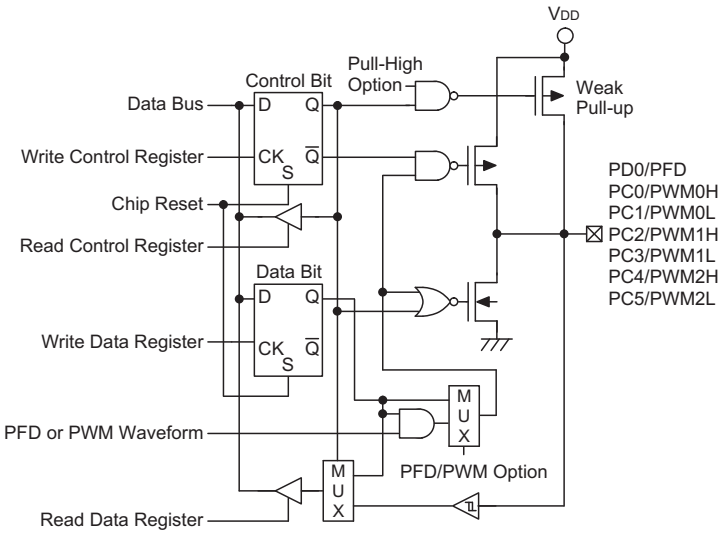
PA0~PA3 Input/Output Ports



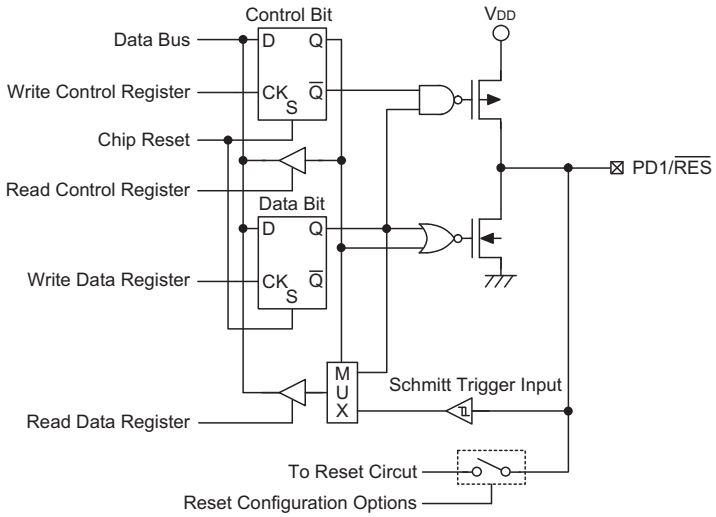
PA4~PA7 Input/Output Ports



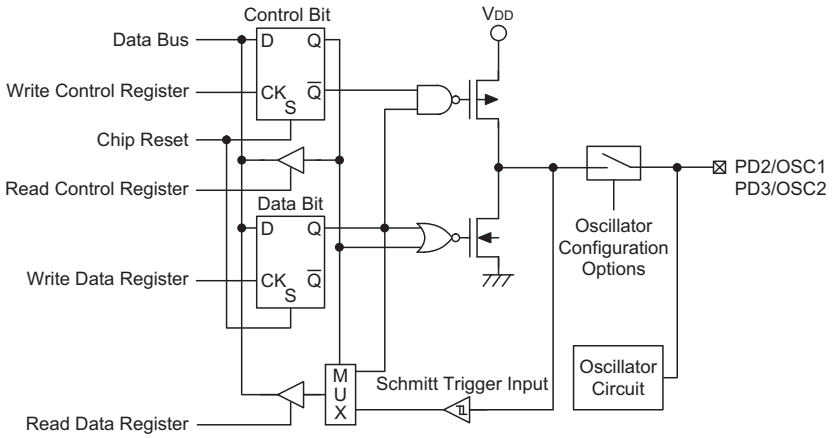
PB Input/Output Ports



PD0/PFD and PC Input/Output Ports



PD1 Input/Output Port



PD2/PD3 Input/Output Ports

- External Timer Inputs
  The external timer pins TMR0 and TMR1 are pin-shared with the I/O pin PB7. To configure them to operate as timer inputs, the corresponding control bits in the timer control register must be correctly set and the pins must also be setup as inputs. Note that the original I/O function will remain even if the pins are setup to be used as external timer inputs.

- PFD Output
  The device contains a PFD function whose single output is pin-shared with PD0. 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, PDC.1, must setup the pin as an output to enable the PFD 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 option, even if the PFD configuration option has been selected.

- PWM Output
  The device contains PWM outputs pin shared with pins PC0~PC5. The PWM output functions are chosen via software options. Note that the corresponding bits in the port control register, PCC, must setup the pins as outputs to enable the PWM output. If the PCC port control register has setup the pins as inputs, then the pins will function as normal logic inputs with the usual pull-high resistor option, even if the PWM software option has been selected.

- A/D Inputs
  The device has eight 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 configuration options remain, however if used as A/D inputs then any pull-high resistor selections associated with these pins will be automatically disconnected.

- Analog Comparator Function
  The device contains an Analog Comparator function whose pins are pin-shared with PA1~PA3. The Analog Comparator function of these pins are chosen using software.

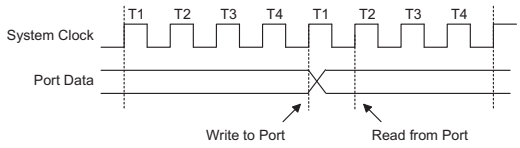
- Operational Amplifier Function
  The device contains an Operational Amplifier function whose pins are pin-shared with PA0, PB2 and PB3. The Operational Amplifier function of these pins are chosen using software.

I/O Pin Structures

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

Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, 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 options have been selected. 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 value into the port control register or by programming individual bits in the port control register using the "SET [m].i" and "CLR [m].i" instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.



Read/Write Timing

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 these 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 an internal 8-bit count-up timer and an internal 16-bit count-up timer. 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 an internal prescaler to the clock circuitry gives added range to the timer/event counters.

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 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 are the timer control registers which defines the timer options and determines how the timer is to be used. This device can have the timer clock configured to come from the inter-

nal clock source. In addition the timer 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 source can originate from either the system clock or from an external clock source. The system clock input timer source is used when the timer is in the timer mode or in the pulse width measurement mode.

The Timer/Event Counter 0 clock also passes through a prescaler, the value of which is conditioned by the bits T0PSC0, T0PSC1 and T0PSC2 in the TMR0C register. The Timer/Event Counter 1 clock also passes through a prescaler, the value of which is conditioned by the bits T1PSC0, T1PSC1 and T1PSC2 in the TMR1C register.

An external clock source is used when the timer is in the event counting mode, the clock source being provided on shared pin PB7/TMR0/TMR1. Depending upon the condition of the T0E or T1E bit, each high to low, or low to high transition on the PB7/TMR0/TMR1 pin will increment the counter by one.

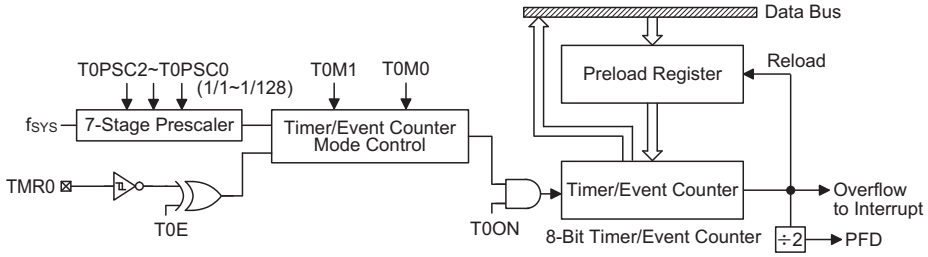
Timer Register – TMR0, TMR1L/TMR1H

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. For the 8-bit timer, this register is known as TMR0. In the case of the 16-bit timer, a pair of 8-bit registers is required to store the 16-bit timer value, and are known as TMR1L and TMR1H.

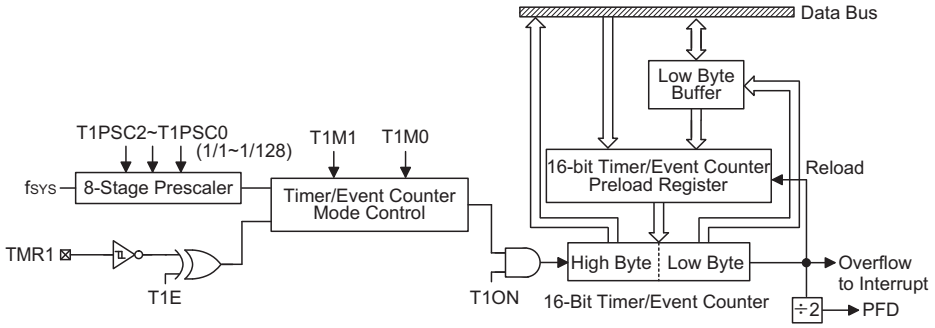
The value in the timer registers increases by one each time an internal clock pulse is received or an external transition occurs on the external timer pin. The timer will count from the initial value loaded by the preload register to the full count of FFH for the 8-bit timer or FFFFH for the 16-bit timers at which point the timer overflows and an internal interrupt signal is generated. The timer value will then be reset with the initial preload register value and continue counting.

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

For device which has an internal 16-bit Timer/Event Counter, and which therefore have both low byte and high byte timer registers, accessing these registers is carried out in a specific way. It must be noted that when using instructions to preload data into the low byte register, namely TMR1L, the data will only be placed in a low byte buffer and not directly into the low byte register. The actual transfer of the data into the low byte register is only carried out when a write to its associated high byte register, namely TMR1H, is executed. On the other



Timer/Event Counter 0



Timer/Event Counter 1

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

Timer Control Register – TMR0C, TMR1C

The flexible features of the Holtek microcontroller Timer/Event Counters enable them to operate in three different modes, the options of which are determined by the contents of their respective control register. There are two timer control registers known as TMR0C and TMR1C. It is the timer control register together with its corresponding timer registers that control the full operation of the Timer/Event Counters. Before the timers can be used, it is essential that the appropriate timer control register is fully programmed with the right data to ensure its correct operation, a process that is normally carried out during program 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 Timer Control Register, which are known as the bit pair T0M1/T0M0 or T1M1/T1M0 respectively, depending upon which timer is used, must be set to the required logic levels. The timer-on bit, which is bit 4 of the Timer Control Register and known as T0ON or T1ON, depending upon which timer is used, provides the basic on/off control of the respective timer. Setting the bit high allows the counter to run, clearing the bit stops the counter. Bits 0~2 of each Timer Control Register determine the division ratio of the input clock prescaler. The prescaler bit settings have no effect if an external clock source is used. If the timer is in the event count or Pulse Width Measurement mode, the active transition edge level type is selected by the logic level of bit 3 of the Timer Control Register which is known as T0E or T1E, depending upon which timer is used.

Configuring the Timer Mode

In this mode, the Timer/Event Counter can be setup 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 in the Timer Control Register must be set to the correct value as shown.

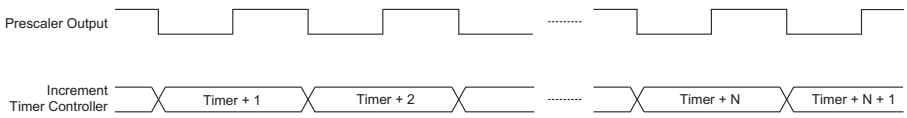
Control Register Operating Mode Select Bits for the Timer Mode	Bit7	Bit6
	1	0

In this mode the internal clock, f_{SYS}, is used as the Timer/Event Counter clock. However, this clock source is further divided by a prescaler, the value of which is determined by the Prescaler Rate Select bits, which are bits 0~2 in the Timer Control Register. After the other bits in the Timer Control Register have been setup, the enable bit, which is bit 4 of the Timer Control Register, can be set high to enable the Timer/Event Counter to run. Each time an internal clock cycle occurs, the Timer/Event Counter increments by one. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the Interrupt Control Register, INTC1, is reset to zero.

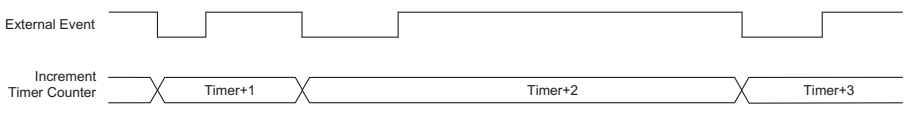
Configuring the Event Counter Mode

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 in the Timer Control Register must be set to the correct value as shown.

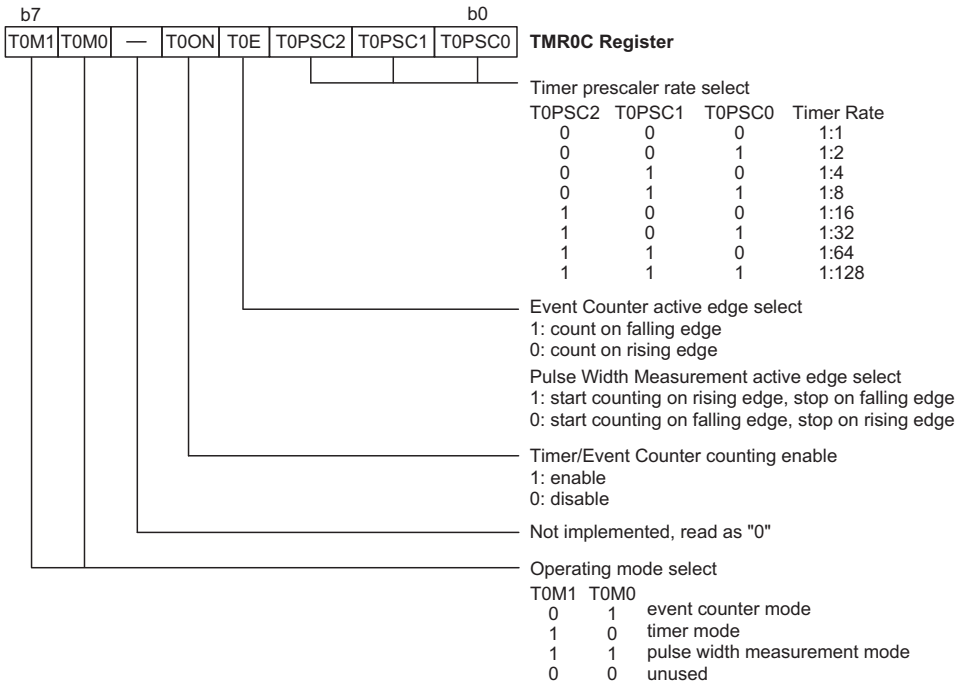
Control Register Operating Mode Select Bits for the Event Counter Mode	Bit7	Bit6
	0	1



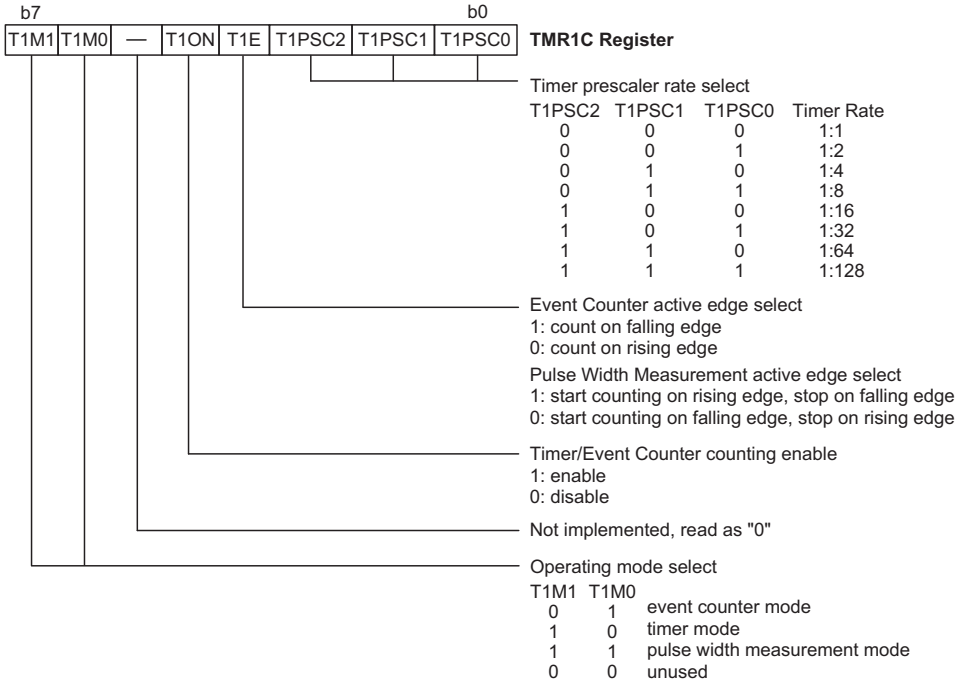
Timer Mode Timing Chart



Event Counter Mode Timing Chart



Timer/Event Counter 0 Control Register



Timer/Event Counter 1 Control Register

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, 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, which is bit 3 of the Timer Control Register, is low, the Timer/Event Counter will increment each time the external timer pin receives a low to high transition. If the Active Edge Select bit is high, the counter will increment each time the external timer pin receives a high to low transition. When it is full and overflows, an interrupt signal is generated and the Timer/Event Counter will reload the value already loaded into the preload register and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the Interrupt Control Register, INTC1, is reset to zero.

As the external timer pin is shared with an I/O pin, PB7/TMR0/TMR1, 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 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

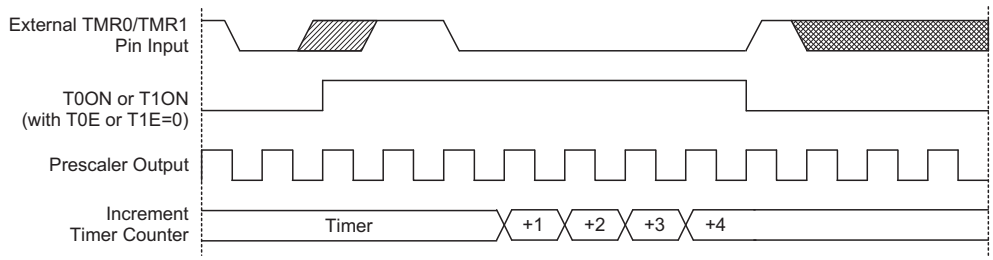
Bit7	Bit6
1	1

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



Prescaler Output is sampled at every falling edge of T1.

Pulse Width Measure Mode Timing Chart

and continue counting. The interrupt can be disabled by ensuring that the Timer/Event Counter Interrupt Enable bit in the Interrupt Control Register 1, INTC1, 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 PFD output is pin-shared with the I/O pin PD0. The PFD on/off function and its timer source are selected via configuration option, however, if not selected, the pin can operate as a normal I/O pin. The timer overflow signal is the clock source for the PFD circuit. The output frequency is controlled by loading the required values into the timer register and if available the timer prescaler registers to give the required frequency. The timer/event counter, driven by the system clock and if applicable, divided by the prescaler value, will begin to count-up from this preloaded register value until full, at which point an overflow signal will be generated, causing the PFD output to change state. The counter will then be automatically reloaded with the preload register value and once again continue counting-up.

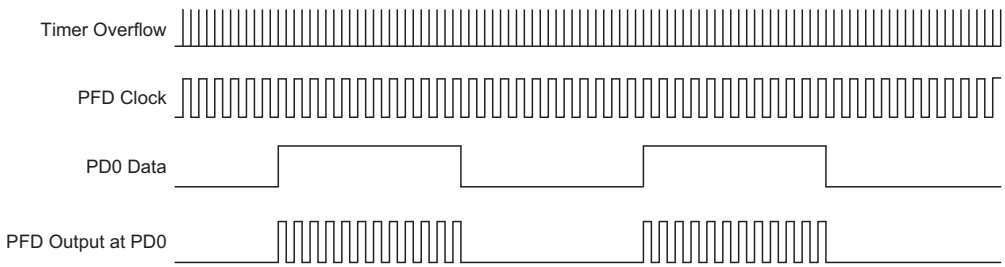
For the PFD output to function, it is essential that the corresponding bit of the Port D control register PDC bit 0 is setup as an output. If setup as an input the PFD output will not function, however, the pin can still be used as a normal input pin. The PFD output will only be activated if bit PD0 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 PD0 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.

Bits T0PSC0~T0PSC2 of the TMR0C register are used to define the pre-scaling stages of the internal clock sources of Timer/Event Counter 0. Bits T1PSC0~T1PSC2 of the TMR1C register are used to define the pre-scaling stages of the internal clock sources of Timer/Event Counter 1. The Timer/Event Counter 0 and Timer/Event Counter 1 overflow signals can be used to generate signals for the PFD and Timer Interrupt.

I/O Interfacing

The Timer/Event Counter 0 and Timer/Event Counter 1, when configured to run in the event counter or pulse width measurement mode, require the use of the external PB7/TMR0/TMR1 pin for correct operation. As this pin is a shared pin it must be configured correctly to ensure it is setup for use as a Timer/Event Counter 0 and Timer/Event Counter 1 input and not as a normal I/O pin. This is implemented by ensuring that the mode select bits in the Timer/Event Counter 0 or Timer/Event Counter 1 control register, selects either the event counter or pulse width measurement mode. Additionally the Port Control Register PBC bit 7 must be set high to ensure that the pin is setup as an input. Any pull-high resistor configuration option on this pin will remain valid even if the pin is used as a Timer/Event Counter 0 or Timer/Event Counter 1 input.



PFD Output Control

Programming Considerations

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

When the Timer/Event Counter is read or if data is written to the registers, the clock is inhibited to avoid errors, however as this may result in a counting error, this should be taken into account by the programmer. Care must be taken to ensure that the timers are properly initialised before using them for the first time. The associated timer enable bits in the interrupt control register must be properly set otherwise the internal interrupt associated with the timer will remain inactive. The edge select, timer mode and clock source control bits in timer control register must also be correctly set to ensure the timer is properly configured for the required application.

It is also important to ensure that an initial value is first loaded into the timer register before the timer is switched on; this is because after power-on the initial value of the timer register is unknown. After the timer has been initialised the timer can be turned on and off by controlling the enable bit in the timer control register. Note that setting the timer enable bit high to turn the timer on, should only be executed after the timer mode bits have been properly setup. Setting the timer enable bit high together with a mode bit modification, may lead to improper timer operation if executed as a single timer control register byte write instruction.

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

Timer Program Example

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

```
org 04h             ; analog comparator interrupt vector
reti
org 08h             ; external interrupt 0 vector
reti
org 0ch             ; multi-function interrupt vector
reti
org 010h            ; PWM interrupt vector
reti
org 014h            ; Timer Counter 0 interrupt vector
jmp tmr0int         ; jump here when Timer 0 overflows
org 018h            ; Timer Counter 1 interrupt vector
jmp tmr1int         ; jump here when Timer 1 overflows
:
:
org 20h             ; main program
:
:
;internal Timer 0 interrupt routine
tmr0int:
:
; Timer 0 main program placed here
:
reti
:
;internal Timer 1 interrupt routine
tmr1int:
:
;Timer 1 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 Timer 1 registers
clr tmr1l
clr tmr1h           ; clear timer register to give maximum count value
mov a,080h          ; setup Timer 1 control register
mov tmr1c,a         ; timer mode and prescaler set to /1
;setup interrupt register
mov a,06h           ; enable Timer 0 and Timer 1 interrupt
mov intc1,a
mov a,01h           ; enable master interrupt
mov intc0,a
:
:
set tmr0c.4  ; start Timer 0
set tmr1c.4  ; start Timer 1
:
```

Pulse Width Modulator

The microcontroller is provided with a three channel 10-bit PWM function. Each channel has a pair of Complementary 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 10-bit values into the corresponding PWM registers.

PWM Clock Source

The PWM clock is sourced from the system clock f_{SYS}. It can be further subdivided using an internal divider to provide a frequency range from f_{SYS} ~ $f_{SYS}/8$ using the PWMPS0~PWMPS2 bits in the PWMC1 register. This clock source is used to drive an internal PWM 10-bit counter which is compared with the programmed PWM data value for duty cycle control. The clock source selection also determines the PWM signal frequency. The PWM frequency is determined by the internal PWM 10-bit counter overflow signal. A PWM clock source of f_{SYS}~$f_{SYS}/8$, gives a fixed frequency output range of $f_{SYS}/1024$~ $f_{SYS}/8192$.

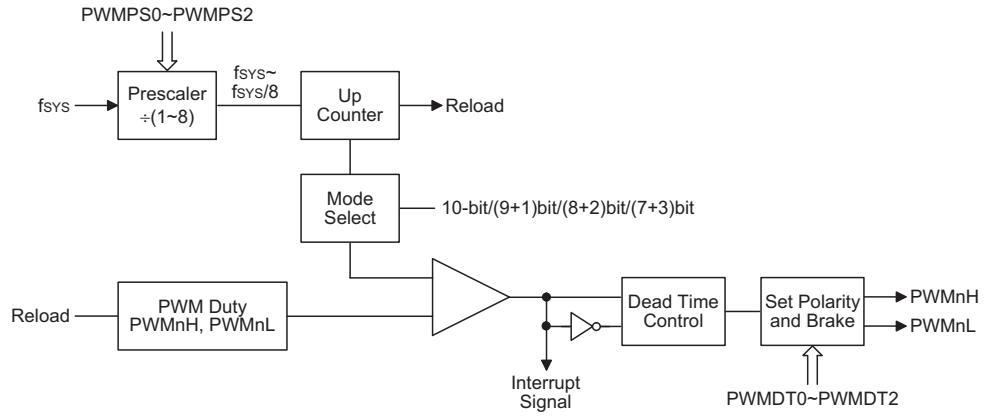
PWM Operation

As the PWM duty cycle value is 10-bits wide, each channel requires a pair of data registers, a low byte and a high byte, named PWMnL and PWMnH . The lower two bits are stored in the PWMnL register while the eight higher order bits are stored in the PWMnH register.

Register	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWMxH	D9	D8	D7	D6	D5	D4	D3	D2
PWMxL	—	—	—	—	—	—	D1	D0

PWMnH, PWMnL Registers

Writing to the PWM data register pair, PWMnL/PWMnH, must be carried out in a specific way. It must be noted that when writing data into the low byte register, namely PWMnL, the data will only be placed in a low byte buffer and not directly into the PWMnL register. The actual transfer of the data into the low byte timer register is only carried out when a write to its associated high byte timer register, namely PWMnH, is executed. However, using instructions to write data into the high byte register, PWMnH, will result in the data being directly written to the register. At the same time the data in the low byte buffer will be transferred into the PWMnL register. For this reason, the low byte register should be written first when preloading data into the PWM register pair. Similary when reading the PWM low register, PWMnL, note that only the low byte buffer value will be read. If the PWM duty has a value between 3F0H to 3FFH, then the PWMxH output will be fully active and the PWMxL output will be fully inactive.



PWM Block Diagram

As the PWM outputs are shared with I/O pins, each pin has a corresponding bit in the PCPWMC register to determine if it is to be used an an I/O pin or as a PWM output pin. The three PWM channels can be chosen to have either single or dual complimentary outputs, chosen via the PWMCEN bit in the PWMC0 register. The Port control register, PCC, must also ensure that the pins are setup as outputs for correct operation.

Once the PWM function is properly setup, its individual outputs can be enabled using bits in the PCPWMD register. When the PWM output function is enabled, the PWMPS0~PWMPS2, PWMxL and PWMxH register values can be written to at any time, even if the PWM is running.

When the BLDC mode is enabled by setting the BLDCMD bit in the DBTC register high, this ensures that the PWMnH/PWMnL and their corresponding complimentary I/O pins can never both be active at the same time. Pins PC0/PC2/PC4 or PC1/PC3/PC5 are active high or active low also by configuration option.

The PWM frequency depends upon the PWM Mode chosen, the system clock frequency f_{SYS} and the condition of the PWMPS0~PWMPS2 bits. By programming suitable values for these bits the user has the flexibility to choose from a wide range of frequencies to suit their application needs. The following equation shows how the frequency can be calculated. The decimal equivalent of the binary bit value should be substituted to calculate the value.

Mode	PWM Frequency
10-bit	$f_{SYS}\div(1024\times((PWMPS2\sim PWMPS0)+1)$
9+1	$f_{SYS}\div(512\times((PWMPS2\sim PWMPS0)+1)$
8+2	$f_{SYS}\div(256\times((PWMPS2\sim PWMPS0)+1)$
7+3	$f_{SYS}\div(128\times((PWMPS2\sim PWMPS0)+1)$

For Example in the 10-bit operating mode if:

- f_{SYS}=12MHz
- PWMPS2~PWMPS0=011 (decimal 3)
  … gives a PWM frequency of 2.93kHz

The accompanying table shows a range of already calculated PWM frequency values for user reference.

- 10-bit PWM Full Frequency

PWMPS2~ PWMPS0	f_{SYS}= 12MHz	f_{SYS}= 16MHz	f_{SYS}= 20MHz	Unit
000	11.72	15.63	19.53	kHz
001	5.86	7.81	9.77	kHz
010	3.91	5.21	6.51	kHz
011	2.93	3.91	4.88	kHz
100	2.34	3.13	3.91	kHz
101	1.95	2.60	3.26	kHz
110	1.67	2.23	2.79	kHz
111	1.46	1.95	2.44	kHz

- (9+1)-bit PWM Full Frequency

PWMPS2~ PWMPS0	f_{SYS}= 12MHz	f_{SYS}= 16MHz	f_{SYS}= 20MHz	Unit
000	23.44	31.25	39.06	kHz
001	11.72	15.63	19.53	kHz
010	7.81	10.42	13.02	kHz
011	5.86	7.81	9.77	kHz
100	4.69	6.25	7.81	kHz
101	3.91	5.21	6.51	kHz
110	3.35	4.46	5.58	kHz
111	2.93	3.91	4.88	kHz

- (8+2)-bit PWM Full Frequency

PWMPS2~ PWMPS0	f_{SYS}= 12MHz	f_{SYS}= 16MHz	f_{SYS}= 20MHz	Unit
000	46.88	62.50	78.13	kHz
001	23.44	31.25	39.06	kHz
010	15.63	20.83	26.04	kHz
011	11.72	15.63	19.53	kHz
100	9.38	12.50	15.63	kHz
101	7.81	10.42	13.02	kHz
110	6.7	8.93	11.16	kHz
111	5.86	7.81	9.77	kHz

- (7+3)-bit PWM Full Frequency

PWMPS2~PWMPS0	f_{SYS}= 12MHz	f_{SYS}= 16MHz	f_{SYS}= 20MHz	Unit
000	93.75	125.00	156.25	kHz
001	46.88	62.50	78.13	kHz
010	31.25	41.67	52.08	kHz
011	23.44	31.25	39.06	kHz
100	18.75	25.00	31.25	kHz
101	15.63	20.83	26.04	kHz
110	13.39	17.86	22.32	kHz
111	11.72	15.63	19.53	kHz

PWM Modes

The PWM output can be chosen to operate in a number of output modes, these are 10-bit, (9+1) bit, (8+2) bit and (7+3) bit output modes. The difference between each Mode is in how the PWM output waveform is subdivided into different sub cycles. The output mode is chosen using a configuration option and applies to all channels.

The (9+1) Mode PWM cycle is divided into two modulation cycles, modulation cycle 0 and modulation cycle 1. Each modulation cycle has 512 PWM input clock periods. In a (9+1) bit PWM function, the contents of the PWM register are divided into two groups. Group 1 of the PWM registers are denoted by DC which are the values of PWMH.7~PWMH.0 and PWML.1 (9 bits from PWMH~PWML.1). Group 2 is denoted by AC which is the value of PWML.0.

Parameter	AC0~AC1	Duty Cycle
Modulation Cycle I (i=0~1)	i<AC	(DC+1)/512
	i≥AC	DC/512

9+1 Mode

The (8+2) bits mode PWM cycle is divided into four modulation cycles, modulation cycle 0~modulation cycle 3. Each modulation cycle has 256 PWM input clock periods. In a (8+2) bit PWM function, the contents of the PWM register are divided into two groups. Group 1 of the PWM register is denoted by DC which is the value of PWMH.7~PWMH.0. Group 2 is denoted by AC which is the value of PWML.1~PWML.0 (PWML).

Parameter	AC0~AC3	Duty Cycle
Modulation Cycle I (i=0~3)	i<AC	(DC+1)/256
	i≥AC	DC/256

8+2 Mode

The (7+3) bits mode PWM cycle is divided into eight modulation cycles, modulation cycle 0~modulation cycle 7. Each modulation cycle has 128 PWM input clock periods. In the (7+3) bit PWM function, the contents of the PWM register are divided into two groups. Group 1 of the PWM register is denoted by DC which is the value of PWMH.7~PWMH.1. Group 2 is denoted by AC which are the value of PWMH.0 and PWML.1~PWML.0 (PWMH.0~PWML.0).

Parameter	AC0~AC7	Duty Cycle
Modulation Cycle I (i=0~7)	i<AC	(DC+1)/128
	i≥AC	DC/128

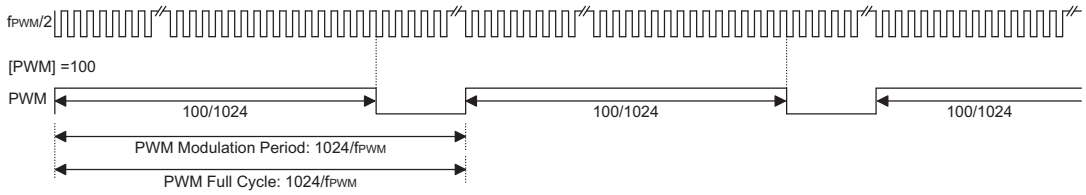
7+3 Mode

The modulation frequency, cycle frequency and cycle duty of the PWM output signal are summarized in the accompanying table.

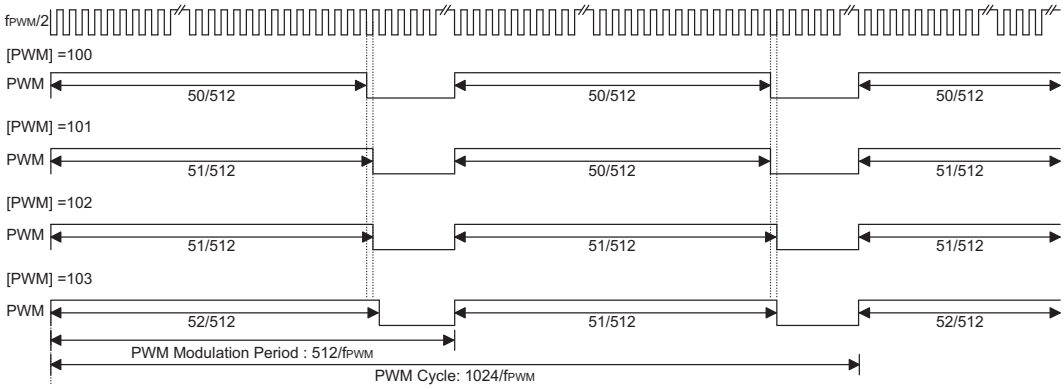
PWM Modulation Frequency	PWM Cycle Frequency	PWM Cycle Duty
f_{PWM}/1024 -10-bits mode f_{PWM}/512 - (9+1) mode f_{PWM}/256 - (8+2) mode f_{PWM}/128 - (7+3) mode	f_{PWM}/1024	[PWM]/1024

$f_{PWM}=f_{SYS}/1024$~$f_{SYS}/8192$

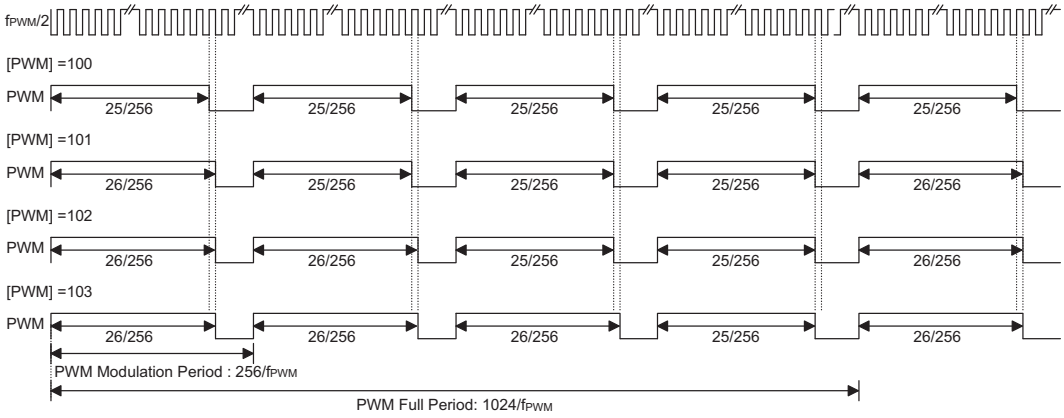
Frequency Table



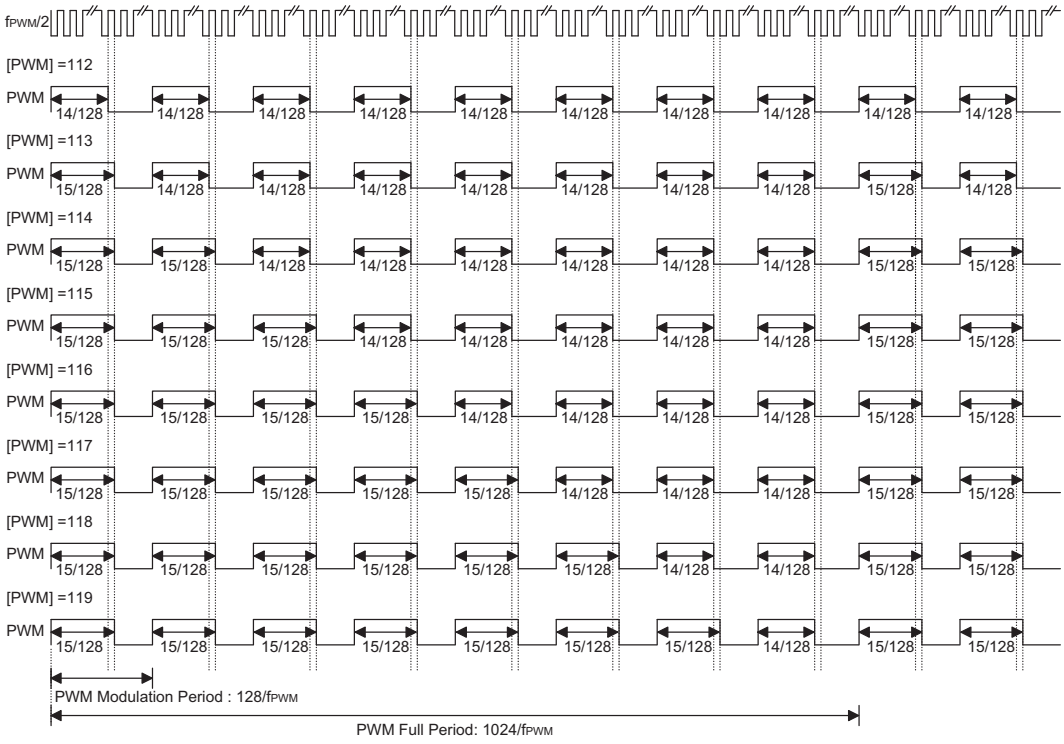
Standard 10 Bit PWM Mode



(9+1) PWM Mode



(8+2) PWM Mode



(7+3) PWM Mode

PWM Output Control

There are a total of six PWM pins, divided into three pairs of complimentary PWM outputs. The PWM outputs are controlled using a combination of register bits, configuration options and external pins. The PWM outputs can be either a single output, PWMH, or a complimentary pair of outputs, PWMH and PWML. Both the PWMH and PWML outputs can be set to be either active high or active low using configuration options.

I/O Pin	PWM Line
PC0	PWM0H
PC1	PWM0L
PC2	PWM1H
PC3	PWM1L
PC4	PWM2H
PC5	PWM2L

The PWMCTRL bit in the PWMC0 register acts as a master control bit for the PWM outputs. When this bit is high the selected PWM outputs will be active and when the bit is low all the PWM outputs will be placed into their inactive state as defined by configuration options. The three PWM pins PWM0H, PWM1H and PWM2H have a control bit, PWMEN, and the complementary PWM pins, PWM0L, PWM1L and PWM2L, have a control bit, PWMCEN. When these bits are set to high their three corresponding PWM pins will be active, when the bits are cleared to zero their PWM pins will be set to the inactive state as defined by configuration options.

As not all of the PWM may be required, the devices offer the flexibility to select which pins are used as PWM pins and which pins are used as I/O pins. This is determined either by bits in the PCPWMC and PCPWMD registers or by the PC6 and PC7 pins. The PWMCM bit in the PWMC2 register selects which method is used for PWM or I/O control.

PWMCM Bit	PWM or I/O Select	PWM Enable
1	PCPWMC register	PCPWMD register PWMEN/PWMCEN bits PWMCTRL bit
0	PC6/PC7 pins PWMEN/PWMCEN bits	PCPWMD register PC.0~PC.5 port data bits, PWMCTRL bit

When the PWMCM bit is zero, external pins PC6 and PC7 will control which pins are PWM output pins and which pins are I/O pins as shown in the following tables:

PWMC=0

PWMEN	Control Pins		Pin Function		
	PC7	PC6	PC4	PC2	PC0
0	X	X	I/O	I/O	I/O
1	0	0	I/O	I/O	0: inactive PWM0H 1: active PWM0H
1	0	1	I/O	0: inactive PWM1H 1: active PWM1H	I/O
1	1	0	0: inactive PWM2H 1: active PWM2H	I/O	I/O
1	1	1	0: inactive PWM2H 1: active PWM2H	0: inactive PWM1H 1: active PWM1H	0: inactive PWM0H 1: active PWM0H

PWM Pin Output Control

PWMC=0

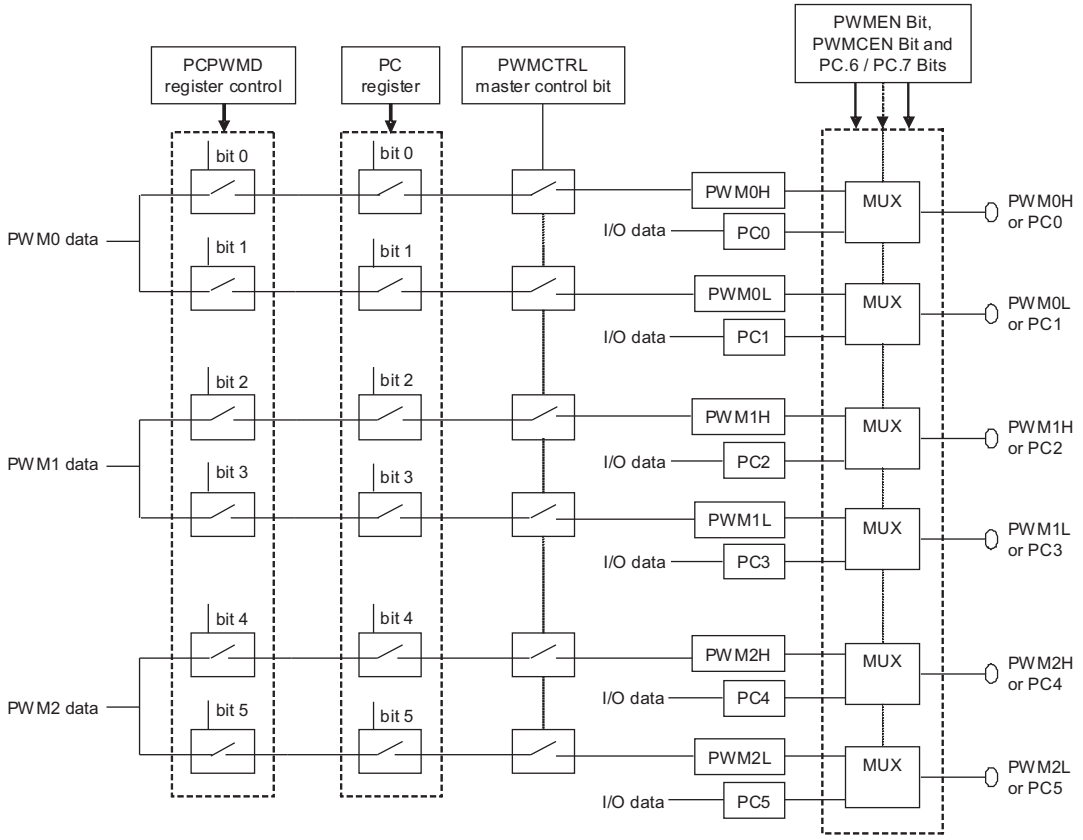
PWMCEN	Control Pins		Pin Function		
	PC7	PC6	PC5	PC3	PC1
0	X	X	I/O	I/O	I/O
1	0	0	I/O	I/O	0: inactive PWM0L 1: active PWM0L
1	0	1	I/O	0: inactive PWM1L 1: active PWM1L	I/O
1	1	0	0: inactive PWM2L 1: active PWM2L	I/O	I/O
1	1	1	0: inactive PWM2L 1: active PWM2L	0: inactive PWM1L 1: active PWM1L	0: inactive PWM0L 1: active PWM0L

Complimentary PWM Pin Output Control

In the above two tables note that the PC data bits PC0~PC5 are used to activate the PWM outputs. Setting these bits to a high level will activate the PWM output while clearing them to zero will set the PWM outputs to their inactive state. Note also that the relevant bits in the PCPWMD register must be set high to activate the PWM output. The port control register PCC must be properly setup to ensure that all the required PWM are setup as outputs.

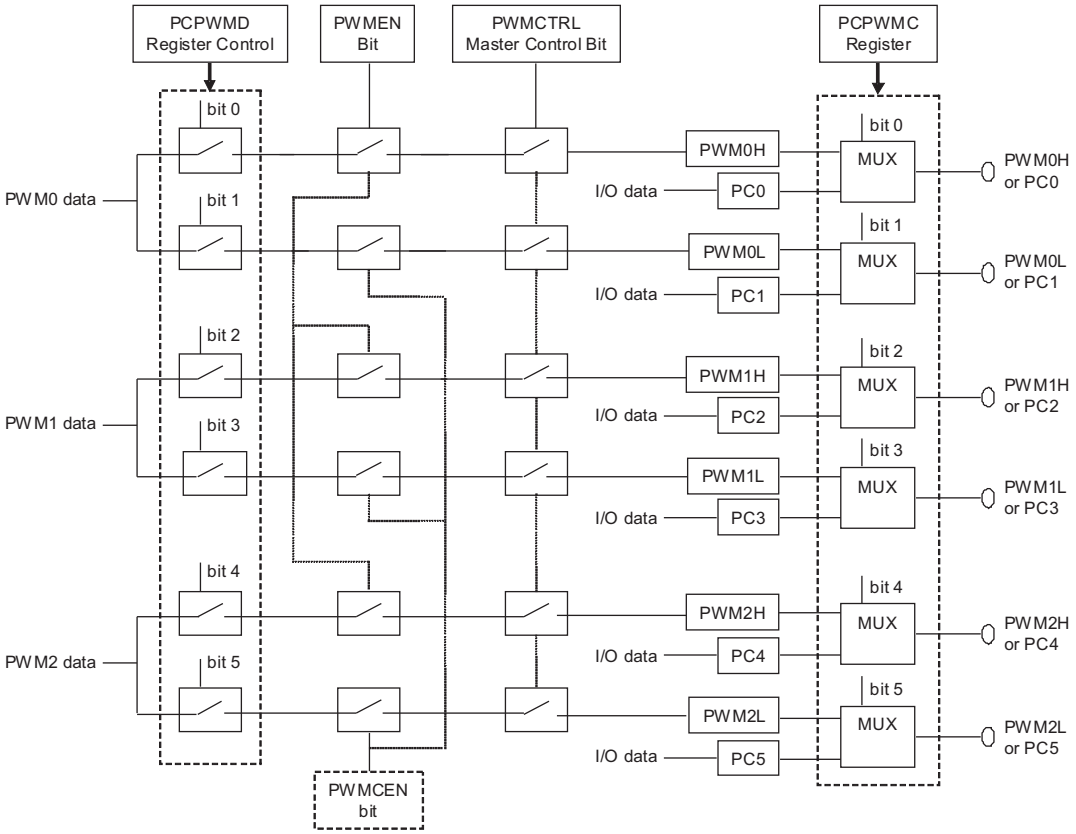
For the situation in the above two tables the following conditions must also be setup to activate the PWM outputs:

PWMCTRL=1, PCPWMD=3FH and PCC=00H.



PWM Output Control for PWMCM = 0

When the PWMCM bit is high, the PCPWMC register will control which pins are PWM output pins and which pins are I/O pins. Bits in the PCPPWMD register are used to activate individual PWM outputs. If these bits are set high then the relevant PWM output will be activated, if the bits are cleared to zero then the relevant PWM output will be set to its inactive state as defined by configuration options.

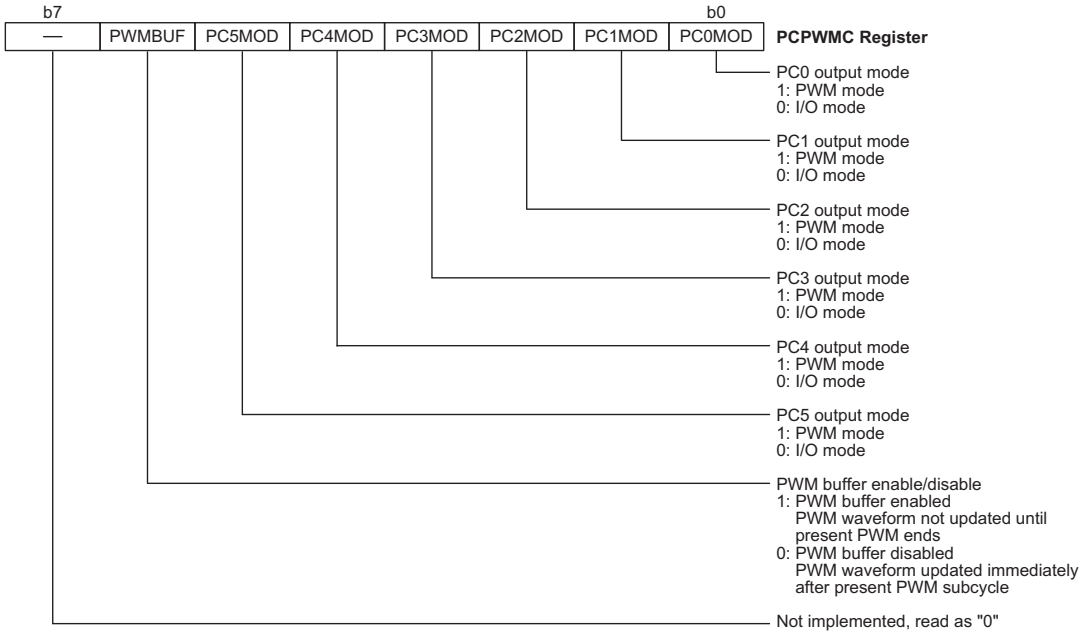


PWM Output Control for PWMCM = 1

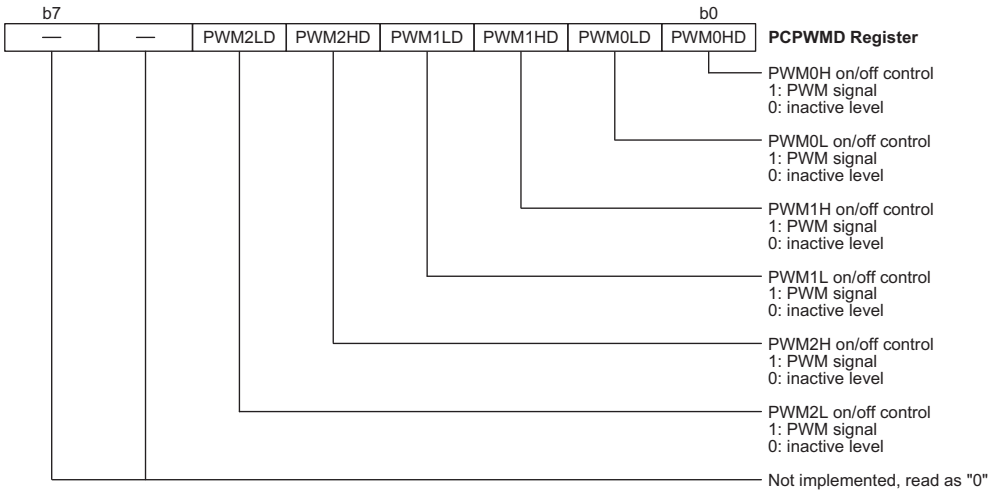
The PWMCTRL bit in the PWMC0 register acts as a PWM master control bit. If this bit is high then all the selected PWM outputs will be active. If the bit is low all the selected PWM outputs will be placed into their inactive condition as determined by configuration options.

The PWMSP0 ~ PWMSP2 bits in the PWMC1 register allow a range of hardware methods to be used to stop the PWM outputs. For some of these methods, after stopping the PWM, recovery is only possible after the PWMCTRL bit is first cleared to zero and then set high again by the application program. Additionally the PWMLEV and PWMCLEV bits in the PWMC1 register, allow the application program to read the status of the two configuration options which determine whether the PWM outputs are active high or active low.

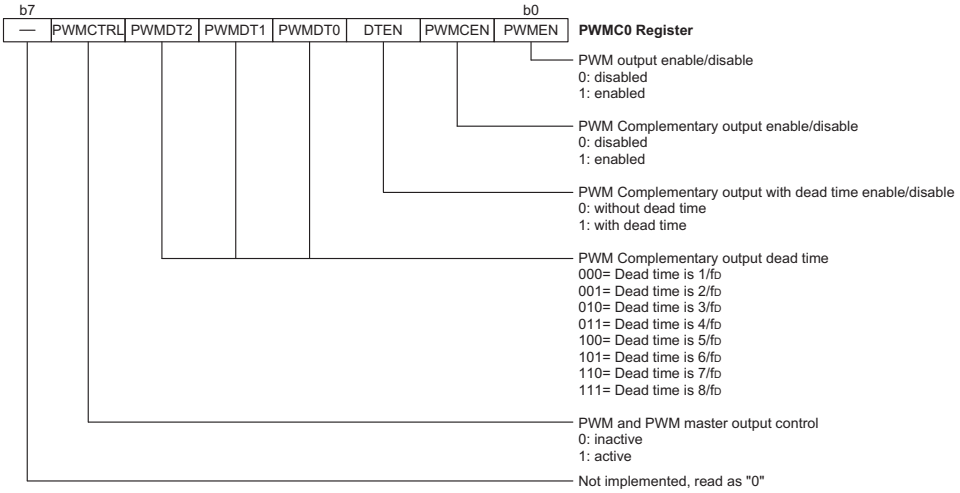
The PWM buffer enable/disabled bit in the PCPWMC register determines how the output is updated when a new duty value is written into the PWM registers. If the PWM buffer is enabled then the output waveform will be updated immediately after the present sub-cycle ends, however if the PWM buffer is disabled then the output waveform will not be updated until the PWM cycle finishes. For example, in the 8+2 mode the sub cycle is 256 clock cycles after which the new PWM value will be reflected in the PWM output waveform if the PWM buffer is enabled.



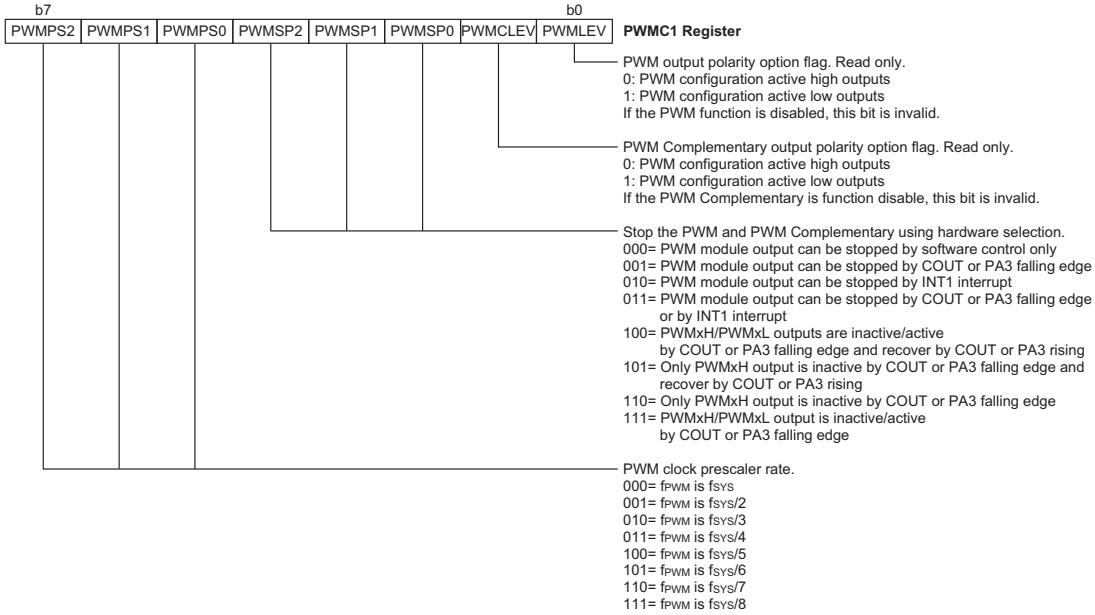
PCPWMC Register



PCPWMD Register

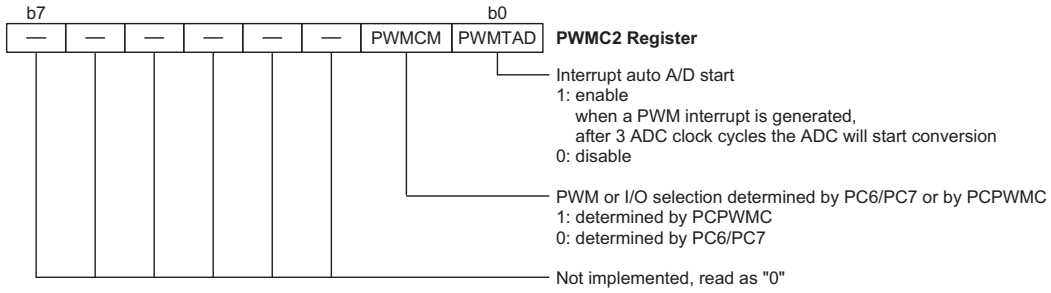


PWMC0 Register

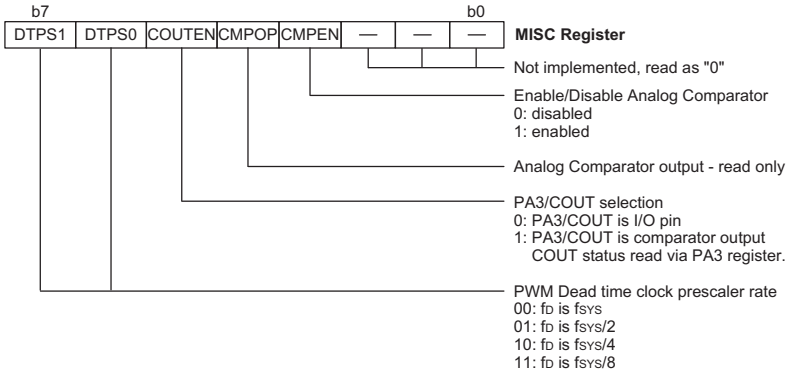


PWMC1 Register

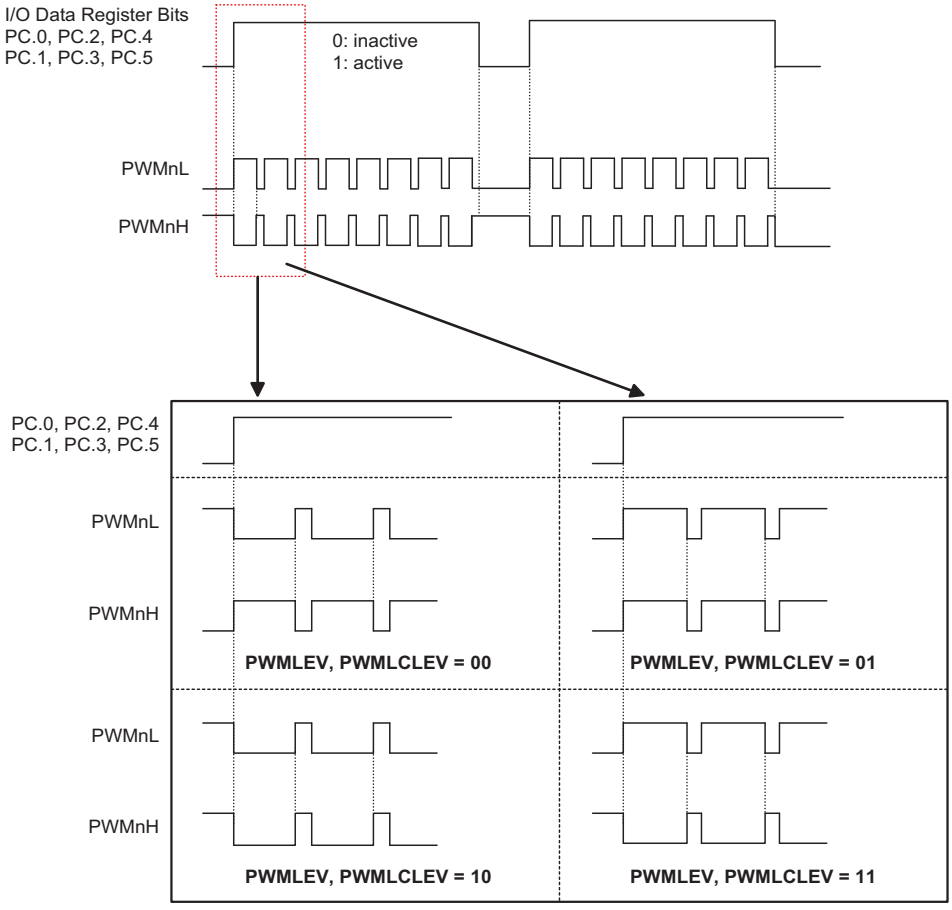
Note: COUT or PA3 is determined by a configuration option. The COUT or PA3 falling edge has a debounce time under software control.



PWMC2 Register



MISC Register



Note: Waveforms do not include dead time, n=0, 1, 2

PWM Output Waveform Control

PWM Dead Time Function

Each PWM output normally drives a pair of push-pull power transistors for load driving. The danger here is that for short periods of time, both both output transistors may be on simultaneously resulting in virtual short circuit conditions across the power supply. To prevent this happening a dead time function is included which ensures that there is a period of time when both output transistors are off when the PWM output changes state.

The dead time can have a value of $1/f_D$, $2/f_D$, $3/f_D$, $4/f_D$, $5/f_D$, $6/f_D$, $7/f_D$ or $8/f_D$ where f_D is $f_{SYS}/1$, $f_{SYS}/2$, $f_{SYS}/4$ or $f_{SYS}/8$. The PWM dead time is determined using the PWMDT2, PWMDT1 and PWMDT0 bits in the PWMC0 register. The DTPS0 and DTPS1 bits in the MISC register select the value for f_D.

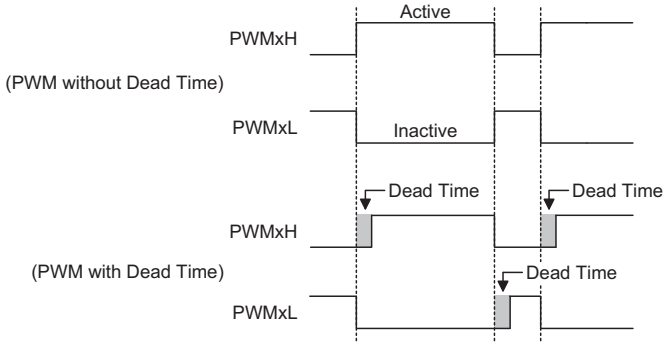
The dead time is a function of the system clock frequency, the DTPSn bits and the PWMDTn bits. By programming suitable values for these bits a wide range of dead times can be chosen. The following equation shows how the dead time can be calculated. The decimal equivalent of the binary bit value should be substituted to calculate the value.

Dead Time value= $(1/f_D)\times((PWMDT2\sim PWMDT0)+1)$

where f_D is determined by the DTPSn bits.

For example:

- f_{SYS}=12MHz
- DTPS1, DTPS0= 11, i.e. $f_D=f_{SYS}/8$
- PWMDT2~PWMDT0 = 101 (decimal 5)
  … gives a dead time of 4μs.



Note: The PWM and Complementary PWM Outputs include Dead Time (Both PWMLEV and PWMCLEV= 0)

PWM Dead Time Timing

The accompanying table shows a range of already calculated dead time values for user reference.

PWMDT2~ PWMDT0	f_{SYS}=12MHz $f_D=f_{SYS}$	f_{SYS}=16MHz $f_D=f_{SYS}$	f_{SYS}=20MHz $f_D=f_{SYS}$	Unit
000	0.08	0.06	0.05	μs
001	0.17	0.13	0.10	μs
010	0.25	0.19	0.15	μs
011	0.33	0.25	0.20	μs
100	0.42	0.31	0.25	μs
101	0.50	0.38	0.30	μs
110	0.58	0.44	0.35	μs
111	0.67	0.50	0.40	μs

PWMDT2~ PWMDT0	f_{SYS}=12MHz $f_D=f_{SYS}/2$	f_{SYS}=16MHz $f_D=f_{SYS}/2$	f_{SYS}=20MHz $f_D=f_{SYS}/2$	Unit
000	0.17	0.13	0.10	μs
001	0.33	0.25	0.20	μs
010	0.50	0.38	0.30	μs
011	0.67	0.50	0.40	μs
100	0.83	0.63	0.50	μs
101	1.00	0.75	0.60	μs
110	1.17	0.88	0.70	μs
111	1.33	1.00	0.80	μs

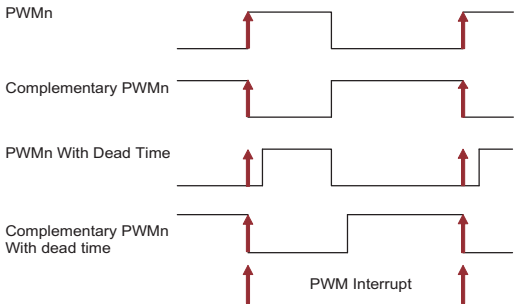
PWMDT2~ PWMDT0	f_{SYS}=12MHz $f_D=f_{SYS}/4$	f_{SYS}=16MHz $f_D=f_{SYS}/4$	f_{SYS}=20MHz $f_D=f_{SYS}/4$	Unit
000	0.33	0.25	0.20	μs
001	0.67	0.50	0.40	μs
010	1.00	0.75	0.60	μs
011	1.33	1.00	0.80	μs
100	1.67	1.25	1.00	μs
101	2.00	1.50	1.20	μs
110	2.33	1.75	1.40	μs
111	2.67	2.00	1.60	μs

PWMDT2~ PWMDT0	f_{SYS}=12MHz $f_D=f_{SYS}/8$	f_{SYS}=16MHz $f_D=f_{SYS}/8$	f_{SYS}=20MHz $f_D=f_{SYS}/8$	Unit
000	0.67	0.50	0.40	μs
001	1.33	1.00	0.80	μs
010	2.00	1.50	1.20	μs
011	2.67	2.00	1.60	μs
100	3.33	2.50	2.00	μs
101	4.00	3.00	2.40	μs
110	4.67	3.50	2.80	μs
111	5.33	4.00	3.20	μs

Dead Time Values

PWM Interrupt

Each time the PWM counter overflows, a PWM interrupt request will be generated. Whether an actual interrupt is generated can be determined by enabling or disabling the EPWMI bit in the INTC1 register. The interrupt request is generated on the leading edge of the PWM signal. The PWMTAD bit in the PWMC2 register, if set, will automatically initiate an A/D converter conversion cycle when a PWM interrupt is generated.

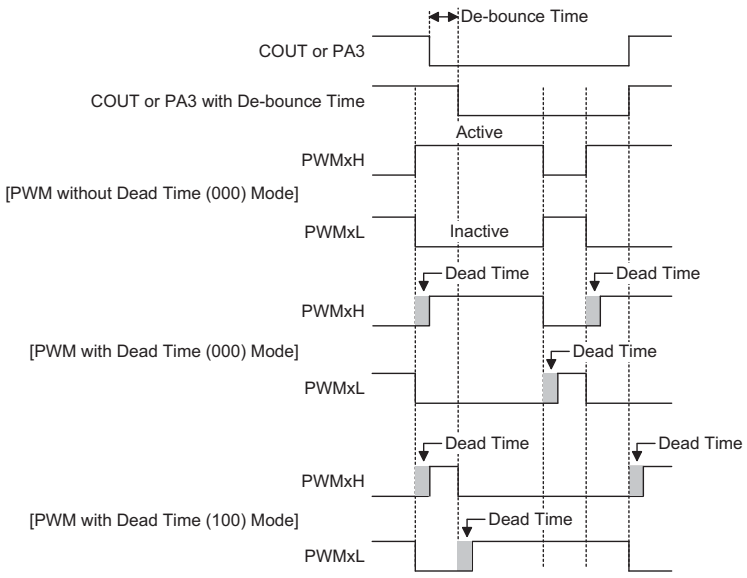


PWM Interrupt Generation

PWM Configuration Options

There are three configuration options associated with the PWM function. One is to determine the modulation type of the PWM waveform which can be chosen to be either 9+1, 8+2 or 7+3. The other two options determine if the PWM outputs are active high or active low, one of these options controls the three non-complementary outputs while the other controls the three complementary outputs. The PWMLEV and PWMCLEV read-only bits in the PWMC1 register reflect the condition of these two configuration option bits and can be read by the application program to indicate the active high or active low condition of the PWM pins.

Configuration Option	Bit	Description
PWM Output Level Selection	PWMLEV	This option determines if the PWM output is Active Low or Active High. If the bit is read as zero then the PWM output has been defined as active high. If the bit is read as high then the PWM output has been defined as active low. If the PWM function is disabled then this bit is invalid.
PWM Complementary Output Level Selection	PWMCLEV	This option determines if the PWM complimentary outputs are Active Low or Active High.If the bit is read as zero then the PWM complementary outputs have been defined as active high. If the bit is read as high then the PWM complementary outputs have been defined as active low.f the PWM function is disabled then this bit is invalid.



Note: PWMxH and PWMxL configuration option is selected active high for example.

Hardware Stop PWM Mode (PWMSP2, PWMSP1, PWMSP0= 1, 0, 0)

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 devices contains a 8-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 9-bit digital value.

Input Channels	Conversion Bits	Input Pins
8	12	PB0~PB7

The following 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 have a 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. For devices which use two A/D Converter Data Registers, note that 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 lower four bits of the 12-bit converted value.

In the following tables, D0~D11 are 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 Register

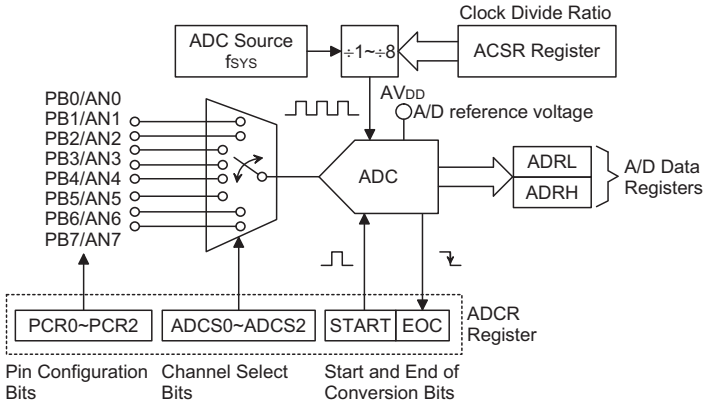
A/D Converter Control Register – ADCR

To control the function and operation of the A/D converter, a control register known as ADCR is provided. This 8-bit register defines 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 as well as controlling the start function and monitoring the A/D converter end of conversion status.

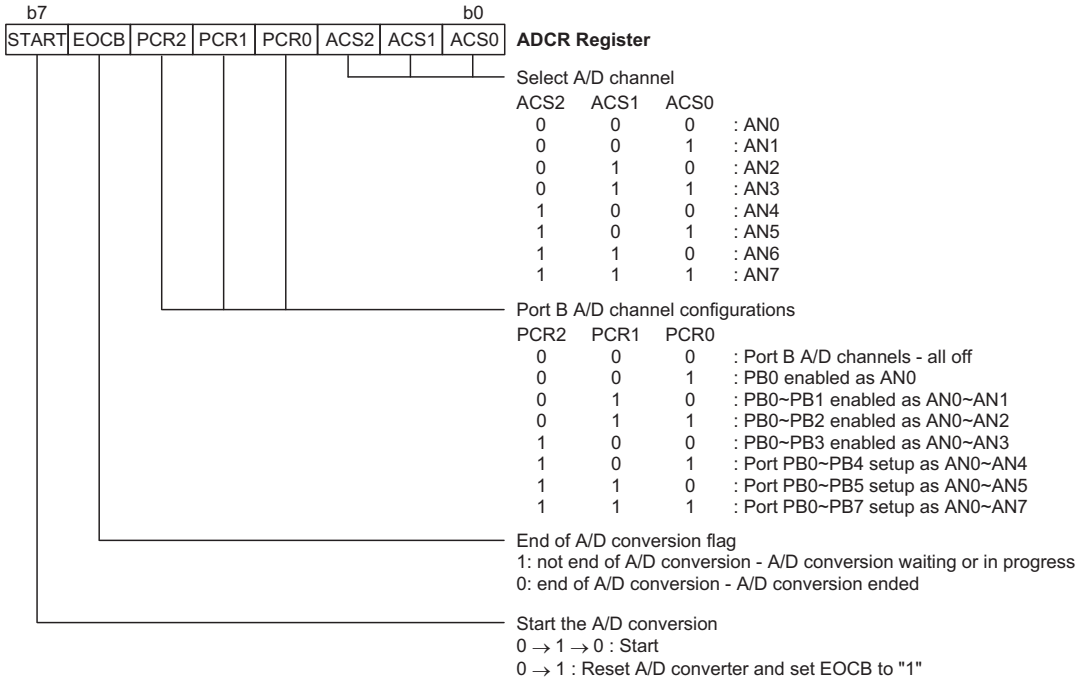
One section of this register contains the bits ACS2~ACS0 which define the channel number. As each of the devices contains only one actual analog to digital converter circuit, each of the individual 8 analog inputs must be routed to the converter. It is the function of the ACS2~ACS0 bits in the ADCR register 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 A 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 ADCR 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 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



A/D Converter Structure



ADCR Register

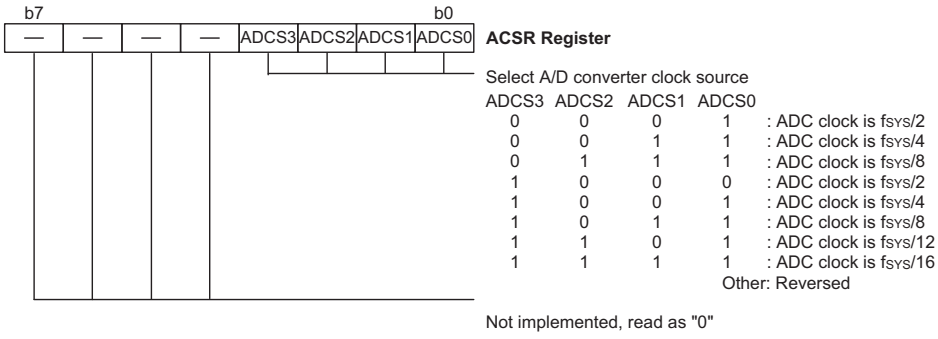
START bit that is used to control the overall on/off operation of the internal analog to digital converter.

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.

A/D Converter Clock Source Register – ACSR

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 bits ADCS0 to ADCS2 in the ACSR register.

Although the A/D clock source is determined by the system clock f_{SYS}, and by bits ADCS0 to ADCS2, 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. Doing so will give an A/D clock period less than the minimum A/D clock period which may result in inaccurate A/D conversion values. Refer to the table for examples, where values marked with an asterisk * show where special care must be taken, as the values are less than the specified minimum A/D Clock Period.



ACSR Register

A/D Clock Period (t_{AD})								
ADCS2	ADCS1	ADSC0	f_{SYS}=1MHz	f_{SYS}=2MHz	f_{SYS}=4MHz	f_{SYS}=8MHz	f_{SYS}=12MHz	Unit
0	0	0	1.00	0.50	*0.25	*0.125	*0.083	μs
0	0	1	2.00	1.00	0.50	*0.25	*0.167	μs
0	1	0	3.00	1.50	0.75	*0.375	*0.25	μs
0	1	1	4.00	2.00	1.00	0.500	*0.333	μs
1	0	0	5.00	2.50	1.25	0.625	*0.417	μs
1	0	1	6.00	3.00	1.50	0.750	0.500	μs
1	1	0	7.00	3.50	1.75	0.875	0.583	μs
1	1	1	8.00	4.00	2.00	1.000	0.667	μs

A/D Clock Period Example

A/D Conversion Time (t_{ADC})								
ADCS2	ADCS1	ADSC0	f_{SYS}=1MHz	f_{SYS}=2MHz	f_{SYS}=4MHz	f_{SYS}=8MHz	f_{SYS}=12MHz	Unit
0	0	0	16.00	8.00	*4.00	*2.00	*1.328	μs
0	0	1	32.00	16.00	8.00	*4.00	*2.672	μs
0	1	0	48.00	24.00	12.00	*6.00	*4.00	μs
0	1	1	64.00	32.00	16.00	8.00	*5.328	μs
1	0	0	80.00	40.00	20.00	10.00	*6.672	μs
1	0	1	96.00	48.00	24.00	12.00	8.00	μs
1	1	0	112.00	56.00	28.00	14.00	9.33	μs
1	1	1	128.00	64.00	32.00	16.00	10.67	μs

A/D Conversion Time Example

f_{SYS}	A/D Clock	ACSR	Division
12MHz	*3MHz	03H	f_{SYS}/4
	1.5MHz	07H	f_{SYS}/\8
	1.5MHz	0BH	f_{SYS}/8
	750kHz	0FH	f_{SYS}/16
16MHz	*4MHz	09H	f_{SYS}/4
	2MHz	07H	f_{SYS}/8
	2MHz	0BH	f_{SYS}/8
	1MHz	0FH	f_{SYS}/16
20MHz	*2.5MHz	07H	f_{SYS}/8
	*2.5MHz	0BH	f_{SYS}/8
	1.67MHz	0DH	f_{SYS}/12
	1.25MHz	0FH	f_{SYS}/16

A/D Clock Frequency

A/D Input Pins

All of the A/D analog input pins are pin-shared with the I/O pins on Port B. Bits PCR0~PCR2 in the ADCR register, not configuration options, 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 via configuration option, 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, when the PCR2~PCR0 bits enable an A/D input, the status of the port control register will be overridden. The AVDD power supply pin is used as the A/D converter reference voltage, and as such analog inputs must not be allowed to exceed this value. Appropriate measures should also be taken to ensure that the AVDD pin remains as stable and noise free as possible.

Initialising the A/D Converter

The internal A/D converter must be initialised in a special way. Each time the 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.

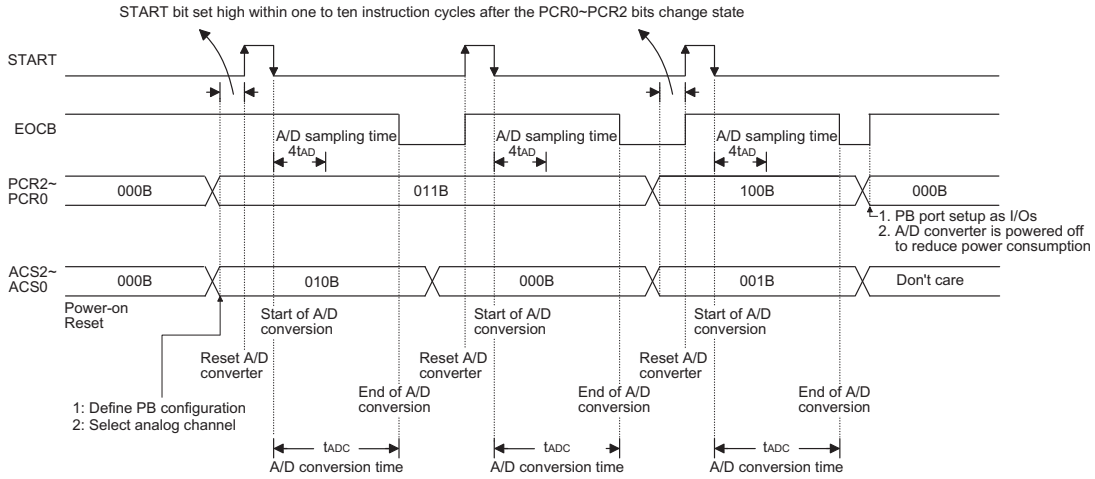
Summary of A/D Conversion Steps

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 ADCS0 to ADCS20 in the ACSR register.
- Step 2
  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 PCR0~PCR2 bits in the ADCR register.
- Step 3
  Select which channel is to be connected to the internal A/D converter by correctly programming the ACS0~ACS2 bits which are also contained in the ADCR register. Note that this step can be combined with Step 2 into a single ADCR register programming operation.
- Step 4
  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 interrupt control register must be set high and the A/D converter interrupt bit, EADI, in the interrupt control register must also be set high.
- Step 5
  The analog to digital conversion process can now be initialised by setting the START bit in the ADCR register from low to high and then low again. Note that this bit should have been originally set to a zero value.
- Step 6
  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 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 following timing diagram shows graphically the various stages involved in an analog to digital conversion process and its associated timing.



Note: A/D clock must be fSYS/1, fSYS/2, fSYS/3, fSYS/4, fSYS/5, fSYS/6, fSYS/7 or fSYS/8

A/D Conversion 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 16 t_{AD} .

Programming Considerations

When programming, special attention must be given to the A/D channel selection bits in the ADCR 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.

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,00000111B
            mov     ACSR,a              ; setup the ACSR register to select fSYS/8 as
                                        ; the A/D clock
            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 A channel bits have changed the
                                        ; following START
                                        ; signal (0-1-0) must be issued within 10
                                        ; 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 conversion result value from the ADR
                                        ; register
            mov     adrl_buffer,a       ; save result to user defined memory
            mov     a,ADRH
            mov     adrh_buffer,a
                    :
                    :
            jmp     start_conversion    ; start next A/D conversion
```

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

```
            clr     EADI                ; disable ADC interrupt
            mov     a,00000111B
            mov     ACSR,a              ; setup the ACSR register to select fSYS/8 as
                                        ; the A/D clock

            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 A channel bits have changed the
                                        ; following START
                                        ; signal (0-1-0) must be issued within 10
                                        ; instruction cycles
                    :
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     EMI                 ; enable global interrupt
                    :
                    :
                    :
; ADC interrupt service routine
ADC_ISR:
            mov     acc_stack,a         ; save ACC to user defined memory
            mov     a,STATUS
            mov     status_stack,a      ; save STATUS to user defined memory
                    :
                    :
            mov     a,ADRL              ; read conversion result value from the ADR
                                        ; register
            mov     adrl_buffer,a       ; save result to user defined register
            mov     a,ADRH
            mov     adrh_buffer,a
                    :
                    :
EXIT_INT_ISR:
            mov     a,status_stack
            mov     STATUS,a            ; restore STATUS from user defined memory
            mov     a,acc_stack         ; restore ACC from user defined memory
            reti
```

A/D Transfer Function

As the device contain an 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 converters.

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.

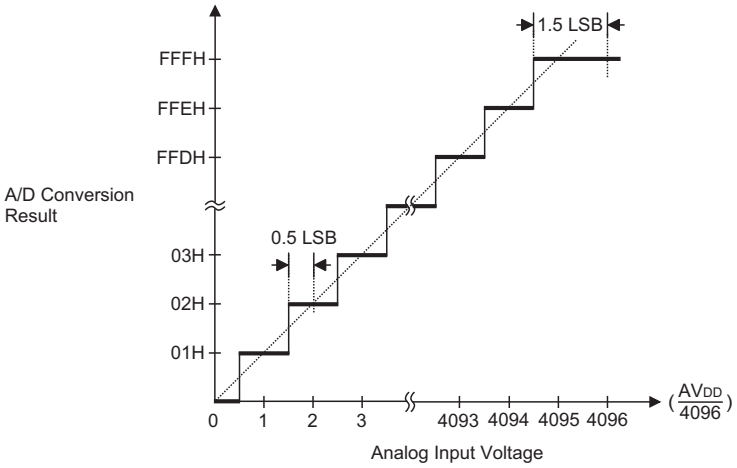
Analog Comparator

The device contains a single fully integrated Analog Comparator function.

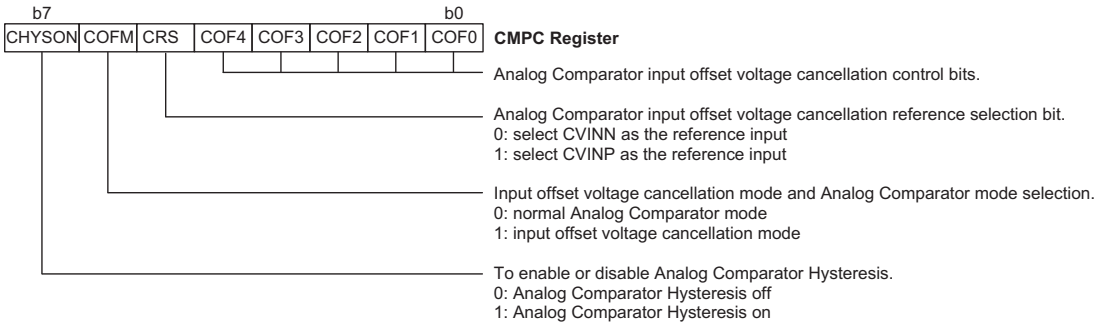
Comparator Operation

The CMPEN bit in the MISC register is used as the enable/disable bit for the Analog Comparator. If the CMPEN bit is cleared to "0", the Analog Comparator is disabled and the PA1/CVINP, PA2/CVINN and PA3/COUT shared function pins can be used as normal I/O pins. If CMPEN is set to "1", the Analog Comparator is enabled and the PA1/CVINP and PA2/CVINN pins will be setup as Analog Comparator input pins and PA3/COUT setup as the Analog Comparator output pin.

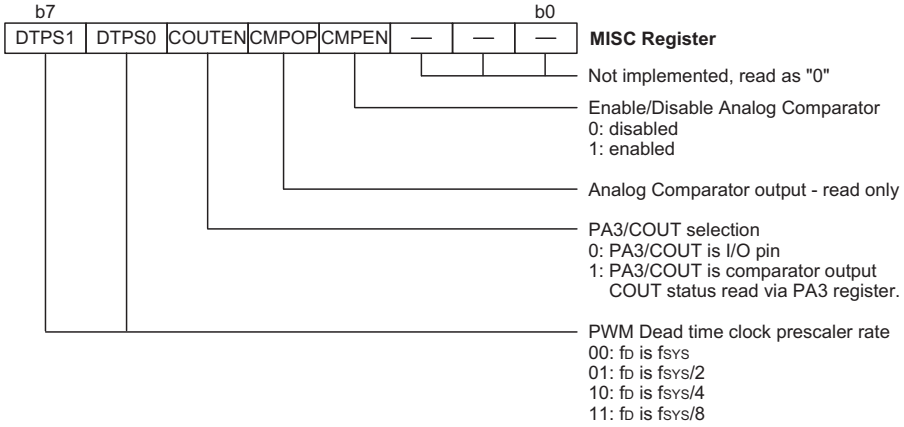
As the CVINP, CVINN and COUT are pin-shared with PA1, PA2 and PA3, once the Analog Comparator function is enabled, the internal registers related to PA1 and PA2 cannot be used, however PA3 can still be used as an input, if the COUTEN bit in the MISC register is set to "1". When COUTEN is set to "1", any pull high resistor options on the pins will be disabled and the PA3 output function will also be disabled. Software instructions fully determine how the Analog Comparator function is to be used.



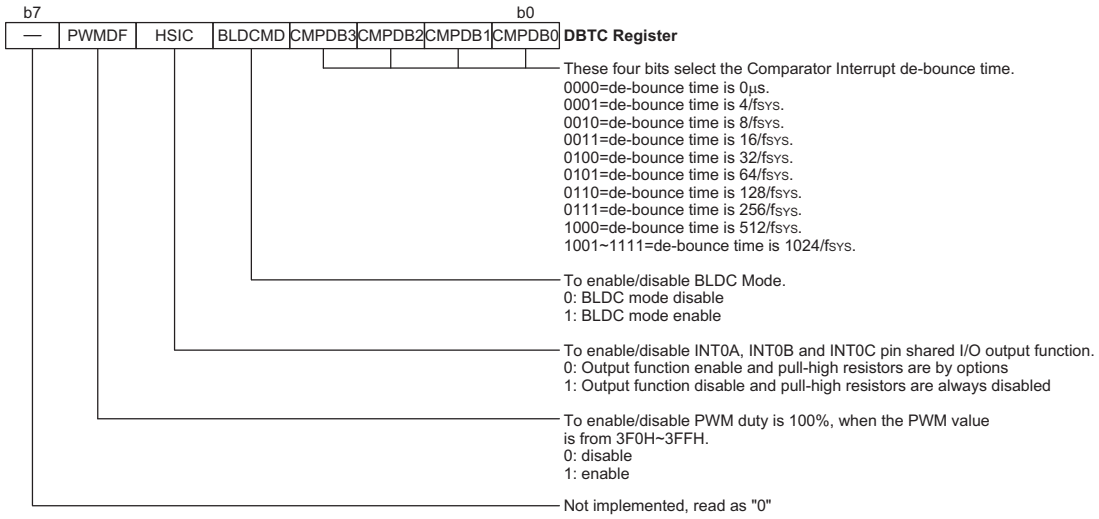
Ideal A/D Transfer Function



CMPC Register



MISC Register



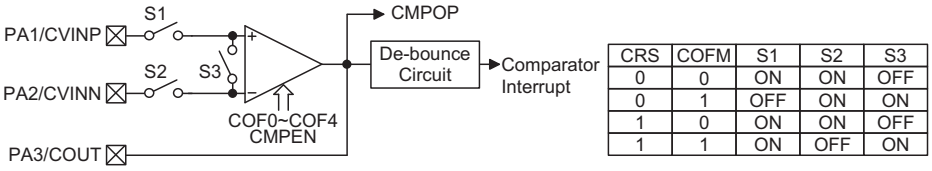
DBTC Register

Note: If PA3/OCUT is selected with a de-bounce time function, then it will not wake up from the power down mode.

CMPEN	COUTEN	PA1, PA2, PA3
0	X	PA1, PA2, PA3 are I/O pins
1	0	PA1, PA2 are Comparator inputs and PA3 is an I/O
1	1	PA1, PA2 are Comparator inputs and PA3 is a Comparator output. PA3 can read the Comparator output status.

The comparator also includes a de-bounce time function. The de-bounce time is from 0, $4/f_{SYS}$, $8/f_{SYS}$, ...to $1024/f_{SYS}$.

COUTEN	PA3/COUT selection 0: PA3/COUT is as an I/O 1: PA3/COUT is a Comparator output, and the status of COUT can be read by reading the PA3 register.



CRS	COFM	S1	S2	S3
0	0	ON	ON	OFF
0	1	OFF	ON	ON
1	0	ON	ON	OFF
1	1	ON	OFF	ON

Analog Comparator Block Diagram

The Analog Comparator can be used as one of the methods of stopping the PWM function. This is implemented using the PWMSP0, PWMSP1 and PWMSP2 bits. The PWM is stopped by clearing the PWMCTRL bit to ″0″.

Comparator Offset Voltage

The Analog Comparator allows its input offset voltage to be adjusted using a common mode input to calibrate the offset value.

The calibration steps are as follows:

- Step1. Set COFM = 1 to select the offset voltage cancellation mode – here S3 is closed.
- Step2. Set CRS = 1 or 0 to select which input pin is the voltage – S1 or S2 is closed
- Step3. Adjust COF0~COF4 until the output status changes
- Step4. Set COFM = 0 to select the normal comparator mode

Operation Amplifier - OPA

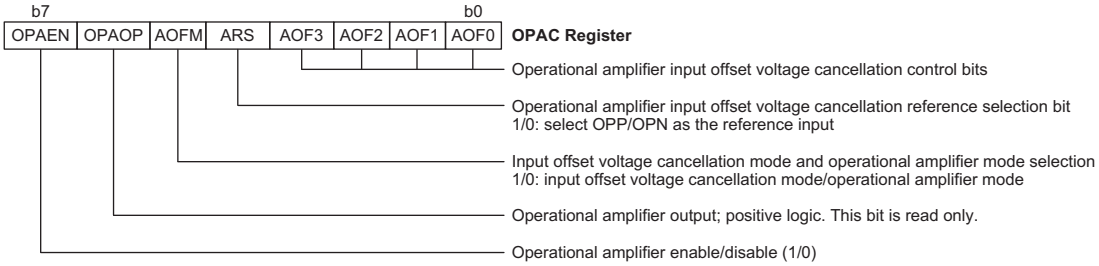
The device contains a fully integrated operational amplifier.

OPA Operation

The OPAEN bit in the OPAC register is used as the enable/disable bit for the Operational Amplifier. If the OPAEN bit is cleared to ″0″, the OPA is disabled and powered off to reduce power consumption. The PB2/AN2/OPOUT, PB3/AN3/OPVINN and PA0/OPVINP pins can all be used as I/O pins. If the OPAEN bit is set to ″1″, the OPA is enabled. Now PB3/AN3/OPVINN and PA0/OPVINP are OPA inverting and non-inverting input pins, and PB2/AN2/OPOUT is the OPA output pin. Any internal pull high resistors connected to, PB2, PB3 and PA0 output will be disabled.

As the OPVINP, OPVINN and OPOUT are pin-shared with PA0, PB3/AN3 and PB2/AN2, once the OPA function is enabled, the internal registers related to PA0, PB3 and PB2 cannot be used and the I/O function and pull-high resistor are disabled automatically. Software instructions fully determine how the OPA is to be used.

OPAEN	PA0, PB3/AN3, PB2/AN2
0	PA0, PB2/AN2 and PB3/AN3 are I/Os or analog ADC inputs.
1	PA0, PB3/AN3 are OPA input pins and PB2/AN2 is an OPA output pin. PB2/AN2 and PB3/AN3 can be analog ADC inputs if the related ADC function is enabled.



OPAC Register (Operational Amplifier Control Register)

Interrupts

Operational amplifier enable/disable (1/0)Interrupts are an important part of any system. When an external event or an internal function such as a Timer/Event Counter, Analog Comparator or ADC 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. This device contains two external interrupts and five internal interrupts functions. The external interrupt is controlled by the action of the external INT0A, INT0B, INT0C or INT1 pins, while the internal interrupts are controlled by the two Timer/Event Counter overflows, the PWM interrupt, the Analog Comparator interrupt and the A/D converter interrupt.

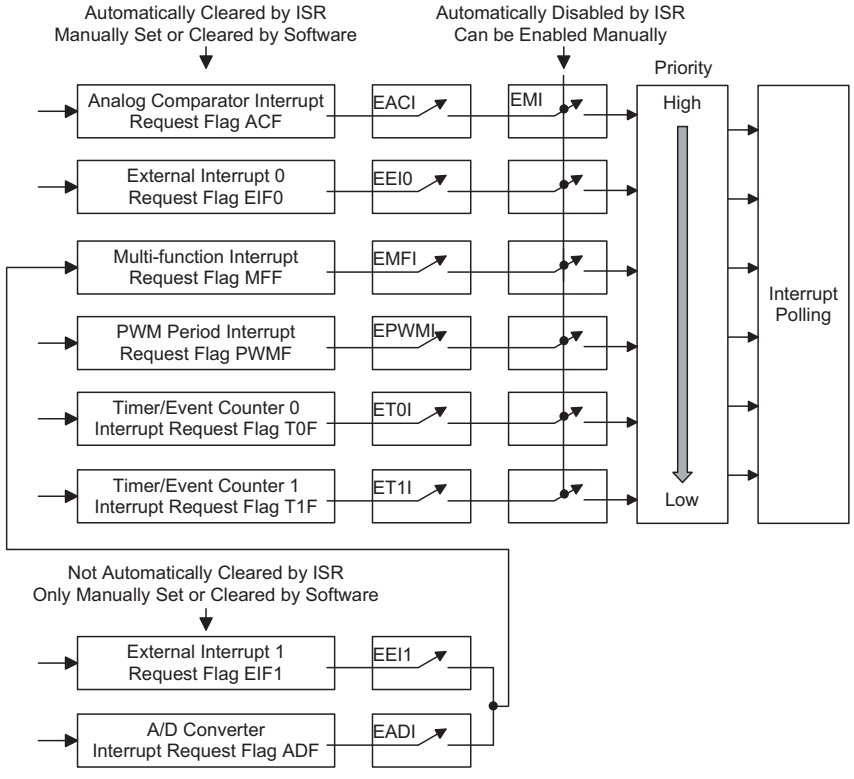
Interrupt Registers

Overall interrupt control, which means interrupt enabling and request flag setting, is controlled by the INTC0, INTC1 and MFIC registers, which are located in the Data Memory. By controlling the appropriate enable bits in these registers each individual interrupt can be enabled or disabled. Also when an interrupt occurs, the corresponding request flag will be set by the microcontroller. The global enable flag if cleared to zero will disable all interrupts.

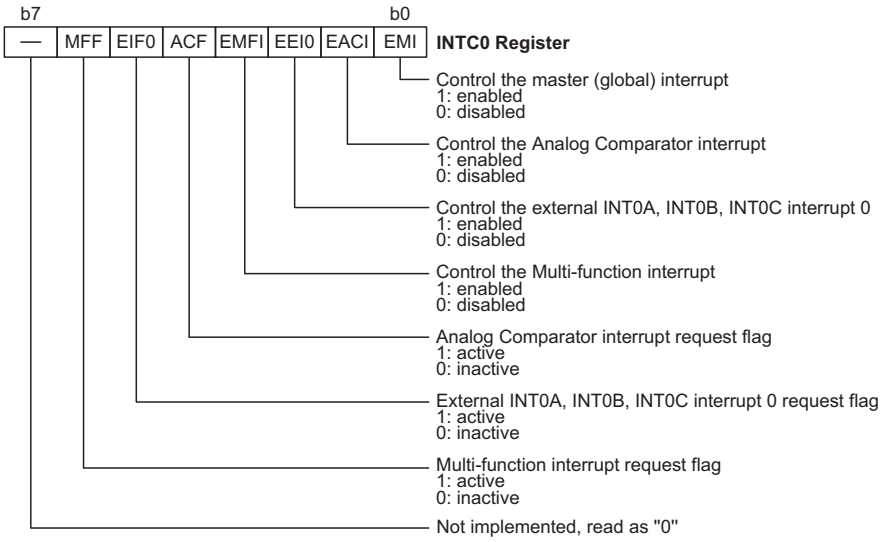
Interrupt Operation

Two external interrupt, one internal 8-bit timer/event counter interrupt, one internal 16-bit timer/event counter interrupt, one Analog Comparator interrupt, one PWM interrupt or one A/D converter interrupt will all generate an interrupt request by setting their corresponding request flag, if their appropriate interrupt enable bit is set. When this happens, the Program Counter, which stores the address of the next instruction to be executed, will be transferred onto the stack. The Program Counter will then be loaded with a new address which will be the value of the corresponding interrupt vector. The microcontroller will then fetch its next instruction from this interrupt vector. The instruction at this vector will usually be a JMP statement which will jump to another section of program which is known as the interrupt service routine. Here is located the code to control the appropriate interrupt. The interrupt service routine must be terminated with a RETI statement, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

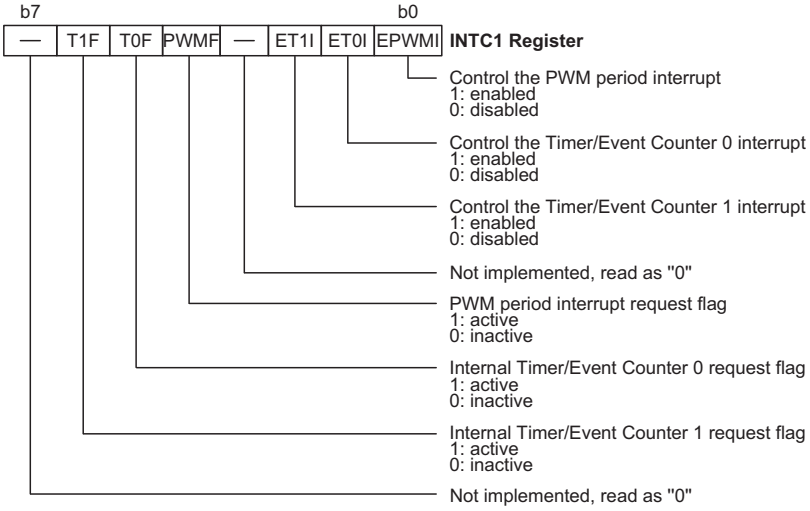
The various interrupt enable bits, together with their associated request flags, are shown in the following diagram with their order of priority.



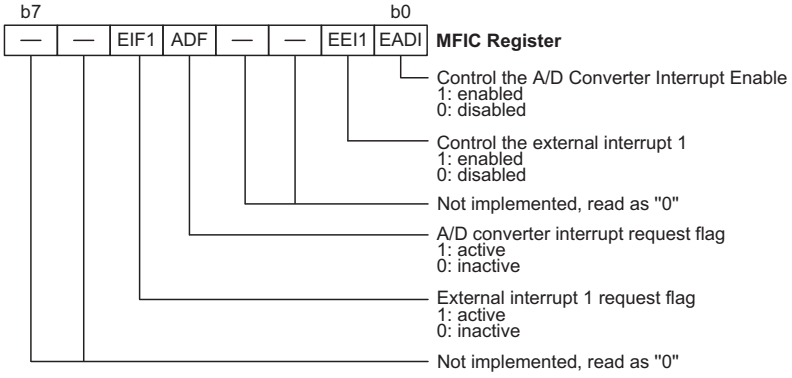
Interrupt Structure



INTC 0 Register



INTC 1 Register



MFIC Register

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

Interrupt Priority

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

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

In cases where both external and internal timer interrupts are enabled and where an external and internal timer interrupt occurs simultaneously, the external interrupt will always have priority and will therefore be serviced first. Suitable masking of the individual interrupts using the INTC0, INTC1 and MFIC register can prevent simultaneous occurrences.

External Interrupt 0

For an External Interrupt 0 to occur, the global interrupt enable bit, EMI, and External Interrupt 0 enable bit, EEIO, in the INTC0 register, must first be set. An actual External Interrupt 0 will take place when the External Interrupt 0 request flag, EIF0, is set, a situation that will occur when a high to low transition appears on either the INT0A, INT0B or INT0C pins. These three external interrupt pins are pin-shared with the I/O pins, PA4, PA5 and PA6 or PB4, PB5 and PB6. The choice of which three pins are used is determined by configuration options. The pins can only be configured as external interrupt pins if the External Interrupt 0 enable bit in the INTC0 register has been set. The pins must also be setup as inputs by setting the corresponding bits in the appropriate port control register. When the interrupt is enabled, the stack is not full and a high to low transition occurs on any the INT0A, INT0B or INT0C pins, a subroutine call to the External Interrupt 0 interrupt vector at location 08H will take place. When the interrupt is serviced, the External Interrupt 0 request flag, EIF0, will be automatically reset and the EMI bit will automatically cleared to disable other interrupts. Note that any pull-high resistor configuration options on these pins will remain valid even if the pins are used as external interrupts.

Timer/Event Counter Interrupt

For a Timer/Event Counter interrupt to occur, the global interrupt enable bit, EMI, and the corresponding Timer/Event Counter interrupt enable bit, ET0I and ET1I in the INTC1 register, must first be set. An actual Timer/Event Counter interrupt will take place when the corresponding Timer/Event Counter interrupt request flag, T0F or T1F in the INTC1 register, 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/Event Counter 0 interrupt vector at location 14H or to the Timer/Event Counter 1 interrupt vector at location 18H will take place. When the interrupt is serviced, the Timer/Event Counter interrupt request flag, T0F or T1F, will be automatically reset and the EMI bit will be automatically cleared to disabled other interrupts.

Analog Comparator Interrupt

For an Analog Comparator Interrupt to occur, the global interrupt enable bit, EMI, and the corresponding analog comparator interrupt enable bit, EACI in the INTC0 register must first be set. An actual Analog Comparator Interrupt will take place when the Analog Comparator request flag, ACF in the INTC0 register, is set, a situation that will occur when a falling edge appears on the comparator output. When the interrupt is enabled, the stack is not full and a falling edge occurs on the comparator output, a subroutine call to the Analog Comparator Interrupt vector at location 04H will take place. When the interrupt is serviced, the Analog Comparator request flag, ACF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

PWM Interrupt

For a PWM Interrupt to occur, the global interrupt enable bit, EMI, and the corresponding PWM interrupt enable bit, EPWMI in the INTC1 register must first be set. An actual PWM Interrupt will take place when the PWM request flag, PWMF in the INTC1 register, is set, a situation that will occur when the PWMxH is from inactive to active. When the interrupt is enabled, the stack is not full and a PWM interrupt occurs, a subroutine call to the PWM Interrupt vector at location 10H will take place. When the interrupt is serviced, the PWM interrupt request flag, PWMF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

Multi-Function Interrupt

The Multi-Function Interrupt handles the interrupt vector for both the A/D converter interrupt and the External Interrupt 1 as these two functions do not have their own independent interrupt vectors. For a Multi-Function Interrupt to occur, the global interrupt enable bit, EMI, and the corresponding Multi-Function Interrupt enable bit, EMFI in the INTC0 register, must first be set. Additionally the A/D Converter Interrupt enable bit, EADI in the MFIC register, and/or the External Interrupt 1 enable bit, EIF1 in the MFIC register, must also be set. An actual Multi-Function Interrupt will take place when the Multi-Function Interrupt request flag, MFF in the INTC1 register, is set. This situation will occur when either the external interrupt pin INT1 experiences an active edge or if the A/D conversion process has completed, resulting in their corresponding interrupt request flags, namely the EIF1 or ADF flags in the MFIC register being set. When the interrupt is enabled, the stack is not full and either an active edge appears on the INT1 pin or the A/D converter process completes, then a subroutine call to the Multi-Function Interrupt vector at location 0CH will take place. When the interrupt is serviced, the Multi-Function Interrupt request flag, MFF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Examination of the External Interrupt 1 and A/D converter request flags will reveal which function was behind the Multi-Function Interrupt. Note that the External Interrupt 1 and A/D converter request flags are not reset automatically when the interrupt is serviced, and have to be reset manually after a Multi-Function Interrupt occurs.

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 or MFIC register until the corresponding interrupt is serviced or until the request flag is cleared by a software instruction.

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

All of these interrupts have the capability of waking up the processor when in the Power Down Mode. Only the Program Counter is pushed onto the stack. If the contents of the register or status register are altered by the interrupt service program, which may corrupt the desired control sequence, then the contents should be saved in advance.

Reset and Initialisation

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

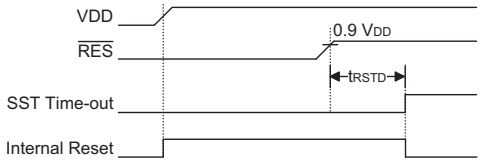
In addition to the power-on reset, situations may arise where it is necessary to forcefully apply a reset condition when the microcontroller is running. One example of this is where after power has been applied and the microcontroller is already running, the $\overline{\text{RES}}$ line is forcefully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

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

Reset Functions

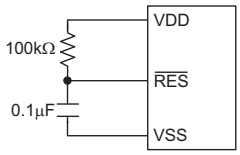
There are five ways in which a microcontroller reset can occur, through events occurring both internally and externally:

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



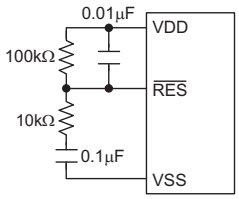
Power-On Reset Timing Chart

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



Basic Reset Circuit

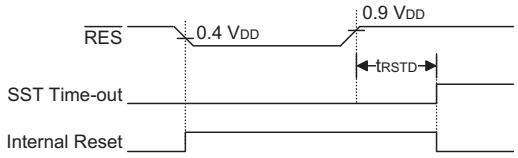
For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.



Enhanced Reset Circuit

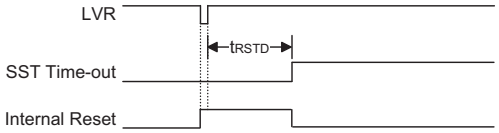
More information regarding external reset circuits is located in Application Note HA0075E on the Holtek website.

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



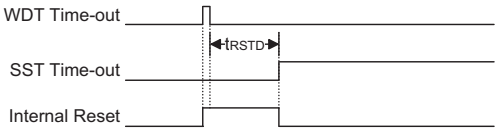
$\overline{RES}$ Reset Timing Chart

- 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 t_{LVR}, the LVR will ignore it and will not perform a reset function.



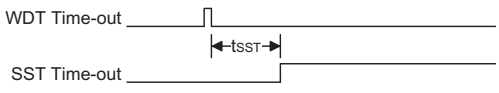
Low Voltage Reset Timing Chart

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



WDT Time-out Reset during Normal Operation Timing Chart

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



WDT Time-out Reset during Power Down Timing Chart

Reset Initial Conditions

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

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

Note: "u" stands for unchanged

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

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

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

Register	Reset (Power-on)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset Normal Operation	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
MP0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
MP1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu	-uuu uuuu
WDTS	0000 0111	0000 0111	0000 0111	0000 0111	uuuu uuuu
STATUS	--00 xxxx	--1u uuuu	--uu uuuu	--01 uuuu	--11 uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
TMR0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR0C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
TMR1H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR1L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
TMR1C	00-0 1000	00-0 1000	00-0 1000	00-0 1000	uu-u uuuu
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PB	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PBC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PC	0011 1111	0011 1111	0011 1111	0011 1111	uuuu uuuu
PCC	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu
PD	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
PDC	---- 1111	---- 1111	---- 1111	---- 1111	---- uuuu
PWM0H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWM0L	---- --00	---- --00	---- --00	---- --00	---- --uu
PWM1H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWM1L	---- --00	---- --00	---- --00	---- --00	---- --uu
PWM2H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWM2L	---- --00	---- --00	---- --00	---- --00	---- --uu
PWMC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
PWMC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PWMC2	---- --00	---- --00	---- --00	---- --00	---- --uu
PCPWMC	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
PCPWMD	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
LVDCTL	--00 -000	--00 -000	--00 -000	--00 -000	--uu -uuu
INTC1	-000 -000	-000 -000	-000 -000	-000 -000	-uuu -uuu
MFIC	--00 --00	--00 --00	--00 --00	--00 --00	--uu --uu
ADRL	xxxx ----	xxxx ----	xxxx ----	xxxx ----	uuuu ----
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu

Register	Reset (Power-on)	WDT Time-out (Normal Operation)	$\overline{RES}$ Reset Normal Operation	$\overline{RES}$ Reset (HALT)	WDT Time-out (HALT)*
ADCR	0100 0000	0100 0000	0100 0000	0100 0000	uuuu uuuu
ACSR	---- -000	---- -000	---- -000	---- -000	---- -uuu
CMPC	1011 0000	1011 0000	1011 0000	1011 0000	uuuu uuuu
MISC	000x 0---	000x 0---	000x 0---	000x 0---	uuuu u---
OPAC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
DBTC	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu

Note: "u" stands for unchanged
"x" stands for unknown
"-" stands for unimplemented

Oscillator

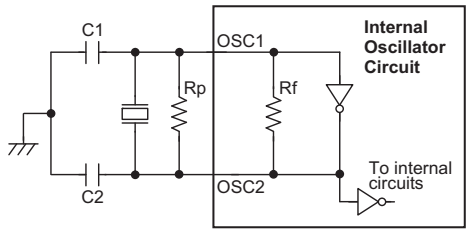
Various oscillator options offer the user a wide range of functions according to their various application requirements. Three types of system clocks can be selected while various clock source options for the Watchdog Timer are provided for maximum flexibility. All oscillator options are selected through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

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

Clock Source Modes

There are three methods of generating the system clock, using an external crystal/ceramic oscillator, an external RC network and an internal RC clock source. One of these three methods must be selected using the configuration options.

- External Crystal/Ceramic Oscillator
  The simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation. However, for some crystals and most resonator types, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.



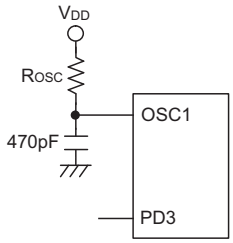
Note: 1. Rp is normally not required. C1 and C2 are required.
2. Although not shown OSC1/OSC2 pins have a parasitic capacitance of around 7pF.

Crystal/Resonator Oscillator – HXT

Crystal Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
12MHz	8pF	10pF
8MHz	8pF	10pF
4MHz	8pF	10pF
1MHz	100pF	100pF
Note: C1 and C2 values are for guidance only.		

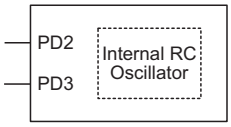
Crystal Recommended Capacitor Values

- External RC Oscillator
  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 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. Note that only the OSC1 pin is used, which is shared with I/O pin PD2, leaving pin PD3 free for use as a normal I/O pin.



External RC Oscillator

- Internal RC Oscillator
  The internal oscillator has three fixed frequencies of either 4MHz, 8MHz or 12MHz, the choice of which is indicated by the suffix marking next to the part number of the device used. This oscillator is fully integrated within the microcontroller and requires no external components. Note that if this internal system clock option is selected, as it requires no external pins for its operation, I/O pins PD2 and PD3 are free for use as normal I/O pins/



Note: PD2/PD3 used as normal I/Os

Internal RC Oscillator

Watchdog Timer Oscillator

The WDT oscillator is a fully integrated free running RC oscillator with a typical period of 65μs at 5V requiring no external components. It is selected via configuration option. If selected, when the device enters the Power Down Mode, the system clock will stop running, however the WDT oscillator continues to run and to keep the watchdog active. However, as the WDT will consume a certain amount of power when in the Power Down Mode, for low power applications, it may be desirable to disable the WDT oscillator by configuration option.

Power Down Mode and Wake-up

Power Down Mode

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

Entering the Power Down Mode

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

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

Standby Current Considerations

As the main reason for entering the Power Down Mode is to keep the current consumption of the MCU to as low a value as possible, perhaps only in the order of several micro-amps, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be 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. Care must also be taken with the loads, which are connected to I/Os, 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 Power Down 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 via an individual configuration option 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 Power Down Mode, the wake-up function of the related interrupt will be disabled.

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

Low Voltage Detector – LVD

Each device has a Low Voltage Detector function, also known as LVD. This enabled the device to monitor the power supply voltage, VDD, and provide a warning signal should it fall below a certain level. This function may be especially useful in battery applications where the supply voltage will gradually reduce as the battery ages, as it allows an early warning battery low signal to be generated. The Low Voltage Detector also has the capability of generating an interrupt signal.

LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDCTL. Three bits in this register, VLVD0~VLVD2, are used to select one of eight fixed voltages below which a low voltage condition will be detemined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the VDD voltage is above the preset low voltage value. The LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to zero will switch off the internal low voltage detector circuits. As the low voltage detector will consume a certain amount of power, it may be desirable to switch off the circuit when not in use, an important consideration in power sensitive battery powered applications.

Bit	7	6	5	4	3	2	1	0
Name	—	—	LVDO	LVDEN	—	VLVD2	VLVD1	VLVD0
R/W	R	R	R	R/W	R	R/W	R/W	R/W

LVD Control Register – LVDCTL

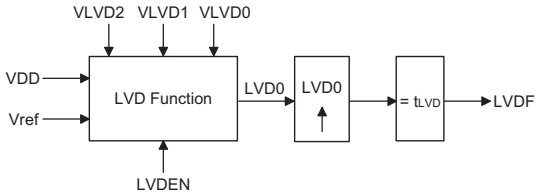
Bit	Name	Description	Condition
0	VLVD0	LVD Voltage Bit 0	See Table
1	VLVD1	LVD Voltage Bit 1	See Table
2	VLVD2	LVD Voltage Bit 2	See Table
3	—	Not used – read as zero	
4	LVDEN	LVD Enable/Disable	1: Enable; 0: Disable
5	LVDO	LVD Output Flag	1: Low Voltage Detect 0: No Low Voltage Detect
6	—	Not used – read as zero	
7	—	Not used – read as zero	

- Bits 0~2: VLVD0~VLVD2
  These bits specify the low voltage detect voltage as shown:

VLVD0	VLVD1	VLVD2	Voltage
0	0	0	Reversed
0	0	1	Reversed
0	1	0	Reversed
0	1	1	Reversed
1	0	0	Reversed
1	0	1	Reversed
1	1	0	Reversed
1	1	1	4.4

LVD Operation

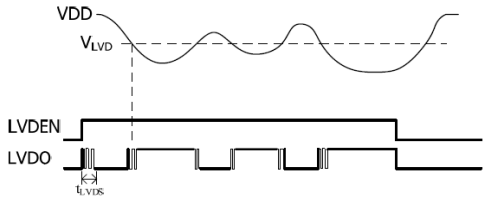
The Low Voltage Detector function operates by comparing the power supply voltage, VDD, with a pre-specified voltage level stored in the LVDCTL register. When the power supply voltage, VDD, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. The Low Voltage Detector function is supplied by a reference voltage which will be automatically enabled. When the device is powered down the low voltage detector will remain active if the LVDEN bit is high.



LVD Block Diagram

After enabling the Low Voltage Detector, a time delay t_{LVDS} should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the VDD voltage may rise and fall rather slowly, at the voltage nears that of VLVD, there may be multiple bit LVDO transitions.

When the device is powered down the Low Voltage Detector will remain active if the LVDEN bit is high.



LVD Operation

Watchdog Timer

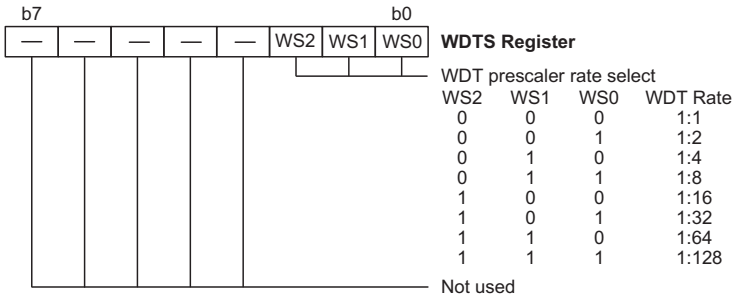
The Watchdog Timer is provided to prevent program malfunctions or sequences from jumping to unknown locations, due to certain uncontrollable external events such as electrical noise. It operates by providing a device reset when the WDT counter overflows. The WDT clock is supplied by one of two sources selected by configuration option: its own self contained dedicated internal WDT oscillator or $f_{SYS}/4$. Note that if the WDT configuration option has been disabled, then any instruction relating to its operation will result in no operation.

The internal WDT oscillator has an approximate period of 65us at a supply voltage of 5V. If selected, it is first divided by 256 via an 8-stage counter to give a nominal period of 17ms. Note that this period can vary with VDD, temperature and process variations. For longer WDT time-out periods the WDT prescaler can be utilised. By writing the required value to bits 0, 1 and 2 of the WDTS register, known as WS0, WS1 and WS2, longer time-out periods can be achieved. With WS0, WS1 and WS2 all equal to 1, the division ratio is 1:128 which gives a maximum time-out period of about 2.1s.

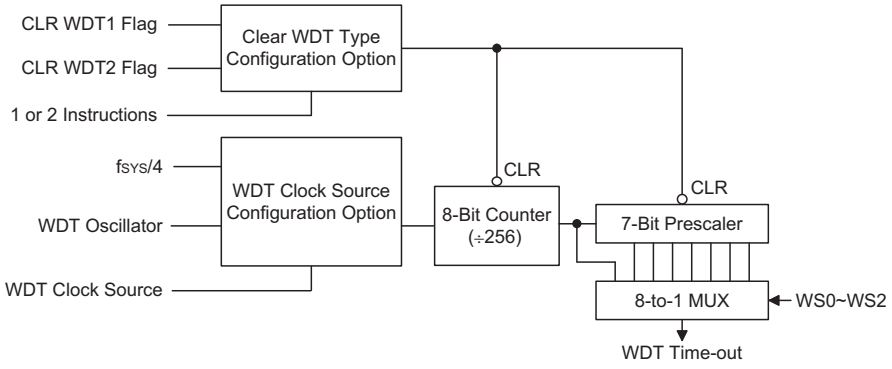
If the $f_{SYS}/4$ clock is used as the WDT clock source, it should be noted that when the system enters the Power Down Mode, then the instruction clock is stopped and the WDT will lose its protecting purposes. For systems that operate in noisy environments, using the internal WDT oscillator is strongly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

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

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



Watchdog Timer Register



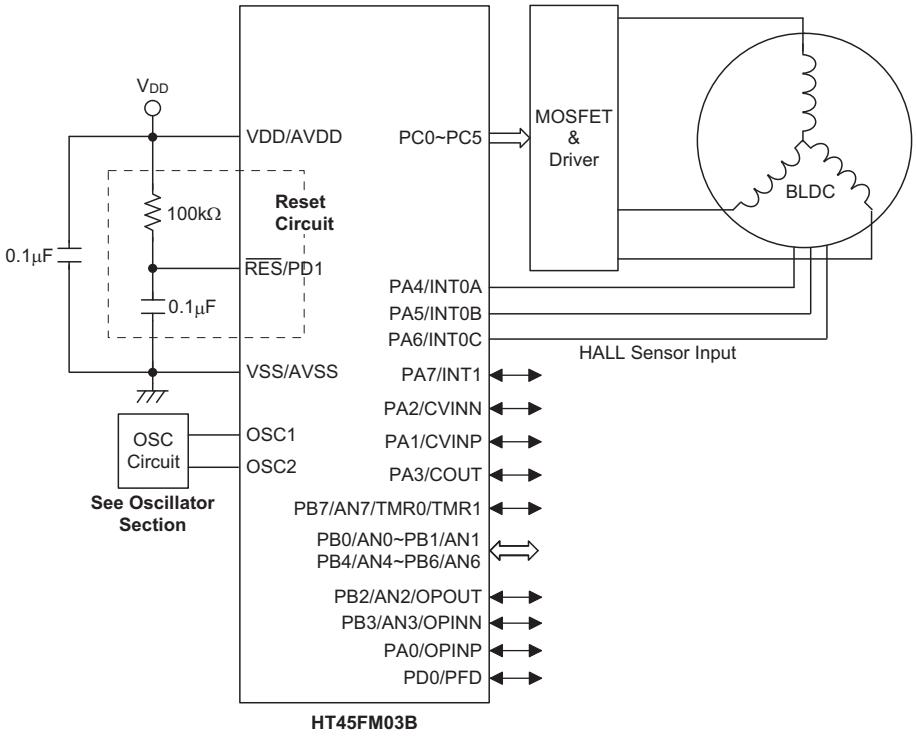
Watchdog Timer

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 by 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.	Function	Description
1	Wake-up PA0~PA7 (bit option)	None wake-up or wake-up
2	Pull-high PA0~PA7 (bit option)	None pull-high or pull-high
3	Pull-high PB0~PB7 (bit option)	None pull-high or pull-high
4	Pull-high PC0~PC5 (port option)	None pull-high or pull-high
5	Pull-high PD0 (bit option)	None pull-high or pull-high
6	PD0/PFD	PD0 or PFD output
7	LVR	Disable or enable
8	PWM mode	10 bits or (9+1) or (8+2) or (7+3) bits mode
9	WDT clock source	WDTOSC or $f_{SYS}/4$
10	WDT	Enable or disable
11	CLRWDT	2 instructions or 1 instruction
12	OSC	External RC + PD3, external XTAL or internal RC + PD2/PD3
13	PD1/$\overline{\text{RES}}$	RESET Pin or PD1
14	PWMLEV	PWMxH outputs are active high or active low
15	PWMCLEV	PWMxL outputs are active high or active low
16	Comparator interrupt source	Comparator output falling edge or PA3 falling edge
17	PFD source	PFD0: timer 0 overflow or PFD1: timer 1 overflow
18	INT1 trigger edge	Disable or rising edge or falling edge or double edge
19	INT0A pin-shared Option	Pin shared with PA4 or PB4
20	INT0B pin-shared Option	Pin shared with PA5 or PB5
21	INT0C pin-shared Option	Pin shared with PA6 or PB6
22	Internal RC OSC	12MHz, 16MHz or 20MHz
23	PWM duty mode	1 PWM duty mode or 3 PWM duty mode

Application Circuits



Instruction Set

Introduction

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

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

Instruction Timing

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

Moving and Transferring Data

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

Arithmetic Operations

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

Logical and Rotate Operations

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

Branches and Control Transfer

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

Bit Operations

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

Table Read Operations

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

Other Operations

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

Instruction Set Summary

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

Table conventions:

x: Bits immediate data

m: Data Memory address

A: Accumulator

i: 0~7 number of bits

addr: Program memory address

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

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

Instruction Definition

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

CALL addr	Subroutine call
Description	Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.
Operation	Stack ← Program Counter + 1 Program Counter ← addr
Affected flag(s)	None
CLR [m]	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] ← 00H
Affected flag(s)	None
CLR [m].i	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i ← 0
Affected flag(s)	None
CLR WDT	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT1	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF
CLR WDT2	Pre-clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.
Operation	WDT cleared TO ← 0 PDF ← 0
Affected flag(s)	TO, PDF

CPL [m] Complement Data Memory

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

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

Affected flag(s) Z

CPLA [m] Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

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

Affected flag(s) C

DEC [m] Decrement Data Memory

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

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

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

HALT Enter power down mode

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

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

Affected flag(s) TO, PDF

INC [m]	Increment Data Memory
Description	Data in the specified Data Memory is incremented by 1.
Operation	[m] ← [m] + 1
Affected flag(s)	Z
INCA [m]	Increment Data Memory with result in ACC
Description	Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	ACC ← [m] + 1
Affected flag(s)	Z
JMP addr	Jump unconditionally
Description	The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.
Operation	Program Counter ← addr
Affected flag(s)	None
MOV A,[m]	Move Data Memory to ACC
Description	The contents of the specified Data Memory are copied to the Accumulator.
Operation	ACC ← [m]
Affected flag(s)	None
MOV A,x	Move immediate data to ACC
Description	The immediate data specified is loaded into the Accumulator.
Operation	ACC ← x
Affected flag(s)	None
MOV [m],A	Move ACC to Data Memory
Description	The contents of the Accumulator are copied to the specified Data Memory.
Operation	[m] ← ACC
Affected flag(s)	None
NOP	No operation
Description	No operation is performed. Execution continues with the next instruction.
Operation	No operation
Affected flag(s)	None
OR A,[m]	Logical OR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″OR″ [m]
Affected flag(s)	Z

OR A,x Logical OR immediate data to ACC

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

Operation: ACC ← ACC ″OR″ x

Affected flag(s): Z

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

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

Operation: [m] ← ACC ″OR″ [m]

Affected flag(s): Z

RET Return from subroutine

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

Operation: Program Counter ← Stack

Affected flag(s): None

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

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

Operation: Program Counter ← Stack
ACC ← x

Affected flag(s): None

RETI Return from interrupt

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

Operation: Program Counter ← Stack
EMI ← 1

Affected flag(s): None

RL [m] Rotate Data Memory left

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

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

Affected flag(s): None

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

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

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

Affected flag(s): None

RLC [m] Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

RR [m] Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

RRC [m] Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

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

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

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

Description The contents of the specified Data Memory and the complement of the carry flag are 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 $[m] \leftarrow ACC - [m] - \overline{C}$

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

SDZ [m] Skip if decrement Data Memory is 0

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

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

Affected flag(s) None

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

Description The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy 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

SET [m] Set Data Memory

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

Operation $[m] \leftarrow FFH$

Affected flag(s) None

SET [m].i Set bit of Data Memory

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

Operation $[m].i \leftarrow 1$

Affected flag(s) None

SIZ [m] Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m] = 0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC = 0

Affected flag(s) None

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

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

Operation Skip if [m].i $\neq$ 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 ← ACC – [m]

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

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

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

Operation [m] ← ACC – [m]

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

SUB A,x Subtract immediate data from ACC

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

Operation ACC ← ACC – x

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

SWAP [m] Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7 ~ [m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

SZ [m] Skip if Data Memory is 0

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

Operation Skip if [m] = 0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m] = 0

Affected flag(s) None

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

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

Operation Skip if [m].i = 0

Affected flag(s) None

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

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

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

Affected flag(s) None

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

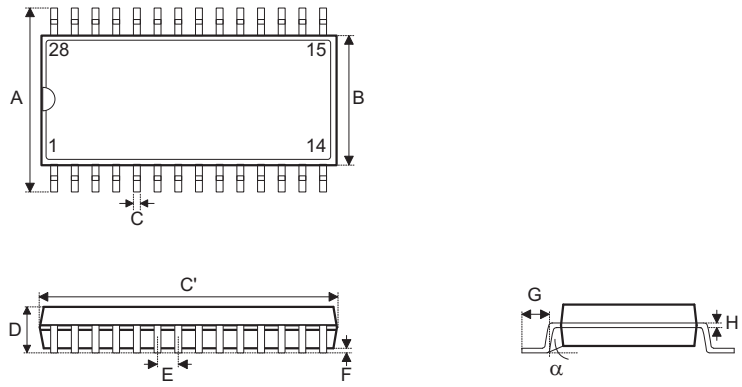
XOR A,[m]	Logical XOR Data Memory to ACC
Description	Data in the Accumulator and the specified Data Memory perform a bitwise logical XOR operation. The result is stored in the Accumulator.
Operation	ACC ← ACC ″XOR″ [m]
Affected flag(s)	Z

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

28-pin SOP (300mil) Outline Dimensions

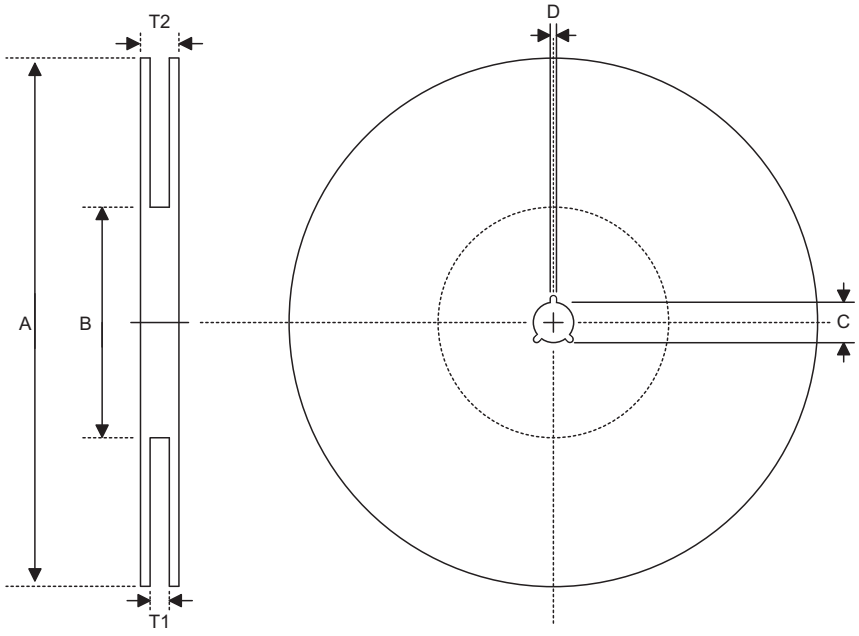


- MS-013

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°

Product Tape and Reel Specifications

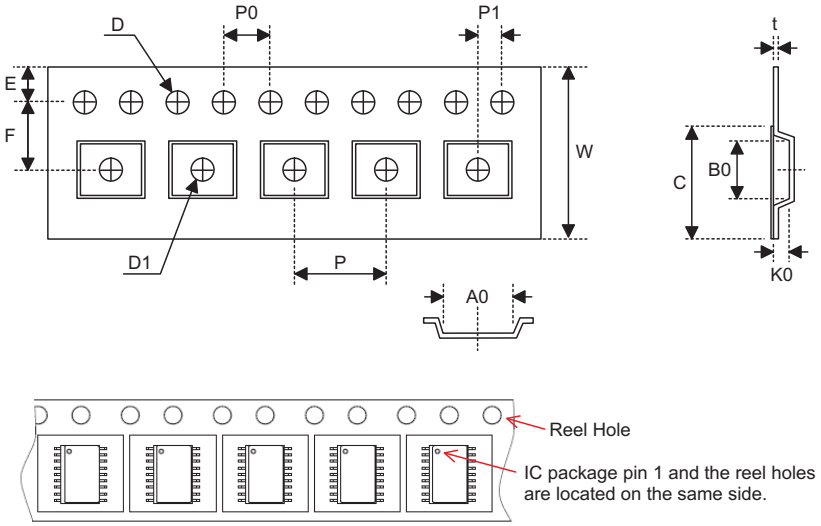
Reel Dimensions



SOP 28W (300mil)

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

Carrier Tape Dimensions



SOP 28W

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

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

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

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

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

Copyright © 2009 by HOLTEK SEMICONDUCTOR INC.

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