Wireless Charger A/D Flash MCU

# HT66FW2230

Revision: V1.40 Date: February 26, 2019

www.holtek.com

# Table of Contents

# Features

## CPU Features

- Operating Voltage : $f_{SYS}$ = 20MHz: 4.0V~5.5V
- Up to 0.2μs instruction cycle with 20MHz system clock at $V_{DD}$=5V
- Power down and wake-up functions to reduce power consumption
- Oscillators
  - External Crystal oscillator – HXT
  - Internal 20MHz RC oscillator – HIRC
  - Internal 32kHz – LIRC
- Multi-mode operation: NORMAL, SLOW, IDLE and SLEEP
- All instructions executed in one or two instruction cycles
- Table read instructions
- 63 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

## Peripheral Features

- Flash Program Memory: 4K×16
- RAM Data Memory: 128×8
- True EEPROM Memory: 64×8
- Watchdog Timer function
- Up to 21 bidirectional I/O lines
- Two pin-shared external interrupts
- Multiple Timer Module for time measure, input capture, compare match output, PWM output function or single pulse output function
- $I^2C$ function
- Over current protection (OCP) and demodulation functions
- Clock generator output:
  - 100kHz~220kHz in 100Hz steps
  - 1MHz
- 2.08V Reference voltage for ADC
- Dual Time-Base functions for generation of fixed time interrupt signals
- Multi-channel 12-bit resolution A/D converter
- Low voltage reset function
- Low voltage detect function
- Package types: 28-pin SSOP/QFN

## General Description

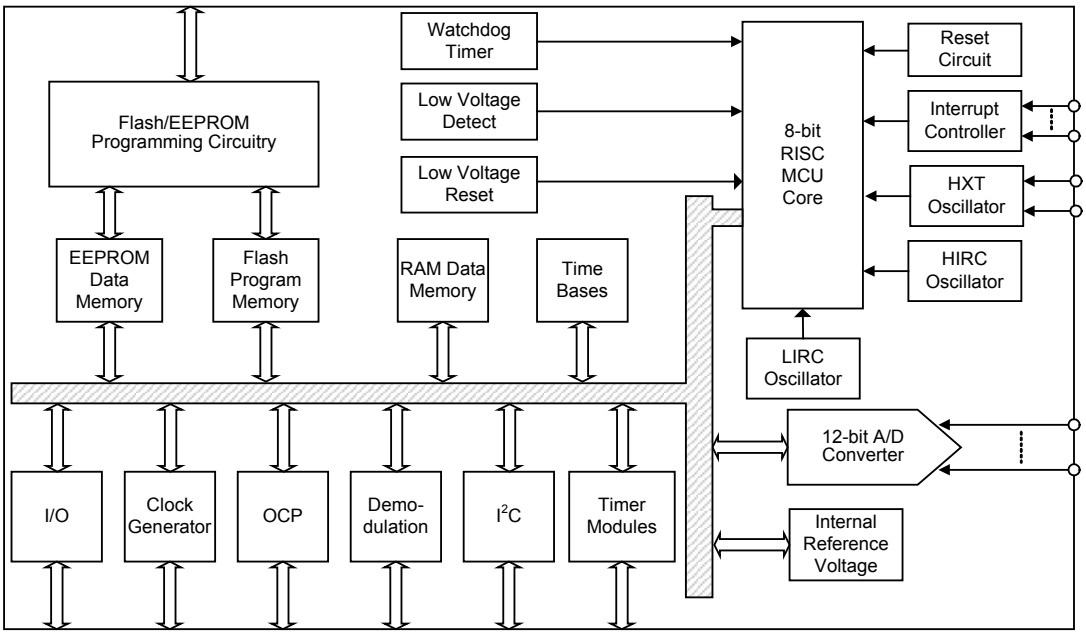
The device is Flash Memory A/D type 8-bit high performance ASSP architecture microcontrollers and specially designed for wireless power transmission control. Offering users the convenience of Flash Memory multi-programming features, this device also includes a wide range of functions and features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

Analog feature includes a multi-channel 12-bit A/D converter and Digital feature includes D/A Converter. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated $I^2C$ interface function, this popular interface which provides designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset, Over Current Protection and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

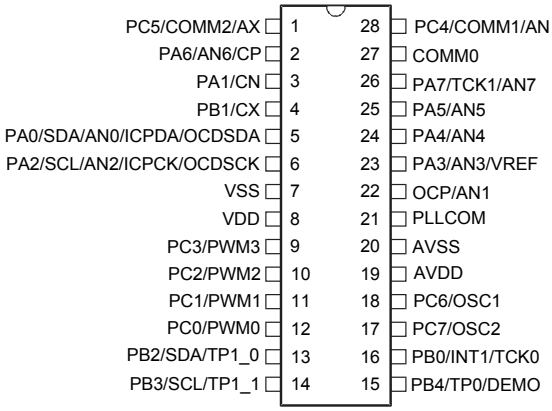
A full choice of HXT, HIRC and LIRC oscillator functions are provided including a fully integrated system oscillator which requires no external components for its implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users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

The inclusion of flexible I/O programming features, PLL, OCP, Demodulator, Clock generator, reference voltage generator, Time-Base functions along with many other features ensure that the device will find excellent use in wireless power transmission applications.

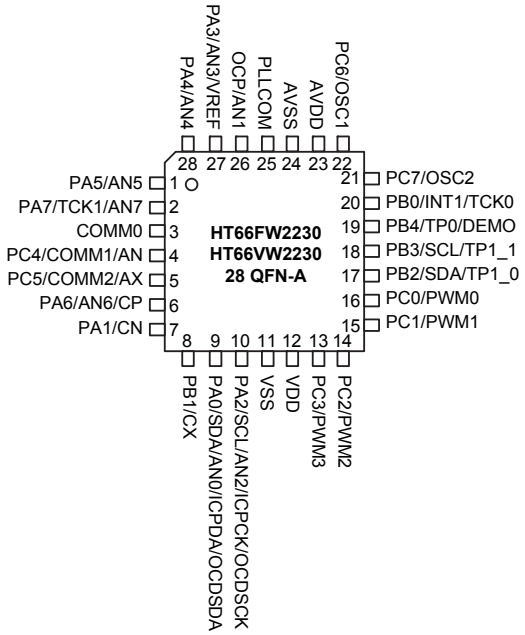
## Block Diagram



## Pin Assignment




**HT66FW2230/HT66VW2230**
**28 SSOP-A**




## Pin Descriptions

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| PA0/SDA/ AN0/ICPDA/ OCDSDA | PA0 | PAPU<br>PAWU<br>PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | SDA | PAS0 | ST | O.D | $I^2C$ data line |
| | AN0 | PAS0 | AN | — | ADC input channel |
| | ICPDA | — | ST | CMOS | ICP Data/Address pin |
| | OCDSDA | — | ST | CMOS | OCDS Data/Address, for EV chip only |
| PA1/CN | PA1 | PAPU<br>PAWU<br>PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | CN | PAS0 | AN | — | Comparator input |

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| OCP/AN1 | OCP | OCPC0 | AN | — | OCP input |
| | AN1 | ADCR0 | AN | — | ADC input channel |
| PA2/SCL/ AN2/ICPCK/ OCDSCK | PA2 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | SCL | PAS0 | ST | O.D | $I^2C$ clock line |
| | AN2 | PAS0 | AN | — | ADC input channel |
| | ICPCK | — | ST | — | ICP clock pin |
| | OCDSCK | — | ST | — | OCDS clock pin, for EV chip only |
| PA3/AN3/ VREF | PA3 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | AN3 | PAS0 | AN | — | ADC input channel |
| | VREF | PAS0 ADCR1 | AN | — | ADC reference voltage input |
| PA4/AN4 | PA4 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | AN4 | PAS1 | AN | — | ADC input channel |
| PA5/AN5 | PA5 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | AN5 | PAS1 | AN | — | ADC input channel |
| PA6/ AN6/ CP | PA6 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | AN6 | PAS1 | AN | — | ADC input channel |
| | CP | PAS1 | AN | — | Comparator input |
| PA7/TCK1/ AN7 | PA7 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. |
| | TCK1 | — | ST | — | TM clock input |
| | AN7 | PAS1 | AN | — | ADC input channel |
| PB0/INT1/ TCK0 | PB0 | RSTC | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | INT1 | INTEG INTC2 | ST | — | External interrupt 1 |
| | TCK0 | — | ST | — | TM clock input |
| PB1/CX | PB1 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | CX | PBS0 | — | CMOS | Comparator output |
| PB2/SDA / TP1_0 | PB2 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | SDA | PBS0 | ST | O.D | $I^2C$ data line |
| | TP1_0 | PBS0 | ST | CMOS | TM1 I/O |
| PB3/SCL / TP1_1 | PB3 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | SCL | PBS0 | ST | O.D | $I^2C$ clock line |
| | TP1_1 | PBS0 | ST | CMOS | TM1 I/O |

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| PB4/TP0/ INT0/ DEMO | PB4 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | TP0 | PBS0 | ST | CMOS | TM0 I/O |
| | INT0 | INTEG INTC0 | ST | — | External interrupt 0 |
| | DEMO | PBS0 | — | CMOS | Demodulation output |
| PC0/PWM0 | PC0 | PCPU PCS0 | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | PWM0 | PCS0 | — | CMOS | PWM0 |
| PC1/PWM1 | PC1 | PCPU PCS0 | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | PWM1 | PCS0 | — | CMOS | PWM1 |
| PC2/PWM2 | PC2 | PCPU PCS0 | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | PWM2 | PCS0 | — | CMOS | PWM2 |
| PC3/PWM3 | PC3 | PCPU PCS0 | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | PWM3 | PCS0 | — | CMOS | PWM3 |
| PC4/ COMM1/ AN | PC4 | PCPU PCS1 | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | COMM1 | DCMISC PCS1 | AN | — | Demodulation input |
| | AN | PCS1 | AN | — | OPA input |
| PC5/ COMM2/ AX | PC5 | PCPU PCS1 | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | COMM2 | DCMISC PCS1 | AN | — | Demodulation input |
| | AX | PCS1 | — | AO | OPA output |
| PC6/OSC1 | PC6 | PCPU PCS1 | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | OSC1 | PCS1 | AN | — | HXT |
| PC7/OSC2 | PC7 | PCPU PCS1 | ST | CMOS | General purpose I/O. Register enabled pull-up. |
| | OSC2 | PCS1 | — | AO | HXT |
| COMM0 | COMM0 | DCMISC | AN | — | Demodulation input |
| PLLCOM | PLLCOM | — | AN | — | PLL compensation (filter) |
| OSC1 | OSC1 | — | HXT | — | Oscillator input |
| OSC2 | OSC2 | — | — | HXT | Oscillator output |
| VDD | VDD | — | PWR | — | Digital positive power supply. |
| VSS | VSS | — | PWR | — | Digital negative power supply. |
| AVDD | AVDD | — | PWR | — | Analog positive power supply. |
| AVSS | AVSS | — | PWR | — | Analog negative power supply. |

Legend: I/T: Input type; O/T: Output type
OPT: Optional by configuration option (CO) or register option
PWR: Power; ST: Schmitt Trigger input
CMOS: CMOS output; AN: Analog input pin
O.D: open drain AO: analog output

## 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 125˚C
Operating Temperature..... -40˚C to 85˚C

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

## D.C. Characteristics

Ta= 25˚C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{DD}$ | Operating Voltage | — | $f_{SYS}$=20MHz | 4.0 | — | 5.5 | V |
| $I_{DD1}$ | Operating Current, Normal Mode, $f_{SYS}=f_H$ | 5V | No load, $f_H$=20MHz, ADC off, WDT enable | — | 5.0 | 7.5 | mA |
| $I_{DD2}$ | Operating Current, Slow Mode, $f_{SYS}= f_L$=LIRC; $f_{SUB}$=LIRC | 5V | No load, $f_{SYS}$=LIRC, ADC off, WDT enable | — | 30 | 50 | μA |
| $I_{IDLE0}$ | IDLE0 Mode Stanby Current (LIRC on) | 5V | No load, ADC off, WDT enable, LVR disable | — | 2.5 | 5.0 | μA |
| $I_{IDLE1}$ | IDLE1 Mode Stanby Current | 5V | No load, ADC off, WDT enable, $f_{SYS}$=20MHz on | — | 2.2 | 3.3 | mA |
| $I_{SLEEP}$ | SLEEP Mode Stanby Current (LIRC on) | 5V | No load, ADC off, WDT enable, LVR disable | — | 2.5 | 5.0 | μA |
| $V_{IL}$ | Input Low Voltage for PA, PB, PC, INTn, TPn | 5V | — | 0 | — | 1.5 | V |
| | | — | — | 0 | — | $0.2V_{DD}$ | V |
| $V_{IH}$ | Input High Voltage for PA, PB, PC, INTn, TPn | 5V | — | 3.5 | — | 5.0 | V |
| | | — | — | $0.8V_{DD}$ | — | $V_{DD}$ | V |
| $V_{LVR}$ | Low Voltage Reset Voltage | — | LVR 2.1V @25˚C | -5% | 2.1 | +5% | V |
| | | | LVR 2.55V @25˚C | -5% | 2.55 | +5% | V |
| | | | LVR 3.15V @25˚C | -5% | 3.15 | +5% | V |
| | | | LVR= 3.8V @25˚C | -5% | 3.8 | -5% | V |
| $I_{OH1}$ | I/O Port Source Current (PA,PB,PC4~PC7) | 5V | $V_{OH}=0.9V_{DD}$ | -5 | -10 | — | mA |
| $I_{OL1}$ | I/O Port Sink Current (PA,PB,PC4~PC7) | 5V | $V_{OL}=0.1V_{DD}$ | 10 | 20 | — | mA |
| $I_{OH2}$ | I/O Source Current (PC0~PC3) | 3V | — | 16 | 32 | — | mA |
| | | 5V | | 40 | 80 | — | |
| $I_{OL2}$ | I/O Sink Current (PC0~PC3) | 3V | — | -16 | -32 | — | mA |
| | | 5V | | -40 | -80 | — | |
| $R_{PH}$ | Pull-high Resistance for I/O Ports | 5V | — | 10 | 30 | 50 | kΩ |

## A.C. Characteristics

Ta= 25˚C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $f_{SYS}$ | System Clock | 4.0V~5.5V | — | — | — | 20 | MHz |
| $f_{SUB}$ | System Clock (LIRC) | 5V | Ta = 25°C | -10% | 32 | +10% | kHz |
| | | $V_{LVR}$~5.5V | Ta = -40°C~85°C | -30% | 32 | +60% | kHz |
| $t_{TIMER}$ | TCKn , TPn Input Pin Pulse Width | — | — | 0.3 | — | — | μs |
| $t_{INT}$ | Interrupt Pulse Width | — | — | 10 | — | — | μs |
| $t_{EERD}$ | EEPROM Read Time | 5V | — | — | 2 | 4 | $t_{SYS}$ |
| $t_{EEWR}$ | EEPROM Write Time | 5V | — | — | 2 | 4 | ms |
| $t_{SST}$ | System Start-up Timer Period (Wake-up From HALT, $f_{SYS}$ Off at HALT state) | 5V | $f_{SYS}=f_{HXT}$ | — | 128 | — | $t_{SYS}$ |
| | | 5V | $f_{SYS}=f_{LIRC}$ | — | 2 | — | $t_{SYS}$ |
| | System Start-up Timer Period (Wake-up From HALT, $f_{SYS}$ on at HALT state) | 5V | — | — | 2 | — | $t_{SYS}$ |
| $t_{RSTD}$ | System Reset Delay Time (Power On Reset, LVR, WDTC/ LVRC S/W reset) | 5V | — | 25 | 50 | 100 | ms |
| | System Reset Delay Time (WDT time-out reset) | 5V | — | 8.3 | 16.7 | 33.3 | ms |

Note: $t_{SYS}= 1/f_{SYS;}\ t_{SUB} = 1/f_{SUB}$

## A/D Characteristics

Ta= 25˚C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $AV_{DD}$ | A/D Converter Operating Voltage | — | — | $V_{LVR}$ | — | 5.5 | V |
| $V_{ADI}$ | A/D Converter Input Voltage | — | — | 0 | — | $V_{REF}$ | V |
| $V_{REF}$ | A/D Converter Reference Voltage | — | — | 2 | — | $AV_{DD}$ | 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 | 5V | No load, $t_{ADCK}$=0.5μs | — | 1.20 | 1.80 | mA |
| $t_{ADCK}$ | A/D Converter Clock Period | — | — | 0.5 | — | 10 | μs |
| $t_{ADC}$ | A/D Conversion Time (Include Sample and Hold Time) | — | 12-bit A/D Converter | — | 16 | — | $t_{ADCK}$ |
| $t_{ADS}$ | A/D Converter Sampling Time | — | — | — | 4 | — | $t_{ADCK}$ |
| $t_{ON2ST}$ | A/D Converter On-to-Start Time | — | — | 2 | — | — | μs |

## PLL Electrical Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $I_{PLL}$ | Power Consumption | 5V | — | — | — | 1.5 | mA |
| $f_{PLL}$ | PLL Frequency Deviation (HXT) | 4.0V~5.5V | $f_{PLL}$=100K~220K, step=100Hz | — | 0.05 | — | % |
| $t_{STB0}$ | Stable Time (Change Frequency) | 5V | $f_{PLL}$=100K→160K | — | 2 | 3 | ms |
| $t_{STB1}$ | Stable Time (PLL off→on) | 5V | — | — | — | 8 | ms |
| Jitter | PLL Timing Jitter | 5V | — | — | — | 0.1 | % |

## OCP Electrical Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $I_{DEM}$ | Demodulation Operating Current | 5V | DEMEN=1, DA $V_{REF}$=2.5V | — | 480 | 710 | μA |
| $I_{OCP}$ | OCP Operating Current | 5V | OCPEN=1, DA $V_{REF}$=2.5V | — | 480 | 710 | μA |
| **Comparator** | | | | | | | |
| $I_{COMP}$ | Comparator Operating Current | 5V | No load | — | 30 | 60 | μA |
| $V_{CMPOS1}$ | Input Offset Voltage | 5V | — | -15 | — | +15 | mV |
| $V_{CMPOS2}$ | Input Offset Voltage | 5V | By calibration | -8 | — | +8 | mV |
| $V_{HYS}$ | Hysteresis Width | 5V | — | 20 | 40 | 60 | mV |
| $V_{CM}$ | Common Mode Voltage Range | 5V | — | $V_{SS}$ | — | $V_{DD}$-1.4V | V |
| $t_{PD}$ | Comparator Response Time | 5V | With 100mV overdrive | — | 370 | 560 | ns |
| **OPA** | | | | | | | |
| $I_{OPA}$ | OPA Operating Current | 5V | No load | — | 200 | 350 | μA |
| $V_{OPOS1}$ | Input Offset Voltage | 5V | — | -15 | — | 15 | mV |
| $V_{OPOS2}$ | Input Offset Voltage | 5V | By calibration | -4 | — | +4 | mV |
| $V_{CM}$ | Common Mode Voltage range | 5V | — | $V_{SS}$ | — | $V_{DD}$-1.4V | V |
| PSRR | Power Supply Rejection Ratio | 5V | — | 60 | 80 | — | dB |
| CMRR | Common Mode Rejection Ratio | 5V | — | 60 | 80 | — | dB |
| SR | Slew Rate +, Slew Rate - | 5V | — | 1.8 | 2.5 | — | V/μs |
| GBW | Gain Band Width | 5V | — | 500 | — | — | KHz |
| ERRG | OPA Gain Error | 5V | Gain=1/5/10/15/20/30/40/50 | -5 | G | +5 | % |
| **DAC for OCPREF/DEMREF** | | | | | | | |
| $I_{DAC}$ | DAC Operating Current | 5V | $V_{REF}$=2.5V | — | 250 | 300 | μA |
| | | | $V_{REF}$=5V | — | 500 | 600 | μA |
| Ro | R2R Output Resistor | 5V | — | — | 10 | — | KΩ |
| DNL | DAC Differential NonLinearity | 5V | — | -0.5 | — | +0.5 | LSB |
| INL | DAC Integral NonLinearity | 5V | — | -1 | — | +1 | LSB |

## Reference Voltage Electrical Characteristics

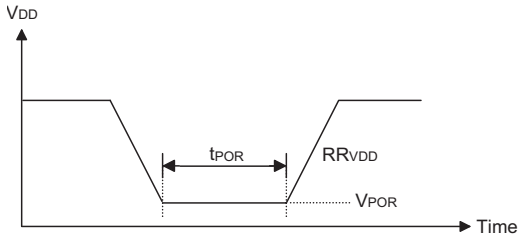
Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $I_{BG}$ | Additional Power Consumption if $V_{BG}$ Reference with Buffer is used | 5V | — | — | 200 | 300 | μA |
| $V_{BG}$ | Reference Voltage | 5V | Ta=25°C | -3% | 1.04 | +3% | V |
| $t_{START}$ | $V_{BG}$ and OPA Turn On Stable Time | — | — | — | — | 150 | μs |
| $V_{REF}$ | Reference Voltage | 5V | Ta=25°C | -3% | 2.08 | +3% | V |
| $I_{OPA}$ | OPA Operating Current | 5V | No load | — | 200 | 350 | μA |

## Power on Reset Electrical 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 |
| $RR_{VDD}$ | $V_{DD}$ Rising Rate to Ensure Power-on Reset | — | — | 0.035 | — | — | V/ms |
| $t_{POR}$ | Minimum Time for $V_{DD}$ Stays at $V_{POR}$ to Ensure Power-on Reset | — | — | 1 | — | — | ms |

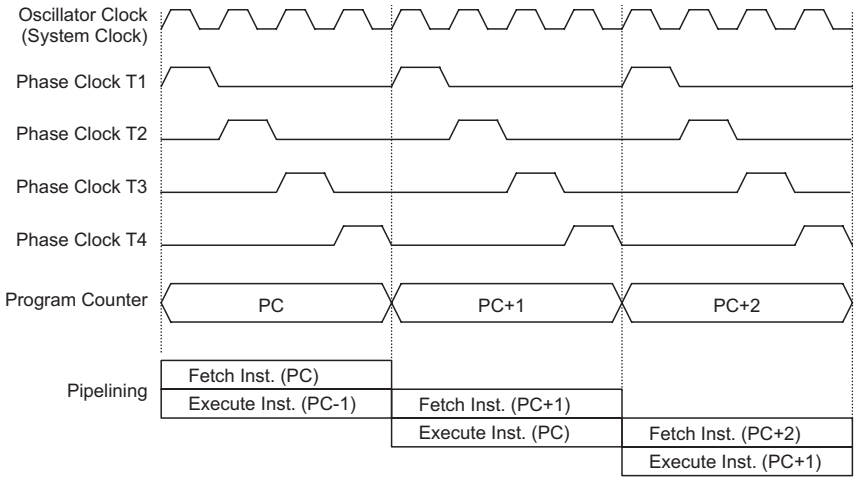


## System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The device takes advantage of the usual features found within RISC microcontrollers providing increased speed of operation and Periodic 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.

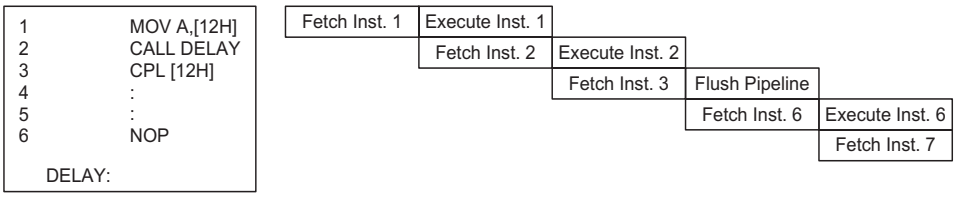
### Clocking and Pipelining

The main system clock, derived from either an HXT/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.




**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.

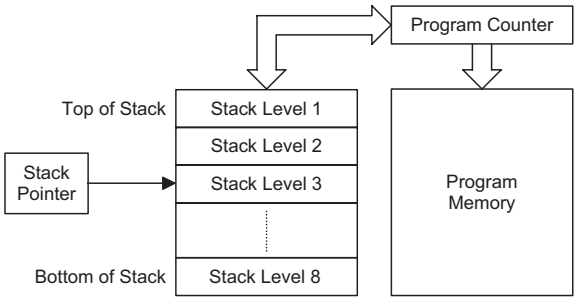
| Program Counter | |
|---|---|
| **Program Counter High byte** | **PCL Register** |
| PC11~PC8 | PCL7~PCL0 |

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

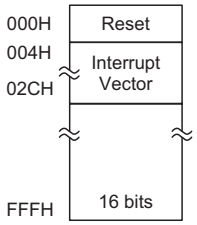
The Program Memory is the location where the user code or program is stored. For this device the Program Memory is Flash type, which means it can be programmed and re-programmed a large number of times, allowing the user the convenience of code modification on the same device. By using the appropriate 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 4K×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.



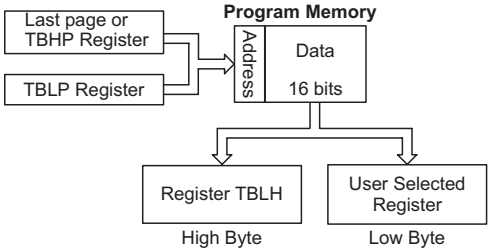

**Program Memory Structure**

## 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 | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | b11 | b10 | b9 | b8 | b7 | b6 | b5 | b4 | b3 | b2 | b1 | b0 |
| TABRD [m] | @11 | @10 | @9 | @8 | @7 | @6 | @5 | @4 | @3 | @2 | @1 | @0 |
| TABRDL [m] | 1 | 1 | 1 | 1 | @7 | @6 | @5 | @4 | @3 | @2 | @1 | @0 |

**Table Location**

Note: b11~b0: Table location bits

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

@11~@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 "F00H" which refers to the start address of the last page within the 4K 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 "F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address of the present page if the "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,0Fh          ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1     ; transfers value in table referenced by table pointer data at program
                   ; memory address "F06H" 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 "F05H" transferred to tempreg2 and TBLH in this
                   ; example the data "1AH" is transferred to tempreg1 and data "0FH" to
                   ; register tempreg2
:
:
org 0F00h          ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:
:
```

## In Circuit Programming – ICP

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.

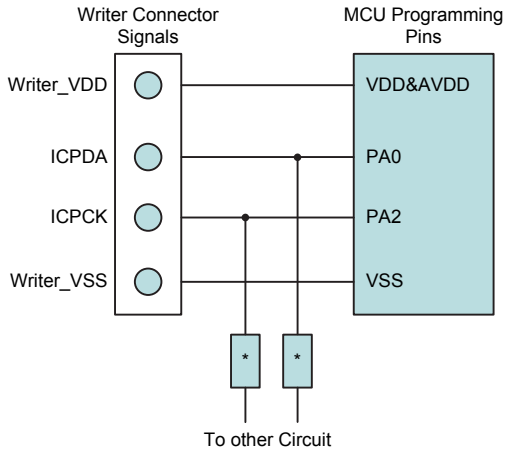
The Holtek Flash MCU to Writer Programming Pin correspondence table is as follows:

| Holtek Write Pins | MCU Programming Pins | Function |
|---|---|---|
| ICPDA | PA0 | Programming Serial Data/Address |
| ICPCK | PA2 | Programming Clock |
| VDD | VDD&AVDD | Power Supply |
| VSS | VSS | Ground |

During the programming process, the user must there take care to ensure that no other outputs are connected to these two pins.

The Program Memory can be programmed serially in-circuit using this 4-wire interface. Data is downloaded and uploaded serially/parallel on several pins 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 is beyond the scope of this document and will be supplied in supplementary literature.

During the programming process, the ICPDA and ICPCK 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 named HT66VW2230 which is used to emulate the HT66FW2230 device. The HT66VW2230 device also provides the "On-Chip Debug" function to debug the HT66FW2230 device during development process. The device, HT66FW2230, is almost functional compatible except the "On-Chip Debug" function and package types. Users can use the HT66VW2230 device to emulate the HT66FW2230 device behaviors 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 HT66VW2230 EV chip for debugging, the corresponding pin functions shared with the OCDSDA and OCDSCK pins in the HT66FW2230 device will have no effect in the HT66VW2230 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&AVDD | Power Supply |
| GND | VSS | Ground |

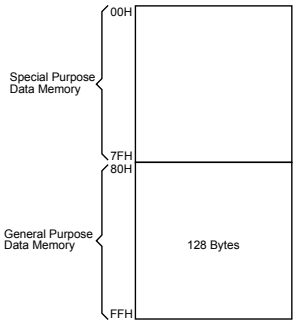
## RAM Data Memory

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.

The overall Data Memory is subdivided into two banks. The Special Purpose Data Memory registers are accessible in all banks, with the exception of the EEC register at address 40H, which is only accessible in Bank 1. Switching between the different Data Memory banks is achieved by setting the Bank Pointer to the correct value. The start address of the Data Memory for the device is the address 00H.




**Data Memory Structure**

| | Bank 0, 1 |
|---|---|
| 00H | IAR0 |
| 01H | MP0 |
| 02H | IAR1 |
| 03H | MP1L |
| 04H | MP1H |
| 05H | ACC |
| 06H | PCL |
| 07H | TBLP |
| 08H | TBLH |
| 09H | TBHP |
| 0AH | STATUS |
| 0BH | Unused |
| 0CH | IAR2 |
| 0DH | MP2L |
| 0EH | MP2H |
| 0FH | Unused |
| 10H | INTC0 |
| 11H | Unused |
| 12H | PA |
| 13H | PAC |
| 14H | PAPU |
| 15H | PAWU |
| 16H | Unused |
| 17H | RSTFC |
| 18H | Unused |
| 19H | Unused |
| 1AH | PB |
| 1BH | PBC |
| 1CH | PBPU |
| 1DH | Unused |
| 1EH | Unused |
| 1FH | Unused |
| 20H | TBC0 |
| 21H | TBC1 |
| 22H | PSCR |
| 23H | WDTC |
| 24H | LVDC |
| 25H | LVRC |
| 26H | RSTC |
| 27H | Unused |
| 28H | ADRL |
| 29H | ADRH |
| 2AH | ADCR0 |
| 2BH | ADCR1 |
| 2CH | SCC |
| 2DH | HXTC |
| 2EH | HIRCC |
| 2FH | Unused |
| 30H | INTEG |
| 31H | INTC1 |
| 32H | INTC2 |
| 33H | MFI0 |
| 34H | MFI1 |
| 35H | MFI2 |
| 36H | Unused |
| 37H | PC |
| 38H | PCC |
| 39H | PCPU |
| 3AH | PAS0 |
| 3BH | PAS1 |
| 3CH | PBS0 |
| 3DH | PCS0 |
| 3EH | IFS0 |
| 3FH | PCS1 |

| | Bank 0 | Bank 1 |
|---|---|---|
| 40H | Unused | EEC |
| 41H | EEA | |
| 42H | EED | |
| 43H | TM0C0 | |
| 44H | TM0C1 | |
| 45H | TM0DL | |
| 46H | TM0DH | |
| 47H | TM0AL | |
| 48H | TM0AH | |
| 49H | TM1C0 | |
| 4AH | TM1C1 | |
| 4BH | TM1DL | |
| 4CH | TM1DH | |
| 4DH | TM1AL | |
| 4EH | TM1AH | |
| 4FH | Unused | |
| 50H | IICC0 | |
| 51H | IICC1 | |
| 52H | IICD | |
| 53H | IICA | |
| 54H | I2CTOC | |
| 55H | Unused | |
| 56H | Unused | |
| 57H | Unused | |
| 58H | Unused | |
| 59H | Unused | |
| 5AH | Unused | |
| 5BH | Unused | |
| 5CH | Unused | |
| 5DH | Unused | |
| 5EH | Unused | |
| 5FH | Unused | |
| 60H | CKGEN | |
| 61H | PLLFL | |
| 62H | PLLFH | |
| 63H | PWMC | |
| 64H | DEMC0 | |
| 65H | DEMC1 | |
| 66H | DEMREF | |
| 67H | DEMACAL | |
| 68H | DEMCCAL | |
| 69H | OCPC0 | |
| 6AH | OCPC1 | |
| 6BH | OCPREF | |
| 6CH | OCPACAL | |
| 6DH | OCPCCAL | |
| 6EH | DCMISC | |
| 6FH | VREFC | |
| 70H | VRACAL | |
| 71H | Unused | |
| 72H | CPR | |
| 73H | Unused | |
| 74H | Unused | |
| 75H | Unused | |
| 76H | Unused | |
| 77H | Unused | |
| 78H | Unused | |
| 79H | Unused | |
| 7AH | Unused | |
| 7BH | Unused | |
| 7CH | Unused | |
| 7DH | Unused | |
| 7EH | Unused | |
| 7FH | Unused | |

**Special Purpose 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, IAR2

The Indirect Addressing Registers, IAR0, IAR1 and IAR2, 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, IAR1 and IAR2 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, MP1L/MP1H or MP2L/MP2H. Acting as a pair, IAR0 and MP0 can together access data only from Bank 0 while the IAR1 register together with MP1L/MP1H register pair and IAR2 register together with MP2L/MP2H register pair can access data from any Data Memory 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, MP1L, MP1H, MP2L, MP2H

Five Memory Pointers, known as MP0, MP1L, MP1H, MP2L and MP2H, are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Bank 0, while MP1L/MP1H together with IAR1 and MP2L/MP2H together with IAR2 are used to access data from all data banks according to the corresponding MP1H or MP2H register. Direct Addressing can be used in all data banks using the corresponding instruction which can address all available data memory space.

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 SC, CZ, 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 SC, CZ, 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.
- SC is the result of the “XOR” operation which is performed by the OV flag and the MSB of the current instruction operation result.
- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.

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 | SC | CZ | TO | PDF | OV | Z | AC | C |
| R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W |
| POR | × | × | 0 | 0 | × | × | × | × |

"×" unknown

Bit 7 **SC:** The result of the “XOR” operation which is performed by the OV flag and the MSB of the instruction operation result.

Bit 6 **CZ:** The operational result of different flags for different instructions.
For SUB/SUBM instructions, the CZ flag is equal to the Z flag.
For SBC/ SBCM instructions, the CZ flag is the “AND” operation result which is performed by the previous operation CZ flag and current operation zero flag Z. For other instructions, the CZ flag will not be affected.

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.

# EEPROM Data Memory

One of the special features in the device is its internal EEPROM Data Memory. EEPROM, which stands for Electrically Erasable Programmable Read Only Memory, is by its nature a non-volatile form of memory, with data retention even when its power supply is removed. By incorporating this kind of data memory, a whole new host of application possibilities are made available to the designer. The availability of EEPROM storage allows information such as product identification numbers, calibration values, specific user data, system setup data or other product information to be stored directly within the product microcontroller. The process of reading and writing data to the EEPROM memory has been reduced to a very trivial affair.

## EEPROM Data Memory Structure

The EEPROM Data Memory capacity is up to 64×8 bits. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped and is therefore not directly accessible in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address and data register in Bank 0 and a single control register in Bank 1.

## EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Bank 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register however, being located in Bank1, cannot be directly addressed directly and can only be read from or written to indirectly using the MP1L and MP1H Memory Pointers and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the MP1L/MP2L Memory Pointer must first be set to the value 40H and the Memory Point, MP1H/MP2H, set to the value, 01H, before any operations on the EEC register are executed.

### EEPROM Control Registers List

| Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EEA | — | — | D5 | D4 | D3 | D2 | D1 | D0 |
| EED | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| EEC | — | — | — | — | WREN | WR | RDEN | RD |

### EEA Register

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

Bit 7 ~ 6 Unimplemented, read as "0"

Bit 5 ~ 0 Data EEPROM address
Data EEPROM address bit 5 ~ bit 0

### EED 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 Data EEPROM data
Data EEPROM data bit 7 ~ bit 0

### EEC Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | WREN | WR | RDEN | RD |
| R/W | — | — | — | — | R/W | R/W | R/W | R/W |
| POR | — | — | — | — | 0 | 0 | 0 | 0 |

Bit 7 ~ 4 Unimplemented, read as "0"

Bit 3 **WREN:** Data EEPROM Write Enable
0: Disable
1: Enable
This is the Data EEPROM Write Enable Bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations.

Bit 2 **WR:** EEPROM Write Control
0: Write cycle has finished
1: Activate a write cycle
This is the Data EEPROM Write Control Bit and when set high by the application program will activate a write cycle. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Setting this bit high will have no effect if the WREN has not first been set high.

Bit 1 **RDEN:** Data EEPROM Read Enable
0: Disable
1: Enable
This is the Data EEPROM Read Enable Bit which must be set high before Data EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data EEPROM read operations.

Bit 0 **RD:** EEPROM Read Control
0: Read cycle has finished
1: Activate a read cycle
This is the Data EEPROM Read Control Bit and when set high by the application program will activate a read cycle. This bit will be automatically reset to zero by the hardware after the read cycle has finished. Setting this bit high will have no effect if the RDEN has not first been set high.

Note: The WREN, WR, RDEN and RD can not be set to “1” at the same time in one instruction.
The WR and RD can not be set to “1” at the same time.

### Reading Data from the EEPROM

To read data from the EEPROM, the read enable bit, RDEN, in the EEC register must first be set high to enable the read function. The EEPROM address of the data to be read must then be placed in the EEA register. If the RD bit in the EEC register is now set high, a read cycle will be initiated. Setting the RD bit high will not initiate a read operation if the RDEN bit has not been set. When the read cycle terminates, the RD bit will be automatically cleared to zero, after which the data can be read from the EED register. The data will remain in the EED register until another read or write operation is executed. The application program can poll the RD bit to determine when the data is valid for reading.

### Writing Data to the EEPROM

To write data to the EEPROM, the write enable bit, WREN, in the EEC register must first be set high to enable the write function. The EEPROM address of the data to be written must then be placed in the EEA register and the data placed in the EED register. If the WR bit in the EEC register is now set high, an internal write cycle will then be initiated. Setting the WR bit high will not initiate a write cycle if the WREN bit has not been set. As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended.

### Write Protection

Protection against inadvertent write operation is provided in several ways. After the device is powered-on the Write Enable bit in the control register will be cleared preventing any write operations. Also at power-on the MP1H/MP2H, will be reset to zero, which means that Data Memory Bank 0 will be selected. As the EEPROM control register is located in Bank 1, this adds a further measure of protection against spurious write operations. During normal program operation, ensuring that the Write Enable bit in the control register is cleared will safeguard against incorrect write operations.

### EEPROM Interrupt

The EEPROM write interrupt is generated when an EEPROM write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. However as the EEPROM is contained within a Multi-function Interrupt, the associated multi-function interrupt enable bit must also be set. When an EEPROM write cycle ends, the DEF request flag and its associated multi-function interrupt request flag will both be set. If the global, EEPROM and Multi-function interrupts are enabled and the stack is not full, a jump to the associated Multi-function Interrupt vector will take place. When the interrupt is serviced only the Multi-function interrupt flag will be automatically reset, the EEPROM interrupt flag must be manually reset by the application program. More details can be obtained in the Interrupt section.

## Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be Periodic by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Memory Pointer could be normally cleared to zero as this would inhibit access to Bank 1 where the EEPROM control register exist. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process. When writing data the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. The global interrupt bit EMI should also be cleared before a write cycle is executed and then re-enabled after the write cycle starts. Note that the device should not enter the IDLE or SLEEP mode until the EEPROM read or write operation is totally complete. Otherwise, the EEPROM read or write operation will fail.

### Programming Examples

- **Reading data from the EEPROM – polling method**

```
MOV A, EEPROM_ADRES          ; user defined address
MOV EEA, A
MOV A, 040H                  ; setup memory pointer MP1
MOV MP1L, A                  ; MP1L points to EEC register
MOV A, 01H                   ; setup Memory Pointer
MOV MP1H, A
SET IAR1.1                   ; set RDEN bit, enable read operations
SET IAR1.0                   ; start Read Cycle - set RD bit
BACK:
SZ IAR1.0                    ; check for read cycle end
JMP BACK
CLR IAR1                     ; disable EEPROM read/write
CLR MP1H
MOV A, EED                   ; move read data to register
MOV READ_DATA, A
```

- **Writing Data to the EEPROM – polling method**

```
MOV A, EEPROM_ADRES          ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA           ; user defined data
MOV EED, A
MOV A, 040H                  ; setup memory pointer MP1
MOV MP1L, A                  ; MP1L points to EEC register
MOV A, 01H                   ; setup Memory Pointer
MOV MP1H, A
CLR EMI
SET IAR1.3                   ; set WREN bit, enable write operations
SET IAR1.2                   ; start Write Cycle - set WR bit
SET EMI
BACK:
SZ IAR1.2                    ; check for write cycle end
JMP BACK
CLR IAR1                     ; disable EEPROM write
CLR MP1H
```

# Oscillator

Various oscillator configurations 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 the 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.

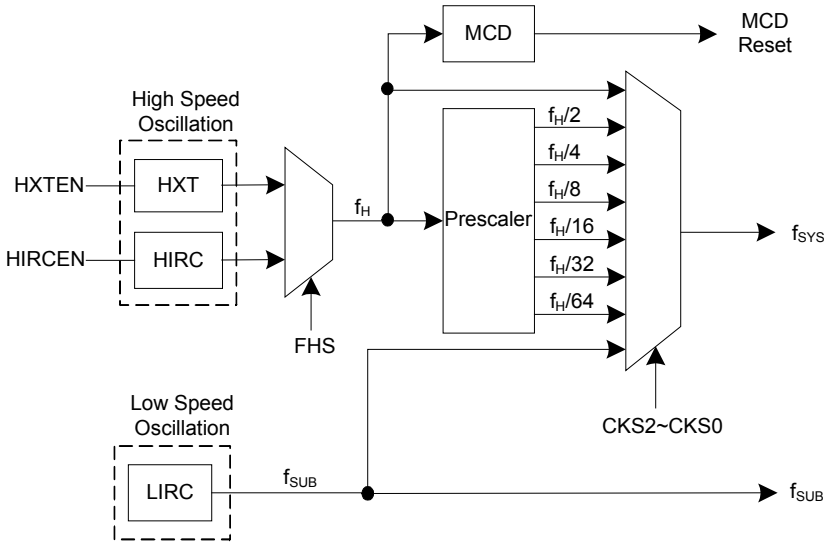
| Type | Name | Freq. |
|---|---|---|
| External Crystal | HXT | 20MHz |
| Internal High Speed RC | HIRC | 20MHz |
| Internal Low Speed RC | LIRC | 32kHz |

**Oscillator Types**

## System Clock Configurations

There are three methods of generating the system clock, two high speed oscillators and a low speed oscillator. The high speed oscillators are the external 20MHz crystal and internal 20MHz RC oscillators. 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 FHS bit and CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.

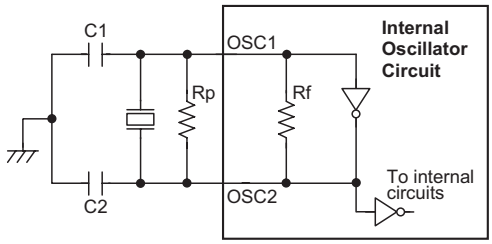
The actual source clock used for the high speed and the low speed oscillators is chosen via register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.



## External Crystal/ Ceramic Oscillator – HXT

The External Crystal/Ceramic System Oscillator is one of the high frequency oscillator choices, which is selected via registers. For most crystal oscillator configurations, the simple connection of a crystal across OSC1 and OSC2 will create the necessary phase shift and feedback for oscillation, without requiring external capacitors. However, for some crystal types and frequencies, to ensure oscillation, it may be necessary to add two small value capacitors, C1 and C2. Using a ceramic resonator will usually require two small value capacitors, C1 and C2, to be connected as shown for oscillation to occur. The values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification.

For oscillator stability and to minimise the effects of noise and crosstalk, it is important to ensure thatthe crystal and any associated resistors andcapacitors along with interconnectinglines are all located as close to the MCUas possible.




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

**Crystal/Resonator Oscillator – HXT**

## 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 20MHz. 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 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.

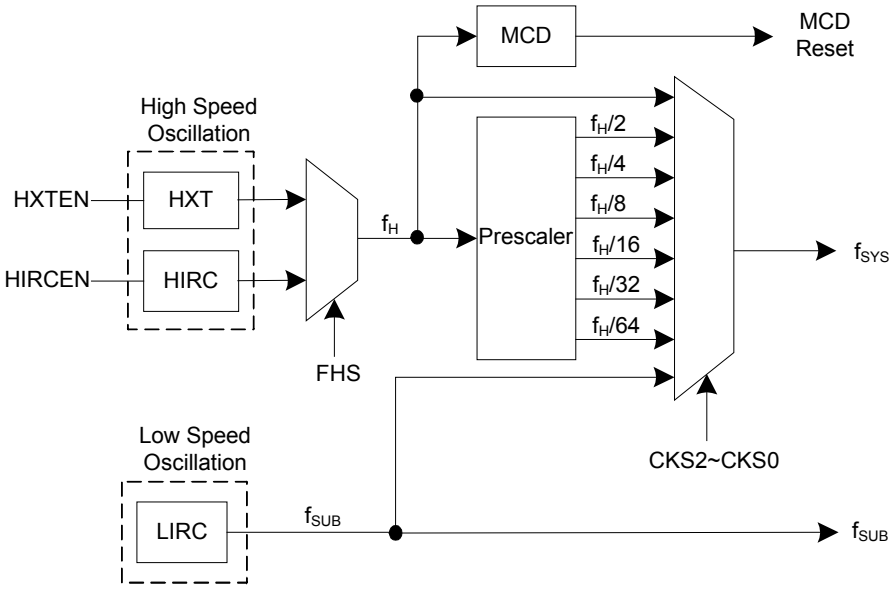
## Operating Modes and System Clocks

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 selections using 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 CKS2~CKS0 bits in the SCC register. The high speed system clock can be sourced from either an HXT or HIRC oscillator by configuring the FHS bit. 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 or continue to oscillate to provide the clock source, $f_H$~$f_H/64$, for peripheral circuit to use, which is determined by configuring the corresponding high speed oscillator enable control bit.

## 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 four modes, the SLEEP, IDLE0, IDLE1 and IDLE2 Mode are used when the microcontroller CPU is switched off to conserve power.

| Operation Mode | CPU | Register Setting | | | $f_{SYS}$ | $f_H$ | $f_{SUB}$ |
|---|---|---|---|---|---|---|---|
| | | FHIDEN | FSIDEN | CKS2~CKS0 | | | |
| NORMAL | On | x | x | 000~110 | $f_H$~$f_H$/64 | On | On |
| SLOW | On | x | x | 111 | $f_{SUB}$ | Off | On |
| IDLE0 | Off | 0 | 1 | 000~110 | Off | Off | On |
| | | | | 111 | On | | |
| IDLE1 | Off | 1 | 1 | x | On | On | On |
| IDLE2 | Off | 1 | 0 | 000~110 | On | On | Off |
| | | | | 111 | Off | | |
| SLEEP | Off | 0 | 0 | x | Off | Off | On |

Note : 1. In the IDLE mode, the system clock is turned on when bit CKS[2:0]=000~110 and FHIDEN=1.

2. In the IDLE mode, the system clock is turned on when bit CKS[2:0]=111 and FSIDEN=1.

3. In sleep mode, the LIRC is turned on since the WDT is enabled.

### 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 one of the high speed oscillators. This mode operates allowing the microcontroller to operate normally with a clock source will come from one of the high speed oscillators, either the HXT or HIRC oscillators. 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 $f_{SUB}$. The $f_{SUB}$ is derived from LIRC.

### SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the FHIDEN and FSIDEN bit are low. In the SLEEP mode the CPU will be stopped. However the $f_{SUB}$ clock still can continue to operate because the WDT function is always enabled.

### IDLE0 Mode

The IDLE0 Mode is entered when a HALT instruction is executed and when the FSIDEN bit in the SCC register is high and the FHIDEN bit in the SCC register is low. In the IDLE0 Mode the CPU will be switched off but the low speed oscillator will be turned on to drive some peripheral functions.

### IDLE1 Mode

The IDLE1 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is high. In the IDLE1 Mode the CPU will be switched off but both the high and low speed oscillators will be turned on to provide a clock source to keep some peripheral functions operational.

### IDLE2 Mode

The IDLE2 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is low. In the IDLE2 Mode the CPU will be switched off but the high speed oscillator will be turned on to provide a clock source to keep some peripheral functions operational.

### MCD (Missing Clock Detector) function

There is a Missing Clock Detector, MCD, in this device. The MCD is used to detect the high speed oscillator operation when the corresponding oscillator is enabled. If the oscillator is enabled and no clock cycle is detected by the MCD in certain period of time, it means that the oscillator does not oscillate successfully and then the MCD will generate a signal to reset the microcontroller.

## Control Register

The registers, SCC, HXTC and HIRCC, are used to control the system clock within the device.

### SCC Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CKS2 | CKS1 | CKS0 | — | FHS | — | FHIDEN | FSIDEN |
| R/W | R/W | R/W | R/W | — | R/W | — | R/W | R/W |
| POR | 0 | 1 | 0 | — | 0 | — | 0 | 0 |

Bit 7~5 **CKS2~CKS0:** The system clock selection
000: $f_H$
001: $f_H/2$
010: $f_H/4$
011: $f_H/8$
100: $f_H/16$
101: $f_H/32$
110: $f_H/64$
111: $f_{SUB}$
These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from $f_H$ or $f_{SUB}$, 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 **FHS:** High frequency clock selection
0: HIRC
1: HXT

Bit 2 Unimplemented, read as "0"

Bit 1 **FHIDEN:** High frequency oscillator control when CPU is switched off
0: disable
1: enable
This bit is used to control whether the high speed oscillator is activated or stopped when the CPU is switched off by executing a HALT instruction.

Bit 0 **FSIDEN:** Low frequency oscillator control when CPU is switched off
0: disable
1: enable
This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing a HALT instruction. When the LIRC oscillator is selected to be the low speed oscillator, the LIRC oscillator will also be controlled by this bit together with the WDT function enable control bit. When this bit is cleared to 0, but the WDT function is enabled, the LIRC oscillator will be enabled.

### HXTC Register

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

Bit 7~4 Unimplemented, read as "0"

Bit 1 **HXTF:** HXT clock stable flag
0: unstable
1: stable
This bit is used to indicate whether the HXT oscillator is stable or not. When the HXTEN bit is set to 1 to enable the HXT oscillator, the HXTF bit will be first cleared to "0" and then set to "1" after the HXT oscillator is stable.

Bit 0 **HXTEN:** HXT oscillator enable control
0: disable
1: enable

### HIRCC Register

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

Bit 7~2 Unimplemented, read as "0"

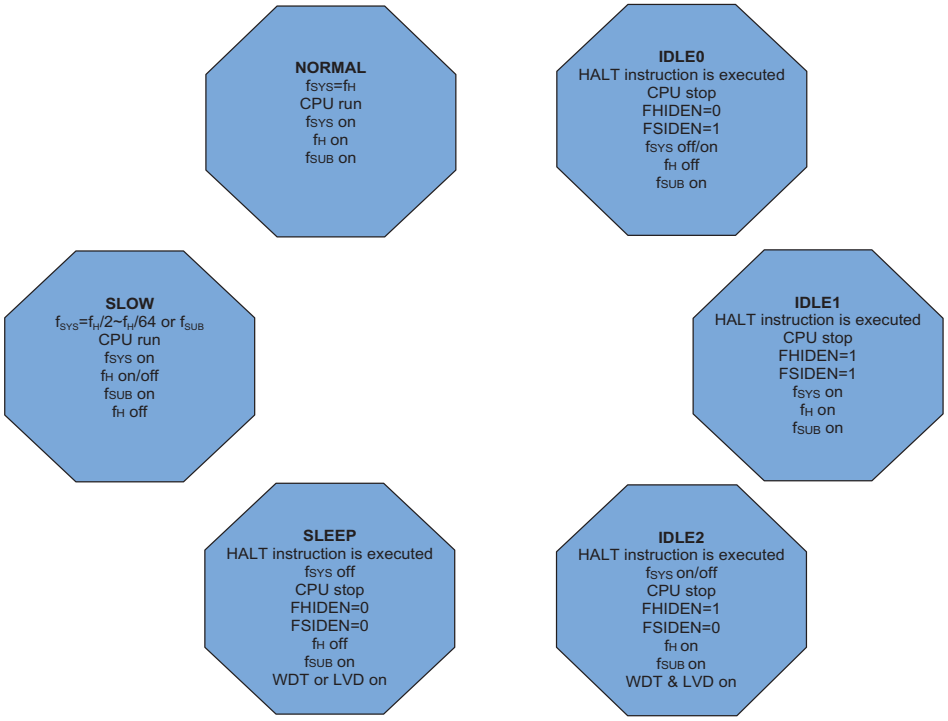
Bit 1 **HIRCF:** HIRC clock stable flag
0: unstable
1: stable
When the HIRC oscillator is enabled and the HIRC frequency is changed by application program, this bit will first be cleared to "0" and then set to "1" after the HIRC oscillator is stable.

Bit 0 **HIRCEN:** HIRC oscillator enable control
0: disable
1: enable

## 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 CKS2~CKS0 bits in the SCC 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 FHIDEN and FSIDEN bit in the SCC register.




## NORMAL Mode to SLOW Mode Switching

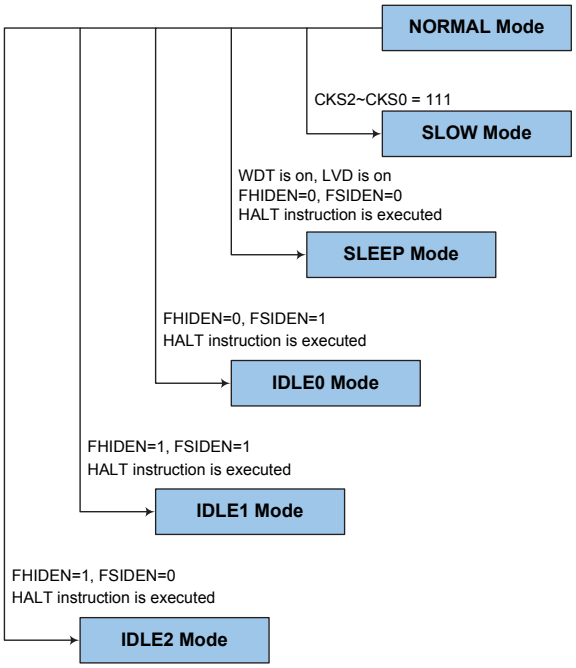
When running in the NORMAL Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by set the CKS2~CKS0 bits to "111" in the SCC register. This will then use the low speed system oscillator which will consume less power. 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.

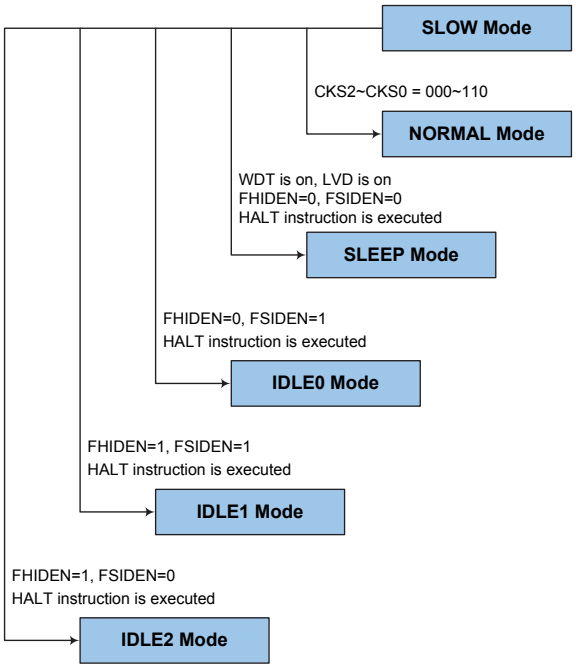
## SLOW Mode to NORMAL Mode Switching

In SLOW mode the system clock is from $f_{SUB}$. When system clock is switched back to the NORMAL mode from $f_{SUB}$, the CKS2~CKS0 bits should be set to "000" ~"110" and then the system clock will respectively be switched to $f_H$~ $f_H/64$.

However, if $f_H$ is not used in SLOW mode, it will take some time to re-oscillate and stabilise. This is monitored using the HXTF/HIRCF bit in the HXTC/HIRCC register. The amount of time required for high speed system oscillator stabilization depends upon which high speed system oscillator type is used.




**NORMAL Mode to SLOW Mode Switching**




**SLOW Mode to NORMAL Mode Switching**

### Entering the SLEEP Mode

There is only one way for the device to enter the SLEEP Mode and that is to execute the “HALT” instruction in the application program with both the FHIDEN and FLIDEN bits in the SCC 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.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting.

### 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 FSIDEN bit in the SCC register equal to “1” and the FHIDEN bit in the SCC register equal to “0”. When this instruction is executed under the conditions described above, the following will occur:

- The $f_H$ clock will be stopped and the application program will stop at the “HALT” instruction, but the $f_{SUB}$ clock will be on.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting.

### 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 both the FHIDEN and FLIDEN bit in the SCC register equal to “1”. When this instruction is executed under the conditions described above, the following will occur:

- The $f_H$ and $f_{SUB}$ clock will be on but the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting.

### Entering the IDLE2 Mode

There is only one way for the device to enter the IDLE2 Mode and that is to execute the "HALT" instruction in the application program with the FHIDEN bit in the SCC register equal to "1" and the FLIDEN bit in SCC register equal to "0". When this instruction is executed under the conditions described above, the following will occur:

- The $f_H$ clock will be on but the $f_{SUB}$ clock will be off and the application program will stop at the "HALT" instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports of peripheral device will keep to work when it is enabled and clock is from $f_{SYS}$.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting.

### 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 and IDLE2 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. Also note that additional standby current will also be required if the LIRC oscillator has enabled.

In the IDLE1 and IDLE 2 Mode the high speed oscillator is on, if the system oscillator is from the high speed oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

## Wake-up

To minimise power consumption the device can enter the SLEEP or any IDLE Mode, where the CPU will be switched off. However when the device is woken up again, it can take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume.

| System Oscillator | Wake-up Time | | | |
|---|---|---|---|---|
| | SLEEP Mode | IDLE 0 Mode | IDLE 1 Mode | IDLE 2 Mode |
| HXT | 1024 HXT cycles | 1024 HXT cycles | 1~2 HXT cycles | 1~2 HXT cycles |
| HIRC | 15~16 HIRC cycles | 15~16 HIRC cycles | 1~2 HIRC cycles | 1~2 HIRC cycles |
| LIRC | 1~2 LIRC cycles | 1~2 LIRC cycles | 1~2 LIRC cycles | 1~2 LIRC cycles |

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

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

## Programming Considerations

The HXT, HIRC and LIRC oscillators use different SST counter. For example, if the system is woken up from the SLEEP Mode and the HXT oscillator needs to start-up from an off state.

- If the device is woken up from the SLEEP Mode to the NORMAL Mode, the high speed system oscillator needs an SST period. The device will execute first instruction after HIRCF/HXTF is "1". The same situation occurs in the power-on state.
- There are peripheral functions, such as TMs, for which the $f_{SYS}$ is used. If the system clock source is switched from $f_H$ to $f_{SUB}$, the clock source to the peripheral functions mentioned above will change accordingly.

# 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 sourced from LIRC oscillators. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with $V_{DD}$, temperature and process variations. 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.

## Watchdog Timer Control Register

A single register, WDTC, controls the required timeout period as well as the enable operation. The WDTC register is initiated to 01010011B at any reset (any reset includes POR reset, LVR reset, LVR software reset, WDT time-out in CPU operating and WDT software 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 or 01010: Enabled
Other values: Reset MCU (reset will be active after 2~3 LIRC clock for debounce time)
If the MCU reset and it's caused by WE[4:0] in WDTC software reset, the WRF flag in RSTFC register will be set after reset.

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: $2^{14}/f_{SUB}$ (default)
100: $2^{15}/f_{SUB}$
101: $2^{16}/f_{SUB}$
110: $2^{17}/f_{SUB}$
111: $2^{18}/f_{SUB}$
These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

### RSTFC Register

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

Bit 7~ 4 Unimplemented, read as "0"

Bit 3 **ERSTF:** Reset caused by RST[7:0] setting
Describe elsewhere

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

Bit 1 Unimplemented, read as "0"

Bit 0 **WRF:** reset caused by WE[4:0] setting
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 instruction. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, this clear instruction will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. With regard to the Watchdog Timer enable function, there are five bits, WE4~WE0, in the WDTC register to offer the enable control and reset control of the Watchdog Timer. The WDT function will be enabled if the WE4~WE0 bits are equal to 10101 or 01010B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after 2~3 LIRC clock cycles. After power on these bits will have a value of 01010B.

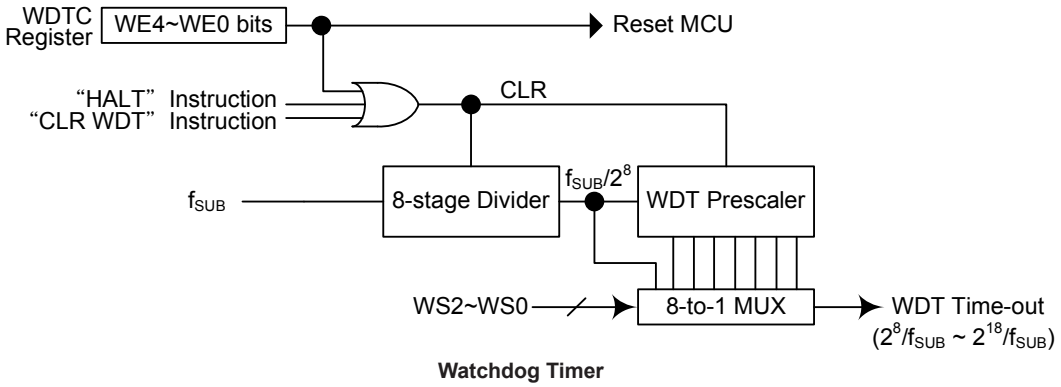
| WE4 ~ WE0 Bits | WDT Function |
|---|---|
| 01010B or 10101B | Enable |
| Any other value | Reset MCU |

**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 software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 field. The second is using the Watchdog Timer software clear instruction. 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 contents.

The maximum time out period is when the $2^{18}$ division ratio is selected. As an example, with a 32 KHz LIRC oscillator as its source clock, this will give a maximum watchdog period of around 8 second 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. There is a number of other hardware and software reset sources that can be implemented dynamically when the device is running.

### Reset Overview

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.

The device provides several reset sources to generate the internal reset signal, providing extended MCU protection. The different types of resets are listed in the accompanying table.

| Reset Name | Abbreviation | Indication Bit | Register | Notes |
|---|---|---|---|---|
| Power-On Reset | POR | — | — | Auto generated at power on |
| RSTC Register Setting Software Reset | — | ERSTF | RSTFC | Write to RSTC register |
| Low-Voltage Reset | LVR | LRF | RSTFC | Low $V_{DD}$ voltage |
| Watchdog Reset | WDT | TO | STATUS | WDT Time-out |
| WDTC Register Setting Software Reset | — | WRF | RSTFC | Write to WDTC register |
| MCD Reset | — | — | — | Missing Clock Detector |

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 which is implemented in situations where the power supply voltage falls below a certain threshold.

## Reset Functions

There are several 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**

### RSTC Register Software Reset

If the device reset caused by RST[7:0] setting, the ERSTF bit in the RSTFC register will be set to 1.

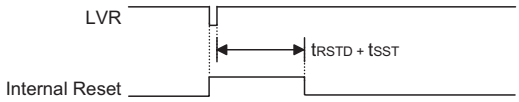
- **RSTC External Reset Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | RST7 | RST6 | RST5 | RST4 | RST3 | RST2 | RST1 | RST0 |
| R/W | R/W | R/W | R/W | R/W | R | R | R/W | R/W |
| POR | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

Bit 7 ~ 0 **RST7 ~ RST0:** GPIO or RESB configuration selection for PB0
01010101and 10101010: configured as GPIO
Other Values: MCU reset
Note: The device is fixed at PB0 before it leaves the factory.

### 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 RSTFC register will also be set to1. For a valid LVR signal, a low voltage, i.e., a voltage in the range between 0.9V~$V_{LVR}$ must exist for greater than the value $t_{LVR}$ specified in the A.C. characteristics. If the low voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. 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=50ms

**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 2 ~ 0 **LVS[7:**0]: LVR voltage select
01010101B: 2.1V (default)
00110011B: 2.55V
10011001B: 3.15V
10101010B: 3.8V
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 set 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.

• **RSTFC Register**

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

Bit 7~ 4 Unimplemented, read as “0”

Bit 3 **ERSTF:** Reset caused by RST[7:0] setting
0: not active
1: active
This bit can be clear to “0”, but can not set to “1”.

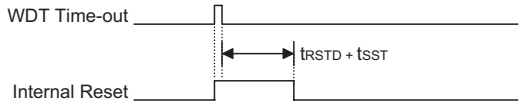
Bit 2 **LVRF:** LVR function reset flag
0: not active
1: active
This bit can be clear to “0”, but can not set to “1”.

Bit 1 Unimplemented, read as “0”

Bit 0 **WRF:** reset caused by WE[4:0] setting
Describe elsewhere

### Watchdog Time-out Reset during Normal Operation

When the Watchdog time-out Reset during normal operation, the Watchdog time-out flag TO will be set to “1”.

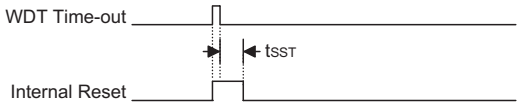



Note: $t_{RSTD}$ is power-on delay, typical time=16.7ms

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

### Watchdog Time-out Reset during SLEEP or IDLE Mode

The Watchdog time-out Reset during SLEEP 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 15~16 clock cycles if the system clock source is provided by HIRC.

The $t_{SST}$ is 128 clock cycles if the system clock source is provided by the HXT.

The $t_{SST}$ is 1~2 clock for the LIRC.

**WDT Time-out Reset during SLEEP or IDLE Timing Chart**

### WDTC Register Software Reset

A WDTC software reset will be generated when a value other than "10101" or "01010", exist in the highest five bits of the WDTC register. The WRF bit in the RSTFC register will be set high when this occurs, thus indicating the generation of a WDTC software reset.

- **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 or 01010: Enabled
Other values: Reset MCU (reset will be active after 2~3 LIRC clock for debounce time)
If the MCU reset and it's caused by WE[4:0] in WDTC software reset, the WRF flag in RSTFC register will be set after reset.

Bit 2~ 0 **WS2 ~ WS0:** WDT Time-out period selection
Described elsewhere

## 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 SLEEP Mode operation |

Note: "u" stands for unchanged

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

| Item | Condition After RESET |
|---|---|
| Program Counter | Reset to zero |
| Interrupts | All interrupts will be disabled |
| WDT | Clear after reset, WDT begins counting |
| Timer Modules | Timer Modules will be turned off |
| Input/Output Ports | I/O ports will be setup as inputs and AN0~AN7 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 | Reset (Power On) | WDT Time-out (Normal Operation) | WDT Time-out (HALT)* |
|---|---|---|---|
| MP0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP1L | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP1H | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP2L | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP2H | 0000 0000 | 0000 0000 | uuuu uuuu |
| ACC | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PCL | 0000 0000 | 0000 0000 | 0000 0000 |
| TBLP | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TBLH | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TBHP | ---- xxxx | ---- uuuu | ---- uuuu |
| STATUS | xx00 xxxx | xx1u uuuu | uu11 uuuu |
| RSTFC | ---- 0x-0 | ---- uu-u | ---- uu-u |
| RSTC | 0101 0101 | 0101 0101 | uuuu uuuu |
| WDTC | 0101 0011 | 0101 0011 | uuuu uuuu |
| PSCR | ---- --00 | ---- --00 | ---- --uu |
| TBC0 | 0--- -000 | 0--- -000 | u--- -uuu |
| TBC1 | 0--- -000 | 0--- -000 | u--- -uuu |
| LVRC | 0101 0101 | 0101 0101 | uuuu uuuu |
| LVDC | --00 -000 | --00 -000 | --uu -uuu |
| INTEG | ---- 0000 | ---- 0000 | ---- uuuu |
| INTC0 | -000 0000 | -000 0000 | -uuu uuuu |

| Register | Reset (Power On) | WDT Time-out (Normal Operation) | WDT Time-out (HALT)* |
|---|---|---|---|
| INTC1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| INTC2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MFI0 | --00 --00 | --00 --00 | --uu --uu |
| MFI1 | --00 --00 | --00 --00 | --uu --uu |
| MFI2 | --00 --00 | --00 --00 | --uu --uu |
| ADRL (ADRFS=0) | xxxx ---- | xxxx ---- | uuuu ---- |
| ADRL (ADRFS=1) | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| ADRH (ADRFS=0) | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| ADRH (ADRFS=1) | ---- xxxx | ---- xxxx | ---- uuuu |
| ADCR0 | 0110 0000 | 0110 0000 | uuuu uuuu |
| ADCR1 | -000 0000 | -000 0000 | -uuu uuuu |
| SCC | 010- 0-00 | 010- 0-00 | uuu- u-uu |
| HXTC | ---- --00 | ---- --00 | ---- --uu |
| HIRCC | ---- --01 | ---- --01 | ---- --uu |
| EEA | --00 0000 | --00 0000 | --uu uuuu |
| EED | 0000 0000 | 0000 0000 | uuuu uuuu |
| EEC | ---- 0000 | ---- 0000 | ---- uuuu |
| PA | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAC | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAPU | 0000 0000 | 0000 0000 | uuuu uuuu |
| PAWU | 0000 0000 | 0000 0000 | uuuu uuuu |
| PB | ---1 1111 | ---1 1111 | ---u uuuu |
| PBC | ---1 1111 | ---1 1111 | ---u uuuu |
| PBPU | ---0 0000 | ---0 0000 | ---u uuuu |
| PC | 1111 1111 | 1111 1111 | 1111 uuuu |
| PCC | 1111 1111 | 1111 1111 | 1111 uuuu |
| PCPU | 0000 0000 | 0000 0000 | 0000 uuuu |
| PAS0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PAS1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PBS0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PCS0 | ---- 0000 | ---- 0000 | ---- uuuu |
| IFS0 | ---- ---0 | ---- ---0 | ---- ---u |
| TM0C0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM0C1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM0DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM0DH | ---- --00 | ---- --00 | ---- --uu |
| TM0AL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM0AH | ---- --00 | ---- --00 | ---- --uu |
| TM1C0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM1C1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM1DL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM1DH | ---- --00 | ---- --00 | ---- --uu |
| TM1AL | 0000 0000 | 0000 0000 | uuuu uuuu |
| TM1AH | ---- --00 | ---- --00 | ---- --uu |

| Register | Reset (Power On) | WDT Time-out (Normal Operation) | WDT Time-out (HALT)* |
|---|---|---|---|
| IICC0 | ---- 000- | ---- 000- | ---- uuu- |
| IICC1 | 1000 0001 | 1000 0001 | uuuu uuuu |
| IICD | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| IICA | 0000 000- | 0000 000- | uuuu uuu- |
| I2CTOC | 0000 0000 | 0000 0000 | uuuu uuuu |
| CKGEN | 0000 ---- | 0000 ---- | uuuu ---- |
| PLLFL | 0000 0000 | 0000 0000 | uuuu uuuu |
| PLLFH | ---- -000 | ---- -000 | ---- -uuu |
| PWMC | 0101 0000 | 0101 0000 | uuuu uuuu |
| DEMC0 | 00-- 0000 | 00-- 0000 | 00-- uuuu |
| DEMC1 | x-00 0000 | x-00 0000 | u-uu uuuu |
| DEMREF | 0000 0000 | 0000 0000 | uuuu uuuu |
| DEMACAL | 0010 0000 | 0010 0000 | uuuu uuuu |
| DEMCCAL | 0001 0000 | 0001 0000 | uuuu uuuu |
| OCPC0 | 00-- ---- | 00-- ---- | 00-- ---- |
| OCPC1 | x-00 0000 | x-00 0000 | u-uu uuuu |
| OCPREF | 0000 0000 | 0000 0000 | uuuu uuuu |
| OCPACAL | 0010 0000 | 0010 0000 | uuuu uuuu |
| OCPCCAL | 0001 0000 | 0001 0000 | uuuu uuuu |
| DCMISC | 000- --00 | 000- --00 | uuu- --uu |
| VREFC | 0--- ---x | 0--- ---x | u--- ---u |
| VRACAL | 0010 0000 | 0010 0000 | uuuu uuuu |
| CPR | 0--0 0000 | 0--0 0000 | u--u uuuu |

Note: "-" not implement

"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, PB and PC. 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 |
| PC | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PCC | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PCPU | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

### 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~PCPU, 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 bit7~ 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

**PCPU 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 C bit 7~ 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~PCC, 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

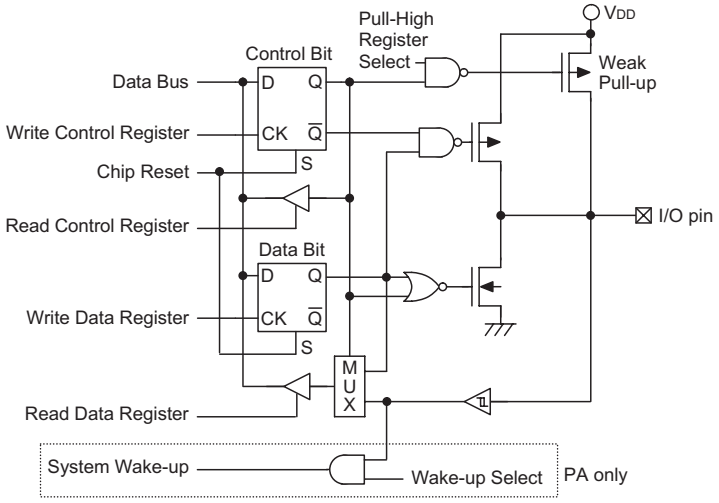
### PCC Register

| 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 |

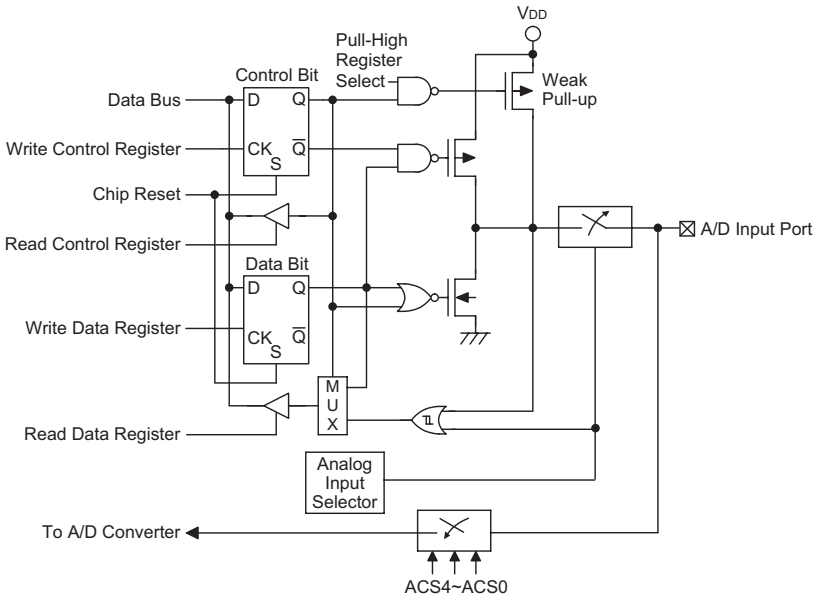
Bit 7 ~ 0 I/O Port C bit 7 ~ bit 0 Input/Output Control
0: Output
1: Input

## 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**

## Pin-sharing Functions

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

### PAS0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PAS07 | PAS06 | PAS05 | PAS04 | PAS03 | PAS02 | PAS01 | PAS00 |
| 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 **PAS07~PAS06:** PA3 Pin Share setting
00: PA3
01: PA3
10: PA3
11: AN3/VREF ( determined by VREFS[1:0] )

Bit 5~4 **PAS05~PAS04:** PA2 Pin Share setting
00: PA2
01: SCL
10: PA2
11: AN2

Bit 3~2 **PAS03~PAS02:** PA1 Pin Share setting
00: PA1
01: PA1
10: CN
11: PA1

Bit 1~0 **PAS01~PAS00:** PA0 Pin Share setting
00: PA0
01: SDA
10: PA0
11: AN0

### PAS1 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PAS17 | PAS16 | PAS15 | PAS14 | PAS13 | PAS12 | PAS11 | PAS10 |
| 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 **PAS17~PAS16:** PA7 Pin Share setting
00: PA7
01: PA7
10: PA7
11: AN7

Bit 5~4 **PAS15~PAS14:** PA6 Pin Share setting
00: PA6
01: PA6
10: CP/AN6
11: AN6

Bit 3~2 **PAS13~PAS12:** PA5 Pin Share setting
00: PA5
01: PA5
10: PA5
11: AN5

Bit 1~0 **PAS11~PAS10:** PA4 Pin Share setting
00: PA4
01: PA4
10: PA4
11: AN4

### PBS0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PBS07 | PBS06 | PBS05 | PBS04 | PBS03 | PBS02 | PBS01 | PBS00 |
| 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 **PBS07~PBS06:** PB4 Pin Share setting
00: PB4
01: TP0
10: DEMO
11: PB4

Bit 5~4 **PBS05~PBS04:** PB3 Pin Share setting
00: PB3
01: TP1_1
10: TP1_1B
11: SCL

Bit 3~2 **PBS03~PBS02:** PB2 Pin Share setting
00: PB2
01: TP1_0
10: TP1_0B
11: SDA

Bit 1~0 **PBS01~PBS00:** PB1 Pin Share setting
00: PB1
01: PB1
10: CX
11: PB1

**PCS0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PCS07 | PCS06 | PCS05 | PCS04 | PCS03 | PCS02 | PCS01 | PCS00 |
| 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 **PCS07~PCS06:** PC3 Pin Share setting
00: PC3
01: PWM3 (=PWM13)
10: PWM3B (=PWM13B)
11: PC3

Bit 5~4 **PCS05~PCS04:** PC2 Pin Share setting
00: PC2
01: PWM2 (=PWM12)
10: PWM2B (=PWM12B)
11: PC2

Bit 3~2 **PCS03~PCS02:** PC1 Pin Share setting
00: PC1
01: PWM1 (=PWM11)
10: PWM1B (=PWM11B)
11: PC1

Bit 1~0 **PCS01~PCS00:** PC0 Pin Share setting
00: PC0
01: PWM0 (=PWM10)
10: PWM0B (=PWM10B)
11: PC0

**PCS1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PCS17 | PCS16 | PCS15 | PCS14 | PCS13 | PCS12 | PCS11 | PCS10 |
| 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 **PCS17~PCS16:** PC7 Pin Share setting
00: PC7
01: OSC2
10: OSC2
11: OSC2 ( Application note: it's suggested to set as 0x11 for HXT )

Bit 5~4 **PCS15~PCS14:** PC6 Pin Share setting
00: PC6
01: OSC1
10: OSC1
11: OSC1 ( Application note: it's suggested to set as 0x11 for HXT )

Bit 3~2 **PCS13~PCS12:** PC5 Pin Share setting
00: PC5
01: COMM2
10: AX ( OPA output )
11: PC5

Bit 1~0 **PCS11~PCS10:** PC4 Pin Share setting
00: PC4
01: COMM1
10: AN ( OPA- )
11: PC4

**IFS0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | IFS02 | IFS01 | IFS00 |
| R/W | — | — | — | — | — | R/W | R/W | R/W |
| POR | — | — | — | — | — | 0 | 0 | 0 |

Bit 7~3 Unimplemented, read as 0

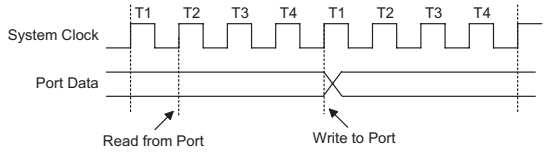
Bit 2 **IFS02:** $I^2C$ SDA input source selection
0: PA0
1: PB2

Bit 1 **IFS01:** $I^2C$ SCL input source selection
0: PA2
1: PB3

Bit 0 **IFS00:** TP0 input source selection
0: TP0
1: DEMO

## 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~PCC, 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~PC, are first programmed. Selecting which pins are inputs and which are outputs can be achieved byte-wide by loading the correct values into the appropriate port control register or by programming individual bits in the port control register using the "SET [m].i" and "CLR [m].i" instructions. Note that when using these bit control instructions, a read-modify-write operation takes place. The microcontroller must first read in the data on the entire port, modify it to the required new bit values and then rewrite this data back to the output ports.




**Read/Write Timing**

Port A has the additional capability of providing wake-up functions. When the device is in the 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 Standard and Compact TM section.

## Introduction

The device contains a 10-bit Standard TM and a 10-bit Compact TM, each TM having a reference name of TM0 and TM1. Although similar in nature, the different TM types vary in their feature complexity. The common features to the Standard and Compact TMs will be described in this section and the detailed operation will be described in corresponding sections. The main features and differences between the two types of TMs are summarised in the accompanying table.

| Function | STM | CTM |
|---|---|---|
| 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 |
|---|---|
| 10-bit STM | 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 either the system clock $f_{SYS}$ or the internal high clock $f_H$, the $f_{SUB}$ clock source or the external TCKn pin. 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 two different types of TMs have two internal interrupts, 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.

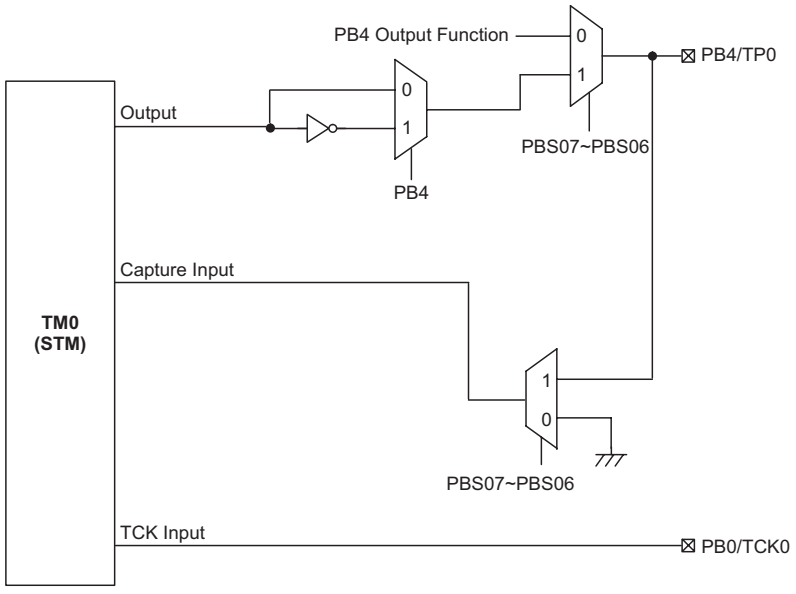
CTM output pin names have a "_n" suffix. Pin names that include a "_0" or "_1" suffix indicate that they are from a TM with multiple output pins. This allows the TM to generate a complimentary output pair, selected using the I/O register data bits.

| TM0 | TM1 |
|---|---|
| TP0 | TP1_0, TP1_1 |

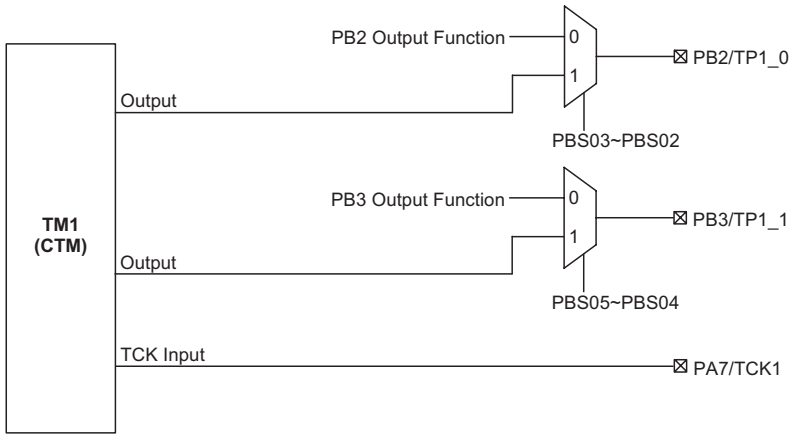
**TM Output Pins**

## TM Input/Output Pin Control Register

Selecting to have a TM input/output or whether to retain its other shared function is implemented using one register, with a single bit in each register corresponding to a TM input/output pin.




**TM0 Function Pin Control Block Diagram**



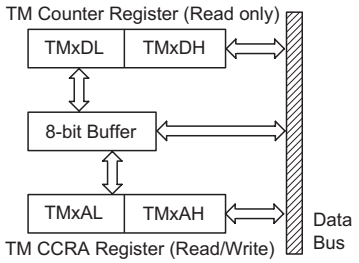

**TM1 Function Pin Control Block Diagram**

Note: 1. The I/O register data bits shown are used for TM output inversion control.
2. In the Capture Input Mode, the TM pin control register must never enable more than one TM input.

## Programming Considerations

The TM Counter Registers, the Capture/Compare CCRA register, being both 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 register is 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 low byte registers, named TMxAL, using the following access procedures. Accessing the CCRA low byte register without following these access procedures will result in unpredictable values.

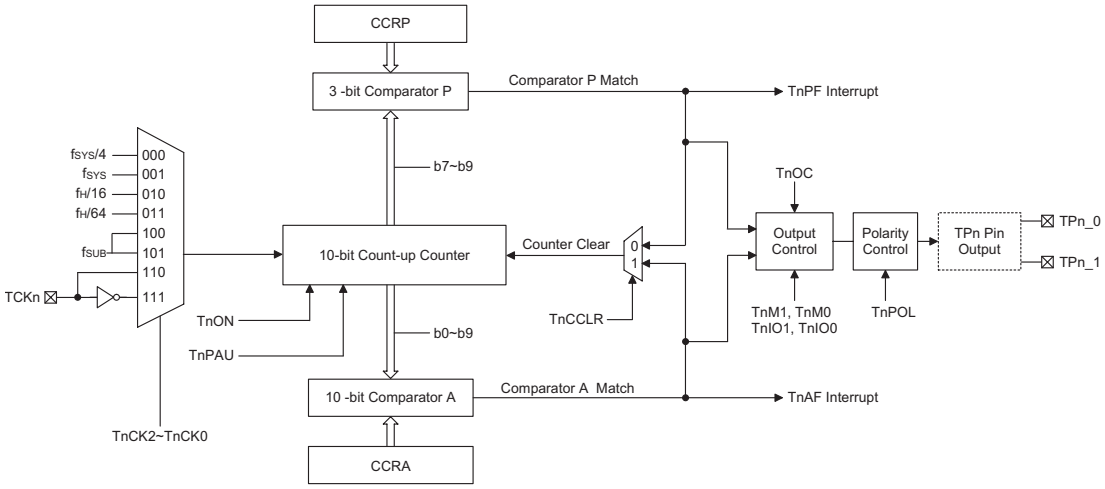



The following steps show the read and write procedures:

- Writing Data to CCRA
  - Step 1. Write data to Low Byte TMxAL
    - note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte TMxAH
    - 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
  - Step 1. Read data from the High Byte TMxDH, TMxAH
    - 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 or TMxAL
    - this step reads data from the 8-bit buffer.

## Compact Type TM – CTM

Although the simplest form of the two 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 two external output pins. These two external output pins can be the same signal or the inverse signal.




**Compact Type TM Block Diagram (n=1)**

### 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 T1ON 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 Register Description

Overall operation of the Compact TM is controlled using a series of 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 or eight CCRP bits.

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TM1C0 | T1PAU | T1CK2 | T1CK1 | T1CK0 | T1ON | T1RP2 | T1RP1 | T1RP0 |
| TM1C1 | T1M1 | T1M0 | T1IO1 | T1IO0 | T1OC | T1POL | T1DPX | T1CCLR |
| TM1DL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TM1DH | — | — | — | — | — | — | D9 | D8 |
| TM1AL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TM1AH | — | — | — | — | — | — | D9 | D8 |

**10-bit Compact TM Register List**

### TM1DL 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 **TM1DL:** TM1 Counter Low Byte Register bit 7~bit 0
TM1 10-bit Counter bit 7~bit 0

### TM1DH 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 **TM1DH:** TM1 Counter High Byte Register bit 1~bit 0
TM1 10-bit Counter bit 9~bit 8

### TM1AL 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 **TM1AL:** TM1 CCRA Low Byte Register bit 7~bit 0
TM1 10-bit CCRA bit 7~bit 0

### TM1AH 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 **TM1AH:** TM1 CCRA High Byte Register bit 1~bit 0
TM1 10-bit CCRA bit 9~bit 8

**TM1C0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | T1PAU | T1CK2 | T1CK1 | T1CK0 | T1ON | T1RP2 | T1RP1 | T1RP0 |
| 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 **T1PAU:** TM1 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 **T1CK2~T1CK0:** Select TM1 Counter clock
000: $f_{SYS}/4$
001: $f_{SYS}$
010: $f_H/16$
011: $f_H/64$
100: $f_{SUB}$
101: $f_{SUB}$
110: TCK1 rising edge clock
111: TCK1 falling edge clock

These three bits are used to select the clock source for the TM1. 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 $f_H$ is the HXT oscillator and $f_{SUB}$ is the internal clock, the details of which can be found in the oscillator section.

Bit 3 **T1ON:** TM1 Counter On/Off Control
0: Off
1: On

This bit controls the overall on/off function of the TM1. Setting the bit high enables the counter to run, clearing the bit disables the TM1. Clearing this bit to zero will stop the counter from counting and turn off the TM1 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 TM1 is in the Compare Match Output Mode then the TM1 output pin will be reset to its initial condition, as specified by the T1OC bit, when the T1ON bit changes from low to high.

Bit 2~0 **T1RP2~T1RP0:** TM1 CCRP 3-bit register, compared with the TM1 Counter bit 9~bit 7 Comparator P Match Period
000: 1024 TM1 clocks
001: 128 TM1 clocks
010: 256 TM1 clocks
011: 384 TM1 clocks
100: 512 TM1 clocks
101: 640 TM1 clocks
110: 768 TM1 clocks
111: 896 TM1 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 T1CCLR bit is set to zero. Setting the T1CCLR 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.

**TM1C1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | T1M1 | T1M0 | T1IO1 | T1IO0 | T1OC | T1POL | T1DPX | T1CCLR |
| 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 **T1M1~T1M0:** Select TM1 Operating Mode
00: Compare Match Output Mode
01: Undefined
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 T1M1 and T1M0 bits. In the Timer/Counter Mode, the TM output pin control must be disabled.

Bit 5~4 **T1IO1~T1IO0:** Select TP1_0, TP1_1 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 TM1 output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the TM1 is running.

In the Compare Match Output Mode, the T1IO1 and T1IO0 bits determine how the TM1 output pin changes state when a compare match occurs from the Comparator A. The TM1 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 TM1 output pin should be setup using the T1OC bit in the TM1C1 register. Note that the output level requested by the T1IO1 and T1IO0 bits must be different from the initial value setup using the T1OC bit otherwise no change will occur on the TM1 output pin when a compare match occurs. After the TM1 output pin changes state it can be reset to its initial level by changing the level of the T1ON bit from low to high.

In the PWM Mode, the T1IO1 and T1IO0 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 only change the values of the T1IO1 and T1IO0 bits only after the TM1 has been switched off. Unpredictable PWM outputs will occur if the T1IO1 and T1IO0 bits are changed when the TM is running.

Bit 3 **T1OC:** TP1_0, TP1_1 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 TM1 output pin. Its operation depends upon whether TM1 is being used in the Compare Match Output Mode or in the PWM Mode. It has no effect if the TM1 is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of he TM1 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 **T1POL:** TP1_0, TP1_1 Output polarity Control
0: Non-invert
1: Invert

This bit controls the polarity of the TP1_0 or TP1_1 output pin. When the bit is set high the TM1 output pin will be inverted and not inverted when the bit is zero. It has no effect if the TM1 is in the Timer/Counter Mode.

Bit 1 **T1DPX:** TM1 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 **T1CCLR:** Select TM1 Counter clear condition
0: TM1 Comparator P match
1: TM1 Comparator A match

This bit is used to select the method which clears the counter. Remember that the Compact TM1 contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the T1CCLR 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 T1CCLR bit is not used in the PWM Mode.

## Compact Type TM Operating Modes

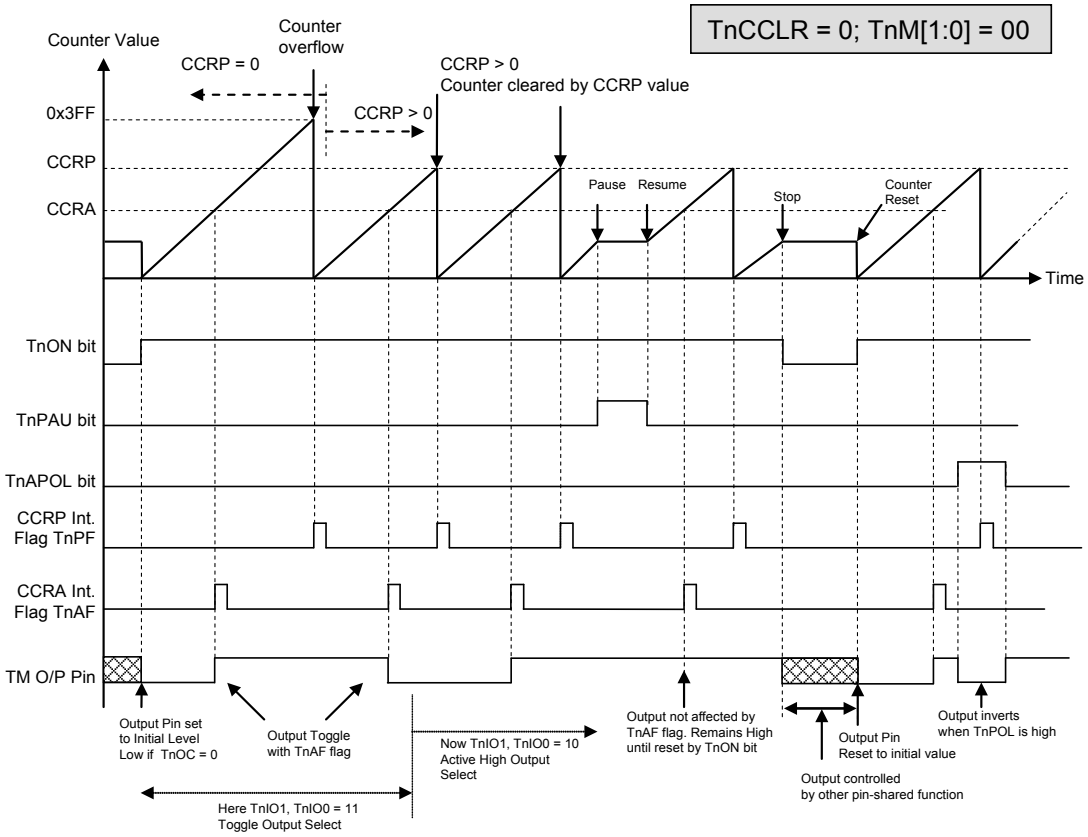
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Mode or Timer/Counter Mode. The operating mode is selected using the T1M1 and T1M0 bits in the TM1C1 register.

## Compare Match Output Mode

To select this mode, bits T1M1 and T1M0 in the TM1C1 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 T1CCLR 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 T1AF and T1PF interrupt request flags for the Comparator A and Comparator P respectively, will both be generated.

If the T1CCLR bit in the TM1C1 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 T1CCLR 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 T1AF 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 T1AF interrupt request flag is generated after a compare match occurs from Comparator A. The T1PF 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 T1IO1 and T1IO0 bits in the TM1C1 register. The TM output pin can be selected using the T1IO1 and T1IO0 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 T1ON bit changes from low to high, is setup using the T1OC bit. Note that if the T1IO1 and T1IO0 bits are zero then no pin change will take place.



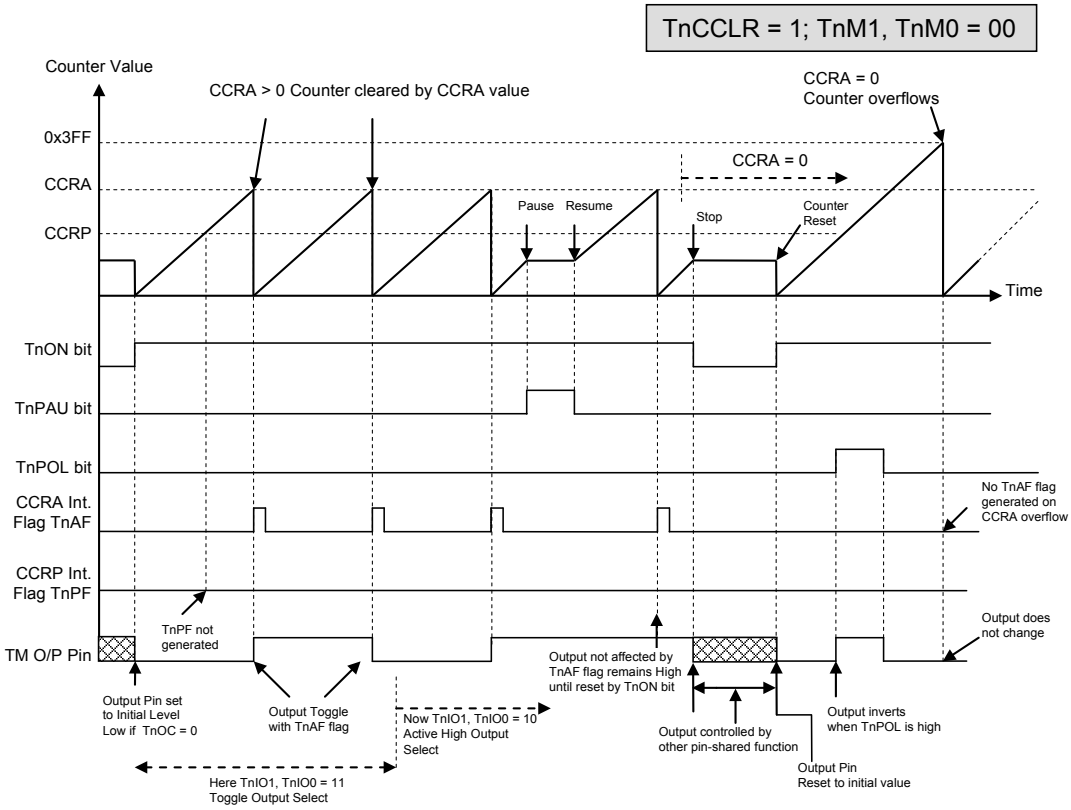

**Compare Match Output Mode – TnCCLR=0**

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

4. n= 1




**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. The TnPF flag is not generated when TnCCLR=1

5. n= 1

### Timer/Counter Mode

To select this mode, bits T1M1 and T1M0 in the TM1C1 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 T1M1 and T1M0 in the TM1C1 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 T1CCLR 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 T1DPX bit in the TM1C1 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 T1OC bit in the TM1C1 register is used to select the required polarity of the PWM waveform while the two T1IO1 and T1IO0 bits are used to enable the PWM output or to force the TM output pin to a fixed high or low level. The T1POL bit is used to reverse the polarity of the PWM output waveform.

#### 10-bit CTM, PWM Mode, Edge-aligned Mode, T1DPX=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,

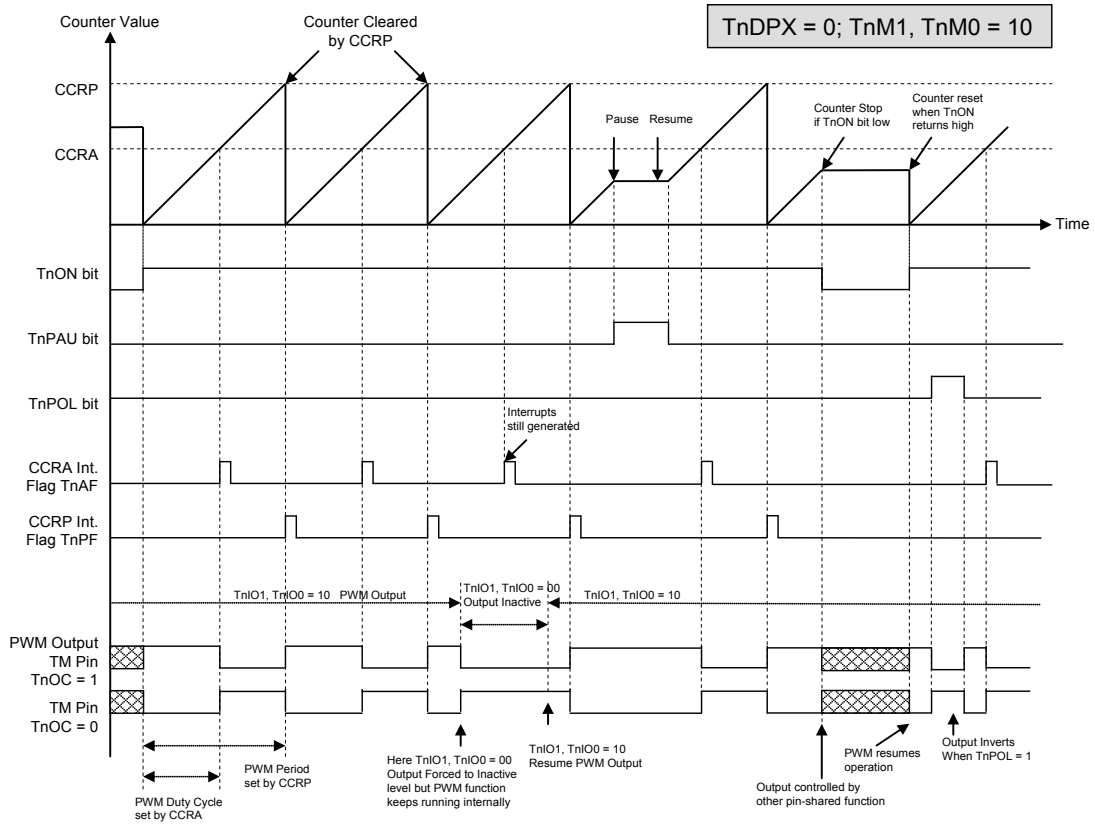
The CTM PWM output frequency=($f_{SYS}$/4)/512=$f_{SYS}$/2048=7.8125 kHz, duty=128/512=25%.

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 CTM, PWM Mode, Edge-aligned Mode, T1DPX=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**

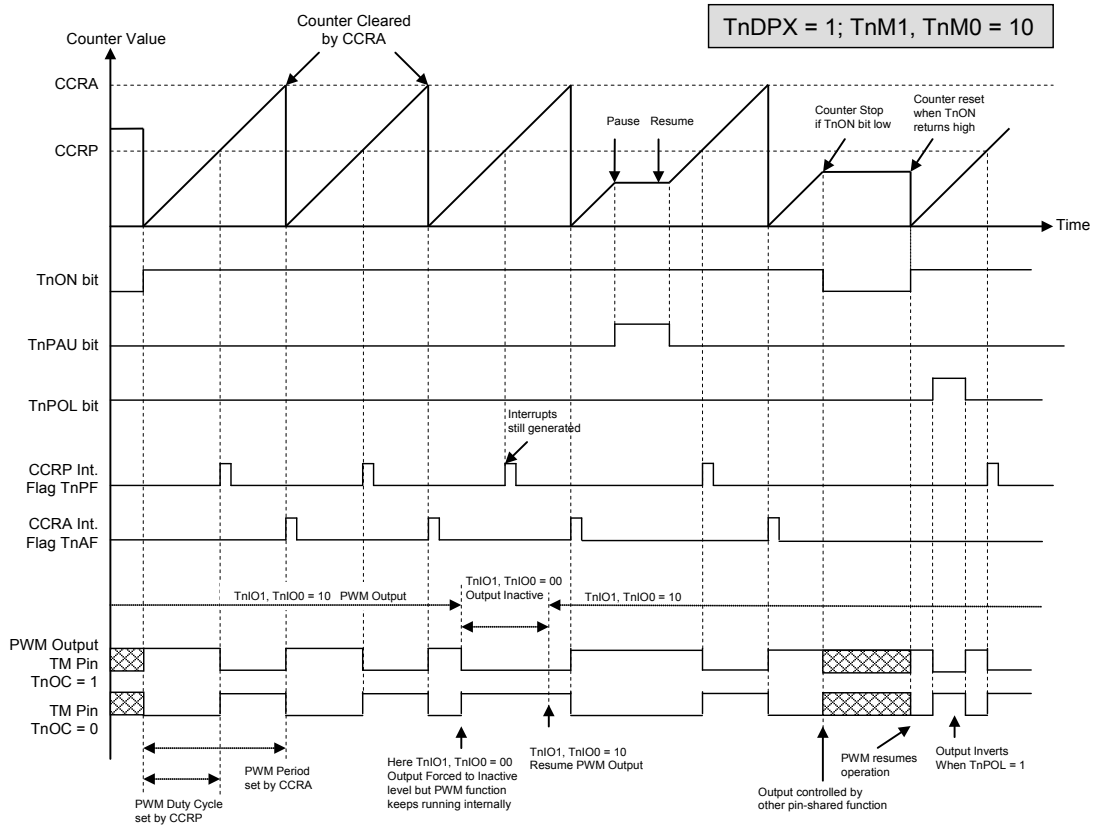
Note: 1. Here TnDPX=0 – Counter cleared by CCRP

2. A counter clear sets the 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

5. n=1




**PWM Mode – TnDPX=1**

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

2. A counter clear sets the PWM Period

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

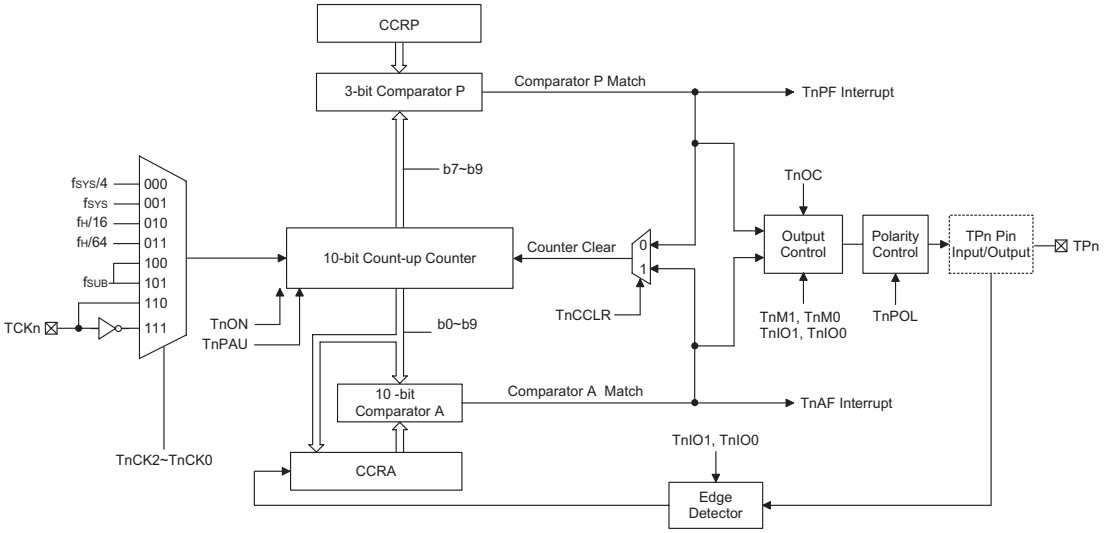
4. The TnCCLR bit has no influence on PWM operation

5. n=1

## Standard Type TM – STM

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

| Name | TM No. | TM Input Pin | TM Output Pin |
|---|---|---|---|
| 10-bit STM | 0 | TCK0 | TP0 |




**Standard Type TM Block Diagram (n=0)**

### Standard 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 3-bits wide whose value is compared with the highest 3 bits in the counter while the CCRA is the 10 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 T2ON 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 Standard 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.

## Standard Type TM Register Description

Overall operation of the Standard TM is controlled using a series of 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 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TM0C0 | T0PAU | T0CK2 | T0CK1 | T0CK0 | T0ON | T0RP2 | T0PR1 | T0PR0 |
| TM0C1 | T0M1 | T0M0 | T0IO1 | T0IO0 | T0OC | T0POL | T0DPX | T0CCLR |
| TM0DL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TM0DH | — | — | — | — | — | — | D9 | D8 |
| TM0AL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| TM0AH | — | — | — | — | — | — | D9 | D8 |

**10-bit Standard TM Register List**

### TM0C0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | T0PAU | T0CK2 | T0CK1 | T0CK0 | T0ON | T0RP2 | T0PR1 | T0PR0 |
| 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 **T0PAU:** TM0 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 **T0CK2~T0CK0:** Select TM0 Counter clock
000: $f_{SYS}/4$
001: $f_{SYS}$
010: $f_H/16$
011: $f_H/64$
100: $f_{SUB}$
101: $f_{SUB}$
110: TCK0 rising edge clock
111: TCK0 falling edge clock

These three bits are used to select the clock source for the TM0. 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 $f_H$ is the HX T oscillator and $f_{SUB}$ is internal clock, the details of which can be found in the oscillator section.

Bit 3 **T0ON:** TM0 Counter On/Off Control
0: Off
1: On

This bit controls the overall on/off function of the TM0. Setting the bit high enables the counter to run, clearing the bit disables the TM0. Clearing this bit to zero will stop the counter from counting and turn off the TM0 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 until the bit returns high again. If the TM0 is in the Compare Match Output Mode then the TM0 output pin will be reset to its initial condition, as specified by the T0OC bit, when the T0ON bit changes from low to high.

Bit 2~0 **T0RP2~T0RP0:** TM0 CCRP 3-bit register, compared with the TM0 Counter bit 9~bit 7 Comparator P Match Period

000: 1024 TM0 clocks
001: 128 TM0 clocks
010: 256 TM0 clocks
011: 384 TM0 clocks
100: 512 TM0 clocks
101: 640 TM0 clocks
110: 768 TM0 clocks
111: 896 TM0 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 T0CCLR bit is set to zero. Setting the T0CCLR 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

**TM0C1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | T0M1 | T0M0 | T0IO1 | T0IO0 | T0OC | T0POL | T0DPX | T0CCLR |
| 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 **T0M1~T0M0:** Select TM0 Operating 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 T0M1 and T0M0 bits. In the Timer/Counter Mode, the TM output pin control must be disabled.

Bit 5~4 **T0IO1~T0IO0:** Select TP0 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 TP0
01: Input capture at falling edge of TP0
10: Input capture at falling/rising edge of TP0
11: Input capture disabled

Timer/counter Mode

unused

These two bits are used to determine how the TM0 output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the TM0 is running.

In the Compare Match Output Mode, the T0IO1 and T0IO0 bits determine how the

TM0 output pin changes state when a compare match occurs from the Comparator A. The TM0 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 TM0 output pin should be setup using the T0OC bit in the TM0C1 register. Note that the output level requested by the T0IO1 and T0IO0 bits must be different from the initial value setup using the T0OC bit otherwise no change will occur on the TM0 output pin when a compare match occurs. After the TM0 output pin changes state it can be reset to its initial level by changing the level of the T0ON bit from low to high.

In the PWM Mode, the T0IO1 and T0IO0 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 only change the values of the T0IO1 and T0IO0 bits only after the TM0 has been switched off. Unpredictable PWM outputs will occur if the T0IO1 and T0IO0 bits are changed when the TM is running.

Bit 3 **T0OC:** TP0 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 TM0 output pin. Its operation depends upon whether TM0 is being used in the Compare Match Output Mode or in the PWM Mode/Single Pulse Output Mode. It has no effect if the TM0 is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of he TM0 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 **T0POL:** TP0 Output polarity Control
0: Non-invert
1: Invert

This bit controls the polarity of the TP0 output pin. When the bit is set high the TM0 output pin will be inverted and not inverted when the bit is zero. It has no effect if the TM0 is in the Timer/Counter Mode.

Bit 1 **T0DPX:** TM0 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 **T0CCLR:** Select TM0 Counter clear condition
0: TM0 Comparator P match
1: TM0 Comparator A match

This bit is used to select the method which clears the counter. Remember that the Compact TM0 contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the T0CCLR 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 T0CCLR bit is not used in the PWM Mode, Single Pulse or input Capture Mode.

**TM0DL 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 **TM0DL:** TM0 Counter Low Byte Register bit 7~bit 0
TM0 10-bit Counter bit 7~bit 0

**TM0DH Register – 10-bit STM**

| 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 **TM0DH:** TM0 Counter High Byte Register bit 1~bit 0
TM0 10-bit Counter bit 9~bit 8

**TM0AL 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 **TM0AL:** TM0 CCRA Low Byte Register bit 7~bit 0
TM0 10-bit CCRA bit 7~bit 0

**TM0AH 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 **TM0AH:** TM0 CCRA High Byte Register bit 1~bit 0
TM0 10-bit CCRA bit 9~bit 8

## Standard Type TM Operating Modes

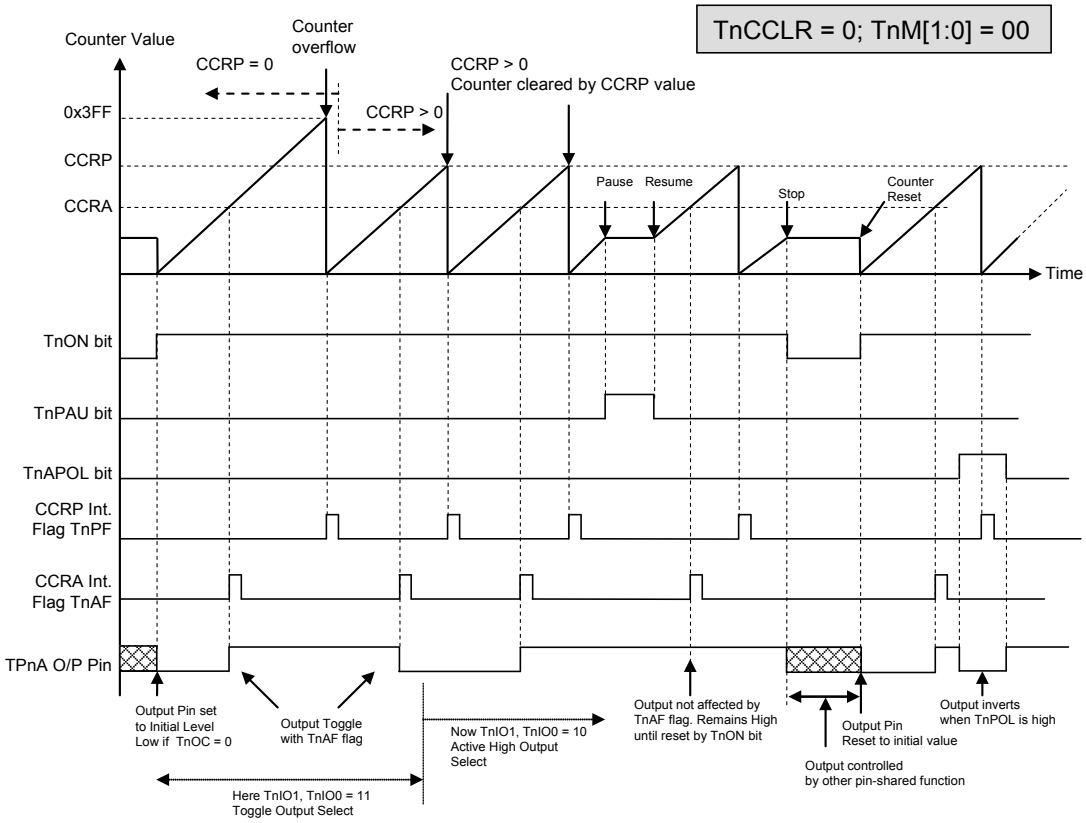
The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the T0M1 and T0M0 bits in the TM0C1 register.

## Compare Output Mode

To select this mode, bits T0M1 and T0M0 in the TM0C1 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 T0CCLR 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 T0AF and T0PF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the T0CCLR bit in the TM0C1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the T0AF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when T0CCLR is high no T0PF 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 T0AF interrupt request flag is generated after a compare match occurs from Comparator A. The T0PF 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 T0IO1 and T0IO0 bits in the TM0C1 register. The TM output pin can be selected using the T0IO1 and T0IO0 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 T0ON bit changes from low to high, is setup using the T0OC bit. Note that if the T0IO1 and T0IO0 bits are zero then no pin change will take place.



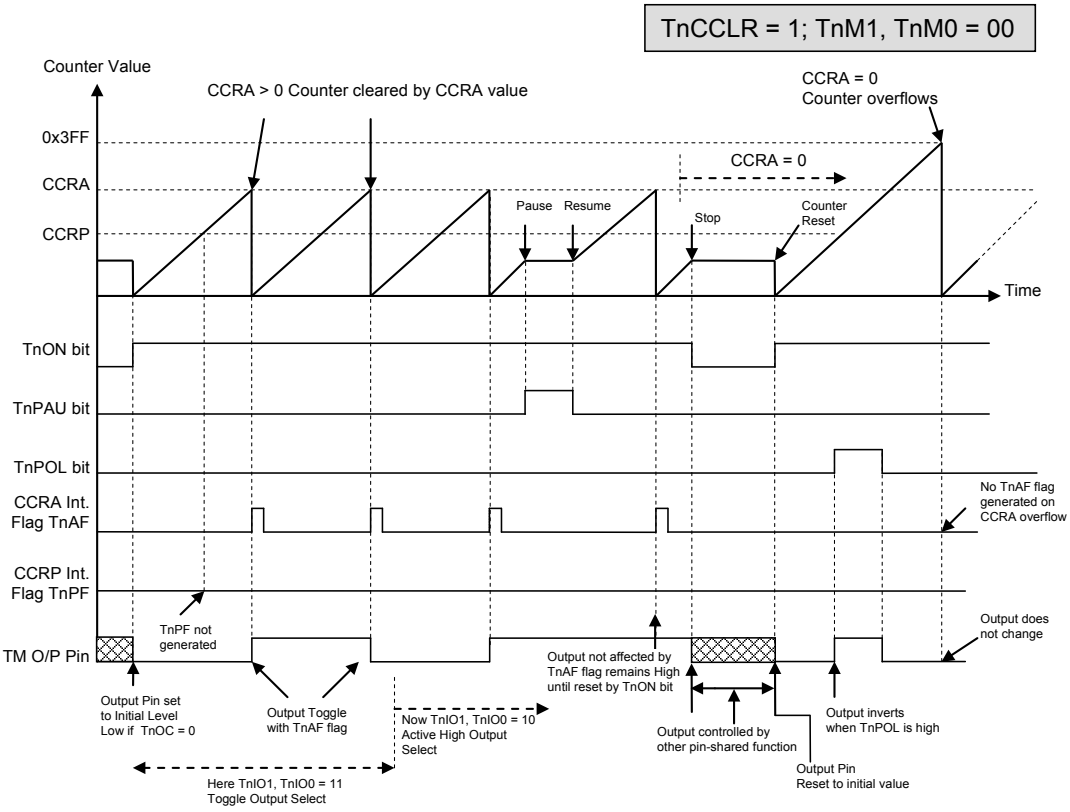

**Compare Match Output Mode – TnCCLR=0**

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

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

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

4. n = 0




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

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

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

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

4. The TnPF flags is not generated when TnCCLR = 1

5. n = 0

### Timer/Counter Mode

To select this mode, bits T0M1 and T0M0 in the TM0C1 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 T0M1 and T0M0 in the TM0C1 register should be set to 10 respectively and also the T0IO1 and T0IO0 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 T0CCLR bit has no effect as the PWM period. 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 T0DPX bit in the TM0C1 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 T0OC bit In the TM0C1 register is used to select the required polarity of the PWM waveform while the two T0IO1 and T0IO0 bits are used to enable the PWM output or to force the TM output pin to a fixed high or low level. The T0POL bit is used to reverse the polarity of the PWM output waveform.

#### 10-bit STM, PWM Mode, Edge-aligned Mode, T0DPX=0

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

If $f_{SYS}$ = 4MHz, TM clock source is $f_{SYS}$, CCRP = 010B and CCRA =128,

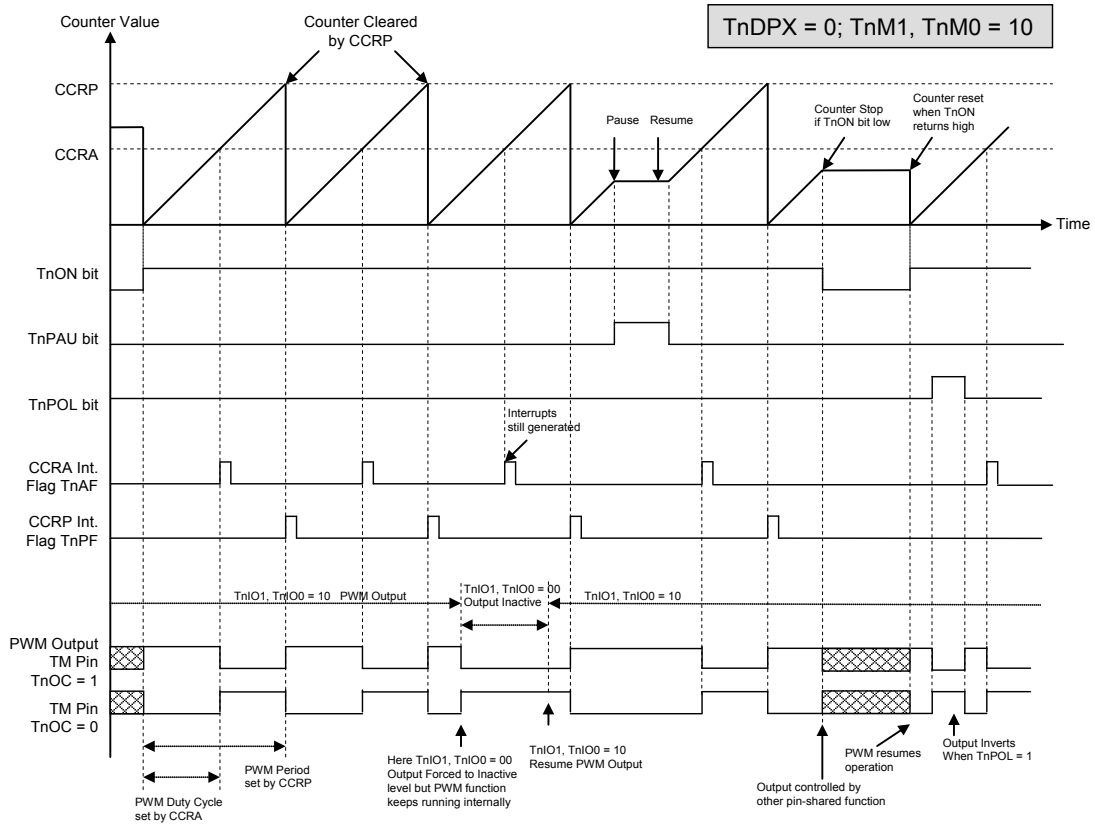
The STM PWM output frequency = $f_{SYS}$ / (2×256) = $f_{SYS}$/512 = 7.8125 kHz, duty = 128/(2×256) = 25%.

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 STM, PWM Mode, Edge-aligned Mode, T0DPX=1

| CCRP | 001 | 010 | 011 | 100 | 101 | 110 | 111 | 000 |
|---|---|---|---|---|---|---|---|---|
| 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**

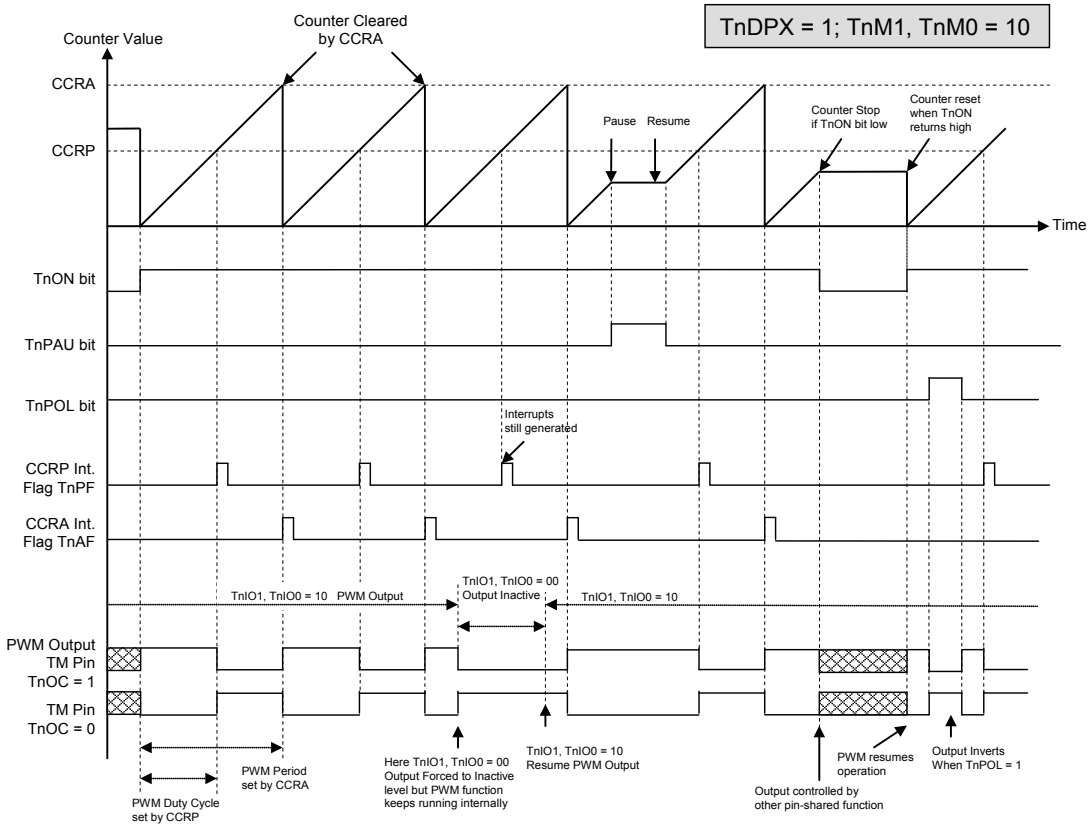
Note: 1. Here TnDPX = 0 - Counter cleared by CCRP

2. A counter clear sets PWM Period

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

4. The TnCCLR bit has no influence on PWM operation

5. n = 0




**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

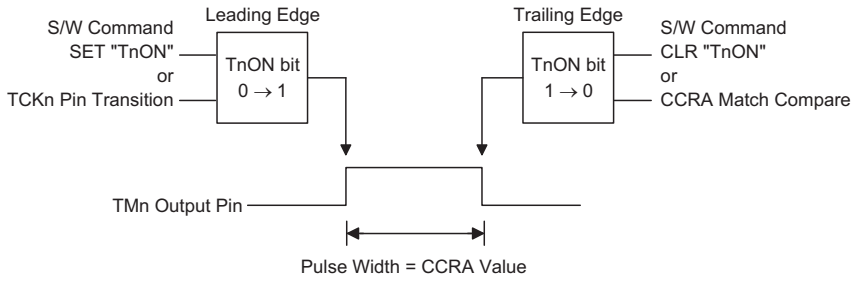
4. The TnCCLR bit has no influence on PWM operation

5. n = 0

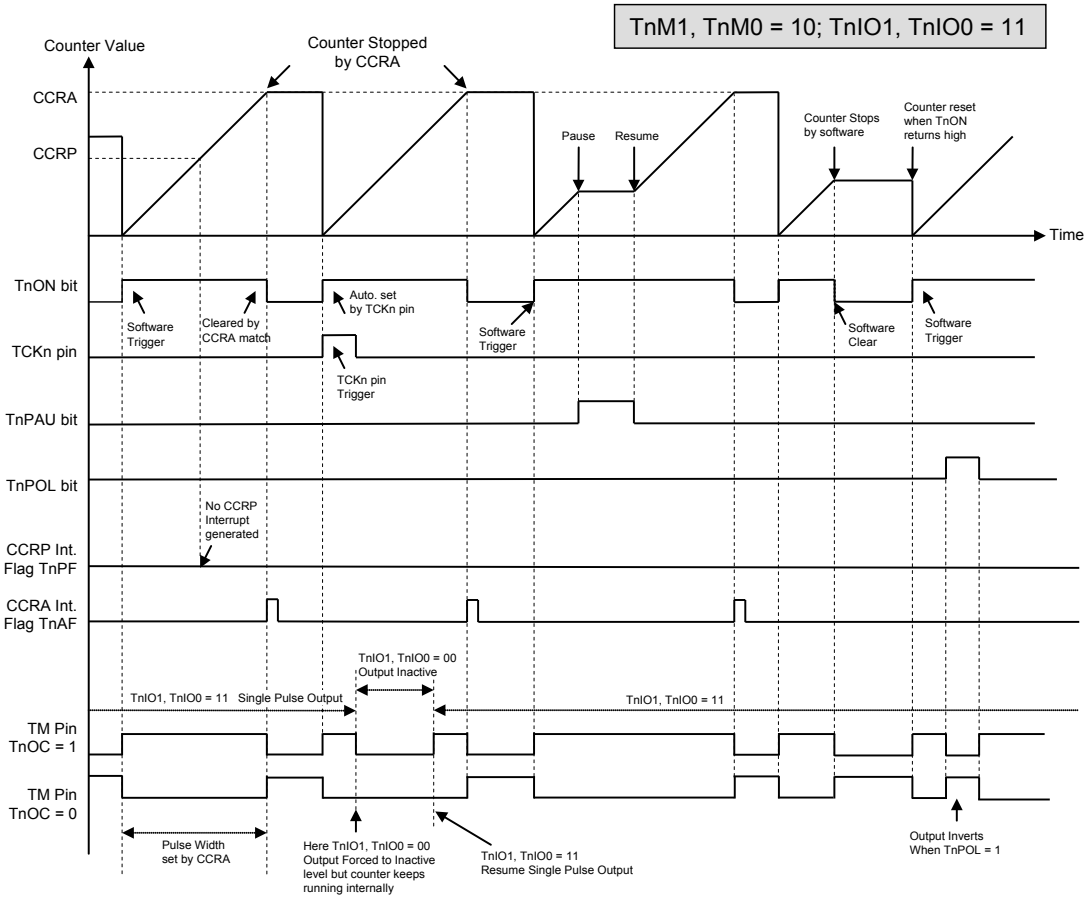
## Single Pulse Mode

To select this mode, bits T0M1 and T0M0 in the TM0C1 register should be set to 10 respectively and also the T0IO1 and T0IO0 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 T0ON bit, which can be implemented using the application program. However in the Single Pulse Mode, the T0ON bit can also be made to automatically change from low to high using the external TCK0 pin, which will in turn initiate the Single Pulse output. When the T0ON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The T0ON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the T0ON bit is cleared to zero, which can be implemented using the application program or when a compare match occurs from Comparator A.




**Single Pulse Generation (n=0)**




**Single Pulse Mode**

Note: 1. Counter stopped by CCRA match

2. CCRP is not used
3. The pulse is triggered by the TCKn pin or 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.
6. n = 0

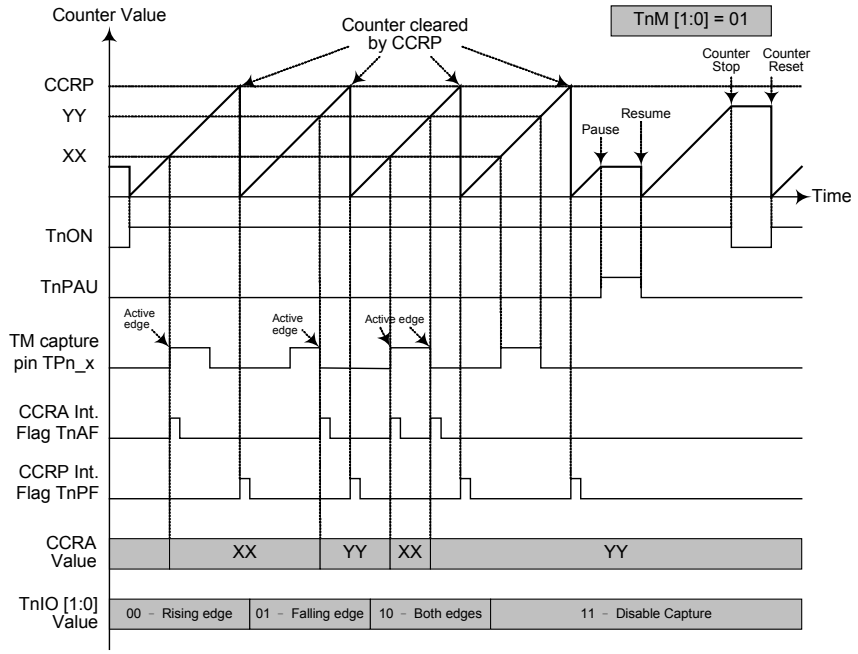
However a compare match from Comparator A will also automatically clear the T0ON bit and thus generate the Single Pulse output trailing edge. In this way the CCRA value can be used to control the pulse width. A compare match from Comparator A will also generate a TM interrupt. The counter can only be reset back to zero when the T0ON bit changes from low to high when the counter restarts. In the Single Pulse Mode CCRP is not used. The T0CCLR and T0DPX bits are not used in this Mode.

## Capture Input Mode

To select this mode bits T0M1 and T0M0 in the TM0C1 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 TP0 pin, whose 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 T0IO1 and T0IO0 bits in the TM0C1 register. The counter is started when the T0ON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the TP0 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 TP0 pin the counter will continue to free run until the T0ON 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 T0IO1 and T0IO0 bits can select the active trigger edge on the TP0 pin to be a rising edge, falling edge or both edge types. If the TnIO1 and T0IO0 bits are both set high, then no capture operation will take place irrespective of what happens on the TP0 pin, however it must be noted that the counter will continue to run.

As the TP0 pin is pin shared with other functions, care must be taken if the TM 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 T0CCLR and T0DPX 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. The TnCCLR bit is 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.
6. n = 0

# Analog to Digital Converter

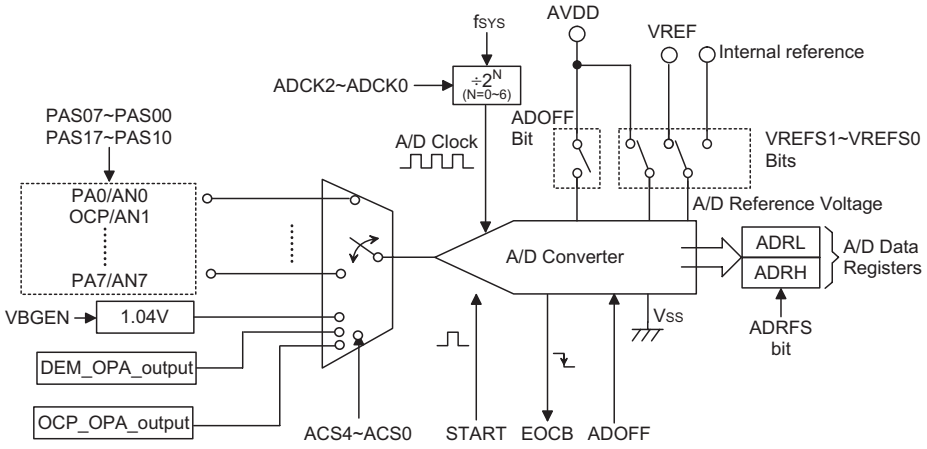
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

## A/D Overview

The device contains an 8-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. Two additional channels are Demodulation and OCP operational amplifier outputs, DEM_OPA_output and OCP_OPA_output.

| Input Channels | A/D Channel Select Bits | Input Pins |
|---|---|---|
| 8+2 | ACS4~ACS0 | AN0~AN7,<br>DEM_OPA_output, OCP_OPA_output |

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 four registers. A read only register pair exists to store the ADC data 12-bit value. The remaining two 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 | ACS4 | ACS3 | ACS2 | ACS1 | ACS0 |
| ADCR1 | — | VBGEN | ADRFS | VREFS1 | VREFS0 | ADCK2 | ADCK1 | ADCK0 |

**A/D Converter Register List**

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

As the device contains 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

To control the function and operation of the A/D converter, three control registers known as ADCR0, ADCR1 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 ACS4~ACS0 bits in the ADCR0 register define the ADC input channel number. As the device contains only one actual analog to digital converter hardware circuit, each of the individual 8 analog inputs must be routed to the converter. It is the function of the ACS4 ~ ACS0 bits to determine which analog channel input signals, DEM_OPA_output, OCP_OPA_output or internal 1.04V is actually connected to the internal A/D converter.

The PAS0 control register contains the PAS07~PAS00 bits and PAS1 control register contains the PAS17~PAS10 bits which determine which pins on Port A 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 | ACS4 | ACS3 | ACS2 | ACS1 | ACS0 |
| R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 1 | 1 | 0 | 0 | 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 ~ 0 **ACS4 ~ ACS0:** Select A/D channel (when ACS4 is "0")
00000: AN0
00001: AN1
00010: AN2
00011: AN3
00100: AN4
00101: AN5
00110: AN6
00111: AN7
01000: AN8 ( DEM_OPA_output )
01001~01111: AN9 ( OCP_OPA_output )
1xxxx: Bandgap Voltage

These are the A/D channel select control bits. As there is only one internal hardware A/D converter each of the eight A/D inputs must be routed to the internal converter using these bits. If bit ACS4 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 | — | VBGEN | ADRFS | VREFS1 | VREFS0 | ADCK2 | ADCK1 | ADCK0 |
| 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 **VBGEN:** Internal reference voltage circuit 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 reference voltage can be used by the A/D converter. If reference voltage 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 reference voltage 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 **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 4 ~ 3 **VREFS1~VREF0:** Select ADC reference voltage
00: Internal ADC power
01: External VREF pin
1x: Internal $V_{REF}$

These bits are used to select the reference voltage for the A/D converter. If these bits are "01" then the A/D converter reference voltage is supplied on the external VREF pin. If these bits are set to "1x" then the A/D converter reference voltage is supplied on the internal VREF. If these bits are set to "00" then the internal reference is used which is taken from the power supply pin AVDD.

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: $f_{SUB}$

These three bits are used to select the clock source for the A/D converter.

## 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 maximum A/D clock source speed 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 system clock frequencies. For example, if the system clock operates at a frequency of 5MHz, the ADCK2~ADCK0 bits should not be set to 000B, 001B 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 ($f_{SUB}$) |
| 5MHz | 200ns* | 400ns* | 800ns | 1.6μs | 3.2μs | 6.4μs | 12.8μs* | Undefined |
| 10MHz | 100ns* | 200ns* | 400ns* | 800ns | 1.6μs | 3.2μs | 6.4μs | Undefined |
| 20MHz | 50ns* | 100ns* | 200ns* | 400ns* | 800ns | 1.6μs | 3.2μs | Undefined |

**A/D Clock Period Examples**

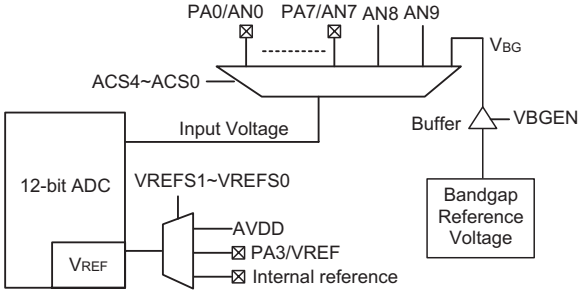
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADOFF bit in the ADCR0 register. This bit must be zero to power on the A/D converter. When the ADOFF bit is cleared to zero to power on the A/D converter internal circuitry a certain delay, as indicated in the 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 PAS07~PAS00 bits in the PAS0 register or the PAS17~PAS10 bits in the PAS1 register or, 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, or from, an internal reference sources. The desired selection is made using the VREFS1~VREFS0 bits. As the VREF pin is pin-shared with other functions, when the VREFS1~VREFS0 bits are set to "01", 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 Port A as well as other functions. The PAS07~PAS00 bits in the PAS0 control register and the PAS17~PAS10 bits in the PAS1 control register, determine whether the input pins are setup as A/D converter analog inputs or whether they have other functions. If these bits for its corresponding pin are set to be correct values 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 PAS07~PAS00 bits or the PAS17~PAS10 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 VREFS1~VREFS0 bits in the ADCR1 register. The analog input values must not be allowed to exceed the value of VREF.




**A/D Input Structure**

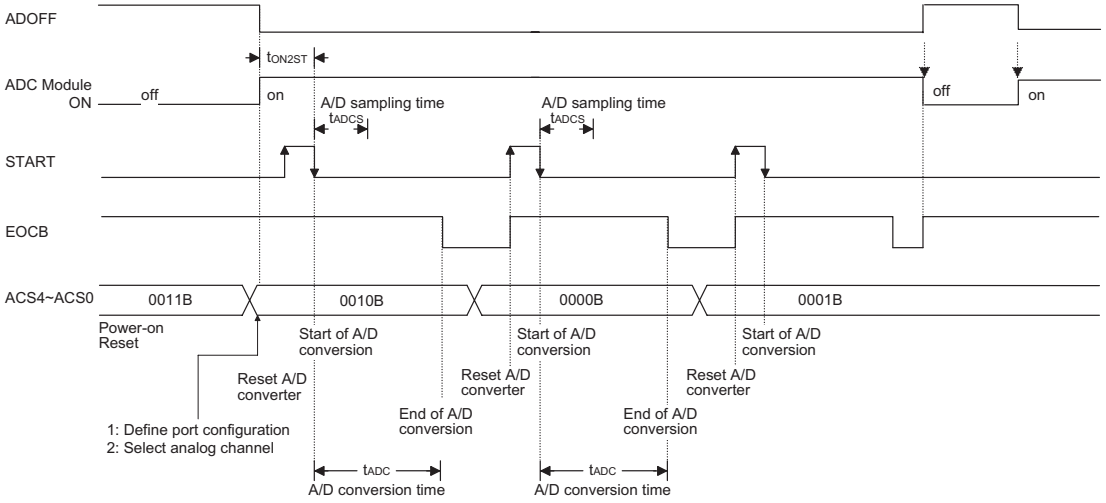
### Summary of A/D Conversion Steps

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~ACS0 bits which are also contained in the ADCR0 register.
- Step 4
  Select which pins are to be used as A/D inputs and configure them by correctly programming the PAS07~PAS00 bits in the PAS0 register and the PAS17~PAS10 bits in the PAS1 register.(AN1does not need to be selected as A/D input, because it is only pin with OCP and these two pins are both as input pins)
- 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

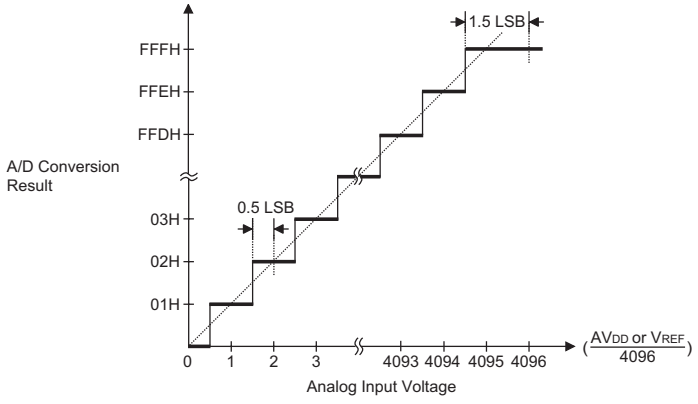
As the device contains a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the $V_{DD}$ or $V_{REF}$ voltage, this gives a single bit analog input value of $V_{DD}$ or $V_{REF}$ divided by 4096.

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

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

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

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 Examples

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**

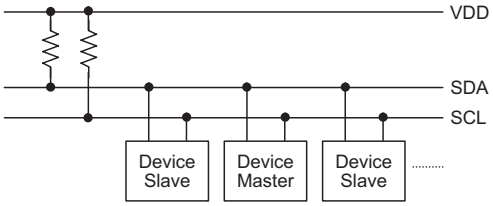
```
clr    ADE             ; disable ADC interrupt
mov    a,03H
mov    ADCR1,a         ; select fSYS/8 as A/D clock and switch off 1.25V
clr    ADOFF
mov    a,0FFh          ; setup PAS0 to configure pins AN0,AN2,AN3
mov    PAS0,a
mov    a,00h
mov    ADCR0,a         ; enable and connect AN0 channel to 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 1.04V
Clr     ADOFF
mov     a,0FFh        ; setup PAS0 to configure pins AN0, AN2, AN3
mov     PAS0,a
mov     a,00h
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
```

## I²C Interface

The I²C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple communication protocol and the ability to accommodate multiple devices on the same bus has made it an extremely popular interface type for many applications.




**I²C Master/Slave Bus Connection**

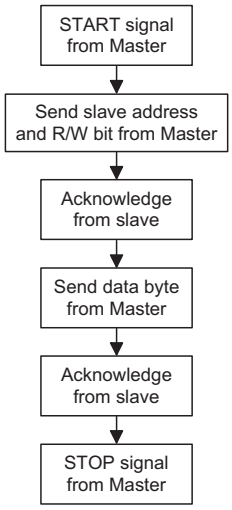
### I²C Interface Operation

The I²C serial interface is a two line interface, a serial data line, SDA, and serial clock line, SCL. As many devices may be connected together on the same bus, their outputs are both open drain types. For this reason it is necessary that external pull-high resistors are connected to these outputs. Note that no chip select line exists, as each device on the I²C bus is identified by a unique address which will be transmitted and received on the I²C bus.

When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For this device, which only operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode.

It is suggested that the user shall not enter the micro processor to HALT mode by application program during processing I²C communication.

If the pin is configured to SDA or SCL function of I²C interface, the pin is configured to open-collect Input/Output port and its pull-up function can be enabled by programming the related Generic Pull-up Control Register.



## I²C Registers

There are four control registers associated with the I²C bus, IICC0, IICC1, IICA and I2CTOC and one data register, IICD. The IICD register, is used to store the data being transmitted and received on the I²C bus. Before the microcontroller writes data to the I²C bus, the actual data to be transmitted must be placed in the IICD register. After the data is received from the I²C bus, the microcontroller can read it from the IICD register. Any transmission or reception of data from the I²C bus must be made via the IICD register.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IICC0 | — | — | — | — | I2CDBNC1 | I2CDBNC0 | I2CEN | — |
| IICC1 | IICHCF | IICHAAS | IICHBB | IICHTX | IICTXAK | IICSRW | IICRNIC | IICRXAK |
| IICD | IICDD7 | IICDD6 | IICDD5 | IICDD4 | IICDD3 | IICDD2 | IICDD1 | IICDD0 |
| IICA | IICA6 | IICA5 | IICA4 | IICA3 | IICA2 | IICA1 | IICA0 | — |
| I2CTOC | I2CTOEN | I2CTOF | I2CTOS5 | I2CTOS4 | I2CTOS3 | I2CTOS2 | I2CTOS1 | I2CTOS0 |

**I²C Registers List**

### IICC0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | I2CDBNC1 | I2CDBNC0 | I2CEN | — |
| R/W | — | — | — | — | R/W | R/W | R/W | — |
| POR | — | — | — | — | 0 | 0 | 0 | — |

Bit 7~4 unimplemented, read as “0”

Bit 3~2 **I2CDBNC1~I2CDBNC0:** I²C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
10: 4 system clock debounce
11: 4 system clock debounce

Bit 1 **I2CEN:** I²C enable
0: Disable (GPIO pin-shared with I²C is I/O function)
1: Enable (GPIO pin-shared with I²C is I²C function)

Bit 0 Unimplemented, read as "0"

I²C function could be turned off or turned on by controlling the related pin-sharing control bit which decides the function of the I/O ports pin-shared the pins SDA and SCL. When the I/O ports pin-shared the pins SDA and SCL are chosen to the functions other than SDA and SCL by pin-sharing control bit, I²C function is turned off and its operating current will be reduced to a minimum value. In contrary, I²C function is turned on when the I/O ports pin-shared the pins SDA and SCL are chosen to the pins SDA and SCL by controlling pin-sharing control bit.

**IICC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | IICHCF | IICHAAS | IICHBB | IICHTX | IICTXAK | IICSRW | IICRNIC | IICRXAK |
| R/W | R | R | R | R/W | R/W | R | R/W | R |
| POR | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Bit 7 **IICHCF:** $I^2C$ Bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-bit data transfer

The IICHCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

Below is an example of the flow of a two-byte $I^2C$ data transfer.

- First, $I^2C$ slave device receive a start signal from $I^2C$ master and then IICHCF bit is automatically cleared to zero.
- Second, $I^2C$ slave device finish receiving the 1st data byte and then IICHCF bit is automatically set to one.
- Third, user read the 1st data byte from IICD register by the application program and then IICHCF bit is automatically cleared to zero.
- Fourth, $I^2C$ slave device finish receiving the 2nd data byte and then IICHCF bit is automatically set to one and so on.
- Finally, $I^2C$ slave device receive a stop signal from $I^2C$ master and then IICHCF bit is automatically set to one.

Bit 6 **IICHAAS:** $I^2C$ Bus address match flag
0: Not address match
1: Address match

The IICHASS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

Bit 5 **IICHBB:** $I^2C$ Bus busy flag
0: $I^2C$ Bus is not busy
1: $I^2C$ Bus is busy

The IICHBB flag is the $I^2C$ busy flag. This flag will be "1" when the $I^2C$ bus is busy which will occur when a START signal is detected. The flag will be set to "0" when the bus is free which will occur when a STOP signal is detected.

Bit 4 **IICHTX:** Select $I^2C$ slave device is transmitter or receiver
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **IICTXAK:** $I^2C$ Bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave do not send acknowledge flag

The IICTXAK bit is the transmit acknowledge flag. After the slave device receipt of 8-bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set IICTXAK bit to "0" before further data is received.

Bit 2 **IICSRW:** $I^2C$ Slave Read/Write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode

The IICSRW flag is the $I^2C$ Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the $I^2C$ bus. When the transmitted address and slave address is match, that is when the IICHAAS flag is set high, the slave device will check the IICSRW flag to determine whether it should be

in transmit mode or receive mode. If the IICSRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the IICSRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

Bit 1 **IICRNIC:** I²C running using Internal Clock Control
0: I²C running using internal clock
1: I²C running not using Internal Clock
The I²C module can run without using internal clock, and generate an interrupt if the I²C interrupt is enabled, which can be used in SLEEP Mode, IDLE(SLOW) Mode.

Bit 0 **IICRXAK:** I²C Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave do not receive acknowledge flag
The IICRXAK flag is the receiver acknowledge flag. When the IICRXAK flag is "0", it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the IICRXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the IICRXAK flag is "1". When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus.

The IICD register is used to store the data being transmitted and received. Before the device writes data to the I²C bus, the actual data to be transmitted must be placed in the IICD register. After the data is received from the I²C bus, the device can read it from the IICD register. Any transmission or reception of data from the I²C bus must be made via the IICD register.

### IICD Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | IICDD7 | IICDD6 | IICDD5 | IICDD4 | IICDD3 | IICDD2 | IICDD1 | IICDD0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | x | x | x | x | x | x | x | x |

"x" unknown

Bit 7~0 **IICDD7~IICDD0:** I²C Data Buffer bit 7~bit 0

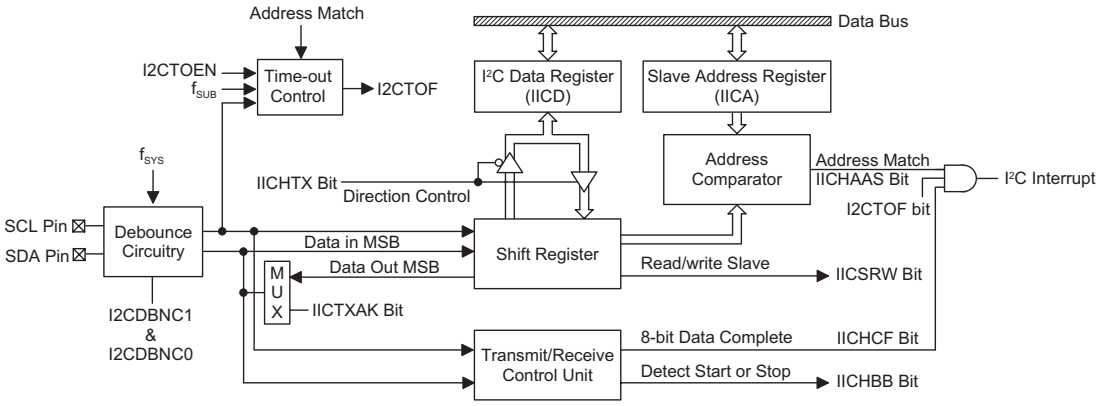
### IICA Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | IICA6 | IICA5 | IICA4 | IICA3 | IICA2 | IICA1 | IICA0 | — |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | — |
| POR | x | x | x | x | x | x | x | — |

"x" unknown

Bit 7~1 **IICA6~IICA0:** I²C slave address
IICA6~ IICA0 is the I²C slave address bit 6 ~ bit 0.
The IICA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~ 1 of the IICA register define the device slave address. Bit 0 is not defined.
When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the IICA register, the slave device will be selected.

Bit 0 Unimplemented, read as "0"

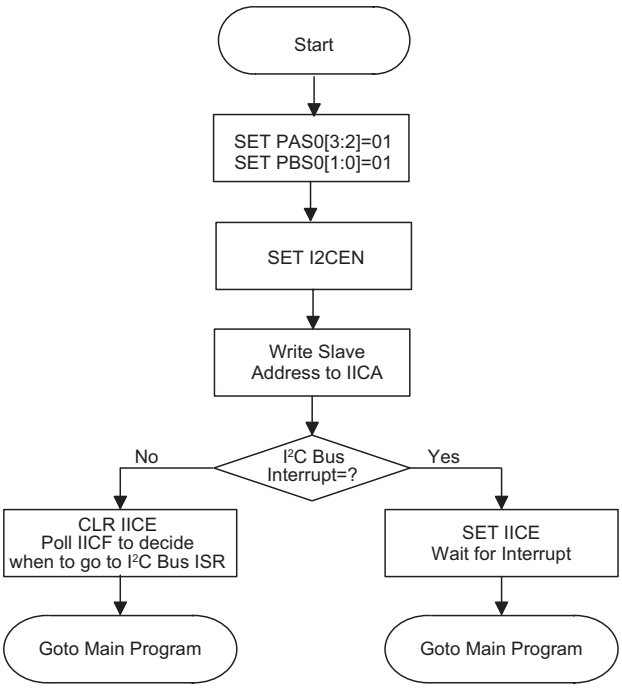



**I²C Block Diagram**

## I²C Bus Communication

Communication on the I²C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I²C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the IICHAAS bit in the IICC1 register will be set and an I²C interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the IICHAAS bit to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1
  Set Configure the pin-shared I/O ports to I²C pin function. (SCL and SDA).
- Step 2
  Set I2CEN bit in the IICC0 register to "1" to enable the I²C bus.
- Step 3
  Write the slave address of the device to the I²C bus address register IICA.
- Step 4
  Set the IICE interrupt enable bit of the interrupt control register to enable the I²C interrupt and Multi-function interrupt.




**I²C Bus Initialisation Flow Chart**

## I²C Bus Start Signal

The START signal can only be generated by the master device connected to the I²C bus and not by the slave device. This START signal will be detected by all devices connected to the I²C bus. When detected, this indicates that the I²C bus is busy and therefore the IICHBB bit will be set. A START condition occurs when a high to low transition on the SDA line takes place when the SCL line remains high.

## Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal I²C bus interrupt signal will be generated. The next bit following the address, which is the 8th bit, defines the read/write status and will be saved to the IICSRW bit of the IICC1 register. The slave device will then transmit an acknowledge bit, which is a low level, as the 9th bit. The slave device will also set the status flag IICHAAS when the addresses match.

As an I²C bus interrupt can come from two sources, when the program enters the interrupt subroutine, the IICHAAS bit should be examined to see whether the interrupt source has come from a matching slave address or from the completion of a data byte transfer. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the IICD register, or in the receive mode where it must implement a dummy read from the IICD register to release the SCL line.

### I²C Bus Read/Write Signal

The IICSRW bit in the IICC1 register defines whether the slave device wishes to read data from the I²C bus or write data to the I²C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the IICSRW flag is "1" then this indicates that the master device wishes to read data from the I²C bus, therefore the slave device must be setup to send data to the I²C bus as a transmitter. If the IICSRW flag is "0" then this indicates that the master wishes to send data to the I²C bus, therefore the slave device must be setup to read data from the I²C bus as a receiver.

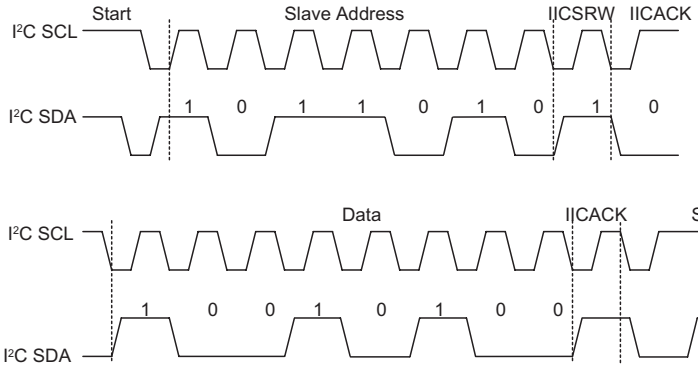
### I²C Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the IICHAAS flag is high, the addresses have matched and the slave device must check the IICSRW flag to determine if it is to be a transmitter or a receiver. If the IICSRW flag is high, the slave device should be setup to be a transmitter so the IICHTX bit in the IICC1 register should be set to "1". If the IICSRW flag is low, then the microcontroller slave device should be setup as a receiver and the IICHTX bit in the IICC1 register should be set to "0".

### I²C Bus Data and Acknowledge Signal

The transmitted data is 8-bits wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8-bits of data, the receiver must transmit an acknowledge signal, level "0", before it can receive the next data byte. If the slave transmitter does not receive an acknowledge bit signal from the master receiver, then the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus. The corresponding data will be stored in the IICD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the IICD register. If setup as a receiver, the slave device must read the transmitted data from the IICD register.

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as IICTXAK, on the 9th clock. The slave device, which is setup as a transmitter will check the IICRXAK bit in the IICC1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.

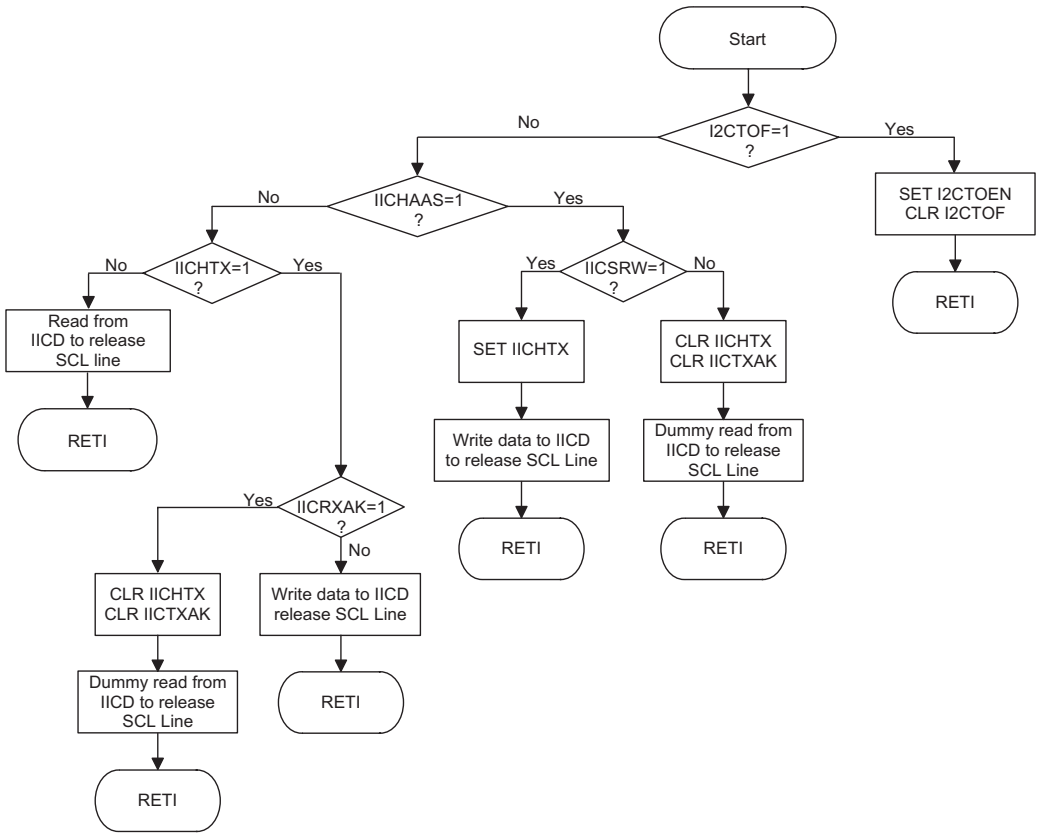



S=Start (1 bit)
SA=Slave Address (7 bits)
SR=IICSRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=IICACK (IICRXAK bit for transmitter, IICTXAK bit for receiver 1 bit)
P=Stop (1 bit)

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

Note: *When a slave address is matched, the device must be placed in either the transmit mode and then write data to the IICD register, or in the receive mode where it must implement a dummy read from the IICD register to release the I²C SCL line.

**I²C Communication Timing Diagram**




**I²C Bus ISR Flow Chart**

## I²C Time-out Control

In order to reduce the problem of I²C lockup due to reception of erroneous clock sources, a time-out function is provided. If the clock source to the I²C is not received then after a fixed time period, the I²C circuitry and registers will be reset.

The time-out counter starts counting on an I²C bus "START" & "address match" condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out setup by the I2CTOC register, then a time-out condition will occur. The time-out function will stop when an I²C "STOP" condition occurs.

When an I²C time-out counter overflow occurs, the counter will stop and the I2CTOEN bit will be cleared to zero and the I2CTOF bit will be set high to indicate that a time-out condition has occurred. The time-out condition will also generate an interrupt which uses the I²C interrupt vector. When an I²C time-out occurs, the I²C internal circuitry will be reset and the registers will be reset into the following condition:

| Register | After I²C Time-out |
|---|---|
| IICD, IICA, IICC0 | No change |
| IICC1 | Reset to POR condition |

**I²C Registers After Time-out**

The I2CTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using bits in the I2CTOC register. The time-out time is given by the formula:

$$((1\sim64) \times 32) / f_{SUB}.$$

This gives a range of about 1ms to 64ms. Note also that the LIRC oscillator is continuously enabled.

### I2CTOC Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | I2CTOEN | I2CTOF | I2CTOS5 | I2CTOS4 | I2CTOS3 | I2CTOS2 | I2CTOS1 | I2CTOS0 |
| 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 |

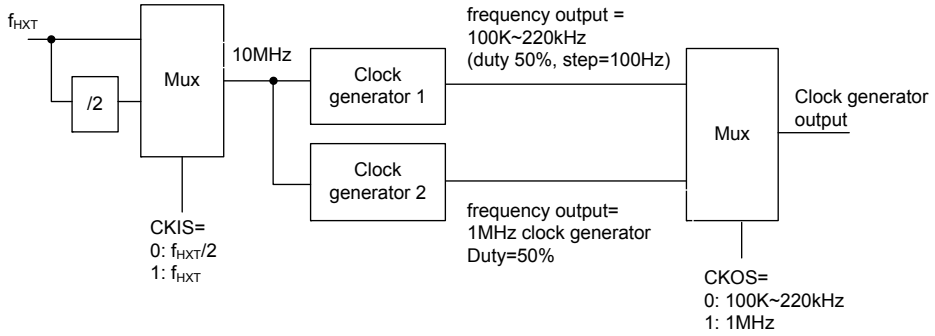
Bit 7 **I2CTOEN:** I²C Time-out Control
0: disable
1: enable

Bit 6 **I2CTOF:** Time-out flag (set by time-out and clear by software)
0: no time-out
1: time-out occurred

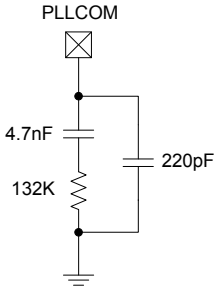
Bit 5~0 **I2CTOS5~I2CTOS0:** Time-out Definition
I²C time-out clock source is $f_{SUB}/32$.
I²C time-out time is given by: ([I2CTOS5 : I2CTOS0]+1) × $(32/f_{SUB})$

## PLL Clock Generator

The device provides a clock generator output which can be used as a PWM driver signal. The accompanying block diagram shows the overall internal structure of the clock generator, together with its associated registers.




**Clock Generator Block Diagram**

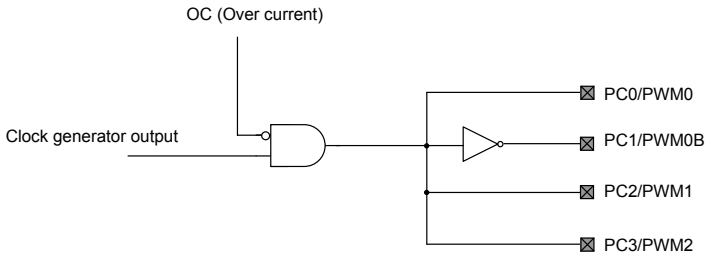



**PLLCOM Pin External Circuit**

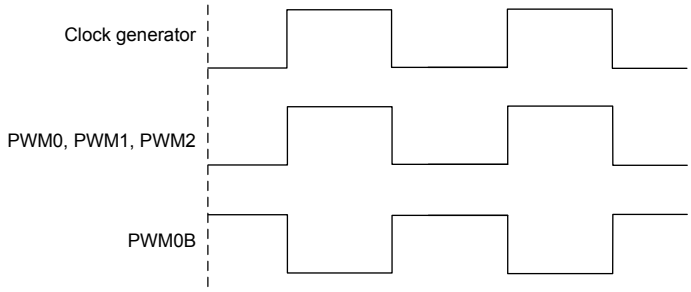
### Clock Generator Operation

The generator clock, can come from either $f_{HXT}$ or $f_{HXT}/2$, and is selected using the CKIS bit in the CKGEN register. The PLLEN and F1MEN bits in the CKGEN register are used to control clock generator 1 and clock generator 2 respectively. The output frequency of the clock generator 1 is within a range of 100K~220K(step=0.1KHz) while the clock generator 2 output frequency is 1MHz. Clock Generator output can come from generator 1 output or generator 2 output which is selected by CKOS bit in the CKGEN register. The output frequency of the clock generator 1 is selected by PLLFL and PLLFH registers.

The clock output can be used as PWM driver signal. The PWM0EN~PWM2EN bits in the PWMC register, determine whether the PWM output function is enabled. The accompanying waveform diagram shows the relationship between the clock generator output and PWM signal for different output pins.




**Clock Generator Output Driver Block Diagram**




**Clock Generator and PWMn Output Waveform**

## Clock Generator Register Description

Three registers control the overall operation of the clock generator. These are the generator overall control register, CKGEN, the generator 1 frequency selection registers, PLLFL and PLLFH, and the PWM output control register, PWMC.

### CKGEN Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PLLEN | F1MEN | CKOS | CKIS | — | — | — | — |
| R/W | R/W | R/W | R/W | R/W | — | — | — | — |
| POR | 0 | 0 | 0 | 0 | — | — | — | — |

Bit 7 **PLLEN:** PLL clock generator 1 enable
0: Disable
1: Enable

Bit 6 **F1MEN:** 1MHz clock generator 2 enable
0: Disable
1: Enable

Bit 5 **CKOS:** Output clock source selection
0: From PLL clock generator(100kHz~220kHz)
1: From 1MHz clock generator

Bit 4 **CKIS:** Input clock source selection
0: $f_{HXT}/2$
1: $f_{HXT}$

Bit 3 ~ 0 Unimplemented, read as 0

### PLLFL Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PFQ7 | PFQ6 | PFQ5 | PFQ4 | PFQ3 | PFQ2 | PFQ1 | PFQ0 |
| 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 |

### PLLFH Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | PFQ10 | PFQ9 | PFQ8 |
| R/W | — | — | — | — | — | R/W | R/W | R/W |
| POR | — | — | — | — | — | 0 | 0 | 0 |

**PFQ10 ~ PFQ0:** PLL frequency control bit
0: 100kHz
1:100.1kHz
2: 100.2kHz
3: 100.3kHz
4: 100.4kHz
… ...
254: 125.4kHz
255: 125.5kHz
256: 125.6kHz
257: 125.7kHz
… …
1197: 219.7kHz
1198: 219.8kHz
1199: 219.9kHz
1200: 220kHz
Other Values: 220kHz

### PWMC Register ( PWM control register)

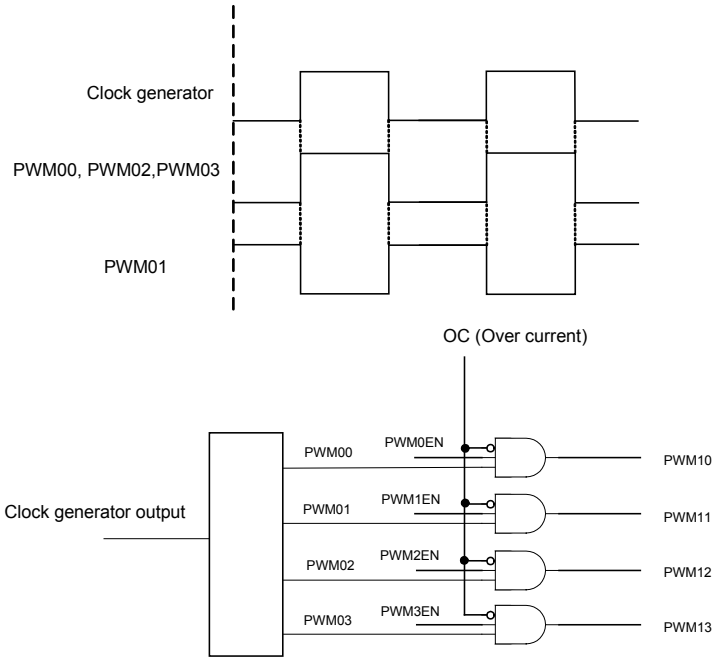
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PMOD3 | PMOD2 | PMOD1 | PMOD0 | PWM3EN | PWM2EN | PWM1EN | PWM0EN |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |

Bit 7~4 **PMOD3~PMOD0:** PWM mode
0101: Mode 0
1010: Mode 1 ( full bridge complementary PWM output with dead time)
Other values: MCU reset (Prevent Interference)

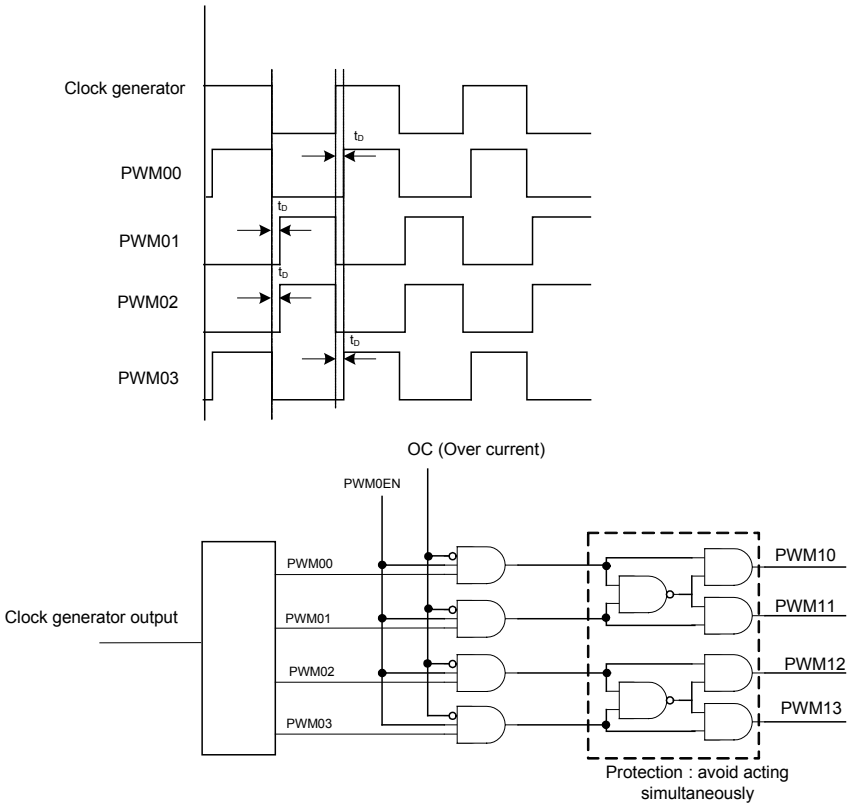
Bit 3~0 **PWM3EN/PWM2EN/PWM1EN/PWM0EN:** PWM3/PWM2/PWM1/PWM0 enable/disable control
0: Disable
1: Enable

## PWM output control

### Mode 0




### Mode 1 (Protection mechanism is only existed in mode1)



**CPR Register (Complementary PWM control register)**

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

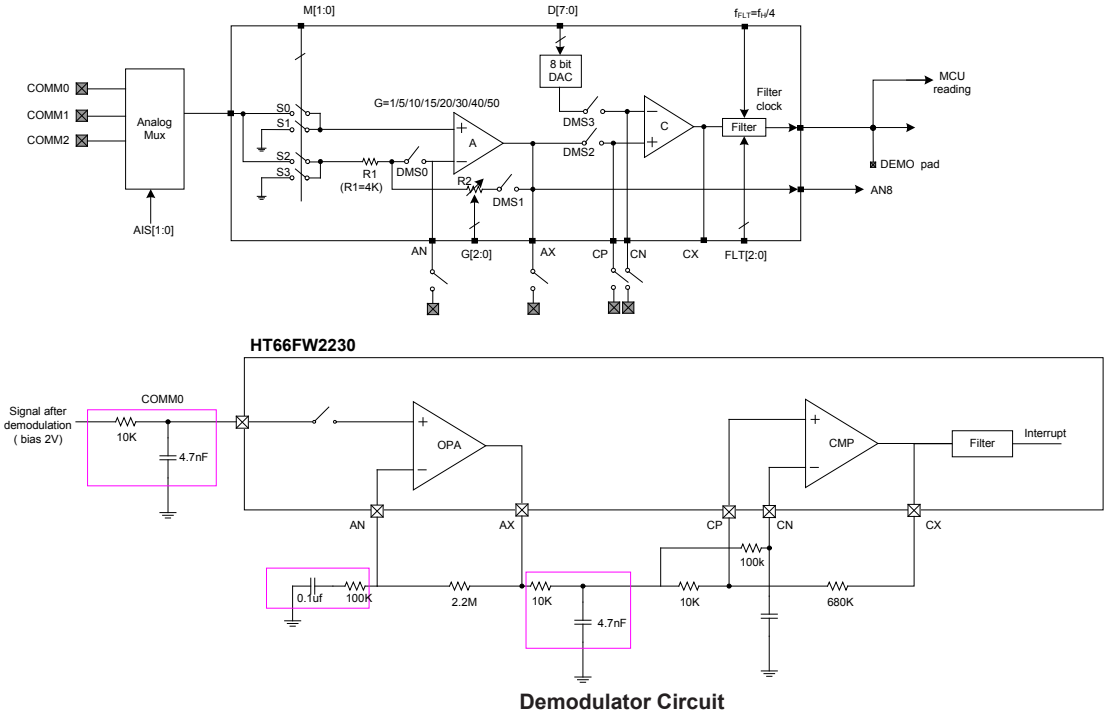
Bit 7~5 **WRPRT :** Write protection for PCS0, PCS1 and PWMC registers
0: These registers are writable
1: These registers can't be changed by writing

Bit 4~3 **DTPSC[1:**0]: Dead Time prescaler
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
$t_D=1/f_D$
000: Dead time is $[(1/f_D)-(1/f_H)] \sim (1/f_D)$
001: Dead time is $[(2/f_D)-(1/f_H)] \sim (2/f_D)$
010: Dead time is $[(3/f_D)-(1/f_H)] \sim (3/f_D)$
011: Dead time is $[(4/f_D)-(1/f_H)] \sim (4/f_D)$
100: Dead time is $[(5/f_D)-(1/f_H)] \sim (5/f_D)$
101: Dead time is $[(6/f_D)-(1/f_H)] \sim (6/f_D)$
110: Dead time is $[(7/f_D)-(1/f_H)] \sim (7/f_D)$
111: Dead time is $[(8/f_D)-(1/f_H)] \sim (8/f_D)$

## Demodulation Function

The Demodulator demodulates the communication signal from the receiver end.




**Demodulator Circuit**

### Demodulator Circuit Operation

The demodulator input is sourced from COMM0~COMM2, selected using the AIS1~AIS0 bits in the DCMISC register. After this, four switches S0~S3, are used for mode selection. An OPAMP and two resistors are used to form a PGA function. The PGA gain can be positive or negative determined by the input voltage connected to the positive input or negative input of the PGA. DEMREF is used to generate reference voltage. The comparator compares this reference voltage with the amplified output. Finally the comparator output is filtered to generate DEMO and DEMINT. These are debounced versions of DEMCX which are used to indicate whether the source current is outwith the specification or not. DEMO is defined as the demodulator output and DEMINT is the demodulator interrupt trigger.

Note that the filter clock; $f_{HXT}$ is the HXT clock. The amplified output voltage also can be read out by means of another ADC from DEMAX. The DAC output voltage is controlled by the DEMREF register and the DAC output is defined as

$$\text{DAC } V_{OUT} = (\text{DAC } V_{REF}/256) \times \text{DEMREF}[7:0] \qquad (1)$$

### Input Voltage Range

The input voltage can be positive or negative, which together with the PGA operating mode, provides for a more flexible application.

(1) If $V_{IN} > 0$ and the PGA operates in the non-inverting mode, the output voltage of the PGA is

$$VO_{PGA} = (1 + R_2 / R_1) \times V_{IN} \quad (2)$$

(2) When the PGA operates in the non-inverter mode, it also provides a unity gain buffer function.

If DEM[1:0]=01 and DEMG[2:0]=000, the PGA gain will be 1 and is configured as unity gain buffer. Switches S2 and S3 will be open internally and the output voltage of the PGA is

$$VO_{PGA} = V_{IN} \quad (3)$$

(3) If $0 > V_{IN} > -0.7V$ and the PGA operates in the inverter mode, the output voltage of the PGA is

$$VO_{PGA} = -(R_2 / R_1) \times V_{IN} \quad (4)$$

Note: if $V_{IN}$ is negative, it should not be lower than -0.7V to avoid leakage current.

### Offset Calibration

The demodulation circuit has 4 operating modes controlled by DEM1~DEM0. One of these modes is the calibration mode (0 V input mode). In the calibration mode, the OP and comparator offset can be calibrated.

OPAMP calibration:

- Step1: Set DEM [1:0] =11, DEMAOFM=1, the demodulator is now in the OPAMP calibration mode.
- Step2: Set DEMAOF [4:0] =00000 then read the DEMAX bit status.
- Step3: Let DEMAOF=DEMAOF+1 then read the DEMAX bit status; if DEMAX is changed, record the register data as VOS1.
- Step4: Set DEMAOF [4:0] 111111 then read the DEMAX bit status.
- Step5: Let DEMAOF=DEMAOF-1 then read the DEMAX bit status; if DEMAX is changed, record the register data as VOS2.
- Step6: Restore VOS = (VOS1 + VOS2)/2 to the DEMAOF register. The calibration is now finished.

Comparator calibration:

- Step1: Set DEM [1:0] =11, DEMCOFM=1, the OCP is now in the comparator calibration status.
- Step2: Set DEMCOF [4:0] =00000 then read the DEMCX bit status.
- Step3: Let DEMCOF=DEMCOF+1 then read the DEMCX bit status; if DEMCX is changed, record the register data as VOS1.
- Step4: Set DEMCOF [4:0] =11111 then read the DEMCX bit status.
- Step5: Let DEMCOF=DEMCOF-1 then read the DEMCX bit status; if DEMCX data is changed, record the register data as VOS2.
- Step6: Restore VOS = (VOS1 + VOS2)/2 to the DEMCOF register. The calibration is now finished.

## Demodulator Register Description

The DEMC0 and DEMC1registers are demodulator control registers which control the demodulator operation mode, PGA and filter functions. The DEMREF register is used to provide the reference voltages for the demodulator. DEMACAL and DEMCCAL are used to cancel out the operational amplifier and comparator input offset.

### DEMC0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | DEM1 | DEM0 | — | — | DMS3 | DMS2 | DMS1 | DMS0 |
| R/W | R/W | R/W | — | — | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | — | — | 0 | 0 | 0 | 0 |

Bit 7~6 **DEM[1:0]**: Mode selection
00: Demodulation function disable, S1, S3 on , S0,S2 off
01: Demodulation function enable in non-Inverter mode, S0, S3 on , S1,S2 off
10: Demodulation function enable in Inverter mode , S1, S2 on , S0,S3 off
11: Demodulation function enable in 0V input mode, S1, S3 on , S0,S2 off
Note: disable means OPA, CMP, DAC, Filter all off & CMP output=low.

Bit 5~4 Unimplemented, read as 0

Bit 3 **DMS3:** demodulation switch 3 control (DMS3)
0: Off (disable 8-bit DAC if DMS3=1)
1: On

Bit 2 **DMS2:** demodulation switch 2 control (DMS2)
0: Off
1: On

Bit 1 **DMS1:** demodulation switch 1 control (DMS1)
0: Off
1: On

Bit 0 **DMS0:** demodulation switch 0 control (DMS0)
0: Off
1: On

### DEMC1 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | DEMO | — | DEMG2 | DEMG1 | DEMG0 | DEMFLT2 | DEMFLT1 | DEMFLT0 |
| R/W | R | — | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | — | 0 | 0 | 0 | 0 | 0 | 0 |

bit 7 **DEMO :** DEMO output (read only)

Bit 6 Unimplemented, read as 0

Bit 5~3 **DEMG2~DEMG0:** OPA Gain

Inverter mode:

000: -1
001: -5
010: -10
011: -15
100: -20
101: -30
110: -40
111: -50

Non-inverter mode:

000: 1
001: 6
010: 11
011: 16
100: 21
101: 31
110: 41
111: 51

bit 2~0 **DEMFLT2~DEMFLT0:** Demodulator filter selection

000: 0 $t_{FLT}$ ( without filter)
001: 1~2 × $t_{FLT}$
010: 3~4 × $t_{FLT}$
011: 7~8 × $t_{FLT}$
100: 15~16 × $t_{FLT}$
101: 31~32 × $t_{FLT}$
110: 63~64 × $t_{FLT}$
111: 127~128 × $t_{FLT}$

Note: $t_{FLT} = f_H/4$, $f_H=f_{HXT}$ (crystal) , $t_{FLT} =1/f_{FLT}$

### DEMREF 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 **DEMREF:** Select reference voltage for over current protect

Reference Voltage= (D/A reference voltage/256) ×(N) , N=DEMR[7:0]

### DEMACAL Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | DEMAOFM | DEMARS | DEMAOF5 | DEMAOF4 | DEMAOF3 | DEMAOF2 | DEMAOF1 | DEMAOF0 |
| 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 **DEMAOFM:** Input offset voltage cancellation mode or normal operating mode selection
0: Normal operating mode
1: Input offset voltage cancellation mode

Bit 6 **DEMARS:** Input offset voltage cancellation reference selection bit
0: Select negative input as the reference input
1: Select positive input as the reference input

Bit 5~0 **DEMAOF5~DEMAOF0:** Input offset voltage calibration control

### DEMCCAL Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | DEMAXCX | DEMCOFM | DEMCRS | DEMCOF4 | DEMCOF3 | DEMCOF2 | DEMCOF1 | DEMCOF0 |
| R/W | R | 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 **DEMAXCX:** OPA/Comparator output for calibration; positive logic (read only)
If DEMAOFM=1, this bit is the OPA output for calibration
If DEMCOFM=1, this bit is the comparator output for calibrationr
Note: DEMAOFM and DEMCOFM can't be 1 simultaneously.

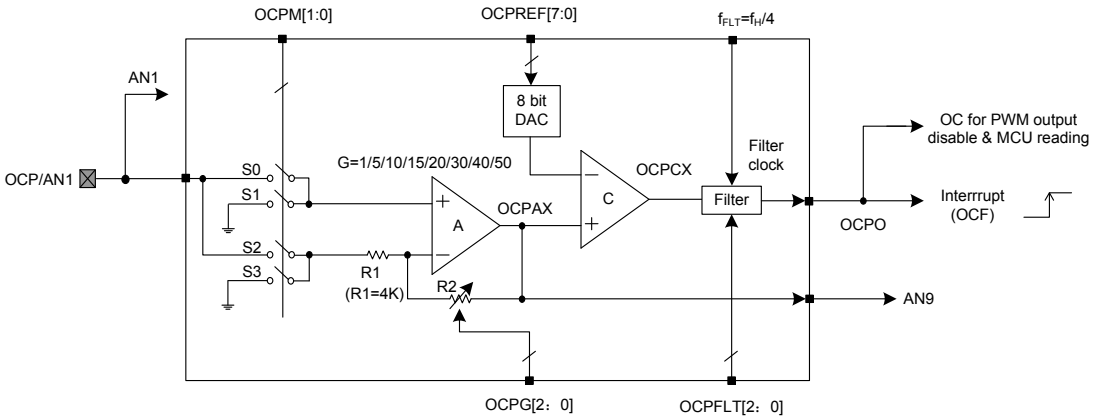
Bit 6 **DEMCOFM:** Input offset voltage cancellation mode or normal operating mode selection
0: Normal operating mode
1: Input offset voltage cancellation mode

Bit 5 **DEMCRS:** Input offset voltage cancellation reference selection bit
0 : Select negative input as the reference input
1 : Select positive input as the reference input

bit 4~0 **DEMCOF4 ~ DEMCOF0:** Input offset voltage calibration control

## OCP Function

OCP is an abbreviation for over current protection. The OCP detects an input voltage which is proportional to the monitored source current. If the input voltage is larger than the reference voltage set by the DAC, the OCP will generate an output signal to indicate that the source current is outwith the specification.




**Over Current Protect Circuit**

### OCP Circuit Operation

The source voltage is sourced from the OCP. Four switches S0~S3 form of a mode select function. An operational amplifier and two resistors form a PGA function. The PGA gain can be positive or negative determined by the input voltage connected to the positive input or negative input of the PGA. The OCPREF is used to generate a reference voltage. The comparator compares the reference voltage and the amplified output voltage. Finally the comparator output is filtered to generate the OCPO output to disable the PWM output and the OCP interrupt trigger. These are debounced versions of DEMCX which are used to indicate whether the source current is outwith the specification or not.

Note that the filter clock; $f_{HXT}$ is the HXT clock. The amplified output voltage also can be read out by means of another ADC from OCPAX. The DAC output voltage is controlled by the OCPREF register and the DAC output is defined as

$$\text{DAC } V_{OUT} = (\text{DAC } V_{REF}/256) \times \text{OCPREF}[7:0] \qquad (1)$$

### Input Voltage Range

The input voltage can be positive or negative, which together with the PGA operating mode, provides for a more flexible application.

(1) If $V_{IN} > 0$ and the PGA operates in the non-inverting mode, the output voltage of the PGA is

$$VO_{PGA} = (1 + R_2 / R_1) \times V_{IN} \qquad (2)$$

(2) When the PGA operates in the non-inverter mode, it also provides a unity gain buffer function.

If OCPM[1:0]=01 and OCPG[2:0]=000, the PGA gain will be 1 and is configured as a unity gain buffer. Switches S2 and S3 will be open internally and the output voltage of the PGA is

$$VO_{PGA} = V_{IN} \qquad (3)$$

(3) If $0 > V_{IN} > -0.7V$ and the PGA operates in inverter mode, the output voltage of the PGA is

$$VO_{PGA} = -(R_2 / R_1) \times V_{IN} \qquad (4)$$

Note: if $V_{IN}$ is negative, it should not be lower than -0.7V to avoid leakage current.

### Offset Calibration

The OCP circuit has 4 operating mode controlled by OCPM1~OCPM0. One of these modes is the calibration mode. In the calibration mode, the OP and comparator offset can be calibrated.

OPAMP calibration:

- Step1: Set OCPM[1:0] =11, OCPAOFM=1, the OCP is now in the OPAMP calibration mode.
- Step2: Set OCPAOF4~OCPAOF0 =00000 then read the OCPAX bit status.
- Step3: Let OCPAOF=OCPAOF+1 then read the OCPAX bit status; if OCPAX is changed, record the register data as VOS1.
- Step4: Set OCPAOF [4:0] 111111 then read the OCPAX bit status.
- Step5: Let OCPAOF=OCPAOF-1 then read the OCPAX bit status; if OCPAX is changed, record the register data as VOS2.
- Step6: Restore VOS = (VOS1 + VOS2)/2 to the OCPAOF register. The calibration is now finished.

Comparator calibration:

- Step1: Set OCPM[1:0] =11, OCPCOFM=1, the OCP is now in the comparator calibration mode
- Step2: Set OCPCOF [4:0] =00000 then read the OCPCX bit status.
- Step3: Let OCPCOF=OCPCOF+1 then read the OCPCX bit status; if OCPCX is changed, record the register data as VOS1.
- Step4: Set OCPCOF [4:0] =11111 then read the OCPCX bit status.
- Step5: Let OCPCOF=OCPCOF-1 then read the OCPCX bit status; if OCPCX data is changed, record the register data as VOS2.
- Step6: Restore VOS = (VOS1 + VOS2)/2 to the OCPCOF register. The calibration is now finished.

### OCP Register Description

The OCP0 and OCP1 registers are the OCP control registers which control the OCP operation mode, PGA and filter functions. The OCPREF register is used to provide the reference voltages for the over current protection. OCPACAL and OCPCCAL are used to cancel out the operational amplifier and comparator input offset.

#### OCPC0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | OCPM1 | OCPM0 | — | — | — | — | — | — |
| R/W | R/W | R/W | — | — | — | — | — | — |
| POR | 0 | 0 | — | — | — | — | — | — |

Bit 7~6 **OCPM1~OCPM0:** Mode selection
00: OCP function disable, S1, S3 on , S0,S2 off
01: OCP function enable in non-Inverter mode, S0, S3 on , S1,S2 off
10: OCP function enable in Inverter mode , S1, S2 on , S0,S3 off
11: OCP function enable in 0V input mode, S1, S3 on , S0,S2 off
Note: Disable means OPA, CMP, DAC, Filter all off & comparator output=low.

Bit 5~0 Unimplemented, read as 0

### OCPC1 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | OCPO | — | OCPG2 | OCPG1 | OCPG0 | OCPFLT2 | OCPFLT1 | OCPFLT0 |
| R/W | R | — | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | — | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7 **OCPO :** OCPO output (read only)

Bit 6 Unimplemented reead as 0

Bit 5~3 **OCPG2~OCPG0:** OPA Gain

Inverter mode:

000: -1
001: -5
010: -10
011: -15
100: -20
101: -30
110: -40
111: -50

Non-inverter mode:

000: 1
001: 6
010: 11
011: 16
100: 21
101: 31
110: 41
111: 51

bit 2~0 **OCPFLT2~OCPFLT0:** Demodulator Filter Selection

000: 0 $t_{FLT}$ ( without filter)
001: 1~2 × $t_{FLT}$
010: 3~4 × $t_{FLT}$
011: 7~8 × $t_{FLT}$
100: 15~16 × $t_{FLT}$
101: 31~32 × $t_{FLT}$
110: 63~64 × $t_{FLT}$
111: 127~128 × $t_{FLT}$

Note: $f_{FLT}$= $f_H$/4, $f_H$=$f_{HXT}$ (crystal) , $t_{FLT}$ =1/$f_{FLT}$

### OCPREF 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 5~0 **OCPREF:** Select reference voltage for over current protect

Reference voltage= (DAC reference voltage /256) × (N) , N=OCPREF[7:0]

### OCPACAL Register – Over Current OPA Calibration Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | OCPAOFM | OCPARS | OCPAOF5 | OCPAOF4 | OCPAOF3 | OCPAOF2 | OCPAOF1 | OCPAOF0 |
| 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 |

Bit7 **OCPAOFM:** Input offset voltage cancellation mode or normal operating mode selection
0: Normal operating mode
1: Input offset voltage cancellation mode

Bit6 **OCPARS:** Input offset voltage cancellation reference selection bit
0: Select negative input as the reference input
1: Select positive input as the reference input

Bit 5~0 **OCPAOF4~OCPAOF0 :** Input offset voltage calibration control

### OCPCCAL Register – Over Current Comparator Calibration Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | OCPAXCX | OCPCOFM | OCPCRS | OCPCOF4 | OCPCOF3 | OCPCOF2 | OCPCOF1 | OCPCOF0 |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit7 **OCPAXCX:** OPA/Comparator output for calibration; positive logic (read only)
If OCPAOFM=1, this bit is the OPA output for calibration
If OCPCOFM=1, this bit is the comparator output for calibration
Note: OCPAOFM and OCPCOFM can't be 1 simultaneously.

Bit6 **OCPCOFM:** Input offset voltage cancellation mode or normal operating mode selection
0: Normal operating mode
1: Input offset voltage cancellation mode

Bit5 **OCPCRS:** Input offset voltage cancellation reference selection bit
0: Select negative input as the reference input
1: Select positive input as the reference input

Bit 4~0 **OCPCOF4~OCPCOF0 :** Input offset voltage calibration control

### DCMISC Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | DEMDAVR | OCDAVR | DEMO | — | — | — | AIS1 | AIS0 |
| R/W | R/W | R/W | R | — | — | — | R/W | R/W |
| POR | 0 | 0 | 0 | — | — | — | 0 | 0 |

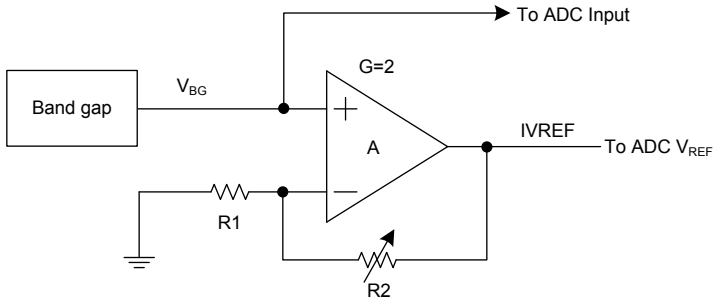
Bit 7 **DEMDAVR :** Demodulation DAC VREF source
0: AVDD
1: Internal VREF

Bit 6 **OCDAVR :** OCP DAC VREF source
0: AVDD
1: Internal VREF

Bit 5 **DEMO :** Demodulation output(read only)

Bit 1~0 **AIS1~AIS0:** Demodulation OPA input selection
00: COMM0
01: COMM1
10: COMM2
11: COMM2

## Internal Reference Voltage – IVREF




**Internal Reference Voltage Circuit**

The bandgap circuit will automatically switch on if either the LVR or LVD is enabled or if the VREF is enabled.

### Demodulator & OCP Miscellaneous Control Register Description

The DCMISC register is used to control the D/A VREF source selection, to store the operational amplifier output status as a logical condition and to select the demodulator OPA input source. The VREFC register is used to control VREF and to store the VREF OPA output status. VRACAL is the VREF OPA calibration register.

#### VREFC register

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

Bit 7 **VREFEN:** VREF enable bit
0: Disable
1: Enable
Note: The bandgap will be enabled if either the LVR, LVD, VBG (ADC) or VREF is enabled.

Bit 0 **VRAX:** VREF OPA output, read only.

#### VRACAL Register – VREF OPA calibration register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | VRAOFM | VRARS | VRAOF5 | VRAOF4 | VRAOF3 | VRAOF2 | VRAOF1 | VRAOF0 |
| 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 |

Bit7 **VRAOFM:** Input offset voltage cancellation mode or normal operating mode selection
0: Normal operating mode
1: Input offset voltage cancellation mode

Bit6 **VRARS:** Input offset voltage cancellation reference selection bit
0: Select negative input as the reference input
1: Select positive input as the reference input

Bit 5~0 **VRAOF5~VRAOF0:** Comparator input offset voltage calibration control

# 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 interrupt is generated by the action of the external INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, Demodulation, OCP, Time Base, LVD, EEPROM 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 number of registers depends upon the device chosen but fall into three categories. The first is the INTC0~INTC2 registers which setup the primary interrupts, the second is the MFI0~MFI2 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 registers 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 Pin | INTnE | INTnF | n=1 |
| OCP | OCPE | OCPF | — |
| Demodulation | DEME | DEMF | — |
| A/D Converter | ADE | ADF | — |
| Multi-function | MFnE | MFnF | n=0~2 |
| Time Base | TBnE | TBnF | n=0 or 1 |
| I²C | IICE | IICF | — |
| LVD | LVE | LVF | — |
| EEPROM | DEE | DEF | — |
| TM | TnPE | TnPF | n=0 or 1 |
| | TnAE | TnAF | |

**Interrupt Register Bit Naming Conventions**

## Interrupt Register Contents

| Name | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| INTEG | — | — | — | — | INT1S1 | INT1S0 | PWMS1 | PWMS0 |
| INTC0 | — | PWMSF | DEMF | OCPF | PWMSE | DEME | OCPE | EMI |
| INTC1 | ADF | MF2F | MF1F | MF0F | ADE | MF2E | MF1E | MF0E |
| INTC2 | INT1F | TB1F | TB0F | IICF | INT1E | TB1E | TB0E | IICE |
| MFI0 | — | — | T0AF | T0PF | — | — | T0AE | T0PE |
| MFI1 | — | — | T1AF | T1PF | — | — | T1AE | T1PE |
| MFI2 | — | — | DEF | LVF | — | — | DEE | LVE |

### INTEG Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | INT1S1 | INT1S0 | PWMS1 | PWMS0 |
| 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 **PWMS1, PWMS0:** Defines PWM synchronization signal interrupt active edge
00: Disabled Interrupt
01: Rising Edge Interrupt
10: Falling Edge Interrupt (must be fixed at this setting)
11: Dual Edge Interrupt
Note: the PWMS[1:0] field must be fixed at "10" if the PWMS interrupt is required. When this bit field is equal to "01" or "11", the function will not be garanteed.

### INTC0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | PWMSF | DEMF | OCPF | PWMSE | DEME | OCPE | 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 **PWMSF:** PWM synchronization signal Interrupt Request Flag
0: No request
1: Interrupt request

Bit 5 **DEMF:** Demodulation interrupt request flag
0: No request
1: Interrupt request

Bit 4 **OCPF:** over current protection interrupt request flag
0: No request
1: Interrupt request

Bit 3 **PWMSE:** PWM synchronization signal Interrupt Control
0: Disable
1: Enable

Bit 2 **DEME:** Demodulation Interrupt Control
0: Disable
1: Enable

Bit 1 **OCPE:** Over current protection 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 | ADF | MF2F | MF1F | MF0F | ADE | MF2E | MF1E | MF0E |
| 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 **ADF :** A/D Converter Interrupt Request Flag
0: No request
1: Interrupt request

Bit 6 **MF2F:** Multi-function Interrupt 2 Request Flag
0: No request
1: Interrupt request

Bit 5 **MF1F:** Multi-function Interrupt 1 Request Flag
0: No request
1: Interrupt request

Bit 4 **MF0F:** Multi-function Interrupt 0 Request Flag
0: No request
1: Interrupt request

Bit 3 **ADE :** A/D Converter Interrupt Control
0: Disable
1: Enable

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

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

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

### INTC2 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | INT1F | TB1F | TB0F | IICF | INT1E | TB1E | TB0E | IICE |
| 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 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 **IICF:** $I^2C$ Interrupt Request Flag
0: No request
1: Interrupt request

Bit 3 **INT1E:** INT1 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 **IICE:** $I^2C$ 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 | — | — | DEF | LVF | — | — | DEE | LVE |
| R/W | — | — | R/W | R/W | — | — | R/W | R/W |
| POR | — | — | 0 | 0 | — | — | 0 | 0 |

Bit 7 ~ 6 Unimplemented, read as "0"
Bit 5 **DEF:** Data EEPROM interrupt request flag
0: No request
1: Interrupt request
Bit 4 **LVF:** LVD interrupt request flag
0: No request
1: Interrupt request
Bit 3 ~ 2 Unimplemented, read as "0"
Bit 1 **DEE:** Data EEPROM Interrupt Control
0: Disable
1: Enable
Bit 0 **LVE:** LVD Interrupt Control
0: Disable
1: Enable

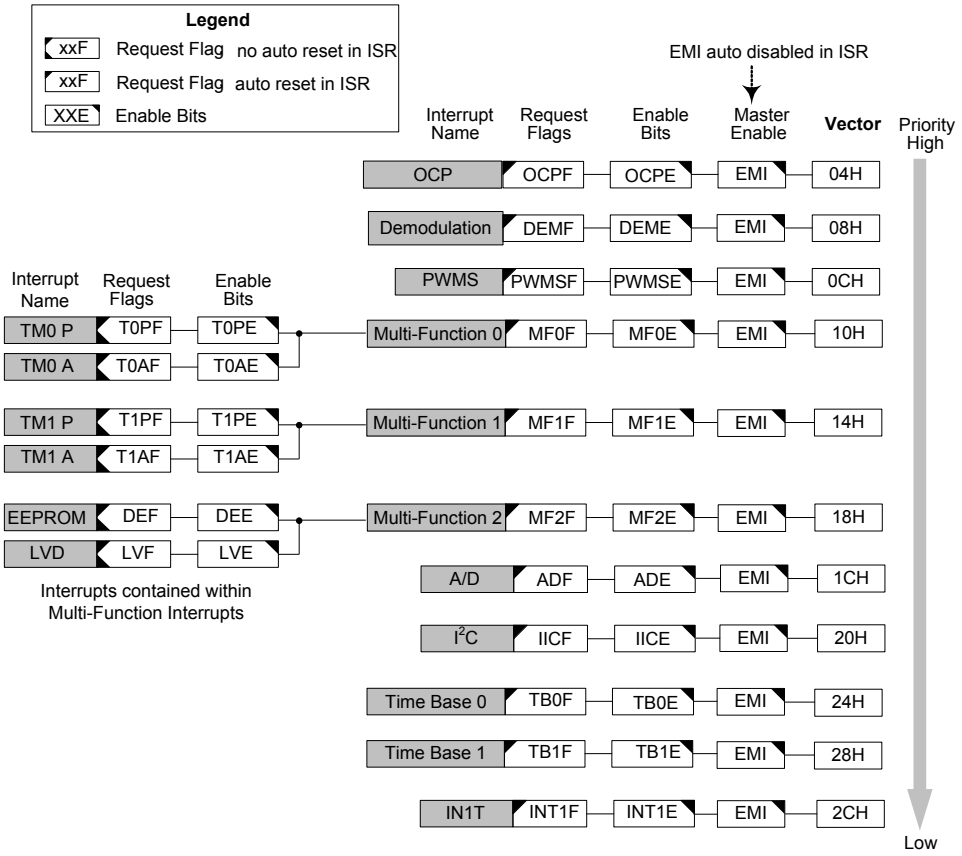
## Interrupt Operation

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 pin INT1. An external interrupt request will take place when the external interrupt request flag, 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, 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 flag, 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.

### Multi-function Interrupt

Within this device there are up to three 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, LVD interrupt and EEPROM Interrupt.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags, MF0F~MF2F 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, LVD interrupt and EEPROM Interrupt will not be automatically reset and must be manually reset by the application program.

### OCP Interrupt

The OCP Interrupt is controlled by detecting a huge current. An OCP Interrupt request will take place when the OCP Interrupt request flag, OCPF, is set, which occurs when a huge current is detected. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and OCP Interrupt enable bit, OCPE, must first be set. When the interrupt is enabled, the stack is not full and a huge current is detected, a subroutine call to the OCP Interrupt vector, will take place. When the interrupt is serviced, the OCP Interrupt flag, OCPF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

### Demodulation Interrupt

The Demodulation Interrupt is controlled by the demodulation process. A demodulation Interrupt request will take place when the Demodulation Interrupt request flag, DEMF, is set, which occurs when the Demodulation process finishes. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and Demodulation Interrupt enable bit, DEME, must first be set. When the interrupt is enabled, the stack is not full and Demodulation process has ended, a subroutine call to the Demodulation Interrupt vector, will take place. When the interrupt is serviced, the Demodulation Interrupt flag, DEMF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

### 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 prescaler, the prescaltion ratio of which is selected by programming the appropriate bits in the PSCR 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 $f_{SYS}$, $f_{SYS}/4$ or $f_{SUB}$ by setting CLKSEL1~CLKSEL0 bits in the PSCR register.

### TBC0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TB0EN | — | — | — | — | TB02 | TB01 | TB00 |
| R/W | R/W | — | — | — | — | R/W | R/W | R/W |
| POR | 0 | — | — | — | — | 0 | 0 | 0 |

Bit 7 **TB0EN:** TB0 Control bit
0: Disable
1: Enable

Bit 6~3 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: $32768/f_{TB}$

### TBC1 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TB1EN | — | — | — | — | TB12 | TB11 | TB10 |
| R/W | R/W | — | — | — | — | R/W | R/W | R/W |
| POR | 0 | — | — | — | — | 0 | 0 | 0 |

Bit 7 **TB1EN:** TB1 Control bit
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

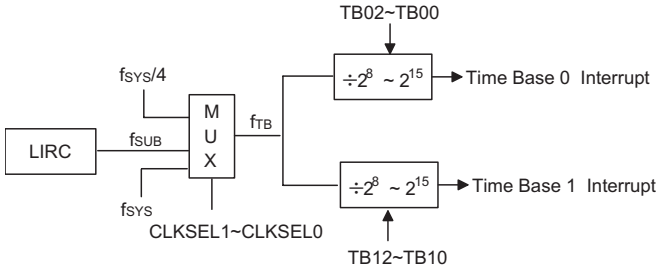
Bit 2 ~ 0 **TB12 ~ TB10:** Select Time Base 1 Time-out Period
000: $256/f_{TB}$
001: $512/f_{TB}$
010: $1024/f_{TB}$
011: $2048/f_{TB}$
100: $4096/f_{TB}$
101: $8192/f_{TB}$
110: $16384/f_{TB}$
111: $32768/f_{TB}$

**PSCR Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | CLKSEL1 | CLKSEL0 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

Bit 7~2 Unimplemented, read as "0"

Bit 1 ~ 0 **CLKSEL1 ~ CLKSEL0:** $f_{TB}$ clock source selection
00: $f_{SYS}$
01: $f_{SYS}/4$
11: $f_{SUB}$




**Time Base Interrupt**

## EEPROM Interrupt

The EEPROM interrupt is contained within the Multi-function Interrupt. An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Interrupt enable bit, DEE, and associated Multi-function interrupt enable bit, MF2E, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Write cycle ends, a subroutine call to the respective EEPROM Interrupt vector, will take place. When the EEPROM 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 DEF flag will not be automatically cleared, it has to be cleared by the application program.

## 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, MF2E, 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 Standard Type TM and Compact Type TM have two interrupts each. All of the TM interrupts are contained within the Multi-function Interrupts. For each of the Standard and Compact Type 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 comparator A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the respective TM Interrupt enable bit, and associated Multi-function interrupt enable bit, MFnF (MF0F or MF1F), 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 TM 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 (MF0F or MF1F) will be automatically cleared. As the TM interrupt request flags will not be automatically cleared, they have to be cleared by the application program.

### I²C Interrupt

An I²C Interrupt request will take place when the I²C Interrupt request flag, IICF, is set, which occurs when a byte of data has been received or transmitted by the I²C interface. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, IICE, must first be set. When the interrupt is enabled, the stack is not full and a byte of data has been transmitted or received by the I²C interface, a subroutine call to the respective Interrupt vector, will take place. When the I²C Interface Interrupt is serviced, the interrupt request flag, IICF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

### 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 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, MF0F~MF2F, 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 determined. 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

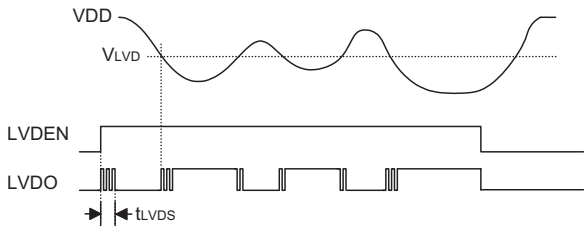
Bit 4 **LVDEN:** Low Voltage Detector Control
0: Disable
1: Enable

Bit 3 Unimplemented, read as "0"

Bit 2~0 **VLVD2 ~ VLVD0:** Select LVD Voltage
000: 2.0V
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 2.0V 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.

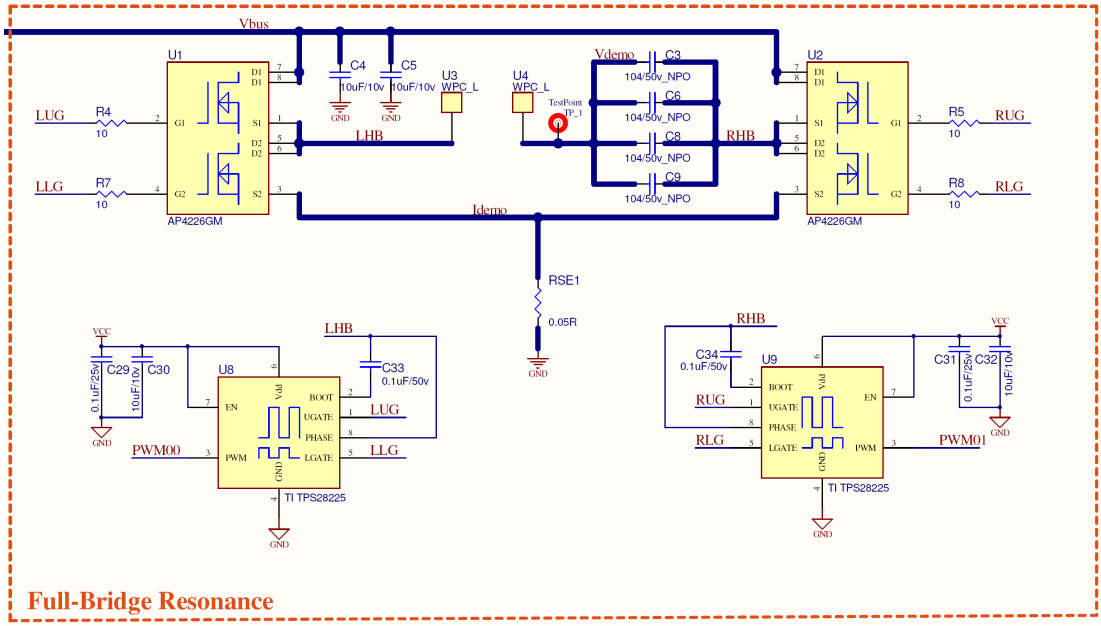


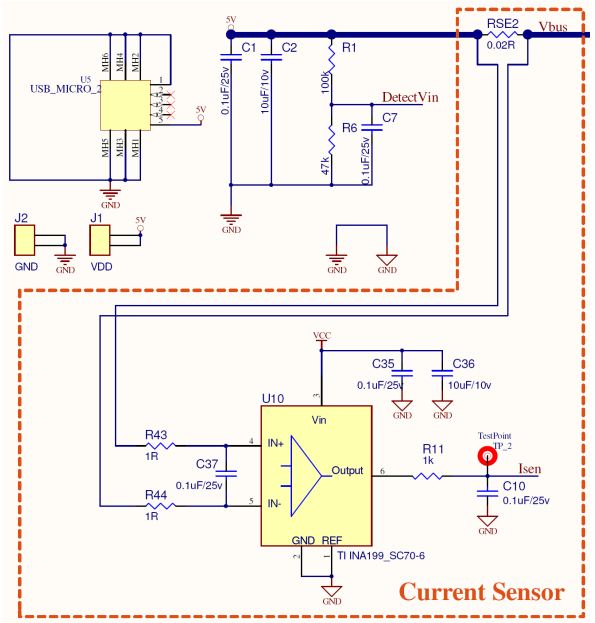

**LVD Operation**

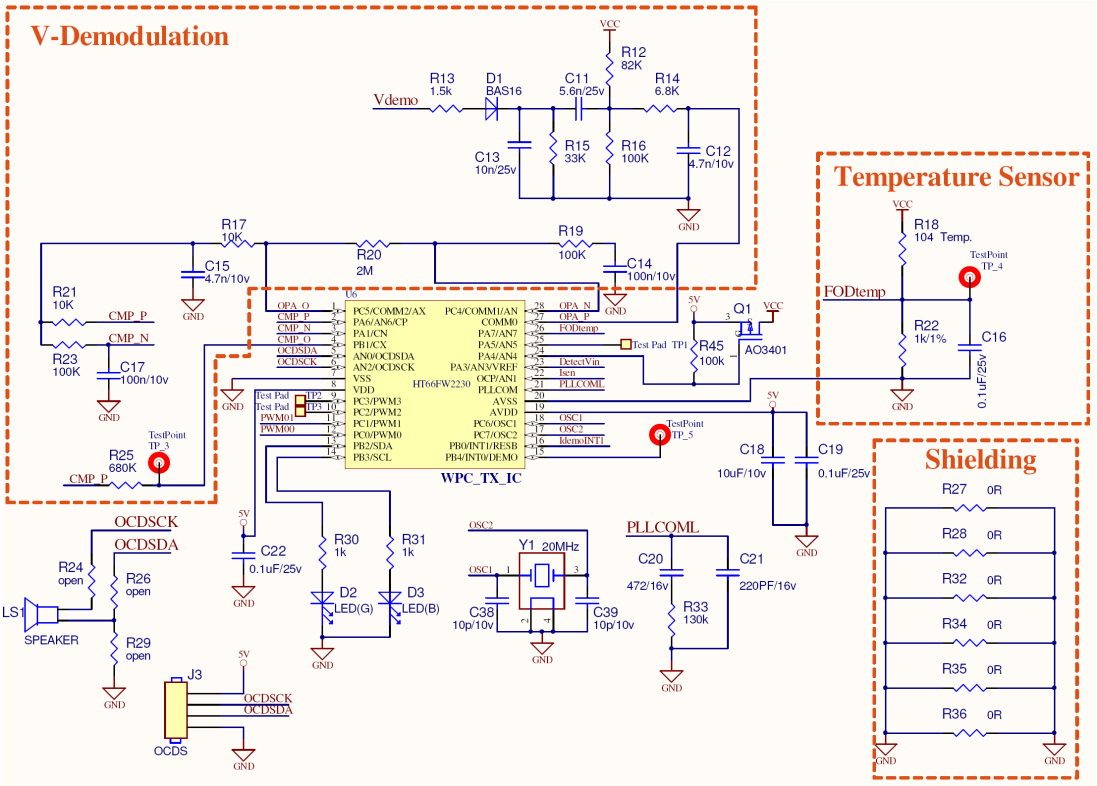
The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of $t_{LVD}$ after the LVDO bit has been 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 VDD 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.

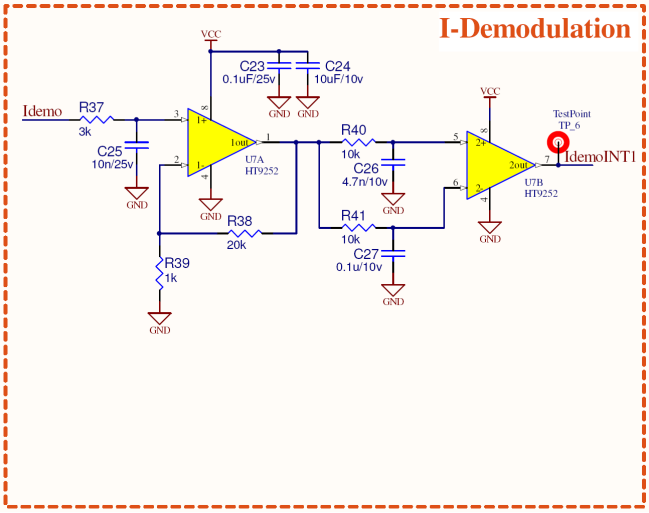
## Application Circuits

### WPC Type A11 Transmitter




V-Demodulation
Temperature Sensor
Shielding
WPC_TX_IC
HT66FW2230
Vdemo
FODtemp
SPEAKER
OCDS
OCDSCK
OCDSDA
PLLCOML
IdemoINT1
R12 82K
R13 1.5k
D1 BAS16
C11 5.6n/25v
R14 6.8K
C13 10n/25v
R15 33K
R16 100K
C12 4.7n/10v
R17 10K
R19 100K
R20 2M
C15 4.7n/10v
C14 100n/10v
R21 10K
R23 100K
C17 100n/10v
R25 680K
R18 104 Temp.
R22 1k/1%
C16 0.1uF/25v
Q1 AO3401
R45 100k
C18 10uF/10v
C19 0.1uF/25v
R27 0R
R28 0R
R32 0R
R34 0R
R35 0R
R36 0R
C22 0.1uF/25v
R30 1k
R31 1k
D2 LED(G)
D3 LED(B)
R24 open
R26 open
R29 open
Y1 20MHz
C38 10p/10v
C39 10p/10v
C20 472/16v
C21 220PF/16v
R33 130k
J3


I-Demodulation
Idemo
R37 3k
C23 0.1uF/25v
C24 10uF/10v
C25 10n/25v
R38 20k
R39 1k
R40 10k
C26 4.7n/10v
R41 10k
C27 0.1u/10v
U7A HT9252
U7B HT9252
IdemoINT1

## Bill of Materials

| Part Name | Designator | Brand | Part Number | Package Type | Quantity |
|---|---|---|---|---|---|
| MCU | U6 | HOLTEK | HT66FW2230 | SSOP28/QFN28 | 1 |
| OPA | U7 | HOLTEK | HT9252 | SOP8 | 1 |
| GATE DRIVER | U8, U9 | TI | TI TPS28225 | SOP8 | 2 |
| OPA | U10 | TI | TI INA199 | SC70 | 1 |
| N-MOSFET | U1, U2 | Advance Power | AP4226GM | SOP8 | 2 |
| INDUCTOR | U3, U4 | GOTREND | GW505006PT-08B10RTN | | 1 |
| USB_MICRO | U5 | FCI | #10103592-0001LF (B type) | | 1 |
| CRYSTAL | Y1 | Abracon Crystal | ABM8G-20.000MHZ-4Y-T3 | | 1 |
| CAPACITOR | C1, C7, C10, C16, C19, C22, C23, C29, C31, C35, C37 | Murata / Walsin | 0.1uF/25V | SMD 0603 | 11 |
| CAPACITOR | C2, C4, C5, C18, C24, C30, C32, C36 | Murata / Walsin | 10uF/10V | SMD 0805 | 8 |
| CAPACITOR | C3, C6, C8, C9 | Murata | 104/50V, NPO, GRM31C5C1H104JA01L | SMD 1206 | 4 |
| CAPACITOR | C11 | Murata / Walsin | 5.6nF/25V | SMD 0603 | 1 |
| CAPACITOR | C12, C15, C26 | Murata / Walsin | 4.7nF/10V | SMD 0603 | 3 |
| CAPACITOR | C13, C25 | Murata / Walsin | 10nF/25V | SMD 0603 | 2 |
| CAPACITOR | C14, C17 | Murata / Walsin | 100nF/10V | SMD 0603 | 2 |
| CAPACITOR | C20 | Murata / Walsin | 472/16V | SMD 0603 | 1 |
| CAPACITOR | C21 | Murata / Walsin | 220pF/16V | SMD 0603 | 1 |
| CAPACITOR | C27 | Murata / Walsin | 0.1u/10V | SMD 0603 | 1 |
| CAPACITOR | C33, C34 | Murata | 0.1uF/50V, X7R ±10% | SMD 0805 | 2 |
| CAPACITOR | C38, C39 | Murata / Walsin | 10pF/10V | SMD 0603 | 2 |
| DIODE | D1 | DIODES | BAV21WS-7-F 250V 200mA | SOD 323 | 1 |
| LED | D2 | EVERLIGHT | LED (GREEN) | SMD 0603 | 1 |
| LED | D3 | EVERLIGHT | LED (BLUE) | SMD 0603 | 1 |
| CONNECTOR | J1 | -- -- | CON2 (VDD) | | 1 |
| CONNECTOR | J2 | -- -- | CON2 (VSS) | | 1 |
| CONNECTOR | J3 | -- -- | CON4 (OCDS) | | 1 |
| BUZZER | LS1 | KINGSTATE | KCG0601(BUZZER) | | 1 |
| P-MOSFET | Q1 | ALPHA & OMEGA | AO3401, PMOS, #85-1001-1-ND | SOT-23 | 1 |
| RESISTOR | R1, R16, R19, R23, R45 | LIKET | 100K | SMD 0603 | 5 |
| RESISTOR | R4, R5, R7, R8 | LIKET | 10 | SMD 0603 | 4 |
| RESISTOR | R6 | LIKET | 47K | SMD 0603 | 1 |
| RESISTOR | R11, R30, R31, R39 | LIKET | 1K | SMD 0603 | 4 |
| RESISTOR | R12 | LIKET | 82K | SMD 0603 | 1 |
| RESISTOR | R13 | LIKET | 1.5K | SMD 0603 | 1 |
| RESISTOR | R14 | LIKET | 6.8K | SMD 0603 | 1 |
| RESISTOR | R15 | LIKET | 33K | SMD 0603 | 1 |
| RESISTOR | R17, R21, R40, R41 | LIKET | 10K | SMD 0603 | 4 |
| TEMP. SENSOR | R18 | Semitec | 10K, NTC #954-103AT-4-80025 | SMD 1210 | 1 |
| RESISTOR | R20 | LIKET | 2M | SMD 0603 | 1 |
| RESISTOR | R22 | LIKET | 1K (1%) | SMD 0603 | 1 |
| RESISTOR | R24, R26, R29 | LIKET | Reserved | SMD 0603 | 3 |
| RESISTOR | R25 | LIKET | 680K | SMD 0603 | 1 |
| RESISTOR | R27, R28, R32, R34, R35, R36 | LIKET | 0 | SMD 0603 | 6 |
| RESISTOR | R33 | LIKET | 130K | SMD 0603 | 1 |
| RESISTOR | R37 | LIKET | 3K | SMD 0603 | 1 |
| RESISTOR | R38 | LIKET | 20K | SMD 0603 | 1 |
| RESISTOR | R43, R44 | LIKET | 1R | SMD 0603 | 2 |
| RESISTOR | RSE1 | Ohmite | 0.05R, 1W #MCS1632R050FER | SMD 1210 | 1 |
| RESISTOR | RSE2 | Ohmite | 0.02R, 1W #MCS1632R020FER | SMD 1210 | 1 |

# Instruction Set

## Introduction

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

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

## Instruction Timing

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

## Moving and Transferring Data

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

## Arithmetic Operations

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

### Logical and Rotate Operation

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 which 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 |

| Mnemonic | Description | Cycles | Flag Affected |
|---|---|---|---|
| **Data Move** | | | |
| MOV A,[m] | Move Data Memory to ACC | 1 | None |
| MOV [m],A | Move ACC to Data Memory | 1Note | None |
| MOV A,x | Move immediate data to ACC | 1 | None |
| **Bit Operation** | | | |
| CLR [m].i | Clear bit of Data Memory | 1Note | None |
| SET [m].i | Set bit of Data Memory | 1Note | None |
| **Branch** | | | |
| JMP addr | Jump unconditionally | 2 | None |
| SZ [m] | Skip if Data Memory is zero | 1Note | None |
| SZA [m] | Skip if Data Memory is zero with data movement to ACC | 1Note | None |
| SZ [m].i | Skip if bit i of Data Memory is zero | 1Note | None |
| SNZ [m].i | Skip if bit i of Data Memory is not zero | 1Note | None |
| SIZ [m] | Skip if increment Data Memory is zero | 1Note | None |
| SDZ [m] | Skip if decrement Data Memory is zero | 1Note | None |
| SIZA [m] | Skip if increment Data Memory is zero with result in ACC | 1Note | None |
| SDZA [m] | Skip if decrement Data Memory is zero with result in ACC | 1Note | None |
| CALL addr | Subroutine call | 2 | None |
| RET | Return from subroutine | 2 | None |
| RET A,x | Return from subroutine and load immediate data to ACC | 2 | None |
| RETI | Return from interrupt | 2 | None |
| **Table Read Operation** | | | |
| TABRD [m] | Read table (specific page or 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 up to 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 $[m] \leftarrow ACC + 00H$ or
$[m] \leftarrow ACC + 06H$ or
$[m] \leftarrow ACC + 60H$ or
$[m] \leftarrow ACC + 66H$

Affected flag(s) C

**DEC [m]** Decrement Data Memory

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

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

Affected flag(s) Z

DECA [m] Decrement Data Memory with result in ACC

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

Operation $ACC \leftarrow [m] - 1$

Affected flag(s) Z

**HALT** Enter power down mode

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

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

Affected flag(s) TO, PDF

**INC [m]** Increment Data Memory

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

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

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 $ACC \leftarrow [m] + 1$

Affected flag(s) Z

**JMP addr** Jump unconditionally

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

Operation Program Counter ← addr

Affected flag(s) None

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

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

Operation ACC ← [m]

Affected flag(s) None

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

Description The immediate data specified is loaded into the Accumulator.

Operation ACC ← x

Affected flag(s) None

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

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

Operation [m] ← ACC

Affected flag(s) None

**NOP** No operation

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

Operation No operation

Affected flag(s) None

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

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

Operation ACC ← ACC ″OR″ [m]

Affected flag(s) Z

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

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

Operation ACC ← ACC ″OR″ x

Affected flag(s) Z

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

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

Operation [m] ← ACC ″OR″ [m]

Affected flag(s) Z

**RET** Return from subroutine

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

Operation Program Counter ← Stack

Affected flag(s) None

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

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

Operation Program Counter ← Stack
ACC ← x

Affected flag(s) None

**RETI** Return from interrupt

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

Operation Program Counter ← Stack
EMI ← 1

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s) C

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

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

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

Affected flag(s) C

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

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

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

Affected flag(s) None

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

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

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

Affected flag(s): None

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

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

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

Affected flag(s): C

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

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

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

Affected flag(s): C

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

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

Operation: ACC ← ACC − [m] − $\overline{C}$

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

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

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

Operation: [m] ← ACC − [m] − $\overline{C}$

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

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

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

Operation:
[m] ← [m] − 1
Skip if [m]=0

Affected flag(s): None

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

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

Operation ACC ← [m] − 1
Skip if ACC=0

Affected flag(s) None

**SET [m]** Set Data Memory

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

Operation [m] ← FFH

Affected flag(s) None

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

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

Operation [m].i ← 1

Affected flag(s) None

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

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

Operation [m] ← [m] + 1
Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m] + 1
Skip if ACC=0

Affected flag(s) None

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

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

Operation Skip if [m].i ≠ 0

Affected flag(s) None

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

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

Operation ACC ← ACC − [m]

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

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

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

Operation [m] ← ACC − [m]

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

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

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

Operation ACC ← ACC − x

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

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

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

Operation [m].3~[m].0 ↔ [m].7~[m].4

Affected flag(s) None

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

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

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

Affected flag(s) None

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

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

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

**TTABRD [m]** Read table (specific page or current page) to TBLH and Data Memory

Description The low byte of the program code addressed by the table pointer (TBHP and TBLP or only TBLP if no TBHP) 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

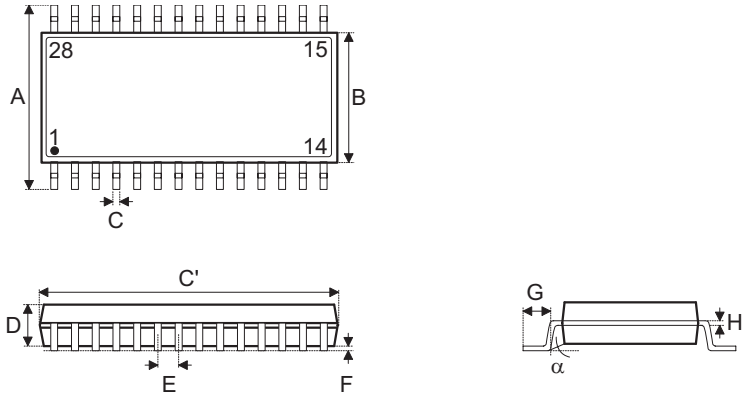
## Package Information

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

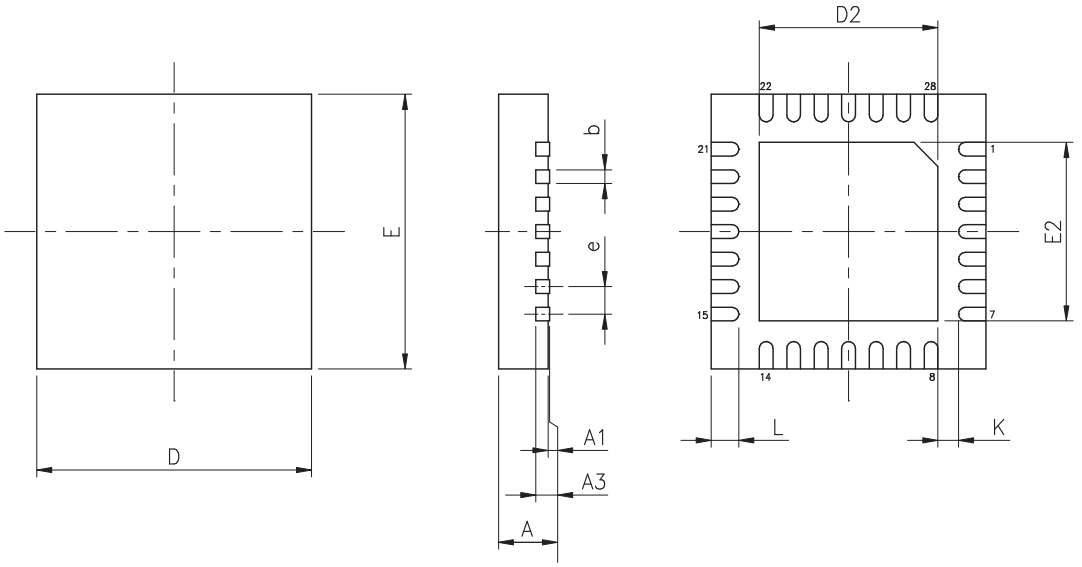
### 28-pin SSOP (150mil) Outline Dimensions




| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | — | 0.236 BSC | — |
| B | — | 0.154 BSC | — |
| C | 0.008 | — | 0.012 |
| C’ | — | 0.390 BSC | — |
| D | — | — | 0.069 |
| E | — | 0.025 BSC | — |
| F | 0.004 | — | 0.0098 |
| G | 0.016 | — | 0.050 |
| H | 0.004 | — | 0.010 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | — | 6.000 BSC | — |
| B | — | 3.900 BSC | — |
| C | 0.20 | — | 0.30 |
| C’ | — | 9.900 BSC | — |
| D | — | — | 1.75 |
| E | — | 0.635 BSC | — |
| F | 0.10 | — | 0.25 |
| G | 0.41 | — | 1.27 |
| H | 0.10 | — | 0.25 |
| α | 0° | — | 8° |

## SAW Type 28-pin QFN (4mm×4mm×0.75mm) Outline Dimensions




| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.028 | 0.030 | 0.031 |
| A1 | 0.000 | 0.001 | 0.002 |
| A3 | — | 0.008 BSC | — |
| b | 0.006 | 0.008 | 0.010 |
| D | — | 0.157 BSC | — |
| E | — | 0.157 BSC | — |
| e | — | 0.016 BSC | — |
| D2 | 0.100 | 0.102 | 0.104 |
| E2 | 0.100 | 0.102 | 0.104 |
| L | 0.012 | 0.016 | 0.020 |
| K | — | — | — |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.70 | 0.75 | 0.80 |
| A1 | 0.00 | 0.02 | 0.05 |
| A3 | — | 0.203 BSC | — |
| b | 0.15 | 0.20 | 0.25 |
| D | — | 4.00 BSC | — |
| E | — | 4.00 BSC | — |
| e | — | 0.40 BSC | — |
| D2 | 2.55 | 2.60 | 2.65 |
| E2 | 2.55 | 2.60 | 2.65 |
| L | 0.30 | 0.40 | 0.50 |
| K | — | — | — |