3D Glasses 8-bit Flash Type MCU

# HT45FH3T

Revision: V1.10 Date: May 17, 2013

www.holtek.com

# Features

## CPU Features

- Operating Voltage:
  - $f_{SYS}$=4MHz: 2.2V~5.5V
- Up to 1μs instruction cycle with 4MHz system clock at $V_{DD}$=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator:
  - Internal 4MHz RC – HIRC
  - Internal 32kHz RC – LIRC
- Fully integrated internal 4MHz oscillator requires no external components
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 4-level subroutine nesting
- Bit manipulation instruction

## Peripheral Features

- Program Memory: 2K×16
- Data Memory: 128×8
- Watchdog Timer function
- 7 bidirectional I/O lines
- One external interrupt line shared with I/O pin
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output function or single pulse output function
- Over voltage protection (OVP) with interrupt
- Dual Time-Base functions for generation of fixed time interrupt signals
- 4-channel 12-bit resolution A/D converter
- Low voltage reset function (enabled @ 2.1V)
- Low voltage detect function
- One integrated LDO: 3V
- 4 level shift output pins
- Package type: 16-pin SSOP

## General Description

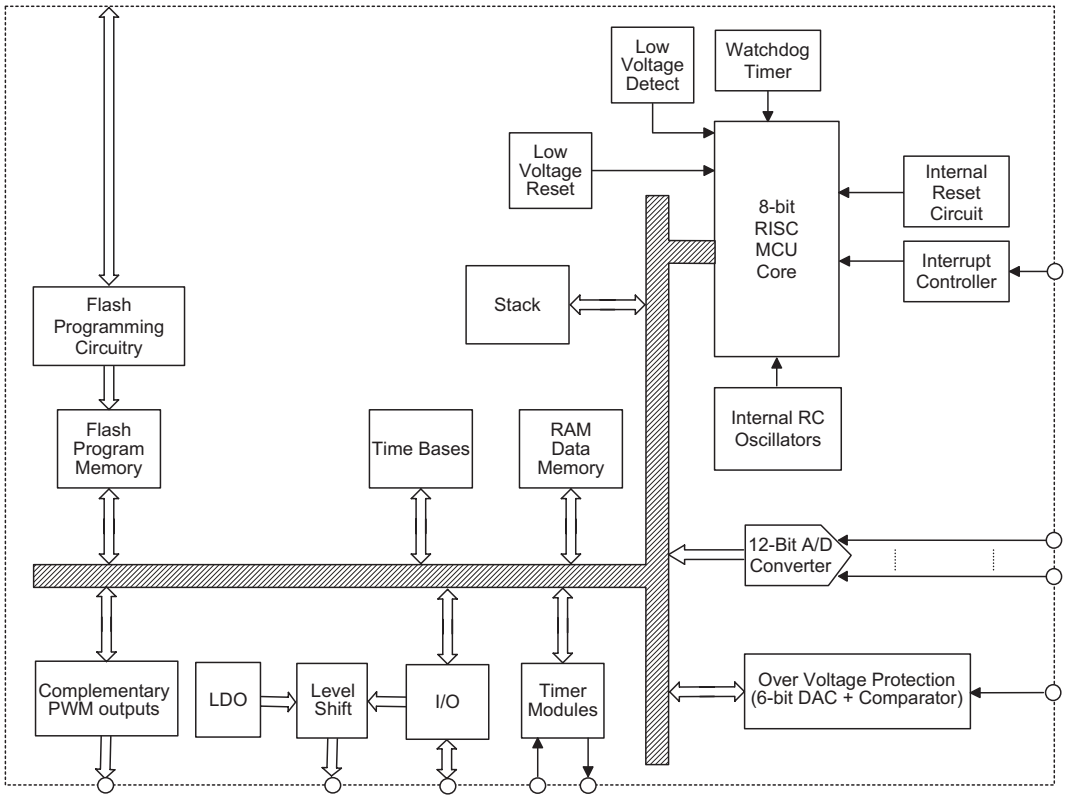
The device is a Flash Memory A/D type 8-bit high performance RISC architecture microcontroller. Offering users the convenience of Flash Memory multi-programming features, this device also includes a wide range of functions and features.

Analog features include a multi-channel 12-bit A/D converter, an over voltage protection function and a LDO regulator. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

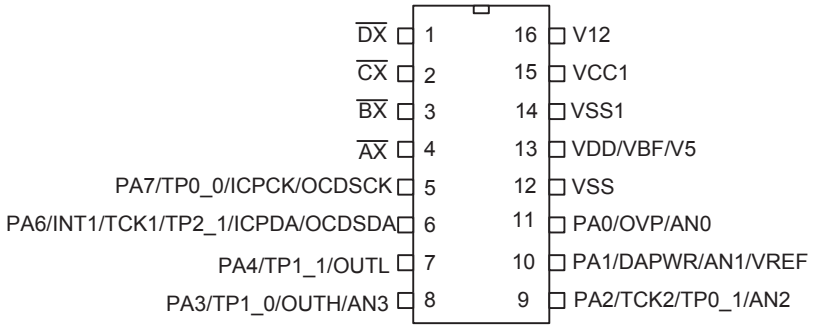
A full choice of HIRC and LIRC oscillator functions are provided including a fully integrated system oscillator which requires no external components for its implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption.

The inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that the device will find excellent use in applications such as electronic metering, environmental monitoring, handheld instruments, household appliances, electronically controlled tools, motor driving in addition to many others.

## Block Diagram



## Pin Assignment




**HT45FH3T 16 SSOP-A**

Note: 1. The HT45FH3T I/O lines, PB0/TCK3, PB1/TP3, PB2~PB4, are internally connected to the level shift inputs, A, B, C, D and $\overline{\text{ENBF}}$ , respectively.

2. If the pin-shared pin functions have multiple outputs simultaneously, its pin names at the right side of the "/" sign can be used for higher priority.

3. The PA5/INT0/TCK0/TP2_0 I/O line is not bonded to the external pin and must be appropriately managed to avoid the power consumption due to the unknown input voltage level.

4. The 16 SSOP-A package is for the real IC, the 16 NSOP-A package is for the OCDS EV IC.

## Pin Description

With the exception of the power pins and some relevant transformer control pins, all pins on these devices can be referenced by their Port name, e.g. PA0, PA1 etc, which refer to the digital I/O function of the pins. However these Port pins are also shared with other function such as the Analog to Digital Converter, Timer Module pins etc. The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

| Pin Name | Function | OPT | I/T | O/T | Pin-Shared Mapping |
|---|---|---|---|---|---|
| PA0~PA4, PA6~PA7 | General purpose I/O port A. Register enabled pull-up and wake-up | PAPU PAWU | ST | CMOS | — |
| OUTH, OUTL | PWM output | TMPC | — | CMOS | PA3, PA4 |
| OVP | Over voltage protection input | OCVPR1 | AN | — | PA0 |
| AN0~AN3 | A/D Converter input 0~3 | ADCR0 ACERL | AN | — | PA0~PA3 |
| VREF | A/D Converter reference voltage input | ADCR1 | AN | — | PA1 |
| DAPWR | D/A Converter power input | OVPREF | PWR | — | PA1 |
| INT1 | External interrupt 1 | INTEG INTC2 | ST | — | PA6 |
| TCK0~TCK2 | TM0~TM2 clock input | — | ST | — | PA5, PA6, PA2 |
| TP0_0, TP0_1 | TM0 I/O | TMPC | ST | CMOS | PA7, PA2 |
| TP1_0, TP1_1 | TM1 I/O | TMPC | ST | CMOS | PA3, PA4 |
| TP2_0, TP2_1 | TM2 I/O | TMPC | ST | CMOS | PA5, PA6 |
| ICPCK | In-circuit programming clock pin | — | ST | — | PA7 |
| ICPDA | In-circuit programming data/address pin | — | ST | CMOS | PA6 |
| OCDSCK | On-chip debug support clock pin | — | ST | — | PA7 |
| OCDSDA | On-chip debug support data/address pin | — | ST | CMOS | PA6 |
| V12 | Level shift output power | — | PWR | — | — |
| VBF | Level shift input power | — | PWR | — | Bond with VDD |
| V5 | LDO output for MCU power supply | — | — | PWR | Bond with VDD |
| VCC1 | LDO positive power supply | — | PWR | — | — |
| VSS1 | LDO negative power supply, ground | — | PWR | — | — |
| VDD | Positive power supply | — | PWR | — | — |
| VSS | Negative power supply, ground | — | PWR | — | — |
| $\overline{AX}$, $\overline{BX}$, $\overline{CX}$, $\overline{DX}$ | Level shift outputs | — | — | — | — |

Note: The I/O line PA5/INT0 is not bonded to the external pin and should be properly configured after power on reset. The functional pin which is pin-shared with PA5 will lose the corresponding pin function.

## Internal Connection Description

| Signal Name | Function | OPT | I/T | O/T | Pin-Shared Mapping |
|---|---|---|---|---|---|
| PB0~PB4 | General purpose input/output. Register enabled pull-up. Internally connected to the level shift inputs respectively and level shift enable. | PBPU | ST | CMOS | — |
| A, B, C, D | Level shift inputs. Internally connected to PB0, PB1, PB2 and PB3 respectively. | — | — | — | — |
| $\overline{ENBF}$ | Level shift enable, active low. Internally connected to PB4 | — | — | — | — |

Note: I/T: Input type; O/T: Output type
OPT: Optional by configuration option (CO) or register option
PWR: Power; AN: analog signal
CMOS: CMOS output; ST: Schmitt Trigger input

## Absolute Maximum Ratings

Supply Voltage ..... $V_{SS}$−0.3V to $V_{SS}$+6.0V
Input Voltage ..... $V_{SS}$−0.3V to $V_{DD}$+0.3V
Storage Temperature ..... -50°C to 150°C
Operating Temperature ..... -40°C to 85°C
$I_{OH}$ Total ..... -80mA
$I_{OL}$ Total ..... 80mA
Total Power Dissipation ..... 500mW

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

## D.C. Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{DD}$ | Operating Voltage | — | $f_{SYS}$=4MHz | 2.2 | — | 5.5 | V |
| $I_{DD1}$ | Operating Current, Normal Mode, $f_{SYS}=f_H$ | 3V | No load, $f_H$=4MHz, ADC off, WDT enable | — | 400 | 600 | μA |
| | | 5V | | — | 840 | 1200 | μA |
| $I_{DD2}$ | Operating Current, Slow Mode, $f_{SYS}=f_L$=LIRC | 3V | No load, $f_{SYS}$=LIRC, ADC off, WDT enable | — | 10 | 20 | μA |
| | | 5V | | — | 20 | 35 | μA |
| $I_{IDLE0}$ | IDLE0 Mode Standby Current (LIRC on) | 3V | No load, ADC off, WDT enable, LVR disable | — | 1.5 | 3.0 | μA |
| | | 5V | | — | 2.5 | 5.0 | μA |
| $I_{IDLE1}$ | IDLE1 Mode Standby Current | 3V | No load, ADC off, WDT enable, $f_{SYS}$=4MHz on | — | 170 | 260 | μA |
| | | 5V | | — | 330 | 500 | μA |
| $I_{SLEEP0}$ | SLEEP0 Mode Standby Current (LIRC off) | 3V | No load, ADC off, WDT disable, LVR disable | — | 0.1 | 1.0 | μA |
| | | 5V | | — | 0.3 | 2.0 | μA |
| $I_{SLEEP1}$ | SLEEP1 Mode Standby Current (LIRC on) | 3V | No load, ADC off, WDT enable, LVR disable | — | 1.5 | 3.0 | μA |
| | | 5V | | — | 2.5 | 5.0 | μA |
| $V_{IL1}$ | Input Low Voltage for I/O Ports or Input Pins | 5V | — | 0 | — | 1.5 | V |
| | | — | — | 0 | — | $0.2V_{DD}$ | V |
| $V_{IH1}$ | Input High Voltage for I/O Ports or Input Pins | 5V | — | 3.5 | — | 5.0 | V |
| | | — | — | $0.8V_{DD}$ | — | $V_{DD}$ | V |
| $I_{OL}$ | I/O Port Sink Current | 3V | $V_{OL}=0.1V_{DD}$ | 4 | 8 | — | mA |
| | | 5V | $V_{OL}=0.1V_{DD}$ | 10 | 20 | — | mA |
| $I_{OH}$ | I/O Port Source Current | 3V | $V_{OH}=0.9V_{DD}$ | -2 | -4 | — | mA |
| | | 5V | $V_{OH}=0.9V_{DD}$ | -5 | -10 | — | mA |
| $R_{PH}$ | Pull-high Resistance for I/O Ports | 3V | — | 20 | 60 | 100 | kΩ |
| | | 5V | — | 10 | 30 | 50 | kΩ |

Note: LVR is aways enabled (HALT mode disabled) fixed @ 2.1V.

## A.C. Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $f_{CPU}$ | Operating Clock | 2.2V~5.5V | — | DC | — | 8 | MHz |
| $f_{SYS}$ | System Clock (HIRC) | 2.2V~5.5V | — | — | — | 8 | MHz |
| $f_{HIRC}$ | HIRC Frequency(note) | 3V | Ta=25°C | -2% | 4 | +2% | MHz |
| | | 2.4V~3.6V | Ta=0°C~70°C | -5% | 4 | +5% | MHz |
| | | 2.4V~3.6V | Ta=-40°C~85°C | -10% | 4 | +10% | MHz |
| $f_{LIRC}$ | System Clock (LIRC) | 5V | Ta=25°C | -10% | 32 | +10% | kHz |
| | | 2.2V~5.5V | Ta=-40°C~85°C | -30% | 32 | +60% | kHz |
| $t_{TIMER}$ | TCKn Input Pin Minimum Pulse Width | — | — | — | 30 | — | ns |
| $t_{INT}$ | Interrupt Minimum Pulse Width | — | — | 1 | 3.3 | 5 | μs |
| $t_{SST}$ | System Start-up Timer Period (Wake-up from HALT, $f_{SYS}$ off at HALT state) | — | $f_{SYS}$=HIRC | — | 16 | — | $t_{SYS}$ |
| | | — | $f_{SYS}$=LIRC | — | 2 | — | $t_{SYS}$ |
| | System Start-up Timer Period (Wake-up from HALT, $f_{SYS}$ on at HALT state) | — | — | — | 2 | — | $t_{SYS}$ |
| $t_{RSTD}$ | System Reset Delay Time (Power On Reset, LVR, WDTC/LVRC SW reset) | — | — | 25 | 50 | 100 | ms |
| | System Reset Delay Time (WDT time-out reset) | — | — | 8.3 | 16.7 | 33.3 | ms |

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

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

## A/D Converter Electrical Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{ADI}$ | A/D Converter Input Voltage | — | — | 0 | — | $V_{REF}$ | V |
| $V_{REF}$ | A/D Converter Reference Voltage | — | — | 2 | — | $V_{DD}$ | V |
| $V_{BG}$ | Reference Voltage | — | — | -3% | 1.19 | +3% | V |
| DNL | Differential Non-linearity | 5V | $t_{ADCK}$=1.0μs | — | ±1 | ±2 | LSB |
| INL | Integral Non-linearity | 5V | $t_{ADCK}$=1.0μs | — | ±2 | ±4 | LSB |
| $I_{ADC}$ | Additional Power Consumption if A/D Converter is used | 3V | No load ($t_{ADCK}$=0.5μs ) | — | 0.9 | 1.35 | mA |
| | | 5V | No load ($t_{ADCK}$=0.5μs ) | — | 1.2 | 1.8 | mA |
| $I_{BG}$ | Additional Power Consumption if $V_{BG}$ Reference with Buffer is used | — | — | — | 200 | 300 | μA |
| $t_{ADCK}$ | A/D Converter Clock Period | — | — | 0.5 | — | 10 | μs |
| $t_{ADC}$ | A/D Conversion Time (Include Sample and Hold Time) | — | 12 bit ADC | — | 16 | — | $t_{ADCK}$ |
| $t_{ADS}$ | A/D Converter Sampling Time | — | — | — | 4 | — | $t_{ADCK}$ |
| $t_{ON2ST}$ | A/D Converter On-to-Start Time | — | — | 2 | — | — | μs |
| $t_{BGS}$ | $V_{BG}$ Turn on Stable Time | — | — | 200 | — | — | μs |

## LVD & LVR Electrical Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{LVR}$ | Low Voltage Reset Voltage | — | LVR Enable | -5%×Typ. | 2.1 | +5%×Typ. | V |
| $V_{LVD1}$ | Low Voltage Detector Voltage | — | LVDEN=1, $V_{LVD}$=2.2V | -5%× Typ. | 2.2 | +5%× Typ. | V |
| $V_{LVD2}$ | | | LVDEN=1, $V_{LVD}$=2.4V | | 2.4 | | V |
| $V_{LVD3}$ | | | LVDEN=1, $V_{LVD}$=2.7V | | 2.7 | | V |
| $V_{LVD4}$ | | | LVDEN=1, $V_{LVD}$=3.0V | | 3.0 | | V |
| $V_{LVD5}$ | | | LVDEN=1, $V_{LVD}$=3.3V | | 3.3 | | V |
| $V_{LVD6}$ | | | LVDEN=1, $V_{LVD}$=3.6V | | 3.6 | | V |
| $V_{LVD7}$ | | | LVDEN=1, $V_{LVD}$=4.0V | | 4.0 | | V |
| $I_{LVR}$ | Low Voltage Reset Current | — | LVR Enable, LVDEN=0 | — | 20 | 30 | μA |
| $I_{LVD}$ | Low Voltage Detector Current | — | LVR enable, LVDEN=1 | — | 30 | 45 | μA |
| $t_{LVR}$ | Low Voltage Width to Reset | — | — | 120 | 240 | 480 | μs |
| $t_{LVD}$ | Low Voltage Width to Interrupt | — | — | 20 | 45 | 90 | μs |
| $t_{LVDS}$ | LVDO stable time | — | For LVR enable, LVD off→on | 15 | — | — | μs |
| | | — | For LVR disable, LVD off→on | 200 | — | — | μs |
| $t_{SRESET}$ | Software Reset Width to Reset | — | — | 45 | 90 | 120 | μs |

Note: LVR is aways enabled (HALT mode disabled) fixed @ 2.1V.

## LDO Regulator Characteristics

V5=3V, $V_{IN}$=$V_{OUT}$ + 1.0V, $I_O$=1mA, Ta=25°C, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $\Delta V_{OUT}$ | Output Voltage Tolerance | — | $I_O$=10mA, Ta=25°C | -2 | — | 2 | % |
| $\Delta V_{LOAD}$ | Load Regulation(Note 1) | — | 1mA≤$I_O$≤30mA, $V_{IN}$=5V | — | 0.09 | 0.18 | %/mA |
| | | — | 1mA≤$I_O$≤12mA, $V_{IN}$=4V | — | | | |
| $V_{DROP}$ | Drop Out Voltage(Note 2) | — | $I_O$=2mA, $\Delta V_O$=2% | — | — | 320 | mV |
| | | — | $I_O$=5mA, $\Delta V_O$=2% | — | — | 450 | mV |
| $I_{SS}$ | Quienscent Current | — | $I_O$=0mA | — | 2 | 4 | μA |
| $\Delta V_{LINE}$ | Line Regulation | — | 2.0V+$V_{OUT}$≤$V_{IN}$≤28V, $I_O$=1mA | — | 0.2 | — | %/V |
| $V_{IN}$ | Input Voltage $V_{CC}$ | — | — | — | — | 28 | V |
| $\Delta V_{OUT}/\Delta Ta$ | Temperature Coefficient | — | $I_O$=10mA, -40°C<Ta<85°C | — | ±0.54 | — | mV/°C |

Note: 1. Load regulation is measured at a constant junction temperature, using pulse testing with a low ON time and is guaranteed up to the maximum power dissipation. Power dissipation is determined by the input/output differential voltage and the output current. Guaranteed maximum power dissipation will not be available over the full input/output range. The maximum allowable power dissipation at any ambient temperature is $P_D$=($T_{J(MAX)}$-Ta)/$\theta_{JA}$.

2. Dropout voltage is defined as the input voltage minus the output voltage that produces a 2% change in the output voltage from the value at $V_{IN}$=$V_{OUT}$+2V.

## Level Converter Characteristics

$V_{CC}$=12V, Ta=25°C, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $I_{source}$ | Output Source Current of $\overline{AX}$, $\overline{BX}$, $\overline{CX}$, $\overline{DX}$ | — | $V_{OH}$=10.4V | -60 | -90 | — | mA |
| $I_{sink}$ | Output Sink Current of $\overline{AX}$, $\overline{BX}$, $\overline{CX}$, $\overline{DX}$ | — | $V_{OL}$=1.6V | 60 | 90 | — | mA |

## Over Voltage circuit Characteristics

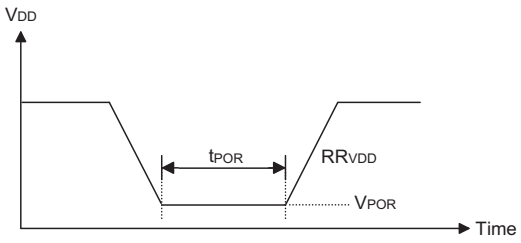
Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $I_{OCVP}$ | Over voltage protection operation current | 3V | OVPEN=1 | — | 200 | 300 | μA |
| | | 5V | | — | 300 | 450 | μA |
| **Compartor (CA)** | | | | | | | |
| $V_{CMPOS1}$ | Comparator input offset voltage | 3V/5V | Without calibration, CAOF[5:0]=100000B | -15 | — | +15 | mV |
| $V_{CMPOS2}$ | Comparator input offset voltage | 3V/5V | With calibration | -4 | — | +4 | mV |
| $V_{HYS}$ | Hysteresis width | 3V/5V | — | 40 | 60 | 80 | mV |
| $V_{CM}$ | Comparator common mode voltage range | 3V/5V | — | $V_{SS}$ | — | $V_{DD}$-1.4V | V |
| $A_{OL}$ | Comparator open loop gain | 3V/5V | — | 60 | 80 | — | dB |
| $t_{PD}$ | Comparator response time | 3V/5V | With 100mV overdrive | — | 370 | 560 | ns |
| **DAC for OVPREF** | | | | | | | |
| $V_{DAPWR}$ | DAC Reference Voltage | — | — | 1.5V | — | $V_{DD}$ | V |
| DNL | DAC Differential NonLinearity | — | — | -1 | — | +1 | LSB |
| INL | DAC Integral NonLinearity | — | — | -2 | — | +2 | LSB |

## Power on Reset Characteristics

Ta=25°C

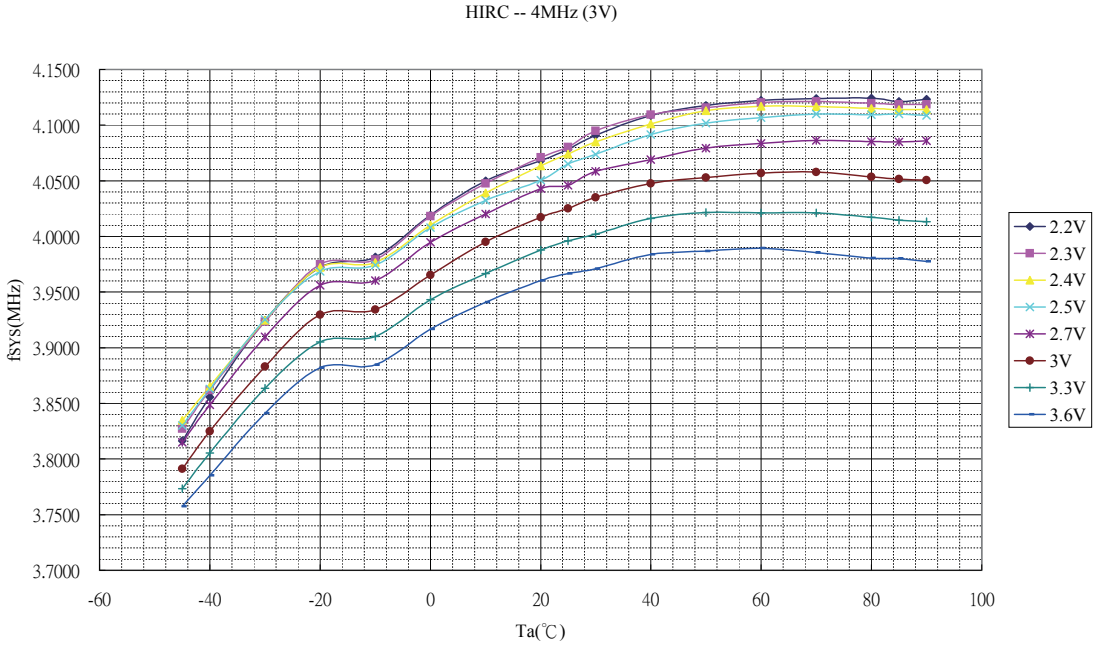
| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{POR}$ | $V_{DD}$ Start Voltage to Ensure Power-on Reset | — | — | — | — | 100 | mV |
| $R_{POR}$ | $V_{DD}$ Rising Rate to Ensure Power-on Reset | — | —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



## Oscillator Temperature/Frequency Characteristics

The following characteristic graphics depicts typical oscillator behavior. The data preseted 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.

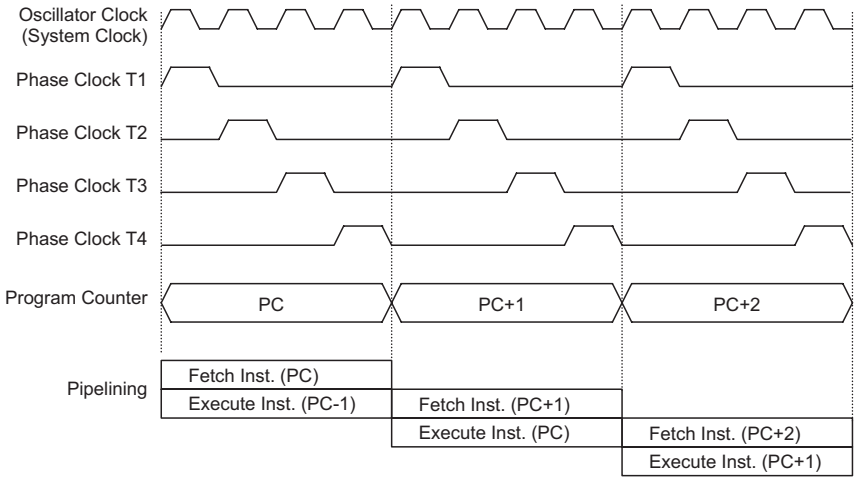


## System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The device takes 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

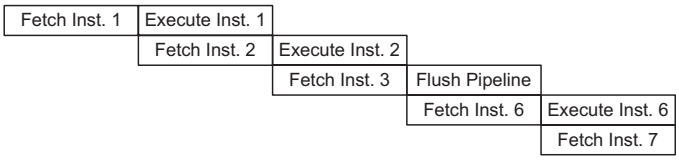
The main system clock, derived from either an HIRC or LIRC oscillator is subdivided into four internally generated non-overlapping clocks, T1~T4. The Program Counter is incremented at the beginning of the T1 clock during which time a new instruction is fetched. The remaining T2~T4 clocks carry out the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




**System Clock and Pipelining**

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.

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




**Instruction Fetching**

## Program Counter

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

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.

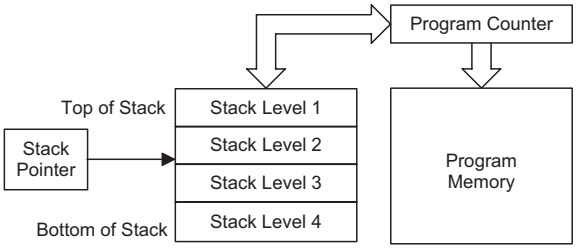
| Program Counter | |
|---|---|
| **Program Counter High byte** | **PCL Register** |
| PC10~PC8 | PCL7~PCL0 |

The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly, however, as only this low byte is available for manipulation, the jumps are limited 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. Manipulating the PCL register may cause program branching, so an extra cycle is needed to pre-fetch.

## Stack

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

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




## Arithmetic and Logic Unit – ALU

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

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

## Flash Program Memory

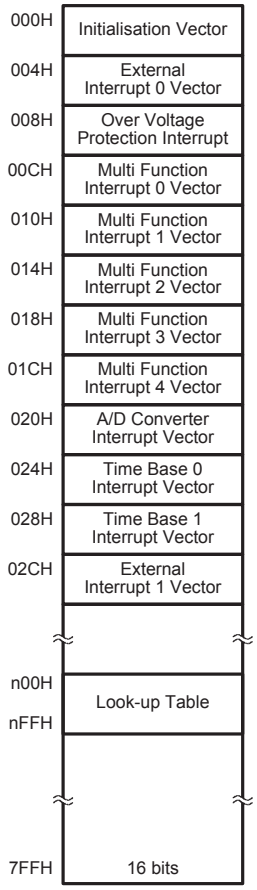
The Program Memory is the location where the user code or program is stored. For this device the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate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

### Structure

The Program Memory has a capacity of 2K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.

### Special Vectors

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



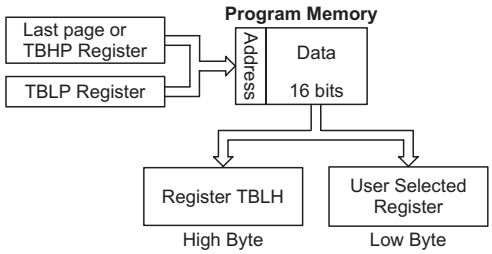

**Program Memory Structure**

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

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

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




| Instruction | Table Location Bits | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| TABRD [m] | @10 | @9 | @8 | @7 | @6 | @5 | @4 | @3 | @2 | @1 | @0 |
| TABRDL [m] | 1 | 1 | 1 | @7 | @6 | @5 | @4 | @3 | @2 | @1 | @0 |

**Table Location**

Note: b10~b0: Table location bits

@7~@0: Table pointer (TBLP) bits

@10~@8: Table pointer (TBHP) bits

## Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "700H" which refers to the start address of the last page within the 2K words 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 "706H" 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 "TABRD [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 "TABRD [m]" instruction is executed.

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

### Table Read Program Example

```
tempreg1 db ?      ; temporary register #1
tempreg2 db ?      ; temporary register #2
:
:
mov a,06h          ; initialise low table pointer - note that this address is referenced
mov tblp,a
mov a,07h          ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1     ; transfers value in table referenced by table pointer data at program
                   ; memory address "706H" transferred to tempreg1 and TBLH
dec tblp           ; reduce value of table pointer by one
tabrd tempreg2  ; transfers value in table referenced by table pointer data at program
                   ; memory address "705H" transferred to tempreg2 and TBLH in this
                   ; example the data "1AH" is transferred to tempreg1 and data "0FH" to
                   ; register tempreg2
:
:
org 700h           ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

## In Circuit Programming

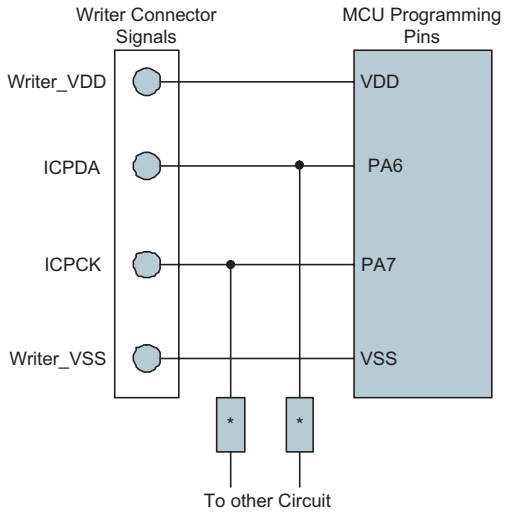
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or upgrading the program at a later stage. This enables product manufacturers to easily keep their manufactured products supplied with the latest program releases without removal and re-insertion of the device.

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

| Holtek Writer Pins | MCU Programming Pins | Pin Description |
|---|---|---|
| ICPDA | PA6 | Programming Serial Data/Address |
| ICPCK | PA7 | Programming Clock |
| VDD | VDD | Power Supply |
| VSS | VSS | Ground |

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

During the programming process, taking control of the PA6 and PA7 pins for data and clock programming purposes. The user must there take care to ensure that no other outputs are connected to these two pins.




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

### On-Chip Debug Support – OCDS

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

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

## RAM Data Memory

The Data Memory is a volatile area of 8-bit wide RAM internal memory and is the location where temporary information is stored.

### Structure

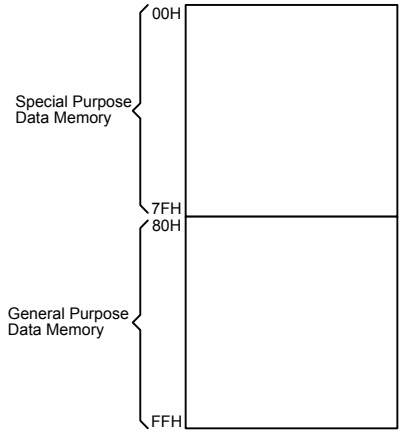
Divided into two sections, the first of these is an area of RAM, known as the Special Function Data Memory. Here are located registers which are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

| | |
|---|---|
| 00H | IAR0 |
| 01H | MP0 |
| 02H | IAR1 |
| 03H | MP1 |
| 04H | Unused |
| 05H | ACC |
| 06H | PCL |
| 07H | TBLP |
| 08H | TBLH |
| 09H | TBHP |
| 0AH | STATUS |
| 0BH | SMOD |
| 0CH | LVDC |
| 0DH | INTEG |
| 0EH | INTC0 |
| 0FH | INTC1 |
| 10H | INTC2 |
| 11H | MFI0 |
| 12H | MFI1 |
| 13H | MFI2 |
| 14H | MFI3 |
| 15H | MFI4 |
| 16H | PA |
| 17H | PAC |
| 18H | PAPU |
| 19H | PAWU |
| 1AH | TMPC |
| 1BH | WDTC |
| 1CH | TBC |
| 1DH ~ 1FH | Unused |
| 20H | ADRL |
| 21H | ADRH |
| 22H | ADCR0 |
| 23H | ADCR1 |
| 24H | ACERL |
| 25H | Unused |
| 26H | CTRL |
| 27H | LVRC |
| 28H | TM0C0 |
| 29H | TM0C1 |
| 2AH | TM0DL |
| 2BH | TM0DH |
| 2CH | TM0AL |
| 2DH | TM0AH |
| 2EH | TM0RPL |
| 2FH | TM0RPH |

| | |
|---|---|
| 30H | TM1C0 |
| 31H | TM1C1 |
| 32H | TM1DL |
| 33H | TM1DH |
| 34H | TM1AL |
| 35H | TM1AH |
| 36H | TM1RPL |
| 37H | TM1RPH |
| 38H ~ 3CH | Unused |
| 3DH | PB |
| 3EH | PBC |
| 3FH | PBPU |
| 40H | Unused |
| 41H | OVPREF |
| 42H | OCVPR0 |
| 43H | OCVPR1 |
| 44H | OCVPR2 |
| 45H | CPR |
| 46H | Unused |
| 47H | Unused |
| 48H | TM2C0 |
| 49H | TM2C1 |
| 4AH | TM2DL |
| 4BH | TM2DH |
| 4CH | TM2AL |
| 4DH | TM2AH |
| 4EH | TM3C0 |
| 4FH | TM3C1 |
| 50H | TM3DL |
| 51H | TM3DH |
| 52H | TM3AL |
| 53H | TM3AH |
| 54H ~ 7FH | Unused |

: Unused, read as 00H

**Special Purpose Data Memory Structure**




**Data Memory Structure**

## Special Function Register Description

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

### Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register 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 Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together access data from Bank 0 while the IAR1 and MP1 register pair can access data from any bank. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of "00H" and writing to the registers indirectly will result in no operation.

### Memory Pointers – MP0, MP1

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

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

**Indirect Addressing Program Example**

```
data .section ´data´
adres1   db ?
adres2   db ?
adres3   db ?
adres4   db ?
block    db ?
code .section at 0 ´code´
org00h
start:
     mov a,04h            ; setup size of block
     mov block,a
     mov a,offset adres1  ; Accumulator loaded with first RAM address
     mov mp0,a            ; setup memory pointer with first RAM address
loop:
     clr IAR0             ; clear the data at address defined by mp0
     inc mp0              ; increment memory pointer
     sdz block            ; check if last memory location has been cleared
     jmp loop
continue:
```

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

### Accumulator – ACC

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

### Program Counter Low Register – PCL

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

### Look-up Table Registers – TBLP, TBHP, TBLH

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

### Status Register – STATUS

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

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

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

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

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

**STATUS Register**

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

"×" unknown

Bit 7~6 Unimplemented, read as "0"

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

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

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

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

Bit 1 **AC:** Auxiliary flag
0: no auxiliary carry
1: an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction

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

# Oscillator

Various oscillator options offer the user a wide range of functions according to their various application requirements. The flexible features of the oscillator functions ensure that the best optimisation can be achieved in terms of speed and power saving. Oscillator selections and operation are selected through a combination of configuration options and registers.

## Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillators. With the capability of dynamically switching between fast and slow system clock, the device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

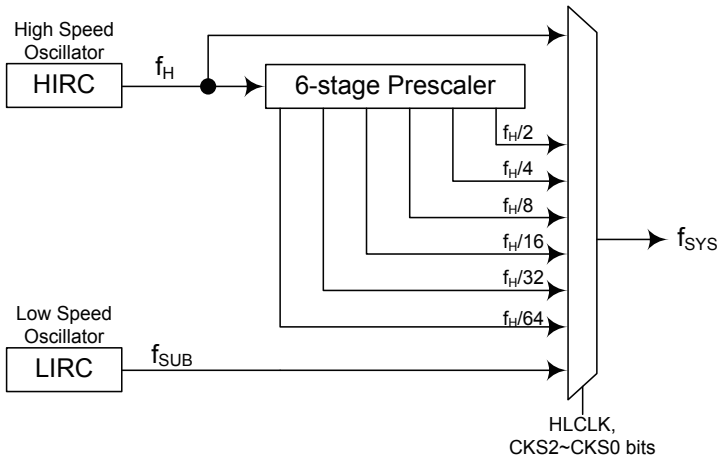
| Type | Name | Freq. |
|---|---|---|
| Internal High Speed RC | HIRC | 4MHz |
| Internal Low Speed RC | LIRC | 32kHz |

**Oscillator Types**

## System Clock Configurations

There are two methods of generating the system clock, a high speed oscillator and a low speed oscillator. The high speed oscillator is the internal 4MHz RC oscillator – HIRC. The low speed oscillator is the internal 32kHz (LIRC) oscillator. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the HLCLK bit and CKS2~CKS0 bits in the SMOD register and as the system clock can be dynamically selected.

The actual source clock used for the high speed and the low speed oscillators is chosen via registers. The frequency of the slow speed or high speed system clock is also determined using the HLCLK bit and CKS2~CKS0 bits in the SMOD register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.




**System Clock Configurations**

### Internal RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 4MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. Note that if this internal system clock option is selected, as it requires no external pins for its operation.

### Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is the low frequency oscillator. It is a fully integrated RC oscillator with a typical frequency of 32kHz at 5V, requiring no external components for its implementation. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

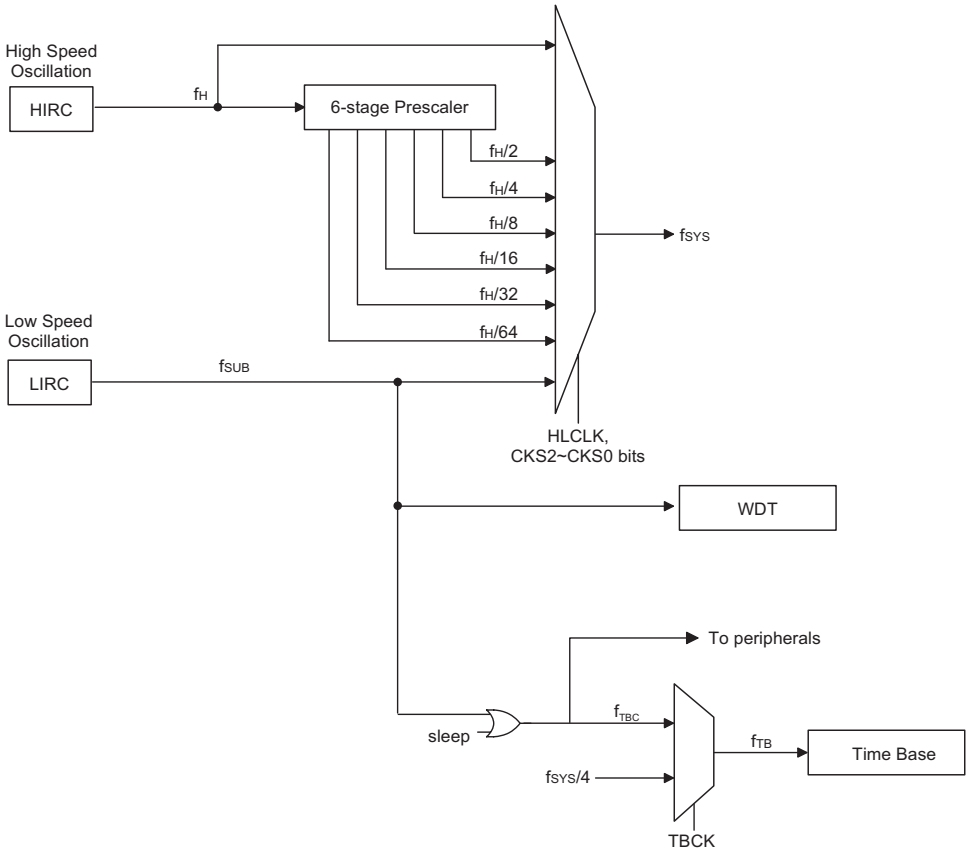
## Operating Modes and System Clocks

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

### System Clocks

The device has many different clock sources for both the CPU and peripheral function operation. By providing the user with a wide range of clock options using configuration options and register programming, a clock system can be configured to obtain maximum application performance.

The main system clock, can come from either a high frequency, $f_H$, or low frequency, $f_{SUB}$, source, and is selected using the HLCLK bit and CKS2~CKS0 bits in the SMOD register. The high speed system clock can be sourced from the HIRC oscillator. The low speed system clock source can be sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.




**System Clock Configurations**

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

## System Operation Modes

There are six different modes of operation for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the NORMAL Mode and SLOW Mode. The remaining three modes, the SLEEP0, SLEEP1, IDLE0 and IDLE1 Mode are used when the microcontroller CPU is switched off to conserve power.

| Operating Mode | Description | | | |
|---|---|---|---|---|
| | CPU | $f_{SYS}$ | $f_{SUB}$ | $f_{TBC}$ |
| NORMAL mode | On | $f_H$~$f_H$/64 | On | On |
| SLOW mode | On | $f_{SUB}$ | On | On |
| ILDE0 mode | Off | Off | On | On |
| IDLE1 mode | Off | On | On | On |
| SLEEP0 mode | Off | Off | Off | Off |
| SLEEP1 mode | Off | Off | On | Off |

### NORMAL Mode

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

### SLOW Mode

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

### SLEEP0 Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP0 mode the CPU will be stopped, and the $f_{SUB}$ clock will be stopped too, and the Watchdog Timer function is disabled. In this mode, the LVDEN is must set to “0”. If the LVDEN is set to “1”, it won’t enter the SLEEP0 Mode.

### SLEEP1 Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the IDLEN bit in the SMOD register is low. In the SLEEP1 mode the CPU will be stopped. However the $f_{SUB}$ clock will continue to operate if the LVDEN is “1” or the Watchdog Timer function is enabled.

### IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the CTRL register is low. In the IDLE0 Mode the system oscillator will be inhibited from driving the CPU, the system oscillator will be stopped, the low frequency clock $f_{SUB}$ will be on.

### IDLE1 Mode

The IDLE1 Mode is entered when a HALT instruction is executed and when the IDLEN bit in the SMOD register is high and the FSYSON bit in the CTRL register is high. In the IDLE1 Mode the system oscillator will be inhibited from driving the CPU, the system oscillator will continue to run, and this system oscillator may be high speed or low speed system oscillator. In the IDLE1 Mode the low frequency clock $f_{SUB}$ will be on.

Note: If LVDEN=1 and the SLEEP or IDLE mode is entered, the LVD and bandgap functions will not be disabled, and the $f_{SUB}$ clock will be forced to enable.

## Control Register

The SMOD register is used to control the internal clocks within the device.

### SMOD Register

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

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

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

Bit 4 Unimplemented, read as 0.

Bit 3 **LTO**: LIRC System OSC SST ready flag
0: Not ready
1: Ready

This is the low speed system oscillator SST ready flag which indicates when the low speed system oscillator is stable after power on reset or a wake-up has occurred. The flag will change to a high level after 1~2 cycles.

Bit 2 **HTO**: HIRC System OSC SST ready flag
0: Not ready
1: Ready

This is the high speed system oscillator SST ready flag which indicates when the high speed system oscillator is stable after a wake-up has occurred. This flag is cleared to "0" by hardware when the device is powered on and then changes to a high level after the high speed system oscillator is stable. Therefore this flag will always be read as "1" by the application program after device power-on. The flag will be low when in the SLEEP or IDLE0 Mode but after power on reset or a wake-up has occurred, the flag will change to a high level after 15~16 clock cycles if the HIRC oscillator is used.

Bit 1 **IDLEN**: IDLE Mode Control
0: Disable
1: Enable
This is the IDLE Mode Control bit and determines what happens when the HALT instruction is executed. If this bit is high, when a HALT instruction is executed the device will enter the IDLE Mode. In the IDLE1 Mode the CPU will stop running but the system clock will continue to keep the peripheral functions operational, if FSYSON bit is high. If FSYSON bit is low, the CPU and the system clock will all stop in IDLE0 mode. If the bit is low the device will enter the SLEEP Mode when a HALT instruction is executed.

Bit 0 **HLCLK**: System Clock Selection
0: $f_H/2$~$f_H/64$ or $f_{SUB}$
1: $f_H$
This bit is used to select if the $f_H$ clock or the $f_H/2$~$f_H/64$ or $f_{SUB}$ clock is used as the system clock. When the bit is high the $f_H$ clock will be selected and if low the $f_H/2$~$f_H/64$ or $f_{SUB}$ clock will be selected. When system clock switches from the $f_H$ clock to the $f_{SUB}$ clock and the $f_H$ clock will be automatically switched off to conserve power.

### CTRL Register

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

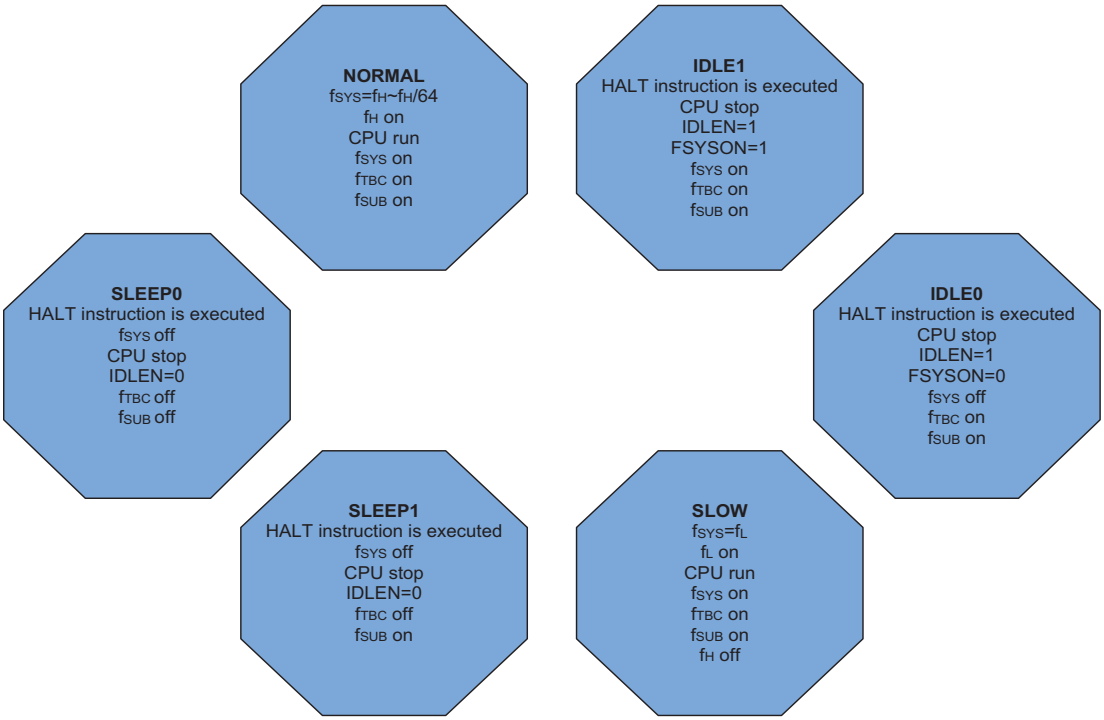
Bit 7 **FSYSON**: $f_{SYS}$ Control in IDLE Mode
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as 0.

Bit 2 **LVRF**: LVR function reset flag
0: Not occur
1: Occurred
This bit is set to 1 when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.

Bit 1 **LRF**: LVR Control register software reset flag
0: Not occur
1: Occurred
This bit is set to 1 if the LVRC register contains any non defined LVR voltage register values. This in effect acts like a software reset function. This bit can only be cleared to 0 by the application program.

Bit 0 **WRF**: WDT Control register software reset flag
0: Not occur
1: Occurred
This bit is set to 1 by the WDT Control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.




## Operating Mode Switching

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

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

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

### NORMAL Mode to SLOW Mode Switching

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

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

### SLOW Mode to NORMAL Mode Switching

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

### Entering the SLEEP0 Mode

There is only one way for the device to enter the SLEEP0 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "0" and the WDT and LVD both off. When this instruction is executed under the conditions described above, the following will occur:

- The system clock, WDT clock and Time Base clock will be stopped 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 stopped.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.

### Entering the SLEEP1 Mode

There is only one way for the device to enter the SLEEP1 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "0" and the WDT or LVD on. When this instruction is executed under the conditions described above, the following will occur:

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

### Entering the IDLE0 Mode

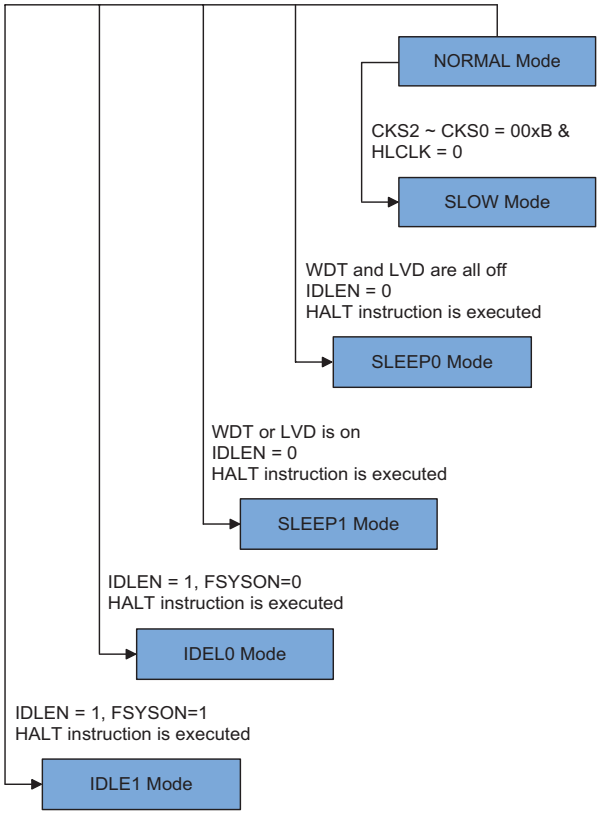
There is only one way for the device to enter the IDLE0 Mode and that is to execute the "HALT" instruction in the application program with the IDLEN bit in SMOD register equal to "1" and the FSYSON bit in CTRL register equal to "0". When this instruction is executed under the conditions described above,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

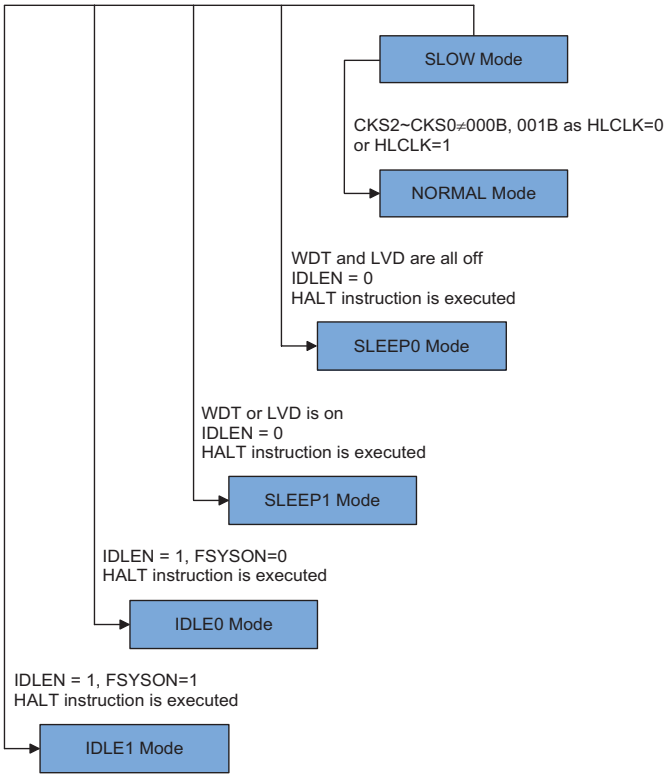
- The system clock will be stopped and the application program will stop at the "HALT" instruction, but the Time Base clock $f_{TBC}$ and the Watchdog Timer clock $f_S$ will be on.
- The Data Memory contents and registers will maintain their present condition.
- The WDT will be cleared and resume counting.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

### Entering the IDLE1 Mode

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

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


NORMAL Mode
CKS2 ~ CKS0 = 00xB &
HLCLK = 0
SLOW Mode
WDT and LVD are all off
IDLEN = 0
HALT instruction is executed
SLEEP0 Mode
WDT or LVD is on
IDLEN = 0
HALT instruction is executed
SLEEP1 Mode
IDLEN = 1, FSYSON=0
HALT instruction is executed
IDEL0 Mode
IDLEN = 1, FSYSON=1
HALT instruction is executed
IDLE1 Mode


SLOW Mode
CKS2~CKS0≠000B, 001B as HLCLK=0
or HLCLK=1
NORMAL Mode
WDT and LVD are all off
IDLEN = 0
HALT instruction is executed
SLEEP0 Mode
WDT or LVD is on
IDLEN = 0
HALT instruction is executed
SLEEP1 Mode
IDLEN = 1, FSYSON=0
HALT instruction is executed
IDLE0 Mode
IDLEN = 1, FSYSON=1
HALT instruction is executed
IDLE1 Mode

## Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to devices which have different package types, as there may be unbonbed pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. In the IDLE1 Mode the system oscillator is on, if the system oscillator is from the high speed system oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps

## Wake-up

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

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

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

Each pin on Port A can be setup using the PAWU register to permit a negative transition on the pin to wake-up the system. When a Port A pin wake-up occurs, the program will resume execution at the instruction following the “HALT” instruction. If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the “HALT” instruction. In this situation, the interrupt which woke-up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

## Watchdog Timer

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

### Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal $f_S$ clock which is in turn supplied by the LIRC oscillator. The Watchdog Timer source clock is then subdivided by a ratio of $2^8$ to $2^{18}$ to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register. The LIRC internal oscillator has an approximate period of 32kHz at a supply voltage of 5V. However, it should be noted that this specified internal clock period can vary with $V_{DD}$, temperature and process variations.

### Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable/disable operation. The WDTC register is initiated to 01010011B at any reset but keeps unchanged at the WDT time-out occurrence in a power down state.

**WDTC Register**

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

Bit 7~3 **WE4~WE0**: WDT function software control
10101: Disabled
01010: Enabled
Other: Reset MCU
When these bits are changed by the environmental noise to reset the microcontroller, the reset operation will be activated after 2~3 LIRC clock cycles and the WRF bit in the CTRL register will be set to 1.

Bit 2~0 **WS2~WS0**: WDT Time-out period selection
000: $2^8/f_{SUB}$
001: $2^{10}/f_{SUB}$
010: $2^{12}/f_{SUB}$
011: $2^{14}/f_{SUB}$
100: $2^{15}/f_{SUB}$
101: $2^{16}/f_{SUB}$
110: $2^{17}/f_{SUB}$
111: $2^{18}/f_{SUB}$

**CTRL Register**

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

Bit 7 **FSYSON:** $f_{SYS}$ Control in IDLE Mode
Describe elsewhere.

Bit 6~3 Unimplemented, read as "0"

Bit 2 **LVRF:** LVR function reset flag
Describe elsewhere.

Bit 1 **LRF:** LVR Control register software reset flag
Describe elsewhere.

Bit 0 **WRF**: WDT Control register software reset flag
0: Not occur
1: Occurred
This bit is set to 1 by the WDT Control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

## Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, the clear WDT instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. There are five bits, WE4~WE0, in the WDTC register to enable/disable the WDT function. The WDT function will be disabled when the WE4~WE0 bits are set to a value of 10101B while the WDT function will be enabled if the WE4~WE0 bits are equal to 01010B. However, if the WE4~WE0 bits are changed to any other values except 01010B and 10101B, which is caused by the environmental noise, it will reset the microcontroller after 2~3 LIRC clock cycles.

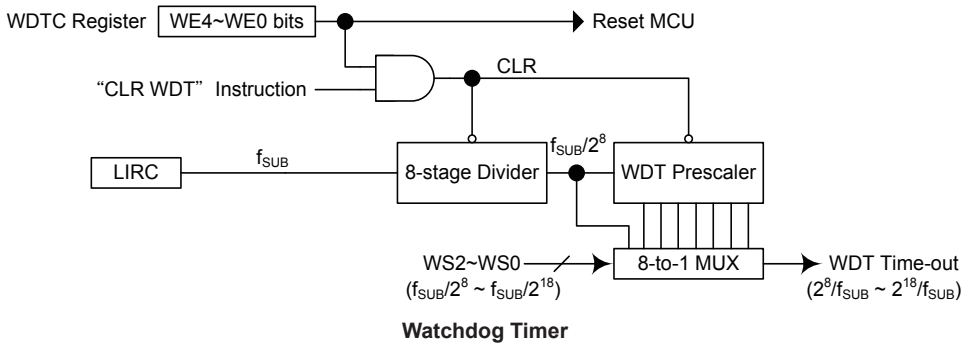
| WE4~WE0 Bits | WDT Function |
|---|---|
| 10101B | Disable |
| 01010B | Enable |
| Any other value | Reset MCU |

**Watchdog Timer Enable/Disable Control**

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

There is only one method of using software instruction to clear the Watchdog Timer. That is to use the single “CLR WDT” instruction to clear the WDT.

The maximum time-out period is when the $2^{18}$ division ratio is selected. As an example, with a 32kHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 seconds for the $2^{18}$ division ratio, and a minimum timeout of 7.8ms for the $2^{8}$ division ration.




**Watchdog Timer**

## Reset and Initialisation

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

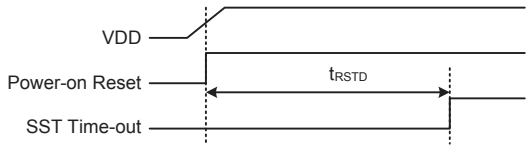
Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold.

### Reset Functions

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

#### Power-on Reset

The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. All the I/O port and port control registers will power up in a high condition ensuring that all pins will be first set to inputs.

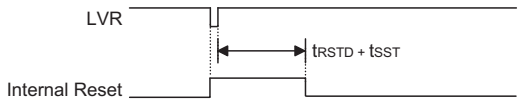



Note: $t_{RSTD}$ is power-on delay, typical time=50ms

**Power-On Reset Timing Chart**

### Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device. The LVR function is always enabled with a specific LVR voltage, $V_{LVR}$. 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 and the LVRF bit in the CTRL register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~$V_{LVR}$ must exist for a time greater than that specified by $t_{LVR}$ in the A.C. characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual $V_{LVR}$ is fixed at a voltage value of 2.1V by the LVS bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by the environmental noise, the LVR will reset the device after 2~3 LIRC clock cycles. When this happens, the LRF bit in the CTRL register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the power down mode.




Note: $t_{RSTD}$ is power-on delay, typical time=16.7ms

**Low Voltage Reset Timing Chart**

- **LVRC Register**

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

Bit 7~0 LVS7~LVS0: LVR voltage select
01010101: 2.1V
00110011: 2.1V
10011001: 2.1V
10101010: 2.1V
Any other value: Generates MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by the above defined LVR voltage value, an MCU reset will be generated. The reset operation will be activated after 2~3 LIRC clock cycles. In this situation this register contents will remain the same after such a reset occurs.

Any register value, other than the four defined values above, will also result in the generation of an MCU reset. The reset operation will be activated after 2~3 LIRC clock cycles. However in this situation this register contents will be reset to the POR value.

• **CTRL Register**

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

Bit 7 **FSYSON:** $f_{SYS}$ Control in IDLE Mode
Describe elsewhere.

Bit 6~3 Unimplemented, read as "0"

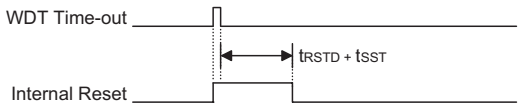
Bit 2 **LVRF**: LVR function reset flag
0: Not occur
1: Occurred
This bit is set to 1 when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.

Bit 1 **LRF**: LVR Control register software reset flag
0: Not occur
1: Occurred
This bit is set to 1 if the LVRC register contains any non defined LVR voltage register values. This in effect acts like a software reset function. This bit can only be cleared to 0 by the application program.

Bit 0 **WRF:** WDT Control register software reset flag
Describe elsewhere.

### Watchdog Time-out Reset During Normal Operation

The Watchdog time-out Reset during normal operation is the same as a LVR reset except that the Watchdog time-out flag TO will be set to "1".

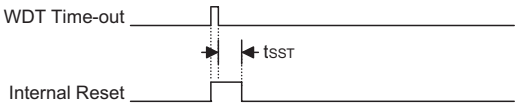



Note: $t_{RSTD}$ is power-on delay, typical time=16.7ms

**WDT Time-out Reset During Normal Operation Timing Chart**

### Watchdog Time-out Reset During SLEEP1 or IDLE Mode

The Watchdog time-out Reset during SLEEP1 or IDLE Mode is a little different from other kinds of reset. Most of the conditions remain unchanged except that the Program Counter and the Stack Pointer will be cleared to "0" and the TO flag will be set to "1". Refer to the A.C. Characteristics for $t_{SST}$ details.




Note: The $t_{SST}$ is 16 clock cycles if the system clock source is provided by the HIRC. The $t_{SST}$ is 2 clock for the LIRC.

**WDT Time-out Reset During SLEEP1 or IDLE 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 SLEEP or IDLE Mode function or Watchdog Timer. The reset flags are shown in the table:

| TO | PDF | RESET Conditions |
|---|---|---|
| 0 | 0 | Power-on reset |
| u | u | LVR reset during NORMAL or SLOW Mode operation |
| 1 | u | WDT time-out reset during NORMAL or SLOW Mode operation |
| 1 | 1 | WDT time-out reset during IDLE or SLEEP1 Mode operation |

"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 Modules | Timer Modules will be turned off |
| Input/Output Ports | I/O ports will be setup as inputs and AN0~AN3 as A/D input pins |
| 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.

| Register | Power On Reset | WDT Time-out (Normal Operation) | LVR Reset | WDT Time-out (IDLE/SLEEP1 mode) |
|---|---|---|---|---|
| MP0 | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| MP1 | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| ACC | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| PCL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 |
| TBLP | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBLH | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBHP | ---- -xxx | ---- -uuu | ---- -uuu | ---- -uuu |
| STATUS | --00 xxxx | --1u uuuu | --uu uuuu | --11 uuuu |
| SMOD | 110- 0010 | 110- 0010 | 110- 0010 | uuu- uuuu |
| LVDC | --00 -000 | --00 -000 | --00 -000 | --uu -uuu |
| INTEG | ---- 0000 | ---- 0000 | ---- 0000 | ---- uuuu |
| INTC0 | -000 0000 | -000 0000 | -000 0000 | -uuu uuuu |
| INTC1 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| INTC2 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MFI0 | --00 --00 | --00 --00 | --00 --00 | --uu --uu |
| MFI1 | --00 --00 | --00 --00 | --00 --00 | --uu --uu |
| MFI2 | --00 --00 | --00 --00 | --00 --00 | --uu --uu |
| MFI3 | --00 --00 | --00 --00 | --00 --00 | --uu --uu |
| MFI4 | ---0 ---0 | ---0 ---0 | ---0 ---0 | ---u ---u |
| PA | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAC | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAPU | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PAWU | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PB | ---1 1111 | ---1 1111 | ---1 1111 | ---u uuuu |
| PBC | ---1 1111 | ---1 1111 | ---1 1111 | ---u uuuu |
| PBPU | ---0 0000 | ---0 0000 | ---0 0000 | ---u uuuu |
| TMPC | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| WDTC | 0101 0011 | 0101 0011 | 0101 0011 | uuuu uuuu |
| TBC | 0011 -111 | 0011 -111 | 0011 -111 | uuuu -uuu |
| ADRL(ADRFS=0) | xxxx ---- | xxxx ---- | xxxx ---- | uuuu ---- |
| ADRL(ADRFS=1) | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| ADRH(ADRFS=0) | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| ADRH(ADRFS=1) | ---- xxxx | ---- xxxx | ---- xxxx | ---- uuuu |
| ADCR0 | 0110 --00 | 0110 --00 | 0110 --00 | uuuu --uu |
| ADCR1 | 00-0 -000 | 00-0 -000 | 00-0 -000 | uu-u -uuu |
| ACERL | ---- 1111 | ---- 1111 | ---- 1111 | ---- uuuu |
| CTRL | 0--- -x00 | 0--- -000 | 0--- -000 | u--- -uuu |
| LVRC | 0101 0101 | 0101 0101 | 0101 0101 | uuuu uuuu |
| TM0C0 | 0000 0--- | 0000 0--- | 0000 0--- | uuuu u--- |
| TM0C1 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM0DL | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM0DH | 0000 0000 | 0000 0000 | 0000 0000 | 0000 uuuu |
| TM0AL | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM0AH | 0000 0000 | 0000 0000 | 0000 0000 | 0000 uuuu |
| TM0RPL | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM0RPH | 0000 0000 | 0000 0000 | 0000 0000 | 0000 uuuu |
| TM1C0 | 0000 0--- | 0000 0--- | 0000 0--- | uuuu u--- |

| Register | Power On Reset | WDT Time-out (Normal Operation) | LVR Reset | WDT Time-out (IDLE/SLEEP1 mode) |
|---|---|---|---|---|
| TM1C1 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM1DL | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM1DH | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| TM1AL | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM1AH | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| TM1RPL | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM1RPH | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| TM2C0 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM2C1 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM2DL | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM2DH | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| TM2AL | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM2AH | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| TM3C0 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM3C1 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM3DL | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM3DH | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| TM3AL | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM3AH | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| CPR | ---0 0000 | ---0 0000 | ---0 0000 | ---u uuuu |
| OVPREF | 0-00 0000 | 0-00 0000 | 0-00 0000 | u-uu uuuu |
| OCVPR0 | -0-- 00-0 | -0-- 00-0 | -0-- 00-0 | -u-- uu-u |
| OCVPR1 | -011 x-00 | -011 x-00 | -011 x-00 | -uuu u-uu |
| OCVPR2 | 0010 0000 | 0010 0000 | 0010 0000 | uuuu uuuu |

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

## Input/Output Ports

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

The device provides bidirectional input/output lines labeled with port names PA and PB. 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.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PA | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PAC | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PAPU | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PAWU | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PB | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PBC | — | — | — | D4 | D3 | D2 | D1 | D0 |
| PBPU | — | — | — | D4 | D3 | D2 | D1 | D0 |

**I/O Resistor Lists**

### Pull-high Resistors

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

**PAPU Register**

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

Bit 7~0 I/O Port A bit 7~bit 0 Pull-High Control
0: Disable
1: Enable

**PBPU Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | D4 | D3 | D2 | D1 | D0 |
| R/W | — | — | — | R/W | R/W | R/W | R/W | R/W |
| POR | — | — | — | 0 | 0 | 0 | 0 | 0 |

Bit 7~5 Unimplemented, read as "0"

Bit 4~0 I/O Port B bit 4~bit 0 Pull-High Control
0: Disable
1: Enable

## Port A Wake-up

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

### PAWU Register

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

Bit 7~0 I/O Port A bit 7~bit 0 Wake Up Control
0: Disable
1: Enable

## I/O Port Control Registers

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

### PAC Register

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

Bit 7~0 I/O Port A bit 7~bit 0 Input/Output Control
0: Output
1: Input

### PBC Register

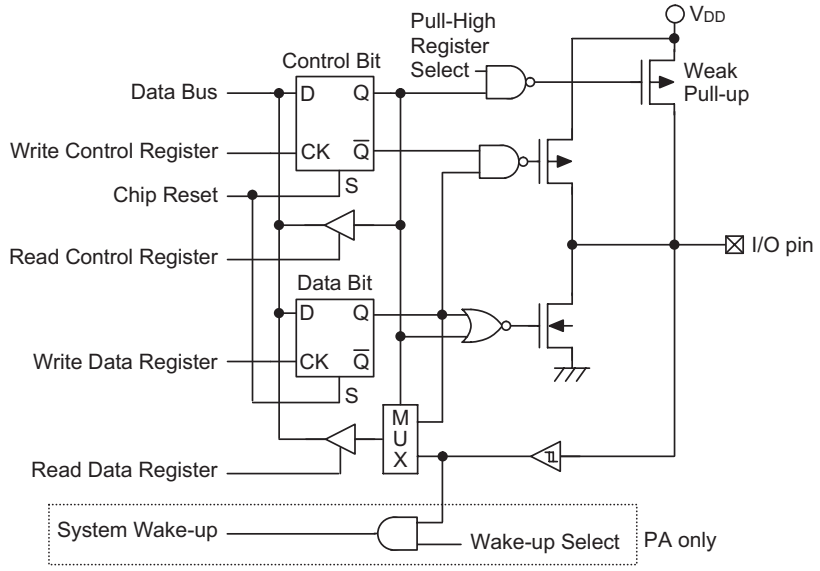
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | D4 | D3 | D2 | D1 | D0 |
| R/W | — | — | — | R/W | R/W | R/W | R/W | R/W |
| POR | — | — | — | 1 | 1 | 1 | 1 | 1 |

Bit 7~5 Unimplemented, read as "0"

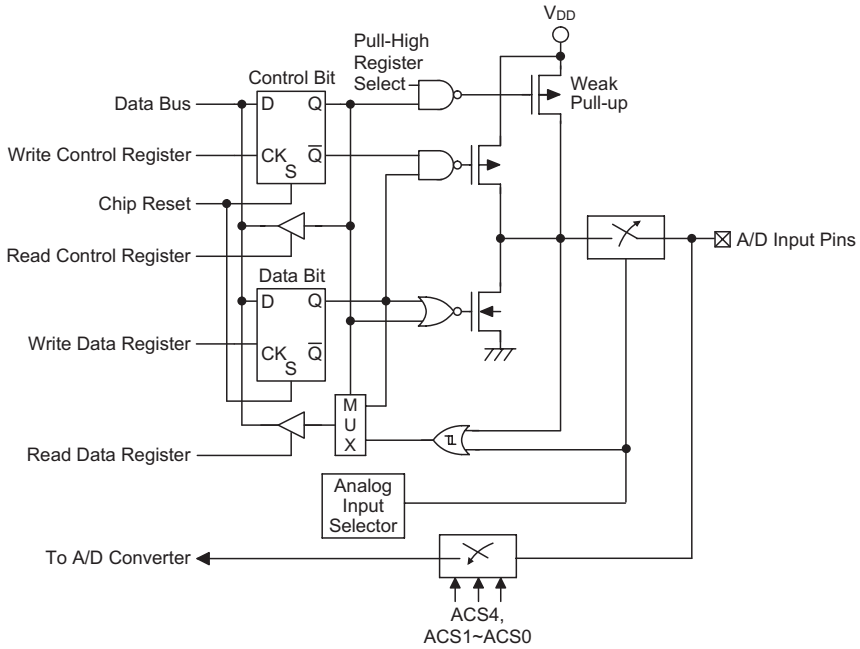
Bit 4~0 I/O Port B bit 4~bit 0 Input/Output Control
0: Output
1: Input

## I/O Pin Structures

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



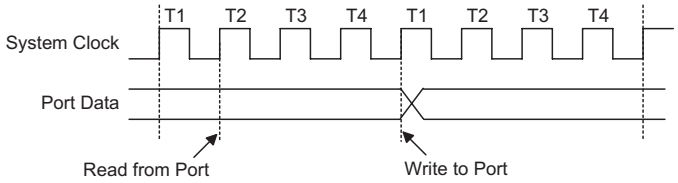

**Generic Input/Output Structure**




**A/D Input/Output Structure**

### Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers, PAC~PBC, 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, are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate port control register or by programming individual bits in the port control register using the "SET [m].i" and "CLR [m].i" instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.




**Read/Wite Timing**

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

## Timer Modules – TM

One of the most fundamental functions in any microcontroller device is the ability to control and measure time. To implement time related functions the device includes several Timer Modules, abbreviated to the name TM. The TMs are multi-purpose timing units and serve to provide operations such as Timer/Counter, Input Capture, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has two individual interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

The common features of the different TM types are described here with more detailed information provided in the individual Periodic and Compact TM sections.

### Introduction

The device contains four TMs with each TM having a reference name of TM0, TM1, TM2 and TM3. The TM0 and TM1 are 16-bit and 10-bit Periodic Type TMs (PTM) respectively while the TM2 and TM3 are 10-bit Compact Type TMs (CTM). The common features to the Periodic and Compact TMs will be described in this section and the detailed operation will be described in corresponding section. The main features of the Compact and Periodic TMs are summarised in the accompanying table.

| Function | CTM | PTM |
|---|---|---|
| Timer/Counter | √ | √ |
| I/P Capture | — | √ |
| Compare Match Output | √ | √ |
| PWM Channels | 1 | 1 |
| Single Pulse Output | — | 1 |
| PWM Alignment | Edge | Edge |
| PWM Adjustment Period & Duty | Duty or Period | Duty or Period |

**TM Function Summary**

| TM0 | TM1 | TM2 | TM3 |
|---|---|---|---|
| 16-bit PTM | 10-bit PTM | 10-bit CTM | 10-bit CTM |

**TM Name/Type Reference**

## TM Operation

The two different types of TMs offer a diverse range of functions, from simple timing operations to PWM signal generation. The key to understanding how the TM operates is to see it in terms of a free running counter whose value is then compared with the value of pre-programmed internal comparators. When the free running counter has the same value as the pre-programmed comparator, known as a compare match situation, a TM interrupt signal will be generated which can clear the counter and perhaps also change the condition of the TM output pin. The internal TM counter is driven by a user selectable clock source, which can be an internal clock or an external pin.

## TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the TnCK2~TnCK0 bits in the TM control registers. The clock source can be a ratio of the system clock $f_{SYS}$ or the internal high clock $f_H$, the $f_{TBC}$ clock source or the external TCKn pin. Note that setting these bits to the value 101 will select a reserved clock input in effect disconnecting the TM clock source except the clock source selection for TM1. The TCKn pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

## TM Interrupts

The Periodic and Compact type TMs both have two internal interrupts, one for each of the internal comparator A or comparator P, which generate a TM interrupt when a compare match condition occurs. When a TM interrupt is generated, it can be used to clear the counter and also to change the state of the TM output pin.

## TM External Pins

Each of the TMs, irrespective of what type, has one TM input pin, with the label TCKn. The TM input pin, is essentially a clock source for the TM and is selected using the TnCK2~TnCK0 bits in the TMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. This external TM input pin is shared with other functions but will be connected to the internal TM if selected using the TnCK2~TnCK0 bits. The TM input pin can be chosen to have either a rising or falling active edge.

The TMs each have one or more output pins. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external TPn output pin is also the pin where the TM generates the PWM output waveform. As the TM output pins are pin-shared with other function, the TM output function must first be setup using registers. A single bit in one of the registers determines if its associated pin is to be used as an external TM output pin or if it is to have another function. The number of output pins for each TM type is different, the details are provided in the accompanying table.

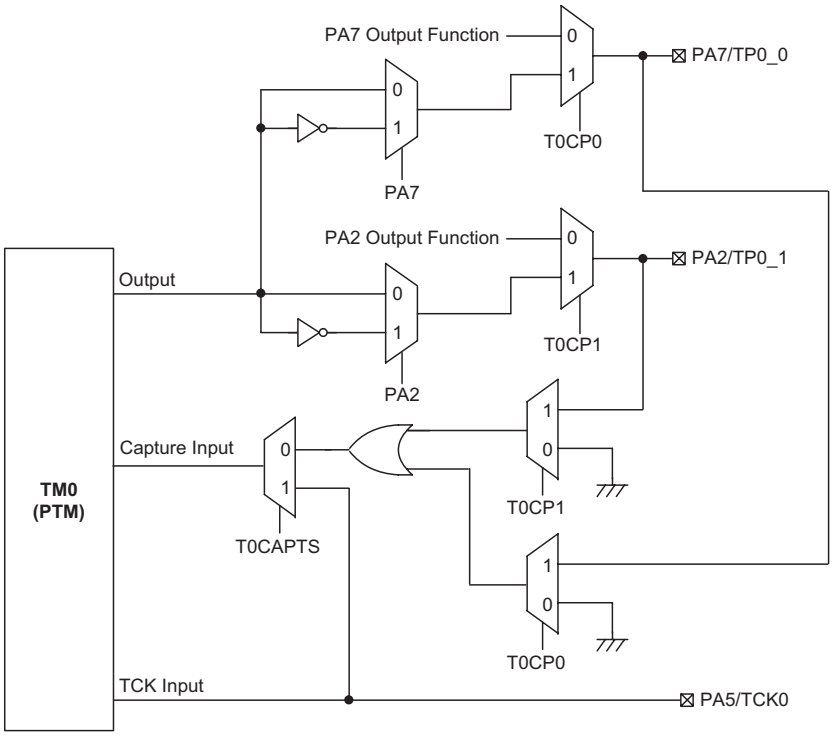
The TM output pin names have a "_n" suffix. Pin names that include a "_0" or "_1" suffix indicate that they are from a TM with multiple output pins. This allows the TM to generate a complimentary output pair, selected using the I/O register data bits.

| TM0 | TM1 | TM2 | TM3 | Register |
|---|---|---|---|---|
| TP0_0, TP0_1 | TP1_0, TP1_1 | TP2_0, TP2_1 | TP3 | TMPC |

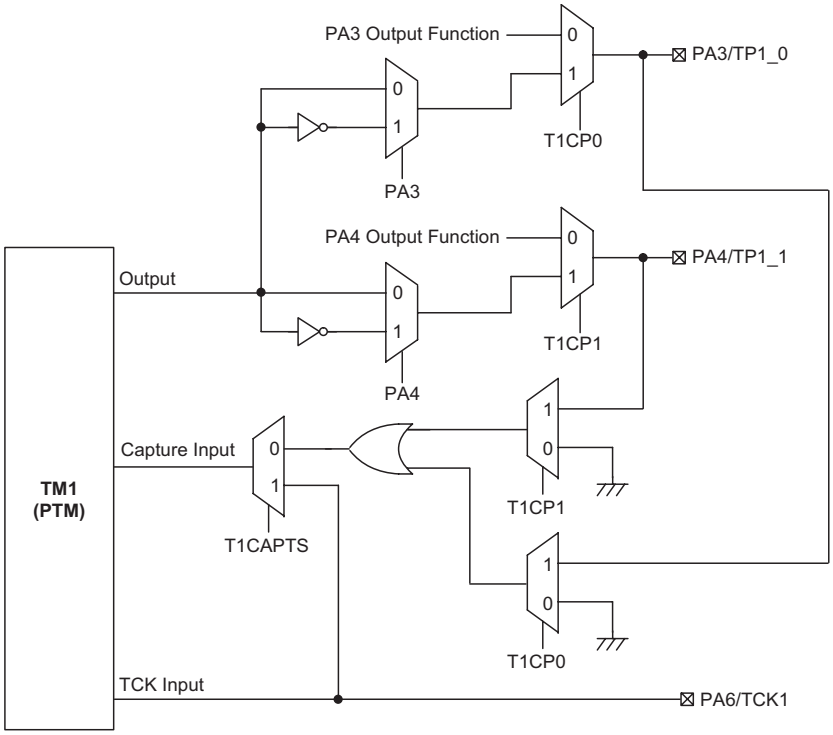
**TM Output Pins**

## TM Input/Output Pin Control Register

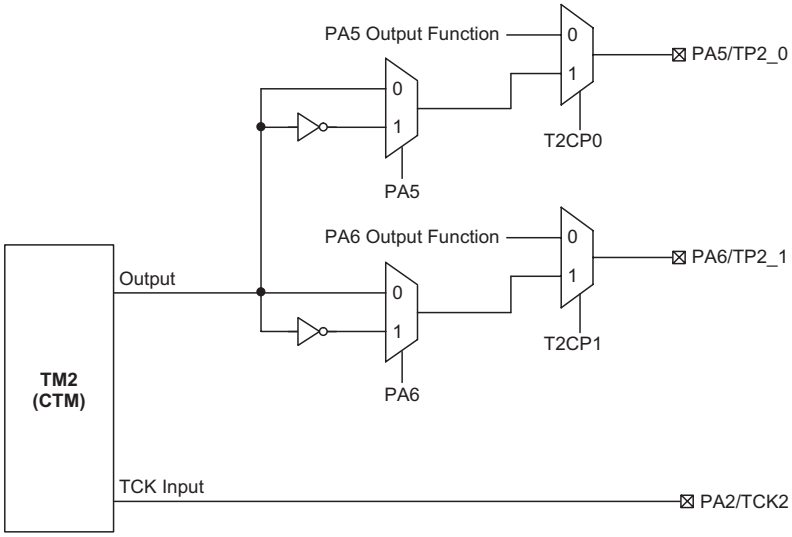
Selecting to have a TM input/output or whether to retain its other shared function is implemented using the control register, with a single bit in the register corresponding to a TM input/output pin. Setting the bit high will setup the corresponding pin as a TM input/output, if reset to zero the pin will retain its original other function.



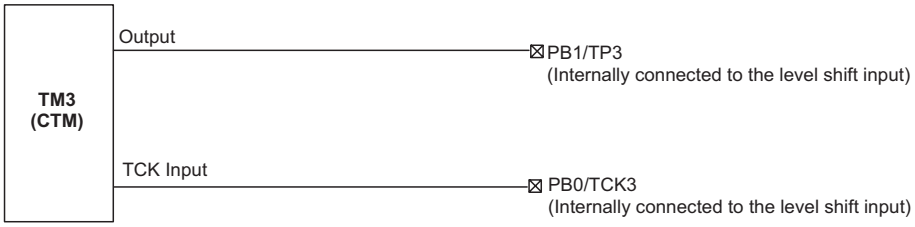

**TM0 Function Pin Control Block Diagram**




**TM1 Function Pin Control Block Diagram**




**TM2 Function Pin Control Block Diagram**




**TM3 Function Pin Control Block Diagram**

### TMPC Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | OUTCP1 | OUTCP0 | T21CP1 | T2CP0 | T1CP1 | T1CP0 | T0CP1 | T0CP0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7~6 **OUTCP [1:0]:** OUTH and OUTL pin control
00: Normal I/O function, i.e., PA3 and PA4
01: OUTH and PA4
10: PA3 and OUTL
11: OUTH and OUTL
If these bits are set to "11", the dead time circuitry will be automatically enabled. If these bits are set to a value except "11", then the dead time circuitry will be automatically disabled.

Bit 5 **T2CP1:** TP2_1 pin control
0: TP2_1 pin is disabled
1: TP2_1 pin is enabled

Bit 4 **T2CP0:** TP2_0 pin control
0: TP2_0 pin is disabled
1: TP2_0 pin is enabled

Bit 3 **T1CP1:** TP1_1 pin control
0: TP1_1 pin is disabled
1: TP1_1 pin is enabled

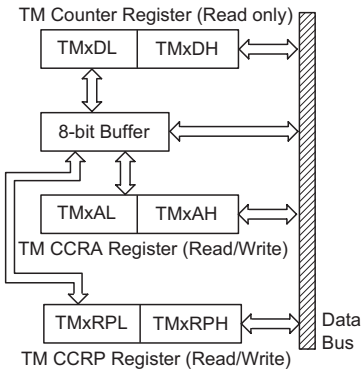
Bit 2 **T1CP0:** TP1_0 pin control
0: TP1_0 pin is disabled
1: TP1_0 pin is enabled

Bit 1 **T0CP1:** TP0_1 pin control
0: TP0_1 pin is disabled
1: TP0_1 pin is enabled

Bit 0 **T0CP0:** TP0_0 pin control
0: TP0_0 pin is disabled
1: TP0_0 pin is enabled

## Programming Considerations

The TM Counter Registers, the Capture/Compare CCRA registers and the CCRP registers, being either 16-bit or 10-bit, all have a low and high byte structure. The high bytes can be directly accessed, but as the low bytes can only be accessed via an internal 8-bit buffer, reading or writing to these register pairs must be carried out in a specific way. The important point to note is that data transfer to and from the 8-bit buffer and its related low byte only takes place when a write or read operation to its corresponding high byte is executed. As the CCRA and CCRP registers are implemented in the way shown in the following diagram and accessing these register pairs is carried out in a specific way described above, it is recommended to use the "MOV" instruction to access the CCRA or CCRP low byte registers, named TMxAL or TMxRPL, using the following access procedures. Accessing the CCRA or CCRP low byte register without following these access procedures will result in unpredictable values.




The following steps show the read and write procedures:

- Writing Data to CCRA or CCRP
  - Step 1. Write data to Low Byte TMxAL or TMxRPL
    – note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte TMxAH or TMxRPH
    – here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and CCRA or CCRP
  - Step 1. Read data from the High Byte TMxDH, TMxAH or TMxRPH
    – here data is read directly from the High Byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
  - Step 2. Read data from the Low Byte TMxDL, TMxAL or TMxRPL
    – this step reads data from the 8-bit buffer.

## Compact Type TM – CTM (TM2, TM3)

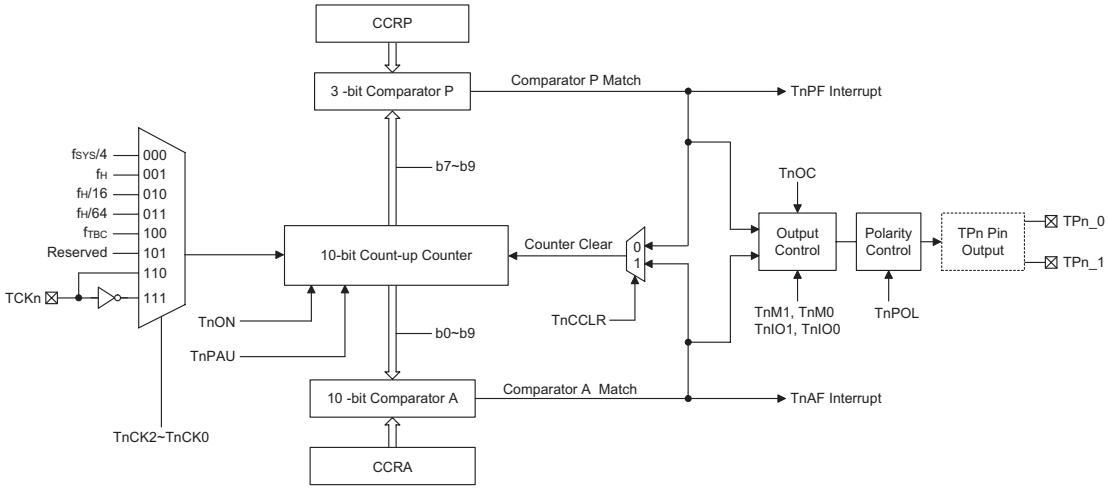
Although the simplest form of the TM types, the Compact TM type still contains three operating modes, which are Compare Match Output, Timer/Event Counter and PWM Output modes. The Compact TM can also be controlled with an external input pin and can drive one or two external output pins. These two external output pins can be the same signal or the inverse signal.

| Name | TM No. | TM Input Pin | TM Output Pin |
|---|---|---|---|
| 10-bit CTM | 2, 3 | TCK2, TCK3 | TP2_0, TP2_1; TP3 |

### Compact TM Operation

At its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP is three bits wide whose value is compared with the highest three bits in the counter while the CCRA is the ten bits and therefore compares with all counter bits.

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the TnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a TM interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control an output pin. All operating setup conditions are selected using relevant internal registers.




**Compact Type TM Block Diagram (n=2 or 3)**

Note: For TM3 there is only one TM output signal named TP3 and the TP3.

## Compact Type TM Register Description

Overall operation of the Compact TM is controlled using six registers. A read only register pair exists to store the internal counter 10-bit value, while a read/write register pair exists to store the internal 10-bit CCRA value. The remaining two registers are control registers which setup the different operating and control modes as well as the three CCRP bits.

| Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| TMnC0 | TnPAU | TnCK2 | TnCK1 | TnCK0 | TnON | TnRP2 | TnRP1 | TnRP0 |
| TMnC1 | TnM1 | TnM0 | TnIO1 | TnIO0 | TnOC | TnPOL | TnDPX | TnCCLR |
| TMnDL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TMnDH | — | — | — | — | — | — | D9 | D8 |
| TMnAL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TMnAH | — | — | — | — | — | — | D9 | D8 |

**Compact TM Register List (n=2 or 3)**

### TMnDL Register

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

Bit 7~0 TMn Counter Low Byte Register bit 7~bit 0
TMn 10-bit Counter bit 7~bit 0

### TMnDH Register

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

Bit 7~2 Unimplemented, read as "0"
Bit 1~0 TMn Counter High Byte Register bit 1~bit 0
TMn 10-bit Counter bit 9~bit 8

### TMnAL Register

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

Bit 7~0 TMn CCRA Low Byte Register bit 7~bit 0
TMn 10-bit CCRA bit 7~bit 0

### TMnAH Register

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

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 TMn CCRA High Byte Register bit 1~bit 0
TMn 10-bit CCRA bit 9~bit 8

### TMnC0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TnPAU | TnCK2 | TnCK1 | TnCK0 | TnON | TnRP2 | TnRP1 | TnRP0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7 **TnPAU:** TMn Counter Pause Control
0: Run
1: Pause
The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the TM will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **TnCK2~TnCK0:** Select TMn Counter clock
000: $f_{SYS}/4$
001: $f_H$
010: $f_H/16$
011: $f_H/64$
100: $f_{TBC}$
101: Undefined
110: TCKn rising edge clock
111: TCKn falling edge clock
These three bits are used to select the clock source for the TM. Selecting the undefined clock input will effectively disable the internal counter. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source $f_{SYS}$ is the system clock, while $f_H$ and $f_{TBC}$ are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **TnON:** TMn Counter On/Off Control
0: Off
1: On
This bit controls the overall on/off function of the TM. Setting the bit high enables the counter to run and clearing the bit disables the TM. Clearing this bit to zero will stop the counter from counting and turn off the TM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value.
If the TM is in the Compare Match Output Mode then the TM output pin will be reset to its initial condition, as specified by the TnOC bit, when the TnON bit changes from low to high.

Bit 2~0 **TnRP2~TnRP0:** TMn CCRP 3-bit register, compare with the TMn counter bit 9~bit 7
000: 1024 TMn clocks
001: 128 TMn clocks
010: 256 TMn clocks
011: 384 TMn clocks
100: 512 TMn clocks
101: 640 TMn clocks
110: 768 TMn clocks
111: 896 TMn clocks

These three bits are used to setup the value on the internal CCRP 3-bit register, which are then compared with the internal counter's highest three bits. The result of this comparison can be selected to clear the internal counter if the TnCCLR bit is set to zero. Setting the TnCCLR bit to zero ensures that a compare match with the CCRP values will reset the internal counter. As the CCRP bits are only compared with the highest three counter bits, the compare values exist in 128 clock cycle multiples. Clearing all three bits to zero is in effect allowing the counter to overflow at its maximum value.

### TMnC1 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TnM1 | TnM0 | TnIO1 | TnIO0 | TnOC | TnPOL | TnDPX | TnCCLR |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7~6 **TnM1~TnM0:** Select TMn Operation Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the TM. To ensure reliable operation the TM should be switched off before any changes are made to the TnM1 and TnM0 bits. In the Timer/Counter Mode, the TM output pin control must be disabled.

Bit 5~4 **TnIO1~TnIO0:** Select TPn_0, TPn_1 or TPn output function
Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output
PWM Mode
00: PWM Output inactive state
01: PWM Output active state
10: PWM output
11: Undefined
Timer/Counter Mode
Unused.

These two bits are used to determine how the TM output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the TM is running.

In the Compare Match Output Mode, the TnIO1 and TnIO0 bits determine how the TM output pin changes state when a compare match occurs from the Comparator A. The TM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the TM output pin should be setup using the TnOC bit in the TMnC1 register. Note that the output level requested by the TnIO1 and TnIO0 bits must be different from the initial value setup using the TnOC bit otherwise no change will occur on the TM output pin when a compare match occurs. After the TM output pin changes state, it can be reset to its initial level by changing the level of the TnON bit from low to high.

In the PWM Mode, the TnIO1 and TnIO0 bits determine how the TM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to change the values of the TnIO1 and TnIO0 bits only after the TM has been switched off. Unpredictable PWM outputs will occur if the TnIO1 and TnIO0 bits are changed when the TM is running.

Bit 3 **TnOC:** TPn_0, TPn_1 or TPn output control bit
Compare Match Output Mode
0: Initial low
1: Initial high
PWM Mode
0: Active low
1: Active high

This is the output control bit for the TM output pin. Its operation depends upon whether TM is being used in the Compare Match Output Mode or in the PWM Mode. It has no effect if the TM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the TM output pin before a compare match occurs. In the PWM Mode it determines if the PWM signal is active high or active low.

Bit 2 **TnPOL:** TPn_0, TPn_1 or TPn output Polarity control
0: Non-invert
1: Invert

This bit controls the polarity of the TPn_0, TPn_1 or TPn output pin. When the bit is set high the TM output pin will be inverted and not inverted when the bit is zero. It has no effect if the TM is in the Timer/Counter Mode.

Bit 1 **TnDPX:** TMn PWM period/duty Control
0: CCRP – period; CCRA – duty
1: CCRP – duty; CCRA – period

This bit determines which of the CCRA and CCRP registers are used for period and duty control of the PWM waveform.

Bit 0 **TnCCLR:** Select TMn Counter clear condition
0: TMn Comparator P match
1: TMn Comparator A match

This bit is used to select the method which clears the counter. Remember that the Compact TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the TnCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The TnCCLR bit is not used in the PWM Mode or Input Capture Mode.

## Compact Type TM Operation Modes

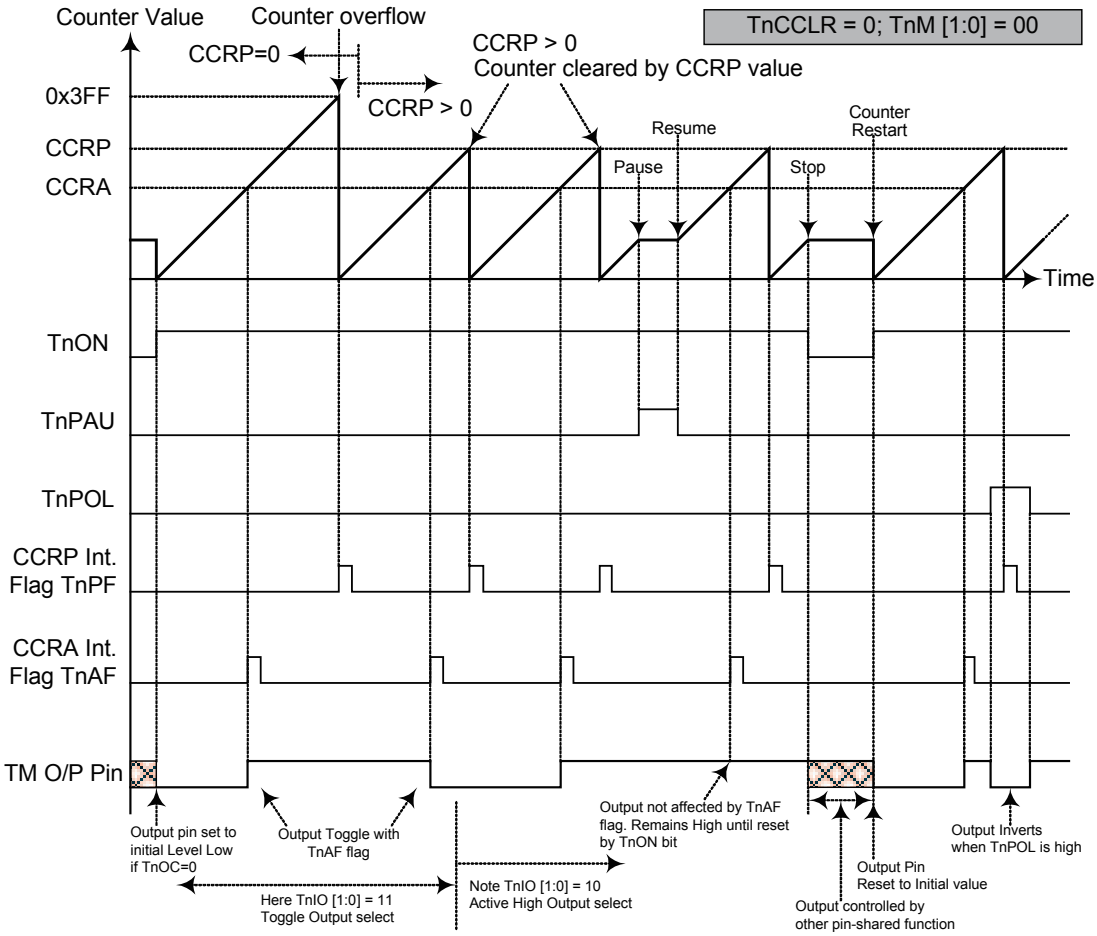
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Mode or Timer/Counter Mode. The operating mode is selected using the TnM1 and TnM0 bits in the TMnC1 register.

### Compare Match Output Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register, should be set to "00" respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the TnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match occurs from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both TnAF and TnPF interrupt request flags for the Comparator A and Comparator P respectively, will both be generated.

If the TnCCLR bit in the TMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the TnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when TnCCLR is high no TnPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, value, however here the TnAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the TM output pin will change state. The TM output pin condition however only changes state when a TnAF interrupt request flag is generated after a compare match occurs from Comparator A. The TnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the TM output pin. The way in which the TM output pin changes state are determined by the condition of the TnIO1 and TnIO0 bits in the TMnC1 register. The TM output pin can be selected using the TnIO1 and TnIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the TM output pin, which is setup after the TnON bit changes from low to high, is setup using the TnOC bit. Note that if the TnIO1 and TnIO0 bits are zero then no pin change will take place.

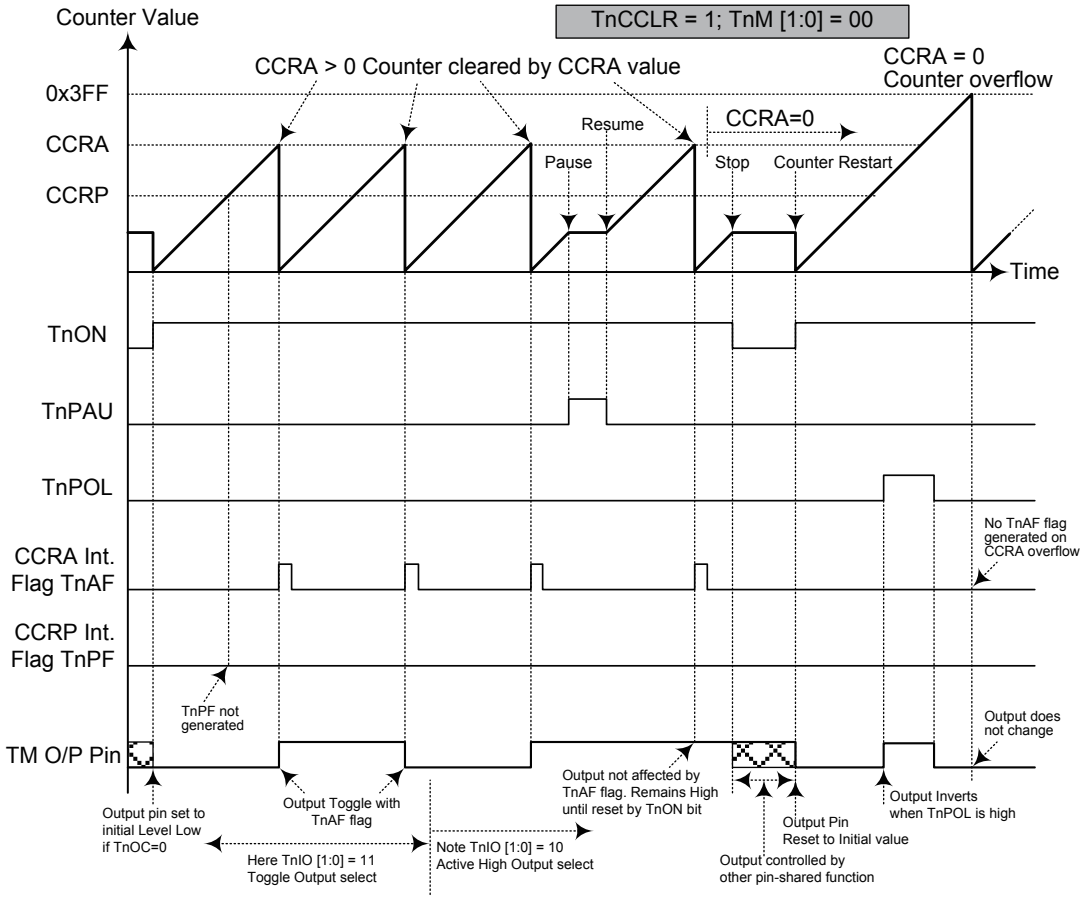



**Compare Match Output Mode – TnCCLR=0**

Note: 1. With TnCCLR=0, a Comparator P match will clear the counter

2. The TM output pin controlled only by TnAF flag

3. The output pin is reset to its initial state by TnON bit rising edge




**Compare Match Output Mode – TnCCLR=1**

Note: 1. With TnCCLR=1, a Comparator A match will clear the counter

2. The TM output pin is controlled only by TnAF flag
3. The TM output pin is reset to initial state by TnON rising edge
4. The TnPF flags is not generated when TnCCLR=1

### Timer/Counter Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the TM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the TM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

### PWM Output Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register should be set to 10 respectively. The PWM function within the TM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the TM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM mode, the TnCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the TnDPX bit in the TMnC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The TnOC bit in the TMnC1 register is used to select the required polarity of the PWM waveform while the two TnIO1 and TnIO0 bits are used to enable the PWM output or to force the TM output pin to a fixed high or low level. The TnPOL bit is used to reverse the polarity of the PWM output waveform.

### CTM, PWM Mode, Edge-aligned Mode, TnDPX=0

| CCRP | 001b | 010b | 011b | 100b | 101b | 110b | 111b | 000b |
|---|---|---|---|---|---|---|---|---|
| Period | 128 | 256 | 384 | 512 | 640 | 768 | 896 | 1024 |
| Duty | CCRA | | | | | | | |

If $f_{SYS}$=16MHz, TM clock source is $f_{SYS}$/4, CCRP=100b and CCRA=128,

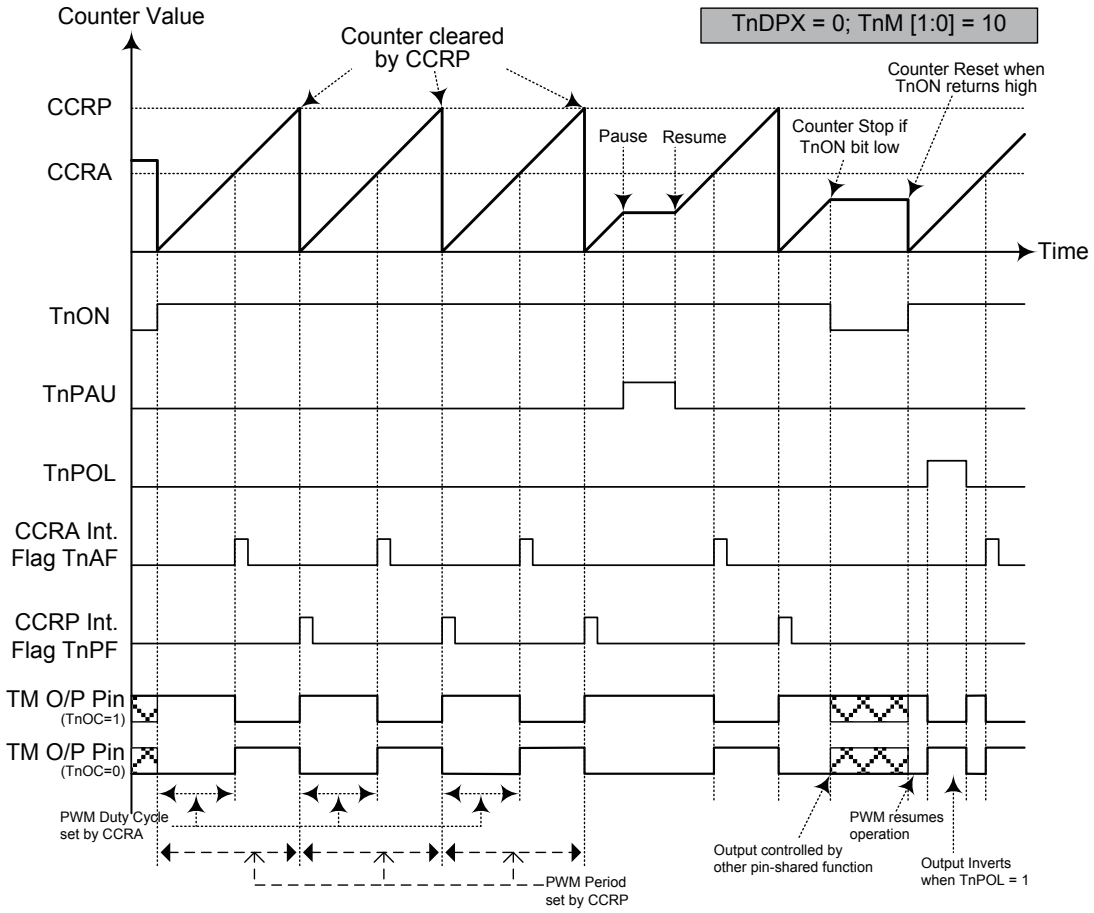
The CTM PWM output frequency=($f_{SYS}$/4)/512=$f_{SYS}$/2048=7.8125kHz, duty=128/512=25%.

If the Duty value defined by the CCRA register is equal to or greater than the Period value, then the PWM output duty is 100%.

### CTM, PWM Mode, Edge-aligned Mode, TnDPX=1

| CCRP | 001b | 010b | 011b | 100b | 101b | 110b | 111b | 000b |
|---|---|---|---|---|---|---|---|---|
| Period | CCRA | | | | | | | |
| Duty | 128 | 256 | 384 | 512 | 640 | 768 | 896 | 1024 |

The PWM output period is determined by the CCRA register value together with the TM clock while the PWM duty cycle is defined by the CCRP register value.



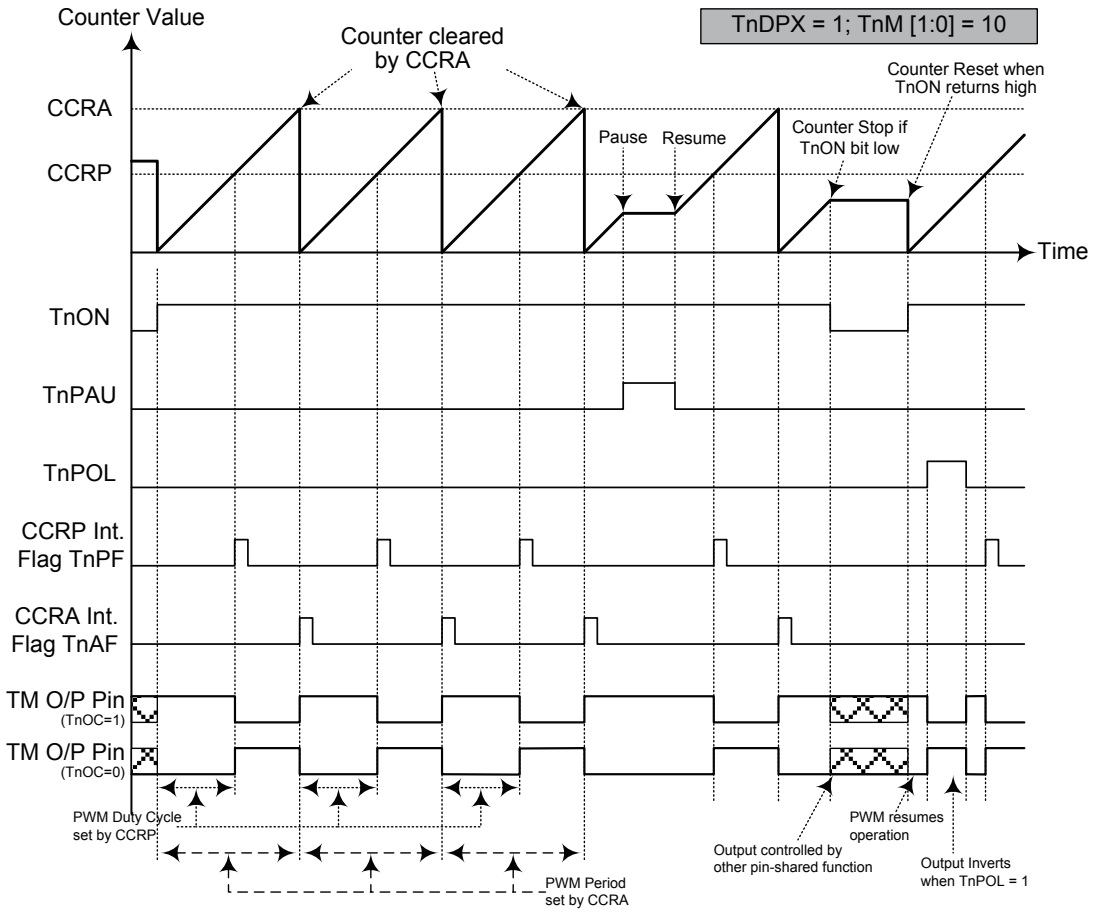

**PWM Mode – TnDPX=0**

Note: 1. Here TnDPX=0 - Counter cleared by CCRP

2. A counter clear sets PWM Period

3. The internal PWM function continues even when TnIO1, TnIO0=00 or 01

4. The TnCCLR bit has no influence on PWM operation




**PWM Mode – TnDPX=1**

Note: 1. Here TnDPX=1 – Counter cleared by CCRA

2. A counter clear sets PWM Period

3. The internal PWM function continues even when TnIO [1:0]=00 or 01

4. The TnCCLR bit has no influence on PWM operation

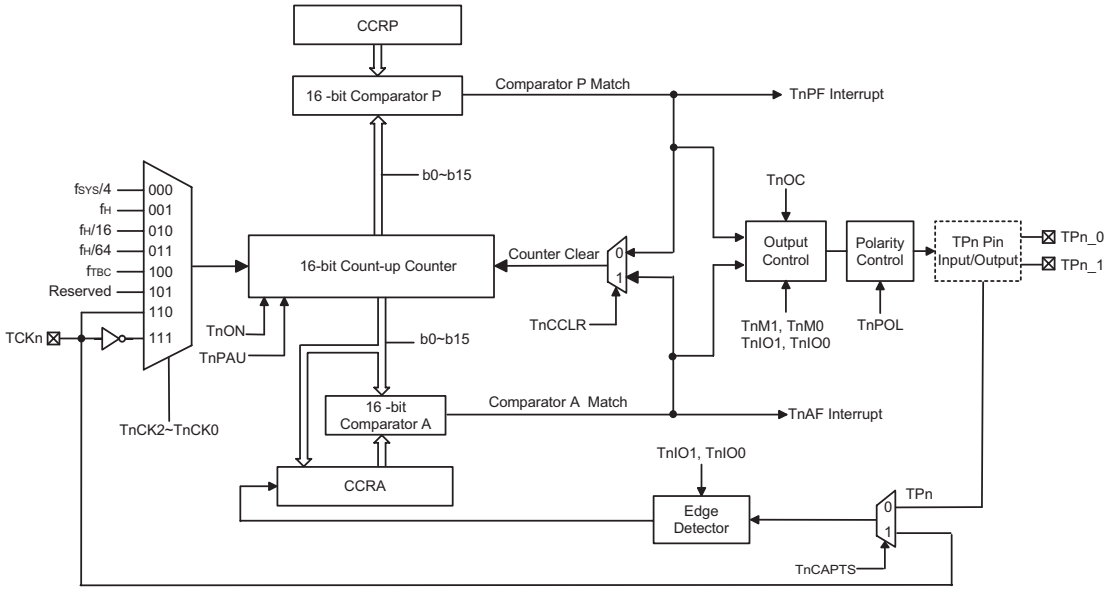
## Periodic Type TM – PTM (TM0, TM1)

The Periodic Type TM contains five operating modes, which are Compare Match Output, Timer/Event Counter, Capture Input, Single Pulse Output and PWM Output modes. The Periodic TM can also be controlled with an external input pin and can drive one external output pin.

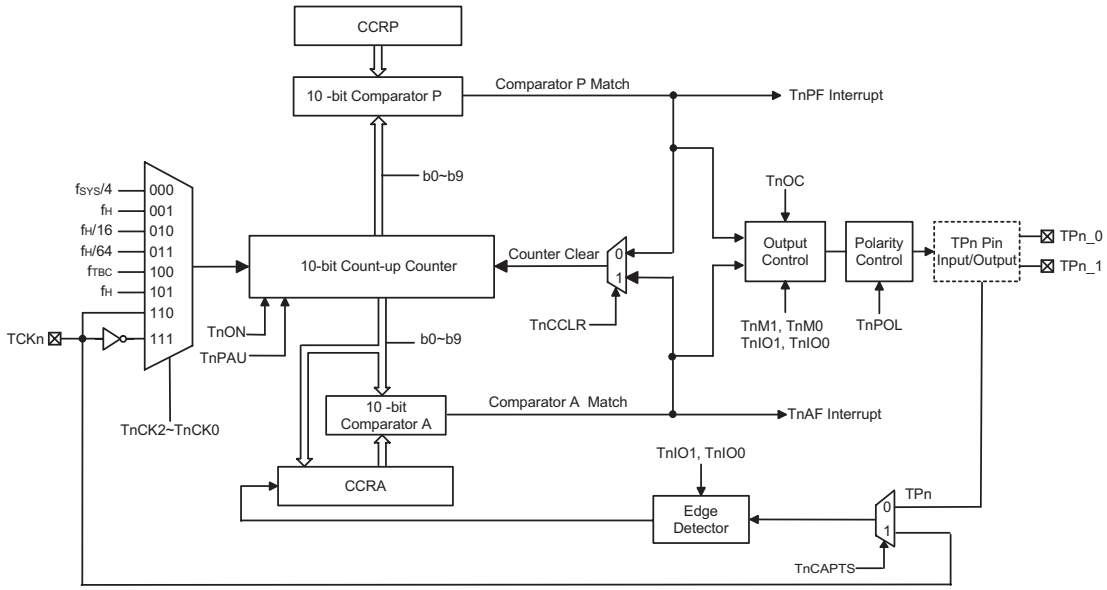
### Periodic TM Operation

There are two sizes of Periodic TMs, one is 10-bit wide and the other is 16-bit wide. At its core is a 10 or 16-bit count-up counter which is driven by a user selectable internal or external clock source. There are two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with the CCRA and CCRP registers.

The only way of changing the value of the 10 or 16-bit counter using the application program, is to clear the counter by changing the TnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a TM interrupt signal will also usually be generated. The Periodic Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control the output pin. All operating setup conditions are selected using relevant internal registers.




**16-bit Periodic Type TM Block Diagram (n=0)**




**10-bit Periodic Type TM Block Diagram (n=1)**

## Periodic Type TM Register Description

Overall operation of the Periodic TM is controlled using a series of registers. A read only register pair exists to store the internal counter 10 or 16-bit value, while two read/write register pairs exist to store the internal 10 or 16-bit CCRA and CCRP value. The remaining two registers are control registers which setup the different operating and control modes.

| Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| TMnC0 | TnPAU | TnCK2 | TnCK1 | TnCK0 | TnON | — | — | — |
| TMnC1 | TnM1 | TnM0 | TnIO1 | TnIO0 | TnOC | TnPOL | TnCAPTS | TnCCLR |
| TMnDL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TMnDH | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| TMnAL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TMnAH | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| TMnRPL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TMnRPH | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |

**16-bit Periodic TM Register List (n=0)**

| Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| TMnC0 | TnPAU | TnCK2 | TnCK1 | TnCK0 | TnON | — | — | — |
| TMnC1 | TnM1 | TnM0 | TnIO1 | TnIO0 | TnOC | TnPOL | TnCAPTS | TnCCLR |
| TMnDL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TMnDH | — | — | — | — | — | — | D9 | D8 |
| TMnAL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TMnAH | — | — | — | — | — | — | D9 | D8 |
| TMnRPL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TMnRPH | — | — | — | — | — | — | D9 | D8 |

**10-bit Periodic TM Register List (n=1)**

**TM0C0 Register – 16-bit PTM (n=0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TnPAU | TnCK2 | TnCK1 | TnCK0 | TnON | — | — | — |
| R/W | R/W | R/W | R/W | R/W | R/W | — | — | — |
| POR | 0 | 0 | 0 | 0 | 0 | — | — | — |

Bit 7 **TnPAU:** TMn Counter Pause Control
0: Run
1: Pause
The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the TM will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **TnCK2~TnCK0:** Select TMn Counter clock
000: $f_{SYS}/4$
001: $f_H$
010: $f_H/16$
011: $f_H/64$
100: $f_{TBC}$
101: Undefined
110: TCKn rising edge clock
111: TCKn falling edge clock
These three bits are used to select the clock source for the TM. Selecting the undefined clock input will effectively disable the internal counter. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source $f_{SYS}$ is the system clock, while $f_H$ and $f_{TBC}$ are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **TnON:** TMn Counter On/Off Control
0: Off
1: On
This bit controls the overall on/off function of the TM. Setting the bit high enables the counter to run and clearing the bit disables the TM. Clearing this bit to zero will stop the counter from counting and turn off the TM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value.
If the TM is in the Compare Match Output Mode then the TM output pin will be reset to its initial condition, as specified by the TnOC bit, when the TnON bit changes from low to high.

Bit 2~0 Unimplemented, read as “0”

### TM1C0 Register – 10-bit PTM (n=1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TnPAU | TnCK2 | TnCK1 | TnCK0 | TnON | — | — | — |
| R/W | R/W | R/W | R/W | R/W | R/W | — | — | — |
| POR | 0 | 0 | 0 | 0 | 0 | — | — | — |

Bit 7 **TnPAU:** TMn Counter Pause Control
0: Run
1: Pause
The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the TM will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **TnCK2~TnCK0:** Select TMn Counter clock
000: $f_{SYS}/4$
001: $f_H$
010: $f_H/16$
011: $f_H/64$
100: $f_{TBC}$
101: $f_H$
110: TCKn rising edge clock
111: TCKn falling edge clock
These three bits are used to select the clock source for the TM. Selecting the undefined clock input will effectively disable the internal counter. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source $f_{SYS}$ is the system clock, while $f_H$ and $f_{TBC}$ are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **TnON:** TMn Counter On/Off Control
0: Off
1: On
This bit controls the overall on/off function of the TM. Setting the bit high enables the counter to run and clearing the bit disables the TM. Clearing this bit to zero will stop the counter from counting and turn off the TM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value.
If the TM is in the Compare Match Output Mode then the TM output pin will be reset to its initial condition, as specified by the TnOC bit, when the TnON bit changes from low to high.

Bit 2~0 Unimplemented, read as “0”

**TM0C1 Register – 16-bit PTM; TM1C1 Register – 10-bit PTM (n=0, 1)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TnM1 | TnM0 | TnIO1 | TnIO0 | TnOC | TnPOL | TnCAPTS | TnCCLR |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7~6 **TnM1~TnM0:** Select TMn Operation Mode
00: Compare Match Output Mode
01: Capture Input Mode
10: PWM Mode or Single Pulse Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the TM. To ensure reliable operation the TM should be switched off before any changes are made to the TnM1 and TnM0 bits. In the Timer/Counter Mode, the TM output pin control must be disabled.

Bit 5~4 **TnIO1~TnIO0:** Select TPn_0, TPn_1 output function
Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output
PWM Mode/Single Pulse Output Mode
00: PWM Output inactive state
01: PWM Output active state
10: PWM output
11: Single pulse output
Capture Input Mode
00: Input capture at rising edge of TPn_0, TPn_1
01: Input capture at falling edge of TPn_0, TPn_1
10: Input capture at falling/rising edge of TPn_0, TPn_1
11: Input capture disabled
Timer/Counter Mode
Unused

These two bits are used to determine how the TM output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the TM is running.

In the Compare Match Output Mode, the TnIO1 and TnIO0 bits determine how the TM output pin changes state when a compare match occurs from the Comparator A. The TM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the TM output pin should be setup using the TnOC bit in the TMnC1 register. Note that the output level requested by the TnIO1 and TnIO0 bits must be different from the initial value setup using the TnOC bit otherwise no change will occur on the TM output pin when a compare match occurs. After the TM output pin changes state, it can be reset to its initial level by changing the level of the TnON bit from low to high.

In the PWM Mode, the TnIO1 and TnIO0 bits determine how the TM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to change the values of the TnIO1 and TnIO0 bits only after the TM has been switched off. Unpredictable PWM outputs will occur if the TnIO1 and TnIO0 bits are changed when the TM is running.

Bit 3 **TnOC:** TPn_0, TPn_1 output control bit
Compare Match Output Mode
0: Initial low
1: Initial high
PWM Mode/Single Pulse Output Mode
0: Active low
1: Active high
This is the output control bit for the TM output pin. Its operation depends upon whether TM is being used in the Compare Match Output Mode or in the PWM Mode/Single Pulse Output Mode. It has no effect if the TM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the TM output pin before a compare match occurs. In the PWM Mode it determines if the PWM signal is active high or active low.

Bit 2 **TnPOL:** TPn_0, TPn_1 output Polarity control
0: Non-invert
1: Invert
This bit controls the polarity of the TPn_0 or TPn_1 output pin. When the bit is set high the TM output pin will be inverted and not inverted when the bit is zero. It has no effect if the TM is in the Timer/Counter Mode.

Bit 1 **TnCAPTS:** TMn capture trigger source select
0: From TPn_0 or TPn_1 pin
1: From TCKn pin

Bit 0 **TnCCLR:** Select TMn Counter clear condition
0: TMn Comparator P match
1: TMn Comparator A match
This bit is used to select the method which clears the counter. Remember that the Periodic TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the TnCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The TnCCLR bit is not used in the PWM Mode, Single Pulse or Input Capture Mode.

### TM0DL Register – 16-bit PTM

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

Bit 7~0 TM0 Counter Low Byte Register bit 7~bit 0
TM0 16-bit Counter bit 7~bit 0

### TM1DL Register – 10-bit PTM

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

Bit 7~0 TM1 Counter Low Byte Register bit 7~bit 0
TM1 10-bit Counter bit 7~bit 0

### TM0DH Register – 16-bit PTM

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

Bit 7~0 TM0 Counter High Byte Register bit 7~bit 0
TM0 16-bit Counter bit 15~bit 8

### TM1DH Register – 10-bit PTM

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

Bit 7~2 Unimplemented, read as 0
Bit 1~0 TM1 Counter High Byte Register bit 1~bit 0
TM1 10-bit Counter bit 9~bit 8

### TM0AL Register – 16-bit PTM

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

Bit 7~0 TM0 CCRA Low Byte Register bit 7~bit 0
TM0 16-bit CCRA bit 7~bit 0

### TM1AL Register – 10-bit PTM

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

Bit 7~0 TM1 CCRA Low Byte Register bit 7~bit 0
TM1 10-bit CCRA bit 7~bit 0

### TM0AH Register – 16-bit PTM

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

Bit 7~0 TM0 CCRA High Byte Register bit 7~bit 0
TM0 16-bit CCRA bit 15~bit 8

### TM1AH Register – 10-bit PTM

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

Bit 7~2 Unimplemented, read as 0

Bit 1~0 TM1 CCRA High Byte Register bit 1~bit 0
TM1 10-bit CCRA bit 9~bit 8

### TM0RPL Register – 16-bit PTM

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

Bit 7~0 TM0 CCRP Low Byte Register bit 7~bit 0
TM0 16-bit CCRP bit 7~bit 0

### TM1RPL Register – 10-bit PTM

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

Bit 7~0 TM1 CCRP Low Byte Register bit 7~bit 0
TM1 10-bit CCRP bit 7~bit 0

### TM0RPH Register – 16-bit PTM

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

Bit 7~0 TM0 CCRP High Byte Register bit 7~bit 0
TM0 16-bit CCRP bit 15~bit 8

**TM1RPH Register – 10-bit PTM**

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

Bit 7~2 Unimplemented, read as 0

Bit 1~0 TM1 CCRP High Byte Register bit 1~bit 0
TM1 10-bit CCRP bit 9~bit 8

## Periodic Type TM Operation Modes

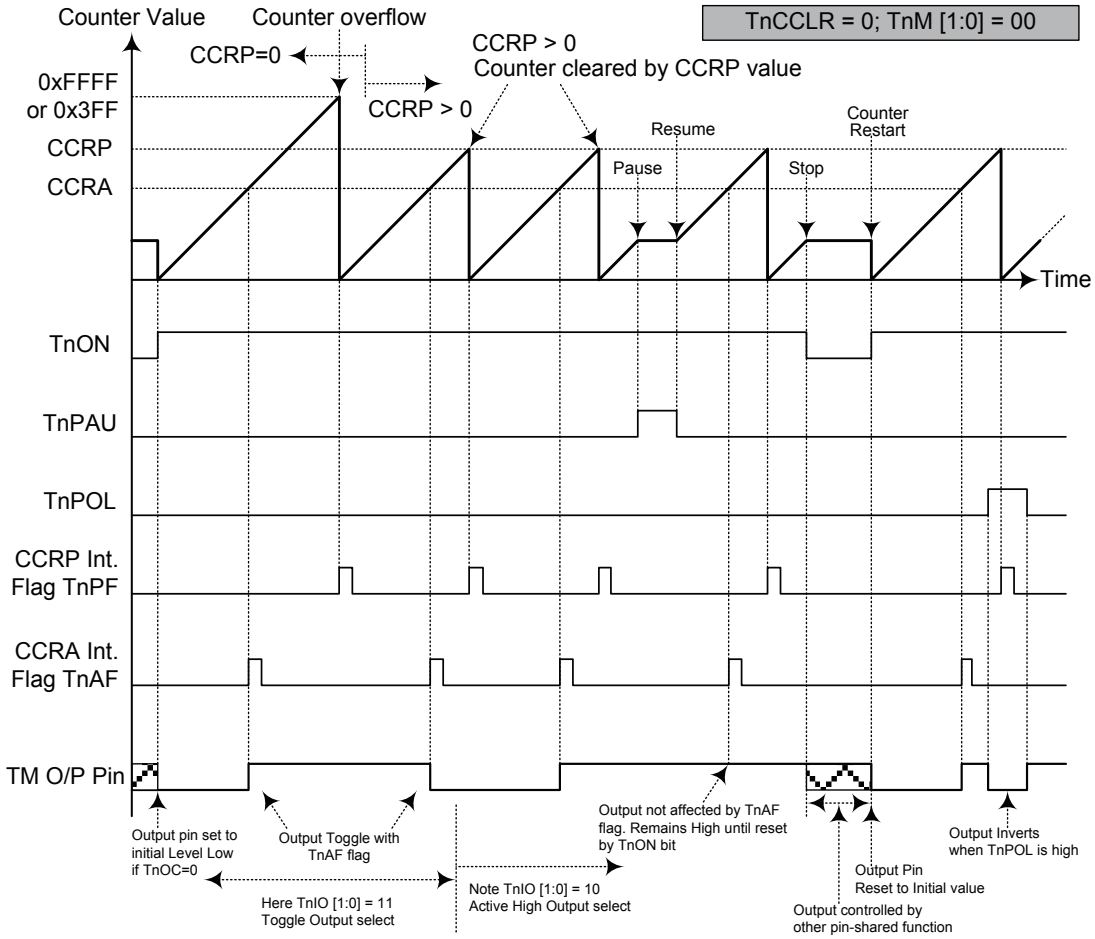
The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the TnM1 and TnM0 bits in the TMnC1 register.

### Compare Match Output Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the TnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both TnAF and TnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the TnCCLR bit in the TMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the TnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when TnCCLR is high no TnPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be set to “0”.

As the name of the mode suggests, after a comparison is made, the TM output pin will change state. The TM output pin condition however only changes state when a TnAF interrupt request flag is generated after a compare match occurs from Comparator A. The TnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the TM output pin. The way in which the TM output pin changes state are determined by the condition of the TnIO1 and TnIO0 bits in the TMnC1 register. The TM output pin can be selected using the TnIO1 and TnIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the TM output pin, which is setup after the TnON bit changes from low to high, is setup using the TnOC bit. Note that if the TnIO1 and TnIO0 bits are zero then no pin change will take place.

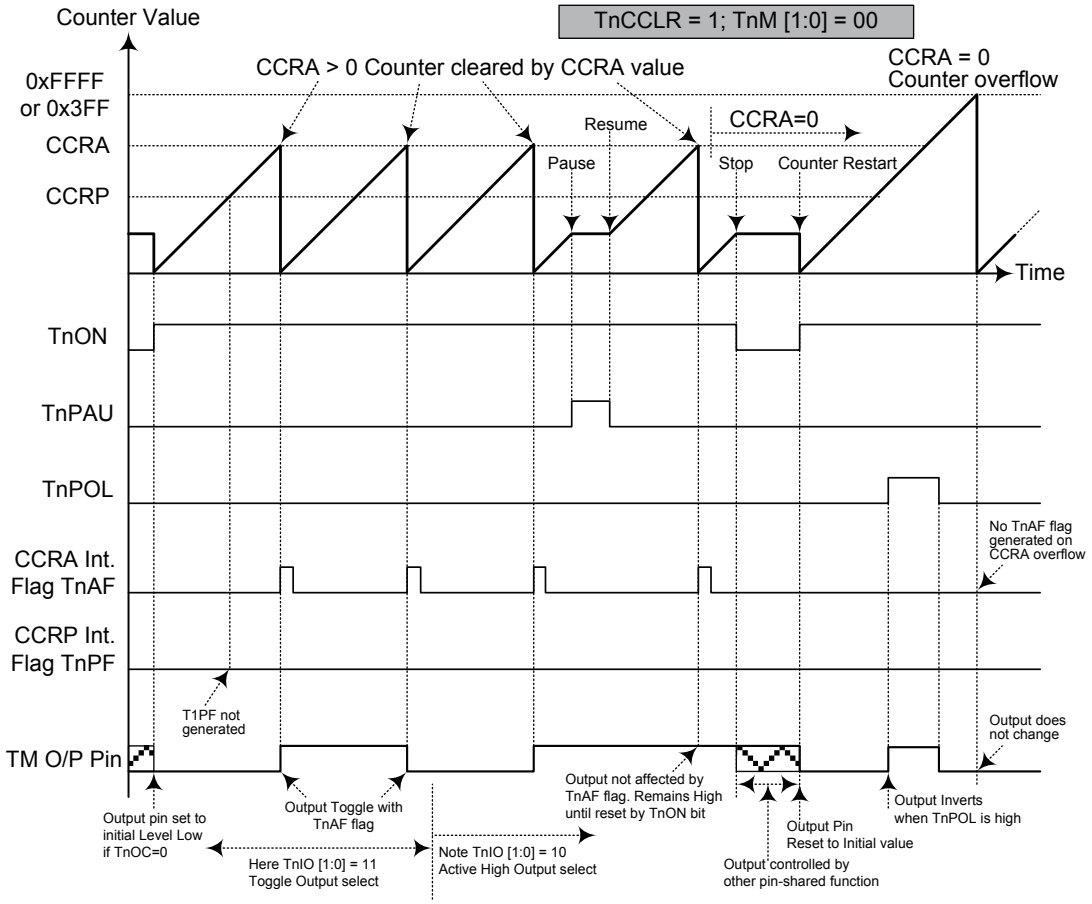



**Compare Match Output Mode – TnCCLR=0**

Note: 1. With TnCCLR=0 – A Comparator P match will clear the counter

2. The TM output pin is controlled only by the TnAF flag

3. The output pin is reset to its initial state by a TnON bit rising edge




**Compare Match Output Mode – TnCCLR=1**

Note: 1. With TnCCLR=1 – A Comparator A match will clear the counter

2. The TM output pin is controlled only by the TnAF flag

3. The output pin is reset to its initial state by a TnON bit rising edge

4. A TnPF flag is not generated when TnCCLR=1

### Timer/Counter Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the TM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the TM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

### PWM Output Mode

To select this mode, bits TnM1 and TnM0 in the TMnC1 register should be set to 10 respectively and also the TnIO1 and TnIO0 bits should be set to 10 respectively. The PWM function within the TM is useful for applications which require functions such as motor control, heating control, illumination control, etc. By providing a signal of fixed frequency but of varying duty cycle on the TM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM mode, the TnCCLR bit has no effect as the PWM period. Both of the CCRP and CCRA registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The TnOC bit in the TMnC1 register is used to select the required polarity of the PWM waveform while the two TnIO1 and TnIO0 bits are used to enable the PWM output or to force the TM output pin to a fixed high or low level. The TnPOL bit is used to reverse the polarity of the PWM output waveform.

- **16-bit PTM, PWM Mode**

| Period | | Duty |
|---|---|---|
| CCRP=0 | CCRP=1~65535 | CCRA |
| 65535 | 1~65535 | |

If $f_{SYS}$=16MHz, TM clock source select $f_{SYS}/4$, CCRP=512 and CCRA=128,

The PTM PWM output frequency=($f_{SYS}/4$)/512=$f_{SYS}/2048$=7.8125kHz, duty=128/512=25%,

If the Duty value defined by the CCRA register is equal to or greater than the Period value, then the PWM output duty is 100%.

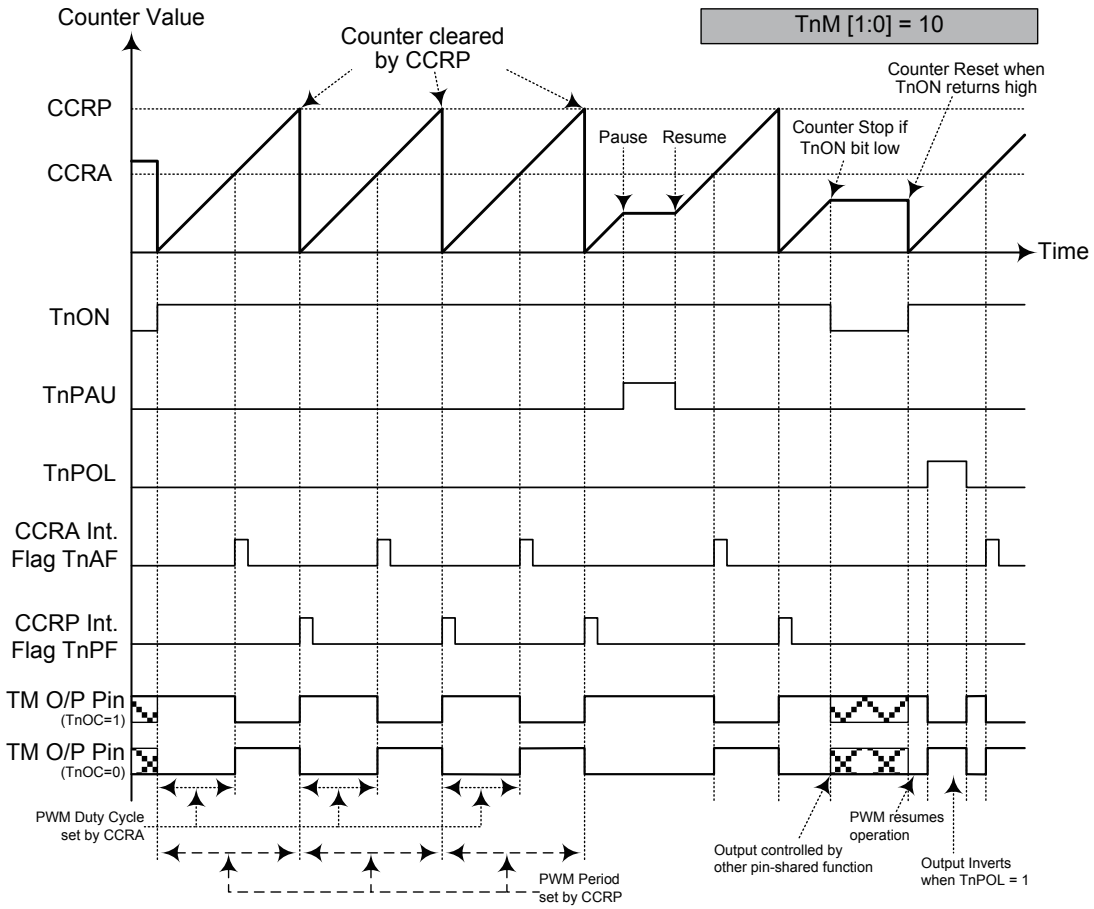
- **10-bit PTM, PWM Mode**

| Period | | Duty |
|---|---|---|
| CCRP=0 | CCRP=1~1023 | CCRA |
| 1024 | 1~1023 | |

If $f_{SYS}$=16MHz, TM clock source select $f_{SYS}/4$, CCRP=512 and CCRA=128,

The PTM PWM output frequency=($f_{SYS}/4$)/512=$f_{SYS}/2048$=7.8125kHz, duty=128/512=25%,

If the Duty value defined by the CCRA register is equal to or greater than the Period value, then the PWM output duty is 100%.




**PWM Mode**

Note: 1. Here Counter cleared by CCRP

2. A counter clear sets the PWM Period

3. The internal PWM function continues running even when TnIO [1:0]=00 or 01

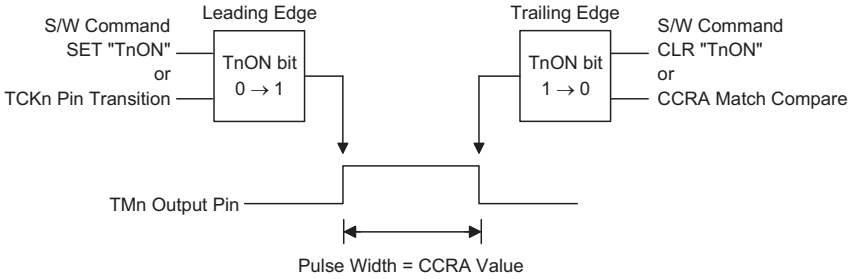
4. The TnCCLR bit has no influence on PWM operation

### Single Pulse Output Mode

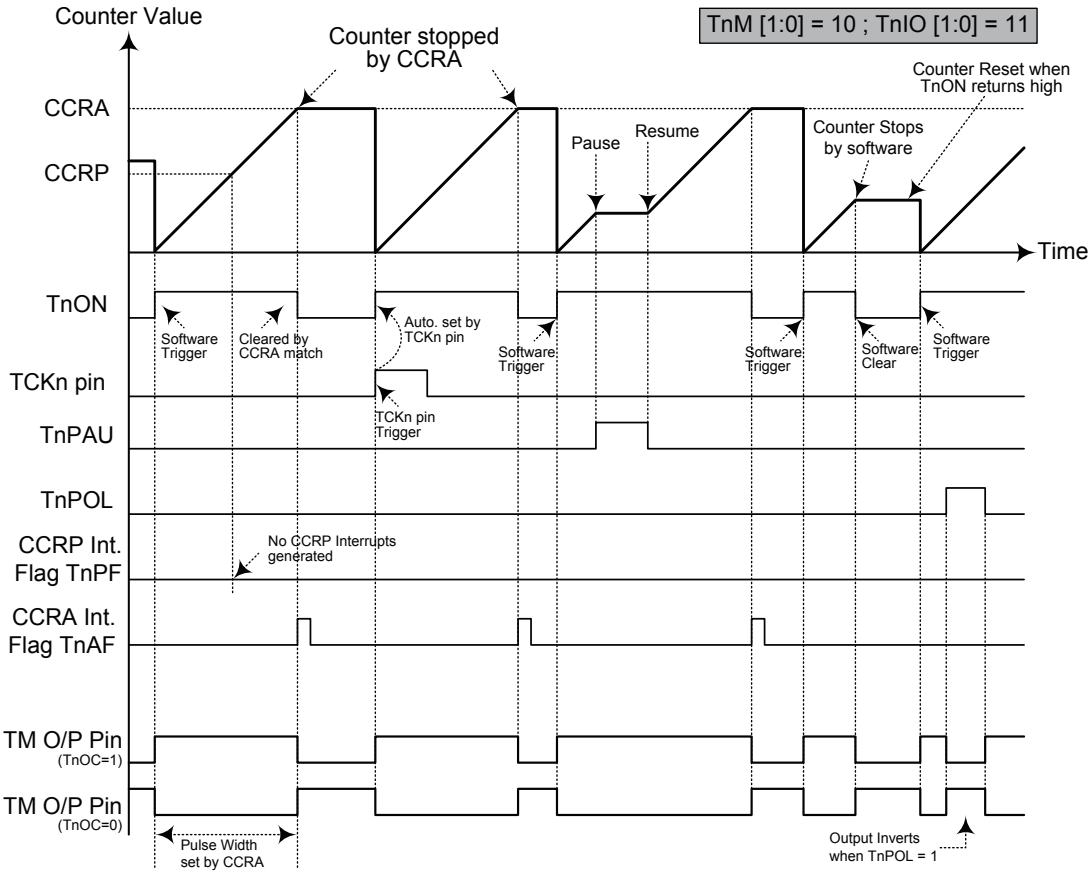
To select this mode, bits TnM1 and TnM0 in the TMnC1 register should be set to 10 respectively and also the TnIO1 and TnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the TM output pin.

The trigger for the pulse output leading edge is a low to high transition of the TnON bit, which can be implemented using the application program. However in the Single Pulse Mode, the TnON bit can also be made to automatically change from low to high using the external TCKn pin, which will in turn initiate the Single Pulse output. When the TnON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The TnON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the TnON bit is cleared to zero, which can be implemented using the application program or when a compare match occurs from Comparator A.

However a compare match from Comparator A will also automatically clear the TnON bit and thus generate the Single Pulse output trailing edge. In this way the CCRA value can be used to control the pulse width. A compare match from Comparator A will also generate a TM interrupt. The counter can only be reset back to zero when the TnON bit changes from low to high when the counter restarts. In the Single Pulse Mode CCRP is not used. The TnCCLR is not used in this Mode.




**Single Pulse Generation**




**Single Pulse Mode**

Note: 1. Counter stopped by CCRA

2. CCRP is not used

3. The pulse is triggered by the TCKn pin or by setting the TnON bit high

4. A TCKn pin active edge will automatically set the TnON bit high

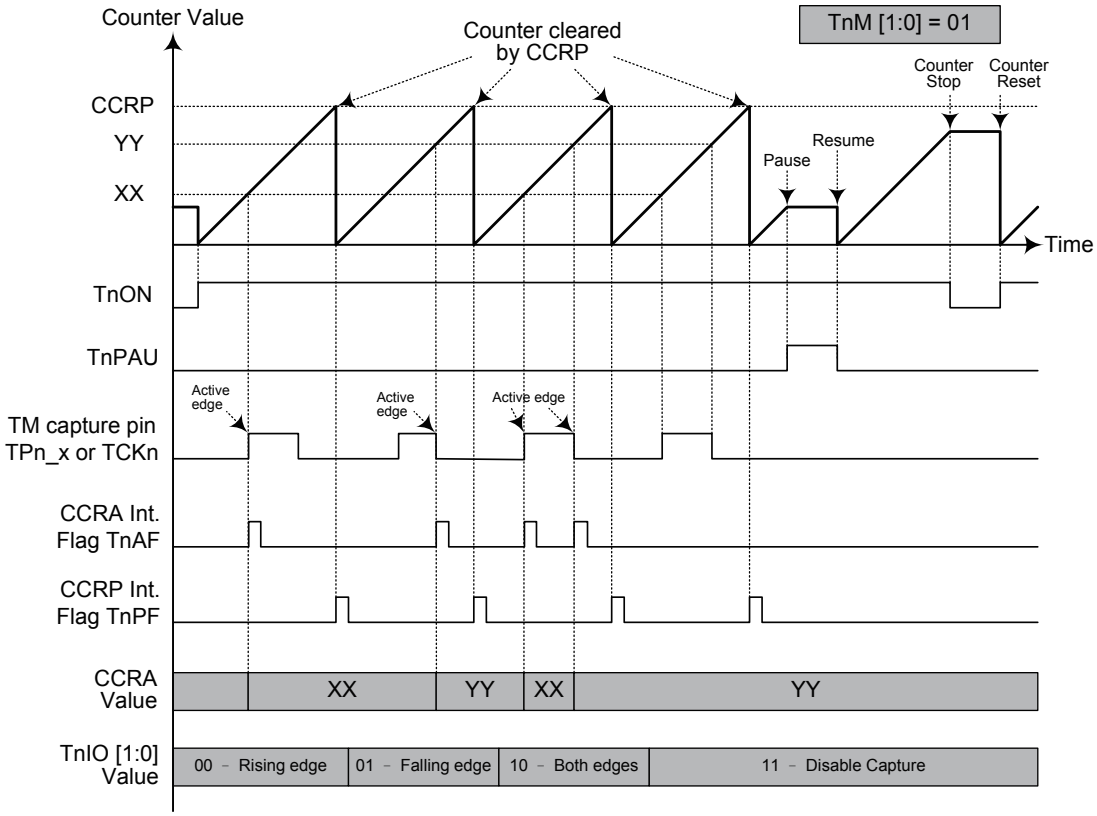
5. In the Single Pulse Mode, TnIO [1:0] must be set to “11” and can not be changed

### Capture Input Mode

To select this mode bits TnM1 and TnM0 in the TMnC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the TPn_0, TPn_1 or TCKn pin, selected by the TnCAPTS bit in the TM1C0 register. The input pin active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the TnIO1 and TnIO0 bits in the TMnC1 register. The counter is started when the TnON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the TPn_0, TPn_1 or TCKn pin the present value in the counter will be latched into the CCRA registers and a TM interrupt generated. Irrespective of what events occur on the TPn_0, TPn_1 or TCKn pin the counter will continue to free run until the TnON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a TM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The TnIO1 and TnIO0 bits can select the active trigger edge on the TPn_0, TPn_1 or TCKn pin to be a rising edge, falling edge or both edge types. If the TnIO1 and TnIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the TPn_0, TPn_1 or TCKn pin, however it must be noted that the counter will continue to run.

As the TPn_0, TPn_1 or TCKn pin is pin shared with other functions, care must be taken if the TM1 is in the Input Capture Mode. This is because if the pin is setup as an output, then any transitions on this pin may cause an input capture operation to be executed. The TnCCLR, TnOC and TnPOL bits are not used in this Mode.




**Capture Input Mode**

Note: 1. TnM [1:0]=01 and active edge set by the TnIO [1:0] bits

2. A TM Capture input pin active edge transfers the counter value to CCRA

3. TnCCLR bit not used

4. No output function – TnOC and TnPOL bits are not used

5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero

## Analog to Digital Converter

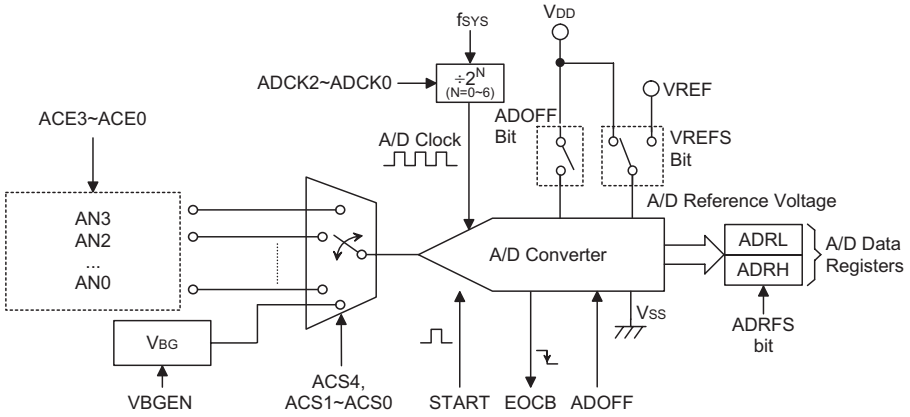
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

### A/D Overview

The device contains a multi-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 a 12-bit digital value.

| Input Channels | Channel Select Bits | External Input Pins |
|---|---|---|
| 4 | ACS4, ACS1~ACS0 | AN0~AN3 |

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




**A/D Converter Structure**

### A/D Converter Register Description

Overall operation of the A/D converter is controlled using five registers. A read only register pair exists to store the ADC data 12-bit value. The remaining three registers are control registers which setup the operating and control function of the A/D converter.

| Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ADRL(ADRFS=0) | D3 | D2 | D1 | D0 | — | — | — | — |
| ADRL(ADRFS=1) | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| ADRH(ADRFS=0) | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 |
| ADRH(ADRFS=1) | — | — | — | — | D11 | D10 | D9 | D8 |
| ADCR0 | START | EOCB | ADOFF | ADRFS | — | — | ACS1 | ACS0 |
| ADCR1 | ACS4 | VBGEN | — | VREFS | — | ADCK2 | ADCK1 | ADCK0 |
| ACERL | — | — | — | — | ACE3 | ACE2 | ACE1 | ACE0 |

**A/D Converter Register List**

## A/D Converter Data Registers – ADRL, ADRH

As the devices contain an internal 12-bit A/D converter, they require two data registers to store the converted value. These are 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. As only 12 bits of the 16-bit register space is utilised, the format in which the data is stored is controlled by the ADRFS bit in the ADCR0 register as shown in the accompanying table. D0~D11 are the A/D conversion result data bits. Any unused bits will be read as zero.

| ADRFS | ADRH | | | | | | | | ADRL | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

**A/D Data Registers**

## A/D Converter Control Registers – ADCR0, ADCR1, ACERL

To control the function and operation of the A/D converter, three control registers known as ADCR0, ADCR1 and ACERL are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter end of conversion status. The ACS1~ACS0 bits in the ADCR0 register and ACS4 bit is the ADCR1 register define the ADC input channel number. As the device contains only one actual analog to digital converter hardware circuit, each of the individual 4 analog inputs must be routed to the converter. It is the function of the ACS1~ACS0 and ACS4 bits to determine which analog channel input pins or internal $V_{BG}$ is actually connected to the internal A/D converter.

The ACERL control register contains the ACE3~ACE0 bits which determine which pins on PA0~PA3 are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. Setting the corresponding bit high will select the A/D input function, clearing the bit to zero will select either the I/O or other pin-shared function. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistors connected to these pins will be automatically removed if the pin is selected to be an A/D input.

**ADCR0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | START | EOCB | ADOFF | ADRFS | — | — | ACS1 | ACS0 |
| R/W | R/W | R | R/W | R/W | — | — | R/W | R/W |
| POR | 0 | 1 | 1 | 0 | — | — | 0 | 0 |

Bit 7 **START**: Start the A/D conversion
0 → 1 → 0: start
0 → 1: reset the A/D converter and set EOCB to "1"
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process. When the bit is set high the A/D converter will be reset.

Bit 6 **EOCB**: End of A/D conversion flag
0: A/D conversion ended
1: A/D conversion in progress
This read only flag is used to indicate when an A/D conversion process has completed. When the conversion process is running the bit will be high.

Bit 5 **ADOFF**: ADC module power on/off control bit
0: ADC module power on
1: ADC module power off
This bit controls the power to the A/D internal function. This bit should be cleared to zero to enable the A/D converter. If the bit is set high then the A/D converter will be switched off reducing the device power consumption. As the A/D converter will consume a limited amount of power, even when not executing a conversion, this may be an important consideration in power sensitive battery powered applications.

Note: 1. it is recommended to set ADOFF=1 before entering IDLE/SLEEP Mode for saving power.
2. ADOFF=1 will power down the ADC module.

Bit 4 **ADRFS**: ADC Data Format Control
0: ADC Data MSB is ADRH bit 7, LSB is ADRL bit 4
1: ADC Data MSB is ADRH bit 3, LSB is ADRL bit 0
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D data register section.

Bit 3~2 Unimplemented, read as "0"

Bit 1~0 **ACS1~ACS0**: Select A/D channel (when ACS4 is "0")
00: AN0
01: AN1
10: AN2
11: AN3
These are the A/D channel select control bits. As there is only one internal hardware A/D converter, each of the four A/D inputs must be routed to the internal converter using these bits. If bit ACS4 in the ADCR1 register is set high, then the internal $V_{BG}$ will be routed to the A/D Converter.

**ADCR1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ACS4 | VBGEN | — | VREFS | — | ADCK2 | ADCK1 | ADCK0 |
| R/W | R/W | R/W | — | R/W | — | R/W | R/W | R/W |
| POR | 0 | 0 | — | 0 | — | 0 | 0 | 0 |

Bit 7 **ACS4**: Selecte Internal $V_{BG}$ as ADC input Control
0: Disable
1: Enable
This bit enables $V_{BG}$ to be connected to the A/D converter. The VBGEN bit must first have been set to enable the bandgap circuit $V_{BG}$ voltage to be used by the A/D converter. When the ACS4 bit is set high, the bandgap $V_{BG}$ voltage will be routed to the A/D converter and the other A/D input channels disconnected.

Bit 6 **VBGEN**: Internal $V_{BG}$ Control
0: Disable
1: Enable
This bit controls the internal Bandgap circuit on/off function to the A/D converter. When the bit is set high the bandgap $V_{BG}$ voltage can be used by the A/D converter. If $V_{BG}$ is not used by the A/D converter and the LVR/LVD function is disabled then the bandgap reference circuit will be automatically switched off to conserve power. When $V_{BG}$ is switched on for use by the A/D converter, a time $t_{BG}$ should be allowed for the bandgap circuit to stabilise before implementing an A/D conversion.

Bit 5 Unimplemented, read as "0"

Bit 4 **VREFS**: Selecte ADC reference voltage
0: Internal ADC power
1: VREF pin
This bit is used to select the reference voltage for the A/D converter. If the bit is high then the A/D converter reference voltage is supplied on the external VREF pin. If the pin is low then the internal reference is used which is taken from the power supply pin VDD.

Bit 3 Unimplemented, read as "0"

Bit 2~0 **ADCK2~ADCK0**: Select ADC clock source
000: $f_{SYS}$
001: $f_{SYS}/2$
010: $f_{SYS}/4$
011: $f_{SYS}/8$
100: $f_{SYS}/16$
101: $f_{SYS}/32$
110: $f_{SYS}/64$
111: Undefined
These three bits are used to select the clock source for the A/D converter.

**ACERL Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | ACE3 | ACE2 | ACE1 | ACE0 |
| R/W | — | — | — | — | R/W | R/W | R/W | R/W |
| POR | — | — | — | — | 1 | 1 | 1 | 1 |

Bit 7~4 Unimplemented, read as "0"

Bit 3 **ACE3**: Define PA3 is A/D input or not
0: Not A/D input
1: A/D input, AN3

Bit 2 **ACE2**: Define PA2 is A/D input or not
0: Not A/D input
1: A/D input, AN2

Bit 1 **ACE1**: Define PA1 is A/D input or not
0: Not A/D input
1: A/D input, AN1

Bit 0 **ACE0**: Define PA0 is A/D input or not
0: Not A/D input
1: A/D input, AN0

## A/D Operation

The START bit in the ADCR0 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 ADCR0 register will be set high and the analog to digital converter will be reset. It is the START bit that is used to control the overall start operation of the internal analog to digital converter.

The EOCB bit in the ADCR0 register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set to "0" 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 ADCR0 register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

The clock source for the A/D converter, which originates from the system clock $f_{SYS}$, can be chosen to be either $f_{SYS}$ or a subdivided version of $f_{SYS}$. The division ratio value is determined by the ADCK2~ADCK0 bits in the ADCR1 register.

Although the A/D clock source is determined by the system clock $f_{SYS}$, and by bits ADCK2~ADCK0, there are some limitations on the A/D clock source speed range that can be selected. As the recommended range of permissible A/D clock period, $t_{ADCK}$, is from 0.5μs to 10μs, care must be taken for selected system clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the ADCK2~ADCK0 bits should not be set to 000B or 110B. Doing so will give A/D clock periods that are less than the minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where, depending upon the device, special care must be taken, as the values may be less than the specified minimum A/D Clock Period.

| $f_{SYS}$ | A/D Clock Period ($t_{ADCK}$) | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | ADCK2, ADCK1, ADCK0 =000 ($f_{SYS}$) | ADCK2, ADCK1, ADCK0 =001 ($f_{SYS}$/2) | ADCK2, ADCK1, ADCK0 =010 ($f_{SYS}$/4) | ADCK2, ADCK1, ADCK0 =011 ($f_{SYS}$/8) | ADCK2, ADCK1, ADCK0 =100 ($f_{SYS}$/16) | ADCK2, ADCK1, ADCK0 =101 ($f_{SYS}$/32) | ADCK2, ADCK1, ADCK0 =110 ($f_{SYS}$/64) | ADCK2, ADCK1, ADCK0 =111 |
| 1MHz | 1μs | 2μs | 4μs | 8μs | 16μs* | 32μs* | 64μs* | Undefined |
| 2MHz | 500ns | 1μs | 2μs | 4μs | 8μs | 16μs* | 32μs* | Undefined |
| 4MHz | 250ns* | 500ns | 1μs | 2μs | 4μs | 8μs | 16μs* | Undefined |
| 8MHz | 125ns* | 250ns* | 500ns | 1μs | 2μs | 4μs | 8μs | Undefined |

**A/D Clock Period Examples**

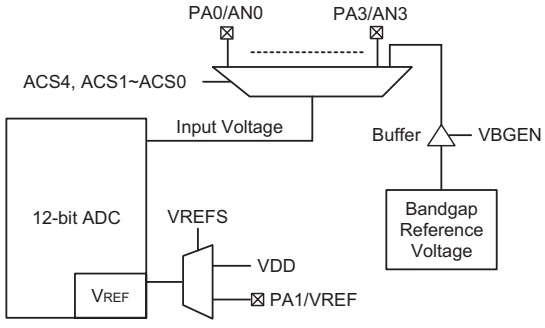
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADOFF bit in the ADCR0 register. This bit must be zero to power on the A/D converter. When the ADOFF bit is cleared to zero to power on the A/D converter internal circuitry a certain delay, as indicated in the timing diagram, must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs by clearing the ACE3~ACE0 bits in the ACERL register, if the ADOFF bit is zero then some power will still be consumed. In power conscious applications it is therefore recommended that the ADOFF is set high to reduce power consumption when the A/D converter function is not being used.

The reference voltage supply to the A/D Converter can be supplied from either the positive power supply pin, VDD, or from an external reference sources supplied on pin VREF. The desired selection is made using the VREFS bit. As the VREF pin is pin-shared with other functions, when the VREFS bit is set high, the VREF pin function will be selected and the other pin functions will be disabled automatically.

## A/D Input Pins

All of the A/D analog input pins are pin-shared with the I/O pins on PA3~PA0 as well as other functions. The ACE3~ACE0 bits in the ACERL register determines whether the input pins are setup as A/D converter analog inputs or whether they have other functions. If the ACE3~ACE0 bits for its corresponding pin is set high then the pin will be setup to be an A/D converter input and the original pin functions disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it is not necessary to first setup the A/D pin as an input in the PAC port control register to enable the A/D input as when the ACE3~ACE0 bits enable an A/D input, the status of the port control register will be overridden.

The A/D converter has its own reference voltage pin VREF however the reference voltage can also be supplied from the power supply pin, a choice which is made through the VREFS bit in the ADCR1 register. The analog input values must not be allowed to exceed the value of $V_{REF}$.




**A/D Input Structure**

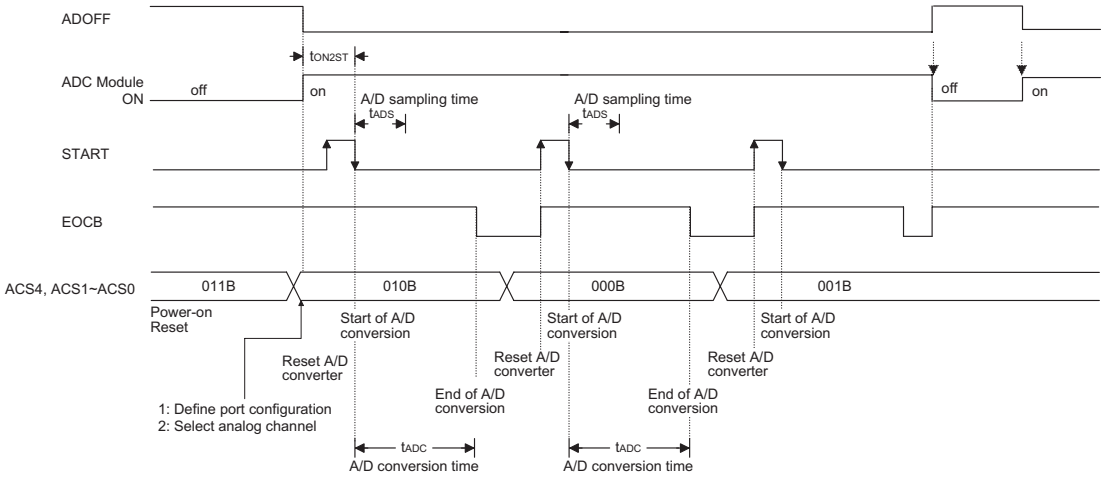
### Summary of A/D Conversion Steps

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 ADCK2~ADCK0 in the ADCR1 register.
- Step 2
  Enable the A/D by clearing the ADOFF bit in the ADCR0 register to zero.
- Step 3
  Select which channel is to be connected to the internal A/D converter by correctly programming the ACS4, ACS1~ACS0 bits which are also contained in the ADCR1 and ADCR0 register.
- Step 4
  Select which pins are to be used as A/D inputs and configure them by correctly programming the ACE3~ACE0 bits in the ACERL register.
- Step 5
  If the interrupts are to be used, the interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt bit, ADE, must both be set high to do this.
- Step 6
  The analog to digital conversion process can now be initialised by setting the START bit in the ADCR0 register from low to high and then low again. Note that this bit should have been originally cleared to zero.
- Step 7
  To check when the analog to digital conversion process is complete, the EOCB bit in the ADCR0 register can be polled. The conversion process is complete when this bit goes low. When this occurs the A/D data registers ADRL and ADRH can be read to obtain the conversion value. As an alternative method, if the interrupts are enabled and the stack is not full, the program can wait for an A/D interrupt to occur.

Note: When checking for the end of the conversion process, if the method of polling the EOCB bit in the ADCR0 register is used, the interrupt enable step above can be omitted.

The accompanying diagram shows graphically the various stages involved in an analog to digital conversion process and its associated timing. 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 $t_{ADCK}$ where $t_{ADCK}$ is equal to the A/D clock period.




**A/D Conversion Timing**

## Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by setting bit ADOFF high in the ADCR0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/Os, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

## A/D Transfer Function

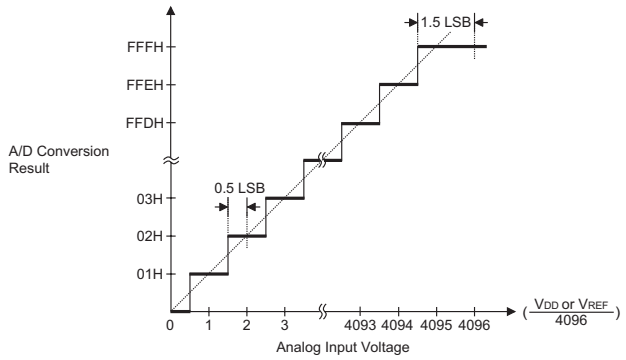
As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the $V_{DD}$ or $V_{REF}$ voltage, this gives a single bit analog input value of $V_{DD}$ or $V_{REF}$ divided by 4096.

$$1\ LSB=(V_{DD}\ or\ V_{REF})/4096$$

The A/D Converter input voltage value can be calculated using the following equation:

$$A/D\ input\ voltage=A/D\ output\ digital\ value \times (V_{DD}\ or\ V_{REF})/4096$$

The diagram shows the ideal transfer function between the analog input value and the digitised output value for the A/D converter. 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 $V_{DD}$ or $V_{REF}$ level.




**Ideal A/D Transfer Function**

## 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 ADCR0 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**

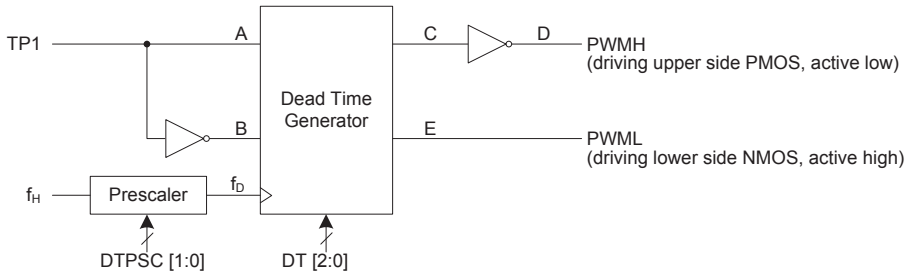
```
clr  ADE                ; disable ADC interrupt
mov  a,03H
mov  ADCR1,a            ; select fSYS/8 as A/D clock and switch off VBG
clr  ADOFF
mov  a,0Fh              ; setup ACERL to configure pins AN0~AN3
mov  ACERL,a
mov  a,01h
mov  ADCR0,a            ; enable and connect AN0 channel to A/D converter
:
start_conversion:
clr  START              ; high pulse on start bit to initiate conversion
set  START              ; reset A/D
clr  START              ; start A/D
polling_EOC:
sz   EOCB               ; poll the ADCR0 register EOCB bit to detect end of A/D conversion
jmp polling_EOC         ; continue polling
mov  a,ADRL             ; read low byte conversion result value
mov  ADRL_buffer,a      ; save result to user defined register
mov  a,ADRH             ; read high byte conversion result value
mov  ADRH_buffer,a      ; save result to user defined register
:
:
jmp  start_conversion   ; start next A/D conversion
```

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

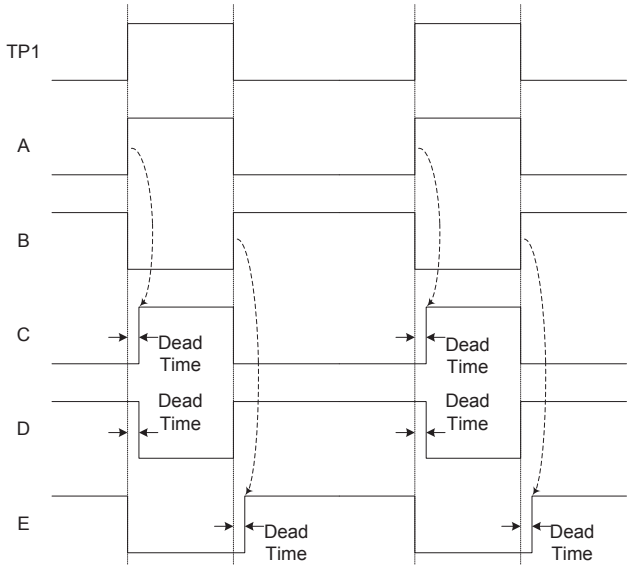
```
clr  ADE                ; disable ADC interrupt
mov  a,03H
mov  ADCR1,a            ; select fSYS/8 as A/D clock and switch off VBG
Clr  ADOFF
mov  a,0Fh              ; setup ACERL to configure pins AN0~AN3
mov  ACERL,a
mov  a,01h
mov  ADCR0,a            ; enable and connect AN0 channel to A/D converter
Start_conversion:
clr  START              ; high pulse on START bit to initiate conversion
set  START              ; reset A/D
clr  START              ; start A/D
clr  ADF                ; clear ADC interrupt request flag
set  ADE                ; 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 low byte conversion result value
mov  adrl_buffer,a      ; save result to user defined register
mov  a,ADRH             ; read high byte conversion result value
mov  adrh_buffer,a      ; save result to user defined register
:
:
EXIT_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
```

## Complementary PWM Output

The device provides a complementary output pair of signals which can be used as a PWM driver signal. The signal is sourced from the TM1 output signal, TP1. For PMOS type upper side driving, the PWM output is an active low signal while for NMOS type lower side driving the PWM output is an active high signal. When these complementary PWM outputs are both used to drive the upper and low sides, the dead time generator will automatically be enabled and a dead time, which is programmable using the DTPSC and DT bits in the CPR register, will be inserted to prevent excessive DC currents. The dead time will be inserted whenever the rising edge of the dead time generator input signal occurs. With a dead time insertion, the output signals are eventually sent out to the external power transistors. The dead time generator will only be enabled if both of the complementary outputs are used, as determined by the OUTCP bits in the TMPC register.




**Complementary PWM Output Block Diagram**




**Complementary PWM Output Waveform**

**CPR Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | DTPSC1 | DTPSC0 | DT2 | DT1 | DT0 |
| R/W | — | — | — | R/W | R/W | R/W | R/W | R/W |
| POR | — | — | — | 0 | 0 | 0 | 0 | 0 |

Bit 7~5 Unimplemented, read as "0"

Bit 4~3 **DTPSC1~DTPSC0**: Dead time prescaler division ratio select
00: $f_D=f_H/1$
01: $f_D=f_H/2$
10: $f_D=f_H/4$
11: $f_D=f_H/8$

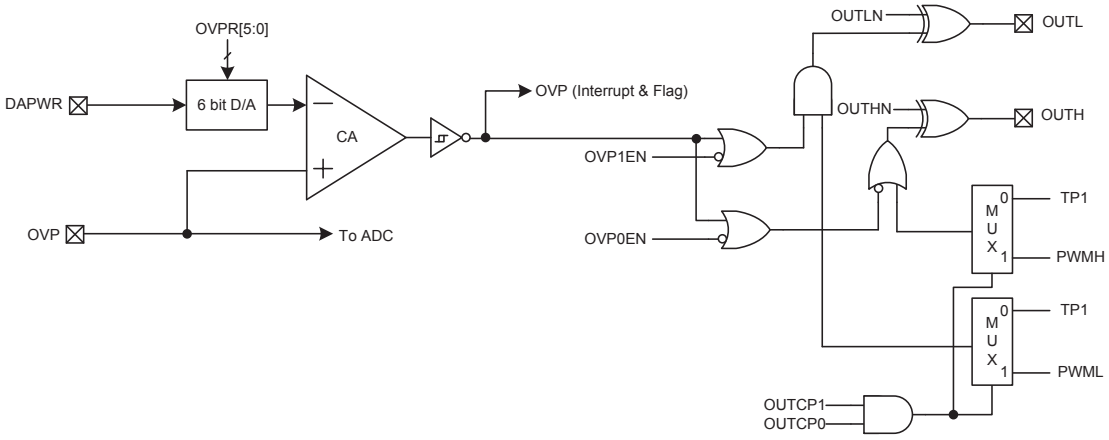
Bit 2~0 **DT2~DT0**: Dead time select, $t_D=1/f_D$
000: dead time is $[(1/f_D)-(1/f_H)]$~$(1/f_D)$
001: dead time is $[(2/f_D)-(1/f_H)]$~$(2/f_D)$
010: dead time is $[(3/f_D)-(1/f_H)]$~$(3/f_D)$
011: dead time is $[(4/f_D)-(1/f_H)]$~$(4/f_D)$
100: dead time is $[(5/f_D)-(1/f_H)]$~$(5/f_D)$
101: dead time is $[(6/f_D)-(1/f_H)]$~$(6/f_D)$
110: dead time is $[(7/f_D)-(1/f_H)]$~$(7/f_D)$
111: dead time is $[(8/f_D)-(1/f_H)]$~$(8/f_D)$

## Over Voltage Protection

The device includes an over voltage protection function which provides a protection mechanism for the applications. To prevent the output voltage from exceeding specific voltage level, the OVP input voltage is compared with a reference voltage generated by a 6-bit D/A converter. The 6-bit D/A converter power is supplied by the external power pin named DAPWR. Once the OVP input voltage is greater than the reference voltage, it will force the OUTH and OUTL signals inactive, i.e., the OUTH signal will be forced into a high state and the OUTL signal will be forced into a low state before the polarity control, to turn the external MOS off for over voltage protection.

The OUTH and OUTL signals can be forced to an inactive state when an over voltage event occurs. If an over voltage event occurs, the corresponding interrupt will be generated. Once the over voltage condition has disappeared, the OUTH and OUTL signals will recover to drive the PWM output.

More information for the OUTH and OUTL signal output control is described in the TMPC register.




**Over Voltage Protection Block Diagram**

## OVP Register

Overall operation of the over voltage protection is controlled using several registers. One register is used to provide the reference voltages for the over voltage protection together with the D/A converter enable control bit. There is one register used to cancel out the comparator input offset. The remaining two registers are control registers which control the OVP function, pin function, complementary PWM output pair polarity and comparator debounce time together with the hysteresis function. For a more detailed description regarding the input offset voltage cancellation procedures, refer to the corresponding application notes on the Holtek website.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OVPREF | DAPC | — | OVPR5 | OVPR4 | OVPR3 | OVPR2 | OVPR1 | OVPR0 |
| OCVPR0 | — | OVPEN | — | — | OVP1EN | OVP0EN | — | CHYAEN |
| OCVPR1 | — | OVPC | OUTHN | OUTLN | CAX | — | DBA1 | DBA0 |
| OCVPR2 | CAOFM | CARS | CAOF5 | CAOF4 | CAOF3 | CAOF2 | CAOF1 | CAOF0 |

**OVP Register Lists**

### OVPREF Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | DAPC | — | OVPR5 | OVPR4 | OVPR3 | OVPR2 | OVPR1 | OVPR0 |
| R/W | R/W | — | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | — | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7 **DAPC:** D/A Converter power source selection
0: From the VDD pin
1: From the DAPWR pin

Bit 6 Unimplemented, read as "0"

Bit 5~0 **OVPR5~OVPR0**: Over voltage Protection reference voltage select
OVP Reference voltage=(DAPWR/64)×OVPR [5:0], where OVPR [5:0] is in decimal notation.

**OCVPR0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | OVPEN | — | — | OVP1EN | OVP0EN | — | CHYAEN |
| R/W | — | R/W | — | — | R/W | R/W | — | R/W |
| POR | — | 0 | — | — | 0 | 0 | — | 0 |

Bit 7 Unimplemented, read as "0"

Bit 6 **OVPEN**: Over Voltage Protection function Enable control
0: Disable
1: Enable
If the OVPEN bit is cleared to 0, the over voltage protection function is disabled and no power will be consumed. This results in the comparator and D/A converter all being switched off.

Bit 5~4 Unimplemented, read as "0"

Bit 3 **OVP1EN**: OUTL Over Voltage Protection Enable control
0: Disable
1: Enable
This bit is used to control whether the OUTL signal is forced into an inactive state when an over voltage condition occurs. If the OVPEN and OVP1EN bits both are set to 1, the OUTL signal will be forced inactive when an over voltage condition occurs. If the OUTL signal protection function is disabled by clearing the OVP1EN bit to 0, the OUTL signal will not be affected when an over voltage condition occurs.

Bit 2 **OVP0EN**: OUTH Over Voltage Protection Enable control
0: Disable
1: Enable
This bit is used to control whether the OUTH signal is forced into an inactive state when an over voltage condition occurs. If the OVPEN and OVP0EN bits both are set to 1, the OUTH signal will be forced inactive when an over voltage condition occurs. If the OUTH signal protection function is disabled by clearing the OVP0EN bit to 0, the OUTH signal will not be affected when an over voltage condition occurs.

Bit 1 Unimplemented, read as “0”.

Bit 0 **CHYAEN**: Over Voltage Protection Comparator Hysteresis Enable control
0: Disable
1: Enable

**OCVPR1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | OVPC | OUTHN | OUTLN | CAX | — | DBA1 | DBA0 |
| R/W | — | R/W | R/W | R/W | R | — | R/W | R/W |
| POR | — | 0 | 1 | 1 | x | — | 0 | 0 |

"x": unknown

Bit 7 Unimplemented, read as "0"

Bit 6 **OVPC**: Over Voltage Protection Pin Control
0: OVP pin is disabled
1: OVP pin is enabled

Bit 5 **OUTHN:** OUTH signal inverting control
0: Non-inverted
1: Inverted
This bit is used to control whether the OUTH signal is inverted or not before output.

Bit 4 **OUTLN:** OUTL signal inverting control
0: Non-inverted
1: Inverted
This bit is used to control whether the OUTL signal is inverted or not before output.

Bit 3 **CAX:** Over Voltage Protection Comparator Digital Output
0: Positive input voltage < negative input voltage
1: Positive input voltage > negative input voltage

Bit 2 Unimplemented, read as "0".

Bit 1~0 **DBA1~DBA0**: Over Voltage Protection Comparator Debounce Time Select
00: No debounce
01: Debounce time=(3~4) × $1/f_H$
10: Debounce time=(7~8) × $1/f_H$
11: Debounce time=(15~16) × $1/f_H$

**OCVPR2 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CAOFM | CARS | CAOF5 | CAOF4 | CAOF3 | CAOF2 | CAOF1 | CAOF0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

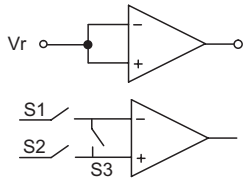
Bit 7 **CAOFM**: Over Voltage Protection Comparator Input Offset Voltage Cancellation Mode Select
0: Comparator mode
1: Input Offset Voltage Cancellation mode

Bit 6 **CARS**: Over Voltage Protection Comparator Offset Voltage Cancellation Reference Input Select
0: Comparator negative input selected
1: Comparator positive input selected

Bit 5~0 **CAOF5~CAOF0**: Over Voltage Protection Comparator Input Voltage Offset Cancellation Setting

## Comparator Cancellation function

Calibrate its input offset voltage before using the comparator.




The calibration steps are as following:

- Set CAOMF=1 to setup the offset cancellation mode, here S3 is closed.
- Set CARS to select which input pin is to be used as the reference voltage – S1 or S2 is closed.
- Adjust CAOF [4:0] until the output status changes.
- Set CAOMF = 0 to enter the normal comparator mode.

# Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupts functions. The external interrupts are generated by the action of the external INT0 and INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, Time Base, LVD, Over Voltage Protection and the A/D converter.

## Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The registers fall into three categories. The first is the INTC0~INTC2 registers which setup the primary interrupts, the second is the MFI0~MFI4 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupt trigger edge type.

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

| Function | Enable Bit | Request Flag | Notes |
|---|---|---|---|
| Global | EMI | — | — |
| INTn Pins | INTnE | INTnF | n=0~1 |
| Over Voltage Protection Interrupt | OVPE | OVPF | — |
| Multi-function | MFnE | MFnF | n=0~4 |
| A/D Converter | ADE | ADF | — |
| Time Base | TBnE | TBnF | n=0~1 |
| LVD | LVE | LVF | — |
| TM | TnPE | TnPF | n=0~3 |
| | TnAE | TnAF | |

**Interrupt Register Bit Naming Conventions**

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTEG | — | — | — | — | INT1S1 | INT1S0 | INT0S1 | INT0S0 |
| INTC0 | — | MF0F | OVPF | INT0F | MF0E | OVPE | INT0E | EMI |
| INTC1 | MF4F | MF3F | MF2F | MF1F | MF4E | MF3E | MF2E | MF1E |
| INTC2 | INT1F | TB1F | TB0F | ADF | INT1E | TB1E | TB0E | ADE |
| MFI0 | — | — | T0AF | T0PF | — | — | T0AE | T0PE |
| MFI1 | — | — | T1AF | T1PF | — | — | T1AE | T1PE |
| MFI2 | — | — | T2AF | T2PF | — | — | T2AE | T2PE |
| MFI3 | — | — | T3AF | T3PF | — | — | T3AE | T3PE |
| MFI4 | — | — | — | LVF | — | — | — | LVE |

**Interrupt Register List**

### INTEG Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | INT1S1 | INT1S0 | INT0S1 | INT0S0 |
| R/W | — | — | — | — | R/W | R/W | R/W | R/W |
| POR | — | — | — | — | 0 | 0 | 0 | 0 |

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **INT1S1, INT1S0**: Defines INT1 interrupt active edge
00: Disabled Interrupt
01: Rising Edge Interrupt
10: Falling Edge Interrupt
11: Dual Edge Interrupt

Bit 1~0 **INT0S1, INT0S0**: Defines INT0 interrupt active edge
00: Disabled Interrupt
01: Rising Edge Interrupt
10: Falling Edge Interrupt
11: Dual Edge Interrupt

### INTC0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | MF0F | OVPF | INT0F | MF0E | OVPE | INT0E | EMI |
| R/W | — | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7 Unimplemented, read as “0”

Bit 6 **MF0F:** Multi-function 0 Interrupt request flag
0: No request
1: Interrupt request

Bit 5 **OVPF:** Over Voltage Protection interrupt request flag
0: No request
1: Interrupt request

Bit 4 **INT0F:** INT0 pin interrupt request flag
0: No request
1: Interrupt request

Bit 3 **MF0E:** Multi-function 0 Interrupt control
0: Disable
1: Enable

Bit 2 **OVPE:** Over Voltage Protection interrupt control
0: Disable
1: Enable

Bit 1 **INT0E:** INT0 pin interrupt control
0: Disable
1: Enable

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

**INTC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | MF4F | MF3F | MF2F | MF1F | MF4E | MF3E | MF2E | MF1E |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7 **MF4F:** Multi-function 4 Interrupt request flag
0: No request
1: Interrupt request

Bit 6 **MF3F:** Multi-function 3 Interrupt request flag
0: No request
1: Interrupt request

Bit 5 **MF2F:** Multi-function 2 Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **MF1F:** Multi-function 1 Interrupt request flag
0: No request
1: Interrupt request

Bit 3 **MF4E:** Multi-function 4 Interrupt control
0: Disable
1: Enable

Bit 2 **MF3E:** Multi-function 3 Interrupt control
0: Disable
1: Enable

Bit 1 **MF2E:** Multi-function 2 Interrupt control
0: Disable
1: Enable

Bit 0 **MF1E:** Multi-function 1 Interrupt control
0: Disable
1: Enable

### INTC2 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | INT1F | TB1F | TB0F | ADF | INT1E | TB1E | TB0E | ADE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7 **INT1F:** INT1 pin interrupt request flag
0: No request
1: Interrupt request

Bit 6 **TB1F:** Time Base 1 Interrupt request flag
0: No request
1: Interrupt request

Bit 5 **TB0F:** Time Base 0 Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **ADF:** A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 3 **INT1E:** INT1 pin interrupt control
0: Disable
1: Enable

Bit 2 **TB1E:** Time Base 1 Interrupt control
0: Disable
1: Enable

Bit 1 **TB0E:** Time Base 0 Interrupt control
0: Disable
1: Enable

Bit 0 **ADE:** A/D Converter interrupt control
0: Disable
1: Enable

### MFI0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | T0AF | T0PF | — | — | T0AE | T0PE |
| R/W | — | — | R/W | R/W | — | — | R/W | R/W |
| POR | — | — | 0 | 0 | — | — | 0 | 0 |

Bit 7~6 Unimplemented, read as "0"

Bit 5 **T0AF:** TM0 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **T0PF:** TM0 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **T0AE:** TM0 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **T0PE:** TM0 Comparator P match Interrupt control
0: Disable
1: Enable

### MFI1 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | T1AF | T1PF | — | — | T1AE | T1PE |
| R/W | — | — | R/W | R/W | — | — | R/W | R/W |
| POR | — | — | 0 | 0 | — | — | 0 | 0 |

Bit 7~6 Unimplemented, read as "0"

Bit 5 **T1AF:** TM1 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **T1PF:** TM1 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **T1AE:** TM1 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **T1PE:** TM1 Comparator P match Interrupt control
0: Disable
1: Enable

### MFI2 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | T2AF | T2PF | — | — | T2AE | T2PE |
| R/W | — | — | R/W | R/W | — | — | R/W | R/W |
| POR | — | — | 0 | 0 | — | — | 0 | 0 |

Bit 7~6 Unimplemented, read as "0"

Bit 5 **T2AF:** TM2 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **T2PF:** TM2 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **T2AE:** TM2 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **T2PE:** TM2 Comparator P match Interrupt control
0: Disable
1: Enable

### MFI3 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | T3AF | T3PF | — | — | T3AE | T3PE |
| R/W | — | — | R/W | R/W | — | — | R/W | R/W |
| POR | — | — | 0 | 0 | — | — | 0 | 0 |

Bit 7~6 Unimplemented, read as "0"

Bit 5 **T3AF:** TM3 Comparator A match Interrupt request flag
0: No request
1: Interrupt request

Bit 4 **T3PF:** TM3 Comparator P match Interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **T3AE:** TM3 Comparator A match Interrupt control
0: Disable
1: Enable

Bit 0 **T3PE:** TM3 Comparator P match Interrupt control
0: Disable
1: Enable

### MFI4 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | LVF | — | — | — | LVE |
| R/W | — | — | — | R/W | — | — | — | R/W |
| POR | — | — | — | 0 | — | — | — | 0 |

Bit 7~5 Unimplemented, read as "0"

Bit 4 **LVF:** LVD Interrupt request flag
0: No request
1: Interrupt request

Bit 3~1 Unimplemented, read as "0"

Bit 0 **LVE:** LVD interrupt control
0: Disable
1: Enable

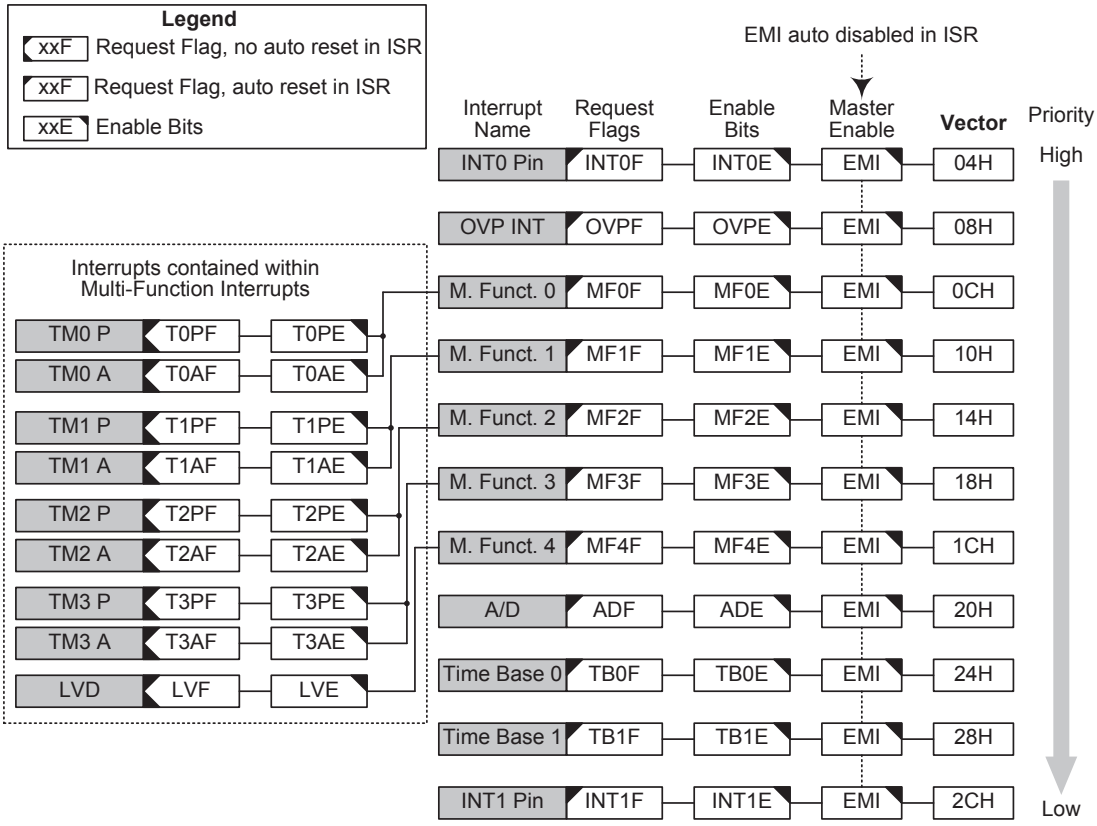
## Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P or Comparator A match or A/D conversion completion etc, the relevant interrupt request flag will be set. Whether the request flag actually generates a program jump to the relevant interrupt vector is determined by the condition of the interrupt enable bit. If the enable bit is set high then the program will jump to its relevant vector; if the enable bit is zero then although the interrupt request flag is set an actual interrupt will not be generated and the program will not jump to the relevant interrupt vector. The global interrupt enable bit, if cleared to zero, will disable all interrupts.

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

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

If an interrupt requires immediate servicing while the program is already in another interrupt service routine, the EMI bit should be set after entering the routine, to allow interrupt nesting. If the stack is full, the interrupt request will not be acknowledged, even if the related interrupt is enabled, until the Stack Pointer is decremented. If immediate service is desired, the stack must be prevented from becoming full. In case of simultaneous requests, the accompanying diagram shows the priority that is applied. All of the interrupt request flags when set will wake-up the device if it is in SLEEP or IDLE Mode, however to prevent a wake-up from occurring the corresponding flag should be set before the device is in SLEEP or IDLE Mode.




**Interrupt Structure**

### External Interrupt

The external interrupts are controlled by signal transitions on the pins INT0, INT1. An external interrupt request will take place when the external interrupt request flags, INT0F, INT1F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E, INT1E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F, INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input. The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

### OVP Interrupt

An OVP Interrupt request will take place when the Over Voltage Protection Interrupt request flag, OVPF, is set, which occurs when the Over Voltage Protection function detects an over voltage condition. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Over Voltage Protection Interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and an over voltage condition occurs, a subroutine call to the OVP Interrupt vector, will take place. When the Over Voltage Protection Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts and the interrupt request flag will be also automatically cleared.

### Multi-function Interrupt

Within the device there are up to five Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM Interrupts and LVD interrupt.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags MFnF are set. The Multi-function interrupt flags will be set when any of their included functions generate an interrupt request flag. To allow the program to branch to its respective interrupt vector address, when the Multi-function interrupt is enabled and the stack is not full, and either one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine call to one of the Multi-function interrupt vectors will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

However, it must be noted that, although the Multi-function Interrupt flags will be automatically reset when the interrupt is serviced, the request flags from the original source of the Multi-function interrupts, namely the TM Interrupts and LVD interrupt, will not be automatically reset and must be manually reset by the application program.

## A/D Converter Interrupt

The A/D Converter Interrupt is controlled by the termination of an A/D conversion process. An A/D Converter Interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D Converter Interrupt vector, will take place. When the interrupt is serviced, the A/D Converter Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

## Time Base Interrupts

The function of the Time Base Interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

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

### TBC Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TBON | TBCK | TB11 | TB10 | — | TB02 | TB01 | TB00 |
| R/W | R/W | R/W | R/W | R/W | — | R/W | R/W | R/W |
| POR | 0 | 0 | 1 | 1 | — | 1 | 1 | 1 |

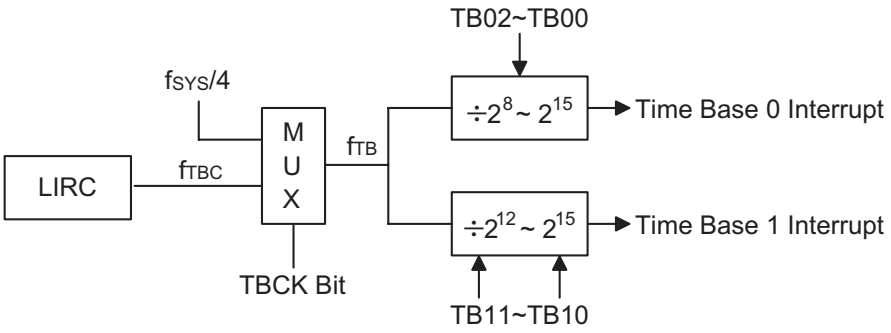
Bit 7 **TBON**: TB0 and TB1 Control bit
0: Disable
1: Enable

Bit 6 **TBCK**: Select $f_{TB}$ Clock
0: $f_{TBC}$
1: $f_{SYS}/4$

Bit 5~4 **TB11~TB10**: Select Time Base 1 Time-out Period
00: $4096/f_{TB}$
01: $8192/f_{TB}$
10: $16384/f_{TB}$
11: $32768/f_{TB}$

Bit 3 Unimplemented, read as "0"

Bit 2~0 **TB02~TB00**: Select Time Base 0 Time-out Period
000: $256/f_{TB}$
001: $512/f_{TB}$
010: $1024/f_{TB}$
011: $2048/f_{TB}$
100: $4096/f_{TB}$
101: $8192/f_{TB}$
110: $16384/f_{TB}$
111: $32768/f_{TB}$




**Time Base Interrupt**

## LVD Interrupt

The Low Voltage Detector Interrupt is contained within the Multi-function Interrupt. An LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Low Voltage Interrupt enable bit, LVE, and associated Multi-function interrupt enable bit, MF4E, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the LVD Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the LVF flag will not be automatically cleared, it has to be cleared by the application program.

## TM Interrupts

The Compact and Periodic Type TMs both have two interrupts each. All of the TM interrupts are contained within the Multi-function Interrupts. For each TM there are two interrupt request flags TnPF and TnAF and two enable bits TnPE and TnAE. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, the respective TM Interrupt enable bit, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant Multi-function Interrupt vector locations, will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the related MFnF flag will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

## Interrupt Wake-up Function

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

## Programming Considerations

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

Where a certain interrupt is contained within a Multi-function interrupt, then when the interrupt service routine is executed, as only the Multi-function interrupt request flags, MFnF, will be automatically cleared, the individual request flag for the function needs to be cleared by the application program.

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

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

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

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

## Low Voltage Detector – LVD

The device has a Low Voltage Detector function, also known as LVD. This enables the device to monitor the power supply voltage, $V_{DD}$, and provides 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 LVDC. Three bits in this register, VLVD2~VLVD0, 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 $V_{DD}$ 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.

#### LVDC Register

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

Bit 7~6 Unimplemented, read as "0"

Bit 5 **LVDO**: LVD Output Flag
0: No low voltage detect
1: Low voltage detect

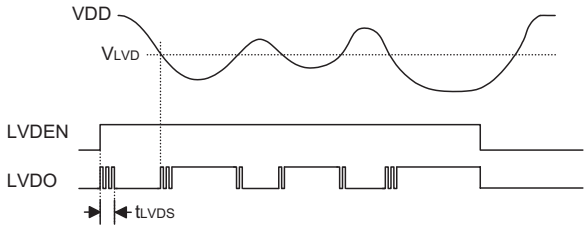
Bit 4 **LVDEN**: Low Voltage Detector Control
0: Disable
1: Enable

Bit 3 Unimplemented, read as "0"

Bit 2~0 **VLVD2~VLVD0**: Select LVD Voltage
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V

## LVD Operation

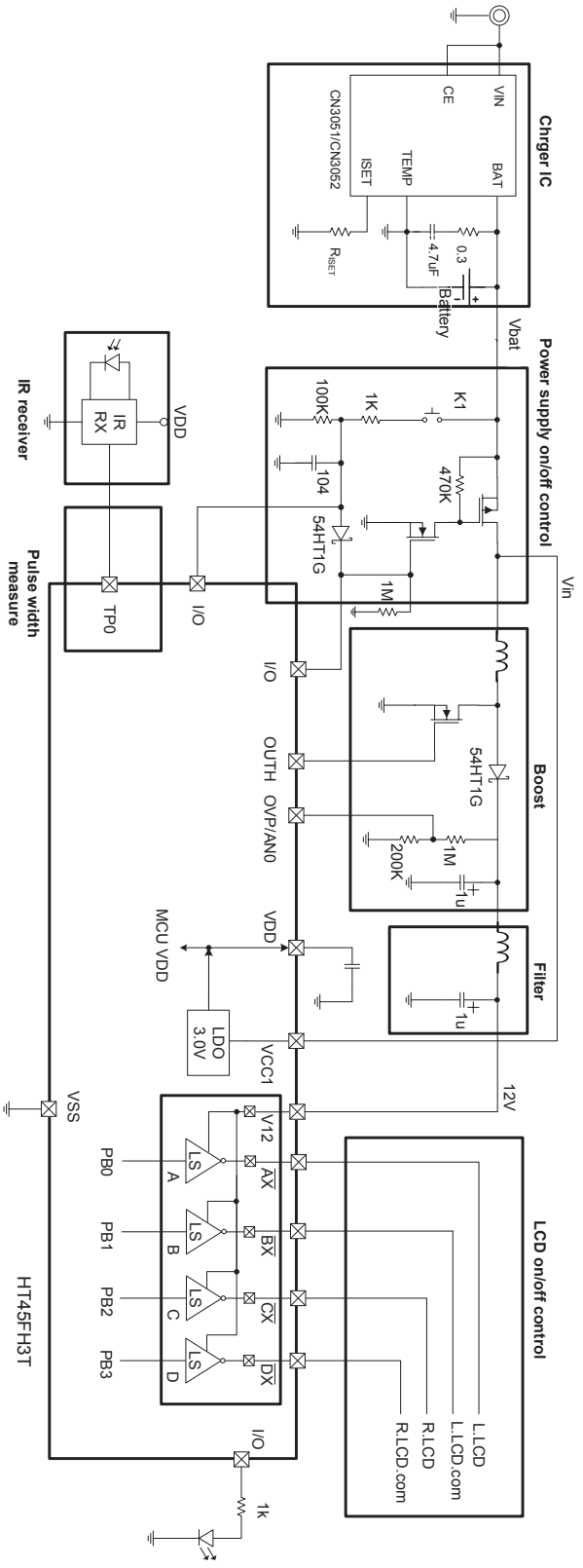
The Low Voltage Detector function operates by comparing the power supply voltage, $V_{DD}$, with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.2V and 4.0V. When the power supply voltage, $V_{DD}$, falls below this pre-determined value, the LVDO bit will be set 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. 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 $V_{DD}$ voltage may rise and fall rather slowly, at the voltage nears that of $V_{LVD}$, there may be multiple bit LVDO transitions.




**LVD Operation**

The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of $t_{LVD}$ after the LVDO bit has been set high by a low voltage condition. When the device is powered down the Low Voltage Detector will remain active if the LVDEN bit is high. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if $V_{DD}$ falls below the preset LVD voltage. This will cause the device to wake-up from the SLEEP or IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVF flag should be first set high before the device enters the SLEEP or IDLE 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 a 4MHz system oscillator, most instructions would be implemented within 1μs and branch or call instructions would be implemented within 2μs. 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 |
| **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 |

| Mnemonic | Description | Cycles | Flag Affected |
|---|---|---|---|
| **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** | | | |
| TABRD [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 bit wise logical AND operation. The result is stored in the Accumulator.

Operation ACC ← ACC "AND" x

Affected flag(s) Z

**ANDM A,[m]** Logical AND ACC to Data Memory

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

Operation [m] ← ACC "AND" [m]

Affected flag(s) Z

**CALL addr** Subroutine call

Description Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction.

Operation Stack ← Program Counter + 1
Program Counter ← addr

Affected flag(s) None

**CLR [m]** Clear Data Memory

Description Each bit of the specified Data Memory is cleared to 0.

Operation [m] ← 00H

Affected flag(s) None

**CLR [m].i** Clear bit of Data Memory

Description Bit i of the specified Data Memory is cleared to 0.

Operation [m].i ← 0

Affected flag(s) None

**CLR WDT** Clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

**CLR WDT1** Pre-clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT2 and must be executed alternately with CLR WDT2 to have effect. Repetitively executing this instruction without alternately executing CLR WDT2 will have no effect.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

**CLR WDT2** Pre-clear Watchdog Timer

Description The TO, PDF flags and the WDT are all cleared. Note that this instruction works in conjunction with CLR WDT1 and must be executed alternately with CLR WDT1 to have effect. Repetitively executing this instruction without alternately executing CLR WDT1 will have no effect.

Operation WDT cleared
TO ← 0
PDF ← 0

Affected flag(s) TO, PDF

**CPL [m]** Complement Data Memory

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

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

**CPLA [m]** Complement Data Memory with result in ACC

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

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

Affected flag(s) Z

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

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

Operation [m] ← ACC + 00H or
[m] ← ACC + 06H or
[m] ← ACC + 60H or
[m] ← ACC + 66H

Affected flag(s) C

**DEC [m]** Decrement Data Memory

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

Operation [m] ← [m] − 1

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation ACC ← [m] − 1

Affected flag(s) Z

**HALT** Enter power down mode

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

Operation TO ← 0
PDF ← 1

Affected flag(s) TO, PDF

**INC [m]** Increment Data Memory

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

Operation [m] ← [m] + 1

Affected flag(s) Z

**INCA [m]** Increment Data Memory with result in ACC

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

Operation ACC ← [m] + 1

Affected flag(s) Z

**JMP addr** Jump unconditionally

Description The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction.

Operation Program Counter ← addr

Affected flag(s) None

**MOV A,[m]** Move Data Memory to ACC

Description The contents of the specified Data Memory are copied to the Accumulator.

Operation ACC ← [m]

Affected flag(s) None

**MOV A,x** Move immediate data to ACC

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

**MOV [m],A** Move ACC to Data Memory

Description The contents of the Accumulator are copied to the specified Data Memory.

Operation [m] ← ACC

Affected flag(s) None

**NOP** No operation

Description No operation is performed. Execution continues with the next instruction.

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

**OR A,x** Logical OR immediate data to ACC

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

**RET** Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

**RETI** Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

**RL [m]** Rotate Data Memory left

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

**RLC [m]** Rotate Data Memory left through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

**RR [m]** Rotate Data Memory right

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

**RRC [m]** Rotate Data Memory right through Carry

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

Operation ACC ← ACC − [m] − C

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

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

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

Operation [m] ← ACC − [m] − C

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

**SDZ [m]** Skip if decrement Data Memory is 0

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

Operation [m] ← [m] − 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

**SET [m]** Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

**SET [m].i** Set bit of Data Memory

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

Operation [m].i ← 1

Affected flag(s) None

**SIZ [m]** Skip if increment Data Memory is 0

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

**SUB A,[m]** Subtract Data Memory from ACC

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

**SUB A,x** Subtract immediate data from ACC

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

Operation ACC ← ACC − x

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

**SWAP [m]** Swap nibbles of Data Memory

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

**SZ [m]** Skip if Data Memory is 0

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

**TABRD [m]** Read table to TBLH and Data Memory

Description The program code addressed by the table pointer (TBHP and 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

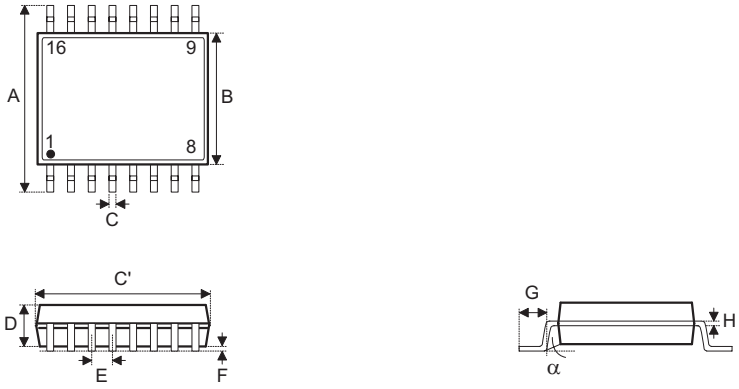
## Package Information

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

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

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

## 16-pin SSOP (150mil) Outline Dimensions




| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.228 | — | 0.244 |
| B | 0.150 | — | 0.157 |
| C | 0.008 | — | 0.012 |
| C' | 0.189 | — | 0.197 |
| D | 0.054 | — | 0.060 |
| E | — | 0.025 | — |
| F | 0.004 | — | 0.010 |
| G | 0.022 | — | 0.028 |
| H | 0.007 | — | 0.010 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 5.79 | — | 6.20 |
| B | 3.81 | — | 3.99 |
| C | 0.20 | — | 0.30 |
| C‘ | 4.80 | — | 5.00 |
| D | 1.37 | — | 1.52 |
| E | — | 0.64 | — |
| F | 0.10 | — | 0.25 |
| G | 0.56 | — | 0.71 |
| H | 0.18 | — | 0.25 |
| α | 0° | — | 8° |

Copyright© 2013 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.