# 24-Bit Delta Sigma A/D Flash MCU
# Integrated Regulator

# BH66F5233

Revision: V1.50 Date: November 10, 2021

www.holtek.com

# Table of Contents

# Features

## CPU Features

- Operating voltage
  - $f_{SYS}$=4MHz: 2.2V~5.5V
  - $f_{SYS}$=8MHz: 2.2V~5.5V
  - $f_{SYS}$=12MHz: 2.7V~5.5V
- Up to 0.33μs instruction cycle with 12MHz system clock at $V_{DD}$=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator types
  - Internal High Speed RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal 4/8/12MHz oscillator requires no external components
- All instructions executed in one or two instruction cycles
- Table read instructions
- 61 powerful instructions
- 4-level subroutine nesting
- Bit manipulation instruction

## Peripheral Features

- Flash Program Memory: 2K×16
- RAM Data Memory: 96×8
- True EEPROM Memory: 32×8
- Watchdog Timer function
- 14 bidirectional I/O lines
- 2 pin-shared external interrupts
- Single Timer Module for time measurement, compare match output or PWM output function
- Serial Interface Module – SIM for SPI or $I^2C$
- Dual Time-Base functions for generation of fixed time interrupt signals
- Single differential or two single-end external channel 24-bit resolution Delta Sigma A/D converter
- Low voltage reset function
- Low voltage detect function
- Package types: 16/20-pin NSOP

## General Description

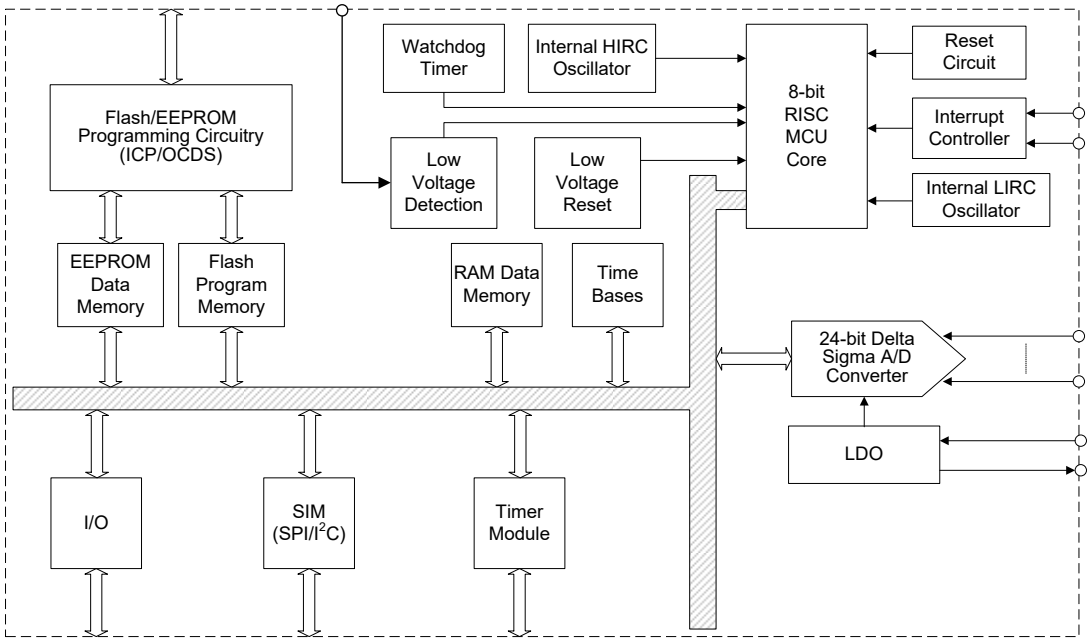
The device is a Flash Memory A/D type 8-bit high performance RISC architecture microcontroller which includes a multi-channel 24-bit Delta Sigma A/D converter designed for applications that interface directly to analog signals and which require a low noise and high accuracy analog to digital converter. 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 features include a multi-channel with 24-bit Delta Sigma A/D converter and programmable gain amplifier (PGA) functions. An extremely flexible Timer Module provides timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated SPI or $I^2C$ interface functions, two popular interfaces which provide designers with a means of easy communication with external peripheral hardware. In addition, an internal LDO function provides various power options to the internal and external devices. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

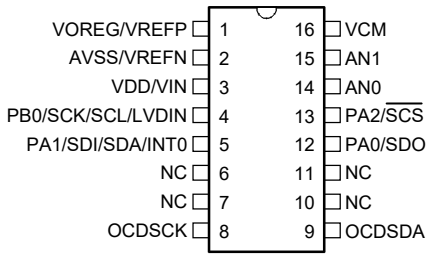
A full choice of internal low and high speed oscillator functions is provided including a fully integrated system oscillator which requires no external components for its implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives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

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

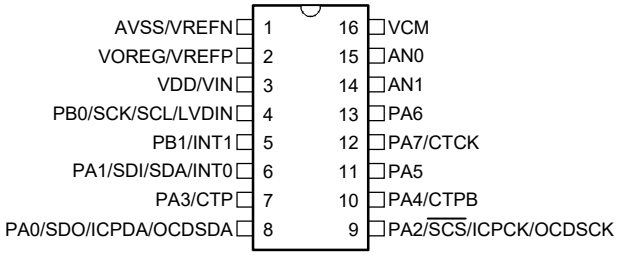
## Block Diagram



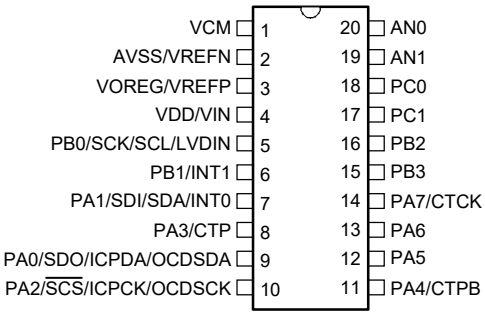
## Pin Assignment




**BH66V5233-10**
**16 NSOP-A**




**BH66F5233/BH66V5233**
**16 NSOP-A**




**BH66F5233/BH66V5233**
**20 NSOP-A**

Note: 1. If the pin-shared pin functions have multiple outputs simultaneously, the desired pin-shared function is determined by the corresponding software control bits.

2. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the BH66V5233 device which is the OCDS EV chip for the BH66F5233 device.

3. For less pin-count package types there will be unbonded pins which should be properly configured to avoid unwanted current consumption resulting from floating input conditions. Refer to the “Standby Current Considerations” and “Input/Output Ports” sections.

## Pin Description

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet.

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| PA0/SDO/ ICPDA/OCDSDA | PA0 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | SDO | PAS0 SIMC2 | — | CMOS | SPI serial data output |
| | ICPDA | — | ST | CMOS | ICP address/data |
| | OCDSDA | — | ST | CMOS | OCDS address/data, for EV chip only. |
| PA1/SDI/SDA/ INT0 | PA1 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | SDI | PAS0 SIMC2 | ST | — | SPI serial data input |
| | SDA | PAS0 SIMC0 | ST | CMOS | $I^2C$ data line |
| | INT0 | INTEG INTC0 | ST | — | External Interrupt 0 input |
| PA2/$\overline{SCS}$/ICPCK/ OCDSCK | PA2 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | $\overline{SCS}$ | PAS0 SIMC2 | ST | CMOS | SPI slave select pin |
| | ICPCK | — | ST | — | ICP clock |
| | OCDSCK | — | ST | — | OCDS clock pin, for EV chip only |
| PA3/CTP | PA3 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | CTP | PAS0 | — | CMOS | CTM output |
| PA4/CTPB | PA4 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | CTPB | PAS1 | — | CMOS | CTM inverting output |
| PA5~PA6 | PA5~PA6 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| PA7/CTCK | PA7 | PAPU PAWU | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | CTCK | — | ST | — | CTM clock input |
| PB0/SCK/SCL/ LVDIN | PB0 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | SCK | PBS0 | ST | CMOS | SPI serial clock |
| | SCL | PBS0 | ST | NMOS | $I^2C$ clock line |
| | LVDIN | PBS0 | AN | — | LVD input |
| PB1/INT1 | PB1 | PBPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | INT1 | INTEG INTC0 | ST | — | External interrupt 1 input |
| PB2~PB3 | PB2~PB3 | PBPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| PC0~PC1 | PC0~PC1 | PCPU | ST | CMOS | General purpose I/O. Register enabled pull-up |
| VOREG/VREFP | VOREG | — | PWR | — | LDO output pin |
| | | | PWR | — | Positive power supply for VCM, A/D converter, PGA |
| | VREFP | — | AN | — | External positive reference input of A/D converter |

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| AVSS/VREFN | AVSS | — | PWR | — | Negative power supply for VCM, A/D converter, PGA |
| | VREFN | — | AN | — | External negative reference input of ADC |
| AN0~AN1 | AN0~AN1 | — | AN | — | A/D Converter input channel |
| VCM | VCM | — | AN | — | External input voltage for A/D Converter Common mode |
| | | | — | AN | A/D Converter Common mode voltage output |
| VDD/VIN | VDD | — | PWR | — | Digital positive power supply |
| | VIN | — | PWR | — | LDO input pin |

Legend: I/T: Input type; O/T: Output type;
OPT: Optional by register option; PWR: Power;
ST: Schmitt Trigger input; CMOS: CMOS output;
NMOS: NMOS output; AN: Analog signal;

## 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
$I_{OL}$ Total ..... 80mA
$I_{OH}$ Total ..... -80mA
Total Power Dissipation ..... 500mW

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

## D.C. Characteristics

For data in the following tables, note that factors such as oscillator type, operating voltage, operating frequency, pin load conditions, temperature and program instruction type, etc., can all exert an influence on the measured values.

### Operating Voltage Characteristics

Ta= -40°C~85°C

| Symbol | Parameter | Test Conditions | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| $V_{DD}$ | Operating Voltage – HIRC | $f_{SYS}$=$f_{HIRC}$=4MHz | 2.2 | — | 5.5 | V |
| | | $f_{SYS}$=$f_{HIRC}$=8MHz | 2.2 | — | 5.5 | |
| | | $f_{SYS}$=$f_{HIRC}$=12MHz | 2.7 | — | 5.5 | |
| | Operating Voltage – LIRC | $f_{SYS}$=$f_{LIRC}$=32kHz | 2.2 | — | 5.5 | V |

### Standby Current Characteristics

Ta=25°C

| Symbol | Standby Mode | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $I_{STB}$ | SLEEP Mode | 3V | WDT off | — | 0.08 | 0.12 | μA |
| | | 5V | | — | 0.15 | 0.29 | |
| | | 3V | WDT on | — | 1.5 | 3.0 | μA |
| | | 5V | | — | 3 | 5 | |
| | IDLE0 Mode | 3V | $f_{SUB}$ on | — | 3 | 5 | μA |
| | | 5V | | — | 5 | 10 | |
| | IDLE1 Mode – HIRC | 3V | $f_{SUB}$ on, $f_{SYS}$=4MHz | — | 0.18 | 0.25 | mA |
| | | 5V | | — | 0.4 | 0.6 | |
| | | 3V | $f_{SUB}$ on, $f_{SYS}$=8MHz | — | 0.36 | 0.50 | mA |
| | | 5V | | — | 0.6 | 0.8 | |
| | | 3V | $f_{SUB}$ on, $f_{SYS}$=12MHz | — | 0.54 | 0.75 | mA |
| | | 5V | | — | 0.8 | 1.2 | |

Notes: When using the characteristic table data, the following notes should be taken into consideration:

- Any digital inputs are setup in a non-floating condition.
- All measurements are taken under conditions of no load and with all peripherals in an off state.
- There are no DC current paths.
- All Standby Current values are taken after a HALT instruction execution thus stopping all instruction execution.

### Operating Current Characteristics

Ta=25°C

| Symbol | Operating Mode | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $I_{DD}$ | Operating Current – LIRC | 3V | $f_{SYS}$=32kHz | — | 10 | 20 | μA |
| | | 5V | | — | 30 | 50 | |
| | Operating Current – HIRC | 3V | $f_{SYS}$=4MHz | — | 0.4 | 0.6 | mA |
| | | 5V | | — | 0.8 | 1.2 | |
| | | 3V | $f_{SYS}$=8MHz | — | 0.8 | 1.2 | mA |
| | | 5V | | — | 1.6 | 2.4 | |
| | | 3V | $f_{SYS}$=12MHz | — | 1.2 | 1.8 | mA |
| | | 5V | | — | 2.4 | 3.6 | |

Notes: When using the characteristic table data, the following notes should be taken into consideration:

- Any digital inputs are setup in a non-floating condition.
- All measurements are taken under conditions of no load and with all peripherals in an off state.
- There are no DC current paths.
- All Operating Current values are measured using a continuous NOP instruction program loop.

## A.C. Characteristics

For data in the following tables, note that factors such as oscillator type, operating voltage, operating frequency and temperature etc., can all exert an influence on the measured values.

### High Speed Internal Oscillator – HIRC – Frequency Accuracy

During the program writing operation the writer will trim the HIRC oscillator at a user selected HIRC frequency and user selected voltage of either 3V or 5V.

**4/8/12MHz**

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Temp. | | | | |
| $f_{HIRC}$ | 4MHz writer trimmed HIRC frequency | 3V/5V | Ta=25°C | -1% | 4 | +1% | MHz |
| | | | Ta= -40°C~85°C | -2% | 4 | +2% | |
| | | 2.2V~5.5V | Ta=25°C | -1.5% | 4 | +1.5% | |
| | | | Ta= -40°C~85°C | -3% | 4 | +3% | |
| | 8MHz writer trimmed HIRC frequency | 3V/5V | Ta=25°C | -1% | 8 | +1% | MHz |
| | | | Ta= -40°C~85°C | -2% | 8 | +2% | |
| | | 2.2V~5.5V | Ta=25°C | -1.5% | 8 | +1.5% | |
| | | | Ta= -40°C~85°C | -3% | 8 | +3% | |
| | 12MHz writer trimmed HIRC frequency | 3V/5V | Ta=25°C | -1% | 12 | +1% | MHz |
| | | | Ta= -40°C~85°C | -2% | 12 | +2% | |
| | | 2.7V~5.5V | Ta=25°C | -1.5% | 12 | +1.5% | |
| | | | Ta= -40°C~85°C | -3% | 12 | +3% | |

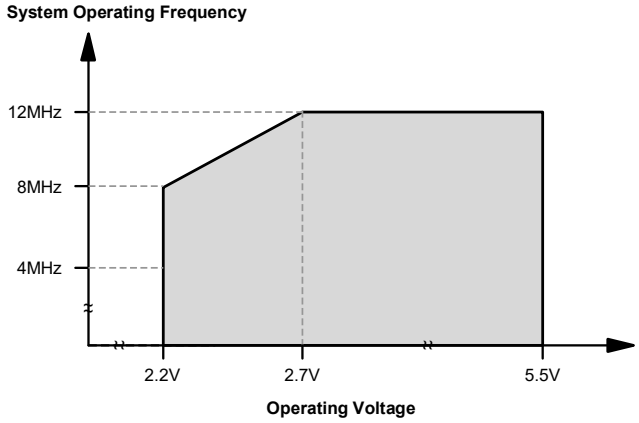
Notes: 1. The 3V/5V values for $V_{DD}$ are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.
2. The row below the 3V/5V trim voltage row is provided to show the values for the full $V_{DD}$ range operating voltage.
3. The minimum and maximum tolerance values provided in the table are only for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will give a frequency tolerance to within ±20%.

### Low Speed Internal Oscillator Characteristics – LIRC

Ta=25°C, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Temp. | | | | |
| $f_{LIRC}$ | LIRC Frequency | 3V | Ta=25°C | -5% | 32 | +5% | kHz |
| | | | Ta= -40°C~85°C | -10% | 32 | +10% | |
| | | 2.2V~5.5V | Ta= -40°C~85°C | -20% | 32 | +20% | |
| Duty Cycle | Duty Cycle | — | — | 10 | — | 90 | % |
| $t_{START}$ | LIRC Start Up Time | 3V/5V | — | — | — | 500 | μs |

## Operating Frequency Characteristic Curves




## System Start Up Time Characteristics

Ta= -40°C~85°C

| Symbol | Parameter | Test Conditions | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| $t_{SST}$ | System Start-up Time Wake-up from Condition where $f_{SYS}$ is off | $f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HIRC}$ | 16 | — | — | $t_{HIRC}$ |
| | | $f_{SYS}=f_{SUB}=f_{LIRC}$ | 2 | — | — | $t_{LIRC}$ |
| | System Start-up Time Wake-up from Condition where $f_{SYS}$ is on | $f_{SYS}=f_H \sim f_H/64$, $f_H=f_{HIRC}$ | 2 | — | — | $t_H$ |
| | | $f_{SYS}=f_{SUB}=f_{LIRC}$ | 2 | — | — | $t_{SUB}$ |
| | System Speed Switch Time FAST to SLOW Mode or SLOW to FAST Mode | $f_{HIRC}$ switches from off → on | 16 | — | — | $t_{HIRC}$ |
| $t_{RSTD}$ | System Reset Delay Time Reset Source from Power-on Reset or LVR Hardware Reset | $RR_{POR}$=5 V/ms | 42 | 48 | 54 | ms |
| | System Reset Delay Time LVRC/WDTC/RSTC Software Reset | — | | | | |
| | System Reset Delay Time Reset Source from WDT Overflow | — | 14 | 16 | 18 | ms |
| $t_{SRESET}$ | Minimum Software Reset Width to Reset | — | 45 | 90 | 120 | μs |

Notes: 1. For the System Start-up time values, whether $f_{SYS}$ is on or off depends upon the mode type and the chosen $f_{SYS}$ system oscillator. Details are provided in the System Operating Modes section.

2. The time units, shown by the symbols $t_{HIRC}$. are the inverse of the corresponding frequency values as provided in the frequency tables. For example $t_{HIRC}=1/f_{HIRC}$, $t_{SYS}=1/f_{SYS}$ etc.

3. If the LIRC is used as the system clock and if it is off when in the SLEEP Mode, then an additional LIRC start up time, $t_{START}$, as provided in the LIRC frequency table, must be added to the $t_{SST}$ time in the table above.

4. The System Speed Switch Time is effectively the time taken for the newly activated oscillator to start up.

## Input/Output Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{IL}$ | Input Low Voltage for I/O Ports | 5V | — | 0 | — | 1.5 | V |
| | | — | — | 0 | — | $0.2V_{DD}$ | |
| $V_{IH}$ | Input High Voltage for I/O Ports | 5V | — | 3.5 | — | 5.0 | V |
| | | — | — | $0.8V_{DD}$ | — | $V_{DD}$ | |
| $V_{OL}$ | Output Low Voltage for I/O Ports | 3V | $I_{OL}$=16mA | — | — | 0.3 | V |
| | | 5V | $I_{OL}$=32mA | — | — | 0.5 | |
| $V_{OH}$ | Output High Voltage for I/O Ports | 3V | $I_{OH}$= -0.7mA, SLEDCn[m+1:m]=00, (n=0, 1, m=0, 2, 4 or 6) | 2.7 | — | — | V |
| | | 5V | $I_{OH}$= -1.5mA, SLEDCn[m+1:m]=00, (n=0, 1, m=0, 2, 4 or 6) | 4.5 | — | — | |
| | | 3V | $I_{OH}$= -1.3mA, SLEDCn[m+1:m]=01, (n=0, 1, m=0, 2, 4 or 6) | 2.7 | — | — | |
| | | 5V | $I_{OH}$= -2.5mA, SLEDCn[m+1:m]=01, (n=0, 1, m=0, 2, 4 or 6) | 4.5 | — | — | |
| | | 3V | $I_{OH}$= -1.8mA, SLEDCn[m+1:m]=10, (n=0, 1, m=0, 2, 4 or 6) | 2.7 | — | — | |
| | | 5V | $I_{OH}$= -3.6mA, SLEDCn[m+1:m]=10, (n=0, 1, m=0, 2, 4 or 6) | 4.5 | — | — | |
| | | 3V | $I_{OH}$= -4mA, SLEDCn[m+1:m]=11, (n=0, 1, m=0, 2, 4 or 6) | 2.7 | — | — | |
| | | 5V | $I_{OH}$= -8mA, SLEDCn[m+1:m]=11, (n=0, 1, m=0, 2, 4 or 6) | 4.5 | — | — | |
| $I_{OL}$ | Sink Current for I/O Pins | 3V | $V_{OL}$=0.1$V_{DD}$ | 16 | 32 | — | mA |
| | | 5V | | 32 | 65 | — | |
| $I_{OH}$ | Source Current for I/O Pins | 3V | $V_{OH}$=0.9$V_{DD}$, SLEDCn[m+1:m]=00, (n=0,1, m=0, 2, 4 or 6) | -0.7 | -1.5 | — | mA |
| | | 5V | | -1.5 | -2.9 | — | |
| | | 3V | $V_{OH}$=0.9$V_{DD}$, SLEDCn[m+1:m]=01, (n=0, 1, m=0, 2, 4 or 6) | -1.3 | -2.5 | — | |
| | | 5V | | -2.5 | -5.1 | — | |
| | | 3V | $V_{OH}$=0.9$V_{DD}$, SLEDCn[m+1:m]=10, (n=0, 1, m=0, 2, 4 or 6) | -1.8 | -3.6 | — | |
| | | 5V | | -3.6 | -7.3 | — | |
| | | 3V | $V_{OH}$=0.9$V_{DD}$, SLEDCn[m+1:m]=11, (n=0, 1, m=0, 2, 4 or 6) | -4 | -8 | — | |
| | | 5V | | -8 | -16 | — | |
| $R_{PH}$ | Pull-high Resistance for I/O Ports(Note) | 3V | — | 20 | 60 | 100 | kΩ |
| | | 5V | — | 10 | 30 | 50 | |
| $I_{LEAK}$ | Input leakage current | 3V | $V_{IN}$=$V_{DD}$ or $V_{IN}$=$V_{SS}$ | — | — | ±1 | μA |
| | | 5V | | — | — | ±1 | |
| $t_{TCK}$ | TM Clock Input Minimum Pulse Width | — | — | 0.3 | — | — | μs |
| $t_{INT}$ | Interrupt Input Pin Minimum Pulse Width | — | — | 10 | — | — | μs |

Note: The $R_{PH}$ internal pull high resistance value is calculated by connecting to ground and enabling the input pin with a pull-high resistor and then measuring the input sink current at the specified supply voltage level. Dividing the voltage by this measured current provides the $R_{PH}$ value.

## Memory Characteristics

Ta= -40°C~85°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{RW}$ | $V_{DD}$ for Read / Write | — | — | $V_{DDmin}$ | — | $V_{DDmax}$ | V |
| **Flash Program / Data EEPROM Memory** | | | | | | | |
| $t_{DEW}$ | Erase / Write cycle time | — | — | 2.2 | 2.5 | 2.8 | ms |
| $I_{DDPGM}$ | Programming / Erase current on $V_{DD}$ | — | — | — | — | 5.0 | mA |
| $E_P$ | Cell Endurance – Flash Program Memory | — | — | 10K | — | — | E/W |
| | Cell Endurance – Data EEPROM Memory | — | — | 100K | — | — | E/W |
| $t_{RETD}$ | ROM Data Retention time | — | Ta=25°C | — | 40 | — | Year |
| **RAM Data Memory** | | | | | | | |
| $V_{DR}$ | RAM Data Retention voltage | — | Device in SLEEP Mode | 1.0 | — | — | V |

## LDO+PGA+ADC+VCM Electrical Characteristics

$V_{DD}=V_{IN}$, Ta=25°C
LDO & VCM Test conditions: MCU enters SLEEP mode, other functions disabled

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{IN}$ | LDO Input Voltage | — | — | 2.6 | — | 5.5 | V |
| $I_Q$ | LDO Quiescent Current | — | LDOVS[1:0]=00B, $V_{IN}$=3.6V, No load | — | 400 | 520 | μA |
| $V_{OUT\_LDO}$ | LDO Output Voltage | — | LDOVS[1:0]=00B, $V_{IN}$=3.6V, $I_{LOAD}$=0.1mA | -5% | 2.4 | +5% | V |
| | | | LDOVS[1:0]=01B, $V_{IN}$=3.6V, $I_{LOAD}$=0.1mA | | 2.6 | | |
| | | | LDOVS[1:0]=10B, $V_{IN}$=3.6V, $I_{LOAD}$=0.1mA | | 2.9 | | |
| | | | LDOVS[1:0]=11B, $V_{IN}$=3.6V, $I_{LOAD}$=0.1mA | | 3.3 | | |
| $\Delta V_{LOAD}$ | LDO Load Regulation(Note 1) | — | LDOVS[1:0]=00B, $V_{IN}=V_{OUT\_LDO}$+0.2V, 0mA≤$I_{LOAD}$≤10mA | — | 0.105 | 0.21 | %/mA |
| $V_{DROP\_LDO}$ | LDO Dropout Voltage(Note 2) | — | LDOVS[1:0]=00B, $V_{IN}$=3.6V, $I_{LOAD}$=10mA, $\Delta V_{OUT\_LDO}$=2% | — | — | 100 | mV |
| | | | LDOVS[1:0]=01B, $V_{IN}$=3.6V, $I_{LOAD}$=10mA, $\Delta V_{OUT\_LDO}$=2% | — | — | 130 | |
| | | | LDOVS[1:0]=10B, $V_{IN}$=3.6V, $I_{LOAD}$=10mA, $\Delta V_{OUT\_LDO}$=2% | — | — | 180 | |
| | | | LDOVS[1:0]=11B, $V_{IN}$=3.6V, $I_{LOAD}$=10mA, $\Delta V_{OUT\_LDO}$=2% | — | — | 200 | |
| $TC_{LDO}$ | LDO Temperature Coefficient | — | Ta= -40°C~85°C, LDOVS[1:0]=00B, $V_{IN}$=3.6V, $I_{LOAD}$=100μA | — | — | 0.48 | mV/°C |
| $\Delta V_{LINE\_LDO}$ | LDO Line Regulation | — | LDOVS[1:0]=00B, 2.6V≤$V_{IN}$≤5.5V, $I_{LOAD}$=100μA | — | — | 0.3 | %/V |
| $V_{OUT\_VCM}$ | VCM Output Voltage | — | VCMS=0, $V_{OREG}$=3.3V, No load | -5% | 1.05 | +5% | V |
| | | — | VCMS=1, $V_{OREG}$=3.3V, No load | -5% | 1.25 | +5% | |
| $TC_{VCM}$ | VCM Temperature Coefficient | — | Ta= -40°C~85°C, VCMS=1, $V_{OREG}$=3.3V, $I_{LOAD}$=10μA | — | — | 0.24 | mV/°C |
| $\Delta V_{LINE\_VCM}$ | VCM Line Regulation | — | VCMS=1, 2.4V≤$V_{OREG}$≤3.3V, No load | — | — | 0.3 | %/V |
| $t_{VCMS}$ | VCM Turn On Stable Time | — | VCMS=1, $V_{OREG}$=3.3V, No load | — | — | 10 | ms |
| $I_{OH}$ | Source Current for VCM Output Pin | — | VCMS=1, $V_{OREG}$=3.3V, $\Delta V_{OUT\_VCM}$= -2% | 1 | — | — | mA |
| $I_{OL}$ | Sink Current for VCM Output Pin | — | VCMS=1, $V_{OREG}$=3.3V, $\Delta V_{OUT\_VCM}$=+2% | 1 | — | — | mA |

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| **ADC & ADC Internal Reference Voltage (Delta Sigma ADC)** | | | | | | | |
| $V_{OREG}$ | Supply Voltage for VCM, ADC, PGA | — | LDOEN=0 | 2.4 | — | 3.3 | V |
| | | | LDOEN=1 | 2.4 | — | 3.3 | |
| $V_{CM\_ADC}$ | Common Mode Voltage Range | — | VCMEN=0 | 1.05 | — | 1.25 | V |
| | | — | VCMEN=1 | 1.05 | — | 1.25 | |
| $I_{ADC}$ | Additional Current for ADC Enable | — | VCM enable, VRBUFP=1 and VRBUFN=1 | — | 750 | 900 | μA |
| | | — | VCM enable, VRBUFP=0 and VRBUFN=0 | — | 600 | 750 | μA |
| | | — | VCM disable, VRBUFP=0 and VRBUFN=0 | — | 500 | 650 | μA |
| $I_{ADSTB}$ | Standby Current | — | MCU enters SLEEP mode,No load | — | — | 1 | μA |
| $N_R$ | Resolution | — | — | — | — | 24 | Bit |
| INL | Integral Non-linearity | — | $V_{OREG}$=3.3V, $V_{REF}$=1.25V, ΔSI=±450mV, PGA gain=1 | — | ±50 | ±200 | ppm |
| NFB | Noise Free Bits | — | PGA gain=128, Data rate=10Hz | — | 15.4 | — | Bit |
| ENOB | Effective Number of Bits | — | PGA gain=128, Data rate=10Hz | — | 18.1 | — | Bit |
| $f_{ADCK}$ | ADC Clock Frequency | — | — | 40.0 | 409.6 | 440.0 | kHz |
| $f_{ADO}$ | ADC Output Data Rate | — | $f_{MCLK}$=4MHz, FLMS[2:0]=000B | 4 | — | 521 | Hz |
| | | — | $f_{MCLK}$=4MHz, FLMS[2:0]=010B | 10 | — | 1302 | Hz |
| $V_{REFP}$ | Reference Input Voltage | — | VGS[1:0]=00B, VREFS=1, VRBUFP=0, VRBUFN=0 | 0.96 | 1.25 | 2.20 | V |
| $V_{REFN}$ | | — | | 0 | 0 | 1 | V |
| $V_{REF}$ | | — | VGS[1:0]=00B, $V_{REF}$=$V_{REFP}$-$V_{REFN}$ | 0.96 | 1.25 | 1.44 | V |
| **PGA** | | | | | | | |
| $V_{CM\_PGA}$ | Common Mode Voltage Range | — | — | 0.4 | — | $V_{OREG}$ -1.1 | V |
| $\Delta D_I$ | Differential Input Voltage Range | — | Gain=PGS×AGS | -$V_{REF}$ /Gain | — | +$V_{REF}$ /Gain | V |
| **Temperature Sensor** | | | | | | | |
| $TC_{TS}$ | Temperature Sensor Temperature Coefficient | — | Ta= -40°C~85°C, $V_{REF}$=1.25V, VGS[1:0]=00B (Gain=1), VRBUFP=0, VRBUFN=0 | — | 175 | — | μV/°C |
| **D/A Converter** | | | | | | | |
| $V_{DACO}$ | Output Voltage Range | — | — | $V_{SS}$ | — | $V_{REF}$ | V |
| $V_{REF}$ | Reference Voltage | — | — | 1.05 | — | $V_{DD}$ | V |
| $I_{DAC}$ | Additional Current for DAC Enable | — | $V_{REF}$=5V | — | — | 450 | μA |
| DNL | Differential Non-linearity | — | 2.4V≤$V_{DD}$≤5.5V | — | — | ±4 | LSB |
| INL | Integral Non-linearity | — | 2.4V≤$V_{DD}$≤5.5V | — | — | ±8 | LSB |

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

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

**Effective Number of Bits (ENOB)**

$V_{OREG}$=3.3V, $V_{REF}$=1.25V, FLMS[2:0]=000

| Data Rate (SPS) | PGA Gain | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 |
| 5 | 19.7 | 19.8 | 19.6 | 19.7 | 19.7 | 19.6 | 19.2 | 18.6 |
| 10 | 19.4 | 19.3 | 19.3 | 19.3 | 19.3 | 19.1 | 18.7 | 18.1 |
| 20 | 19.0 | 18.8 | 18.7 | 18.9 | 18.8 | 18.6 | 18.2 | 17.5 |
| 40 | 18.4 | 18.3 | 18.3 | 18.3 | 18.3 | 18.1 | 17.7 | 17.0 |
| 80 | 18.1 | 17.9 | 18.0 | 17.9 | 17.9 | 17.6 | 17.2 | 16.5 |
| 160 | 17.6 | 17.4 | 17.4 | 17.4 | 17.3 | 17.1 | 16.6 | 15.9 |
| 320 | 15.8 | 15.8 | 15.9 | 15.8 | 15.9 | 15.9 | 15.8 | 15.3 |
| 640 | 14.1 | 14.0 | 14.0 | 14.1 | 14.1 | 14.0 | 14.1 | 14.4 |

**Effective Number of Bits (ENOB)**

$V_{OREG}$=3.3V, $V_{REF}$=1.25V, FLMS[2:0]=010

| Data Rate (SPS) | PGA Gain | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 4 | 8 | 16 | 32 | 64 | 128 |
| 12.5 | 19.4 | 18.8 | 18.7 | 18.8 | 18.8 | 18.7 | 18.9 | 18.1 |
| 25 | 19.0 | 18.3 | 18.3 | 18.3 | 18.3 | 18.2 | 17.9 | 17.3 |
| 50 | 18.5 | 17.8 | 17.8 | 17.8 | 17.9 | 17.7 | 17.4 | 16.8 |
| 100 | 18.2 | 18.2 | 18.1 | 18.2 | 18.1 | 17.8 | 17.2 | 16.4 |
| 200 | 17.9 | 17.8 | 17.8 | 17.8 | 17.6 | 17.3 | 16.7 | 15.9 |
| 400 | 17.4 | 17.2 | 17.2 | 17.2 | 17.1 | 16.8 | 16.2 | 15.4 |
| 800 | 16.2 | 16.1 | 16.1 | 16.1 | 16.1 | 15.9 | 15.5 | 14.8 |
| 1600 | 14.5 | 14.5 | 14.5 | 14.4 | 14.5 | 14.5 | 14.3 | 14.0 |

## LVR/LVD Electrical Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{DD}$ | Operating Voltage | — | — | 2.2 | — | 5.5 | V |
| $V_{LVR}$ | Low Voltage Reset Voltage | — | LVR enable, voltage select 2.1V | -5% | 2.1 | +5% | V |
| | | — | LVR enable, voltage select 2.55V | | 2.55 | | |
| | | — | LVR enable, voltage select 3.15V | | 3.15 | | |
| | | — | LVR enable, voltage select 3.8V | | 3.8 | | |
| $V_{LVD}$ | Low Voltage Detection Voltage | — | LVD enable, voltage select 1.04V | -10% | 1.04 | +10% | V |
| | | — | LVD enable, voltage select 2.2V | -5% | 2.2 | +5% | |
| | | — | LVD enable, voltage select 2.4V | | 2.4 | | |
| | | — | LVD enable, voltage select 2.7V | | 2.7 | | |
| | | — | LVD enable, voltage select 3.0V | | 3.0 | | |
| | | — | LVD enable, voltage select 3.3V | | 3.3 | | |
| | | — | LVD enable, voltage select 3.6V | | 3.6 | | |
| | | — | LVD enable, voltage select 4.0V | | 4.0 | | |
| $I_{LVRLVDBG}$ | Operating Current | 3V | LVD enable, LVR enable, VBGEN=0 | — | — | 18 | μA |
| | | 5V | | — | 20 | 25 | |
| | | 3V | LVD enable, LVR enable, VBGEN=1 | — | — | 150 | μA |
| | | 5V | | — | 25 | 30 | |

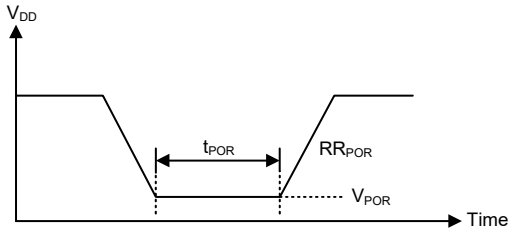
| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $I_{LVR}$ | Additional Current for LVR Enable | — | LVD disable, VBGEN=0 | — | — | 24 | μA |
| $I_{LVD}$ | Additional Current for LVD Enable | — | LVR disable, VBGEN=0 | — | — | 24 | μA |
| $I_{BG}$ | Additional current for bandgap reference with buffer | — | LVR disable, LVD disable | — | — | 180 | μA |
| $t_{LVDS}$ | LVDO Stable Time | — | For LVR enable, VBGEN=0, LVD off → on | — | — | 15 | μs |
| | | — | For LVR disable, VBGEN=0, LVD off → on | — | — | 150 | μs |
| $t_{LVR}$ | Minimum Low Voltage Width to Reset | — | — | 120 | 240 | 480 | μs |
| $t_{LVD}$ | Minimum Low Voltage Width to Interrupt | — | — | 60 | 120 | 240 | μs |

Note: If $V_{LVD}$=1.04V, it is used to detect the LVDIN pin input voltage. Other $V_{LVD}$ choices are used to detect the power supply $V_{DD}$.

## Power-on Reset Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{POR}$ | $V_{DD}$ Start Voltage to Ensure Power-on Reset | — | — | — | — | 100 | mV |
| $RR_{POR}$ | $V_{DD}$ 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 |



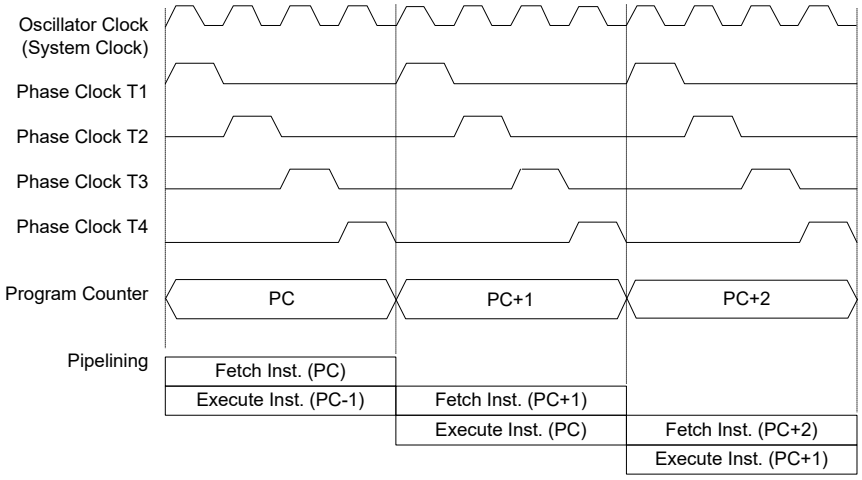
## System Architecture

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.

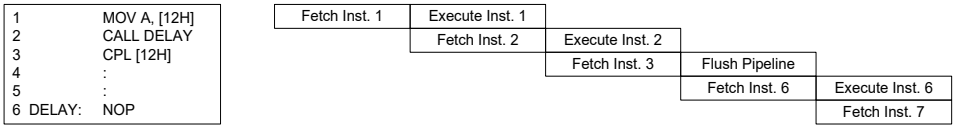
### Clocking and Pipelining

The main system clock, derived from either a 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.

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




**System Clocking and Pipelining**




**Instruction Fetching**

## Program Counter

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

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

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

**Program Counter**

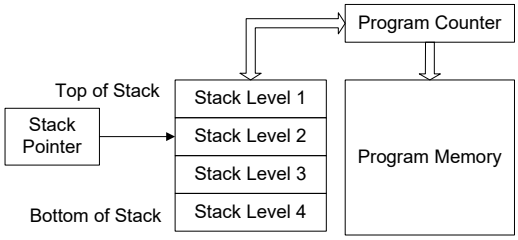
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available 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 organized into 4 levels and 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:

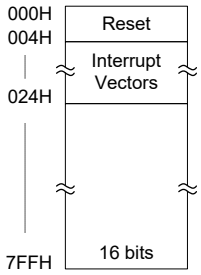
- Arithmetic operations:
  ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,
- Logic operations:
  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
- Rotation:
  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,
- Increment and Decrement:
  INCA, INC, DECA, DEC,
- Branch decision:
  JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA,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

## Flash Program Memory

The Program Memory is the location where the user code or program is stored. For the 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, the Flash device offers users the flexibility to conveniently debug and develop their applications while also offering a means of field programming and updating.

### Structure

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




**Program Memory Structure**

### Special Vectors

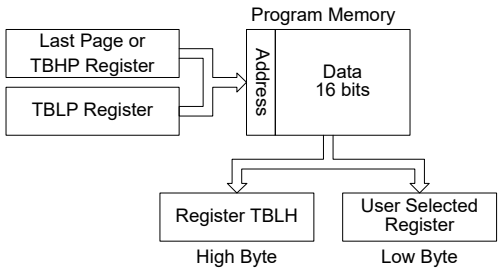
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

### 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.




## Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is "700H" which refers to the start address of the last page within the 2K Program Memory of the microcontroller. The table pointer low byte register is setup here to have an initial value of "06H". This will ensure that the first data read from the data table will be at the Program Memory address "706H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the first address specified by TBLP and TBHP 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      ; to the last page or the page that tbhp pointed
mov a,07h       ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1  ; transfers value in table referenced by table pointer data at program
                ; memory address "706H" transferred to tempreg1 and TBLH
dec tblp        ; reduce value of table pointer by one
tabrd tempreg2  ; transfers value in table referenced by table pointer data at program
                ; memory address "705H" transferred to tempreg2 and TBLH in this example
                ; the data "1AH" is transferred to tempreg1 and data "0FH" to register
                ; tempreg2
:
:
org 0700h       ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
```

## In Circuit Programming – ICP

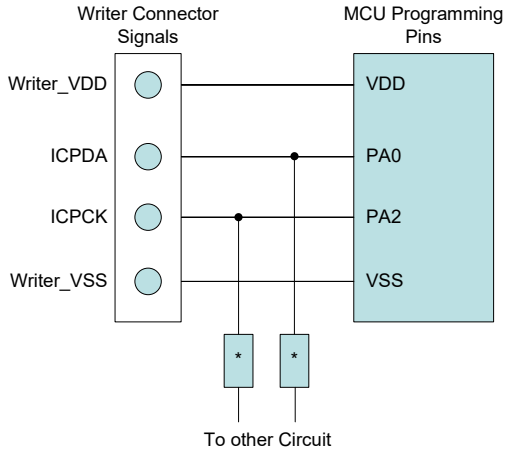
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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.

| Holtek Writer Pins | MCU Programming Pins | Pin Description |
|---|---|---|
| ICPDA | PA0 | Programming Serial Data/Address |
| ICPCK | PA2 | Programming Clock |
| VDD | VDD | Power Supply |
| VSS | VSS | Ground |

The Program Memory and EEPROM Data Memory can be programmed serially in-circuit using this 4-wire interface. Data is downloaded and uploaded serially on a single pin with an additional line for the clock. Two additional lines are required for the power supply and one line for the reset. 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 user must take care of the ICPDA and ICPCK pins for data and clock programming purposes to ensure that no other outputs are connected to these two pins.




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

## On-Chip Debug Support – OCDS

There is an EV chip named BH66V5233 which is used to emulate the BH66F5233 device. The EV chip device also provides an "On-Chip Debug" function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for "On-Chip Debug" function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCDSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named "Holtek e-Link for 8-bit MCU OCDS User's Guide".

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

# 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.

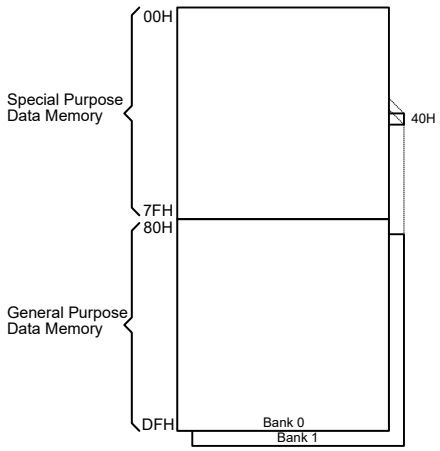
## Structure

Divided into two types, 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 addressed from 00H~DFH in Data Memory are common and accessible in Bank 0 and the EEC register at the address 40H 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.

| Special Purpose Data Memory | | General Purpose Data Memory | |
|---|---|---|---|
| Available Banks | Banks | Capacity | Banks |
| 0,1 | Bank 0: 00H~7FH<br>Bank 1: 40H (EEC) | 96 × 8 | Bank 0: 80H~DFH |

**Data Memory Summary**



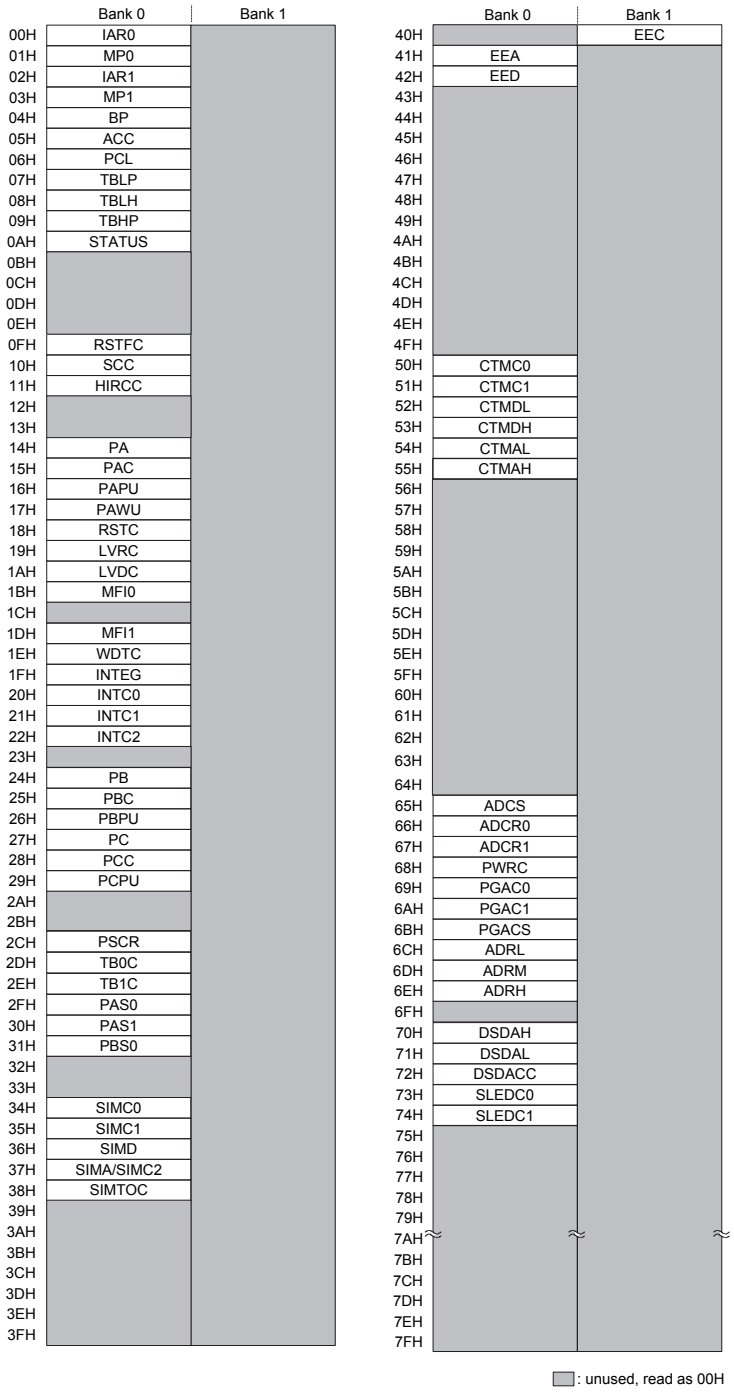

**Data Memory Structure**

## General Purpose Data Memory

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

## Special Purpose Data Memory

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




**Special Purpose Data Memory**

## Special Function Register Description

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

### Indirect Addressing Registers – IAR0, IAR1

The Indirect Addressing Registers, IAR0 and IAR1, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0 and IAR1 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0 or MP1. Acting as a pair, IAR0 and MP0 can together access data from Bank 0 while the IAR1 and MP1 register pair can access data from any bank. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of "00H" and writing to the registers indirectly will result in no operation.

### Memory Pointers – MP0, MP1

Two Memory Pointers, known as MP0 and MP1 are provided. These Memory Pointers are physically implemented in the Data Memory and can be manipulated in the same way as normal registers providing a convenient way with which to address and track data. When any operation to the relevant Indirect Addressing Registers is carried out, the actual address that the microcontroller is directed to is the address specified by the related Memory Pointer. MP0, together with Indirect Addressing Register, IAR0, are used to access data from Bank 0, while MP1 and IAR1 are used to access data from all banks according to BP register. Direct Addressing can only be used with Bank 0, all other Banks must be addressed indirectly using MP1 and IAR1.

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´
org 00h
start:
     mov a, 04h              ; setup size of block
     mov block, a
     mov a, offset adres1    ; Accumulator loaded with first RAM address
     mov mp0, a              ; setup memory pointer with first RAM address
loop:
     clr IAR0                ; clear the data at address defined by MP0
     inc mp0                 ; increment memory pointer
     sdz block               ; check if last memory location has been cleared
     jmp loop
continue:
```

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

## Bank Pointer – BP

For this device, the Data Memory is divided into two banks, Bank0 and Bank1. Selecting the required Data Memory area is achieved using the Bank Pointer. Bit 0 of the Bank Pointer is used to select Data Memory Banks 0~1.

The Data Memory is initialised to Bank 0 after a reset, except for a WDT time-out reset in the SLEEP or IDLE Mode, in which case, the Data Memory bank remains unaffected. Directly addressing the Data Memory will always result in Bank 0 being accessed irrespective of the value of the Bank Pointer. Accessing data from Bank1 must be implemented using Indirect Addressing.

### BP Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | — | DMBP0 |
| R/W | — | — | — | — | — | — | — | R/W |
| POR | — | — | — | — | — | — | — | 0 |

Bit 7~1 Unimplemented, read as "0"

Bit 0 **DMBP0**: Select Data Memory Banks
0: Bank 0
1: Bank 1

## Accumulator – ACC

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

## Program Counter Low Register – PCL

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

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

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

### Status Register – STATUS

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

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

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

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

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

### STATUS Register

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

"x": unknown

Bit 7~6 Unimplemented, read as "0"

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

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

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

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

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

Bit 0 **C**: Carry flag
0: No carry-out
1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation

The "C" flag is also affected by a rotate through carry instruction.

# EEPROM Data Memory

This device contains an area of internal EEPROM Data Memory. EEPROM, which stands for Electrically Erasable Programmable Read Only Memory, is by its nature a non-volatile form of re-programmable 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 32×8 bits for the device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable 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 a 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 registers, 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 MP1 Memory Pointer and Indirect Addressing Register, IAR1. Because the EEC control register is located at address 40H in Bank 1, the MP1 Memory Pointer must first be set to the value 40H and the Bank Pointer register, BP, set to the value, 01H, before any operations on the EEC register are executed.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EEA | — | — | — | EEA4 | EEA3 | EEA2 | EEA1 | EEA0 |
| EED | EED7 | EED6 | EED5 | EED4 | EED3 | EED2 | EED1 | EED0 |
| EEC | — | — | — | — | WREN | WR | RDEN | RD |

**EEPROM Registers List**

### EEA Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | EEA4 | EEA3 | EEA2 | EEA1 | EEA0 |
| 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 **EEA4~EEA0**: Data EEPROM address bit 4~bit 0

### EED Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | EED7 | EED6 | EED5 | EED4 | EED3 | EED2 | EED1 | EED0 |
| 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 **EED7~EED0**: 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 cannot be set high at the same time in one instruction. The WR and RD cannot be set high 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

The EEPROM address of the data to be written must first be placed in the EEA register and the data placed in the EED register. 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. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle. These two instructions must be executed consecutively. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set again after the write cycle has started. Note that 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 Bank Pointer, BP, 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 enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Bank Pointer could be normally cleared to zero as this would inhibit access to Bank 1 where the EEPROM control register exists. 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 MP1, A                ; MP1 points to EEC register
MOV A, 01H                ; setup Bank Pointer
MOV BP, 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 write
CLR BP
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 MP1, A                ; MP1 points to EEC register
MOV A, 01H                ; setup Bank Pointer
MOV BP, A
CLR EMI
SET IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                ; start Write Cycle - set WR bit— executed immediately after set
                          ; WREN bit
SET EMI
BACK:
SZ  IAR1.2                ; check for write cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM write
CLR BP
```

# Oscillators

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

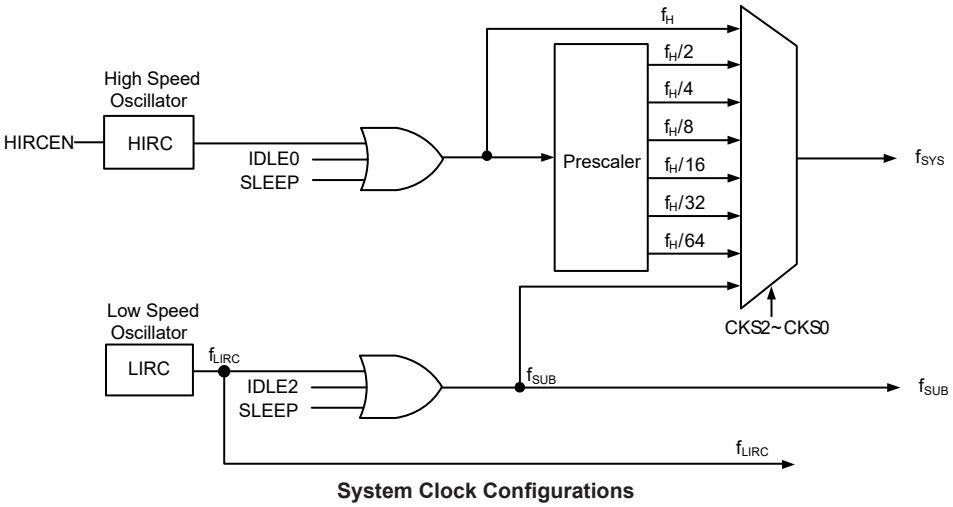
| Type | Name | Frequency |
|---|---|---|
| Internal High Speed RC | HIRC | 4/8/12MHz |
| Internal Low Speed RC | LIRC | 32kHz |

**Oscillator Types**

## System Clock Configurations

There are two oscillator sources, a high speed oscillator and a low speed oscillator. The high speed oscillator is the internal 4/8/12MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC.

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




**System Clock Configurations**

### Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 4MHz, 8MHz and 12MHz. 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.

### 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.

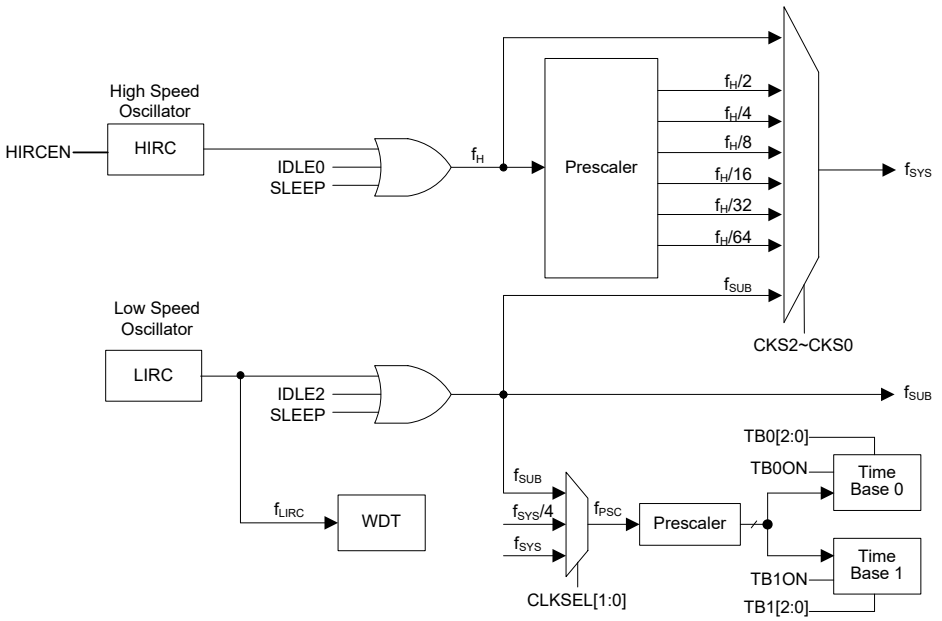
## Operating Modes and System Clocks

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

The main system clock, can come from 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 HIRC oscillator. The low speed system clock source can be sourced from the internal clock $f_{SUB}$. If $f_{SUB}$ is selected then it can be sourced by LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.




**Device Clock Configurations**

Note: When the system clock source $f_{SYS}$ is switched to $f_{SUB}$ from $f_H$, the high speed oscillator 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 FAST 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 | | | $f_{SYS}$ | $f_H$ | $f_{SUB}$ | $f_{LIRC}$ |
|---|---|---|---|---|---|---|---|---|
| | | FHIDEN | FSIDEN | CKS2~CKS0 | | | | |
| FAST | On | x | x | 000~110 | $f_H$~$f_H/64$ | On | On | On |
| SLOW | On | x | x | 111 | $f_{SUB}$ | On/Off (1) | On | On |
| IDLE0 | Off | 0 | 1 | 000~110 | Off | Off | On | On |
| | | | | 111 | On | | | |
| IDLE1 | Off | 1 | 1 | xxx | On | On | On | On |
| IDLE2 | Off | 1 | 0 | 000~110 | On | On | Off | On |
| | | | | 111 | Off | | | |
| SLEEP | Off | 0 | 0 | xxx | Off | Off | Off | On/Off (2) |

"x": Don't care

Note: 1. The $f_H$ clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.

2. The $f_{LIRC}$ clock can be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

### FAST 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 HIRC oscillator. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current.

### SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from $f_{SUB}$. The $f_{SUB}$ clock is derived from the LIRC oscillator.

### SLEEP Mode

The SLEEP Mode is entered when an HALT instruction is executed and when the FHIDEN and FSIDEN bit are low. In the SLEEP mode the CPU will be stopped. The $f_{SUB}$ clock provided to the peripheral function will also be stopped, too. However the $f_{LIRC}$ clock can continues to operate if the WDT function is enabled.

### IDLE0 Mode

The IDLE0 Mode is entered when an HALT instruction is executed and when the FHIDEN bit in the SCC register is low and the FSIDEN bit in the SCC register is high. 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 an 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 an 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.

## Control Registers

The registers, SCC and HIRCC are used to control the system clock and the corresponding oscillator configurations.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SCC | CKS2 | CKS1 | CKS0 | — | — | — | FHIDEN | FSIDEN |
| HIRCC | — | — | — | — | HIRC1 | HIRC0 | HIRCF | HIRCEN |

**System Operating Mode Control Registers List**

### SCC Register

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

Bit 7~5 **CKS2~CKS0**: 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 4~2 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 an "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 an "HALT" instruction. The LIRC oscillator is controlled by this bit together with the WDT function enable control. If this bit is cleared to 0 but the WDT function is enabled, the $f_{LIRC}$ clock will also be enabled.

### HIRCC Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | HIRC1 | HIRC0 | HIRCF | HIRCEN |
| R/W | — | — | — | — | R/W | R/W | R | R/W |
| POR | — | — | — | — | 0 | 0 | 0 | 1 |

Bit 7~4 Unimplemented, read as 0.

Bit 3~2 **HIRC1~HIRC0**: HIRC frequency selection
00: 4MHz
01: 8MHz
10: 12MHz
11: 4MHz

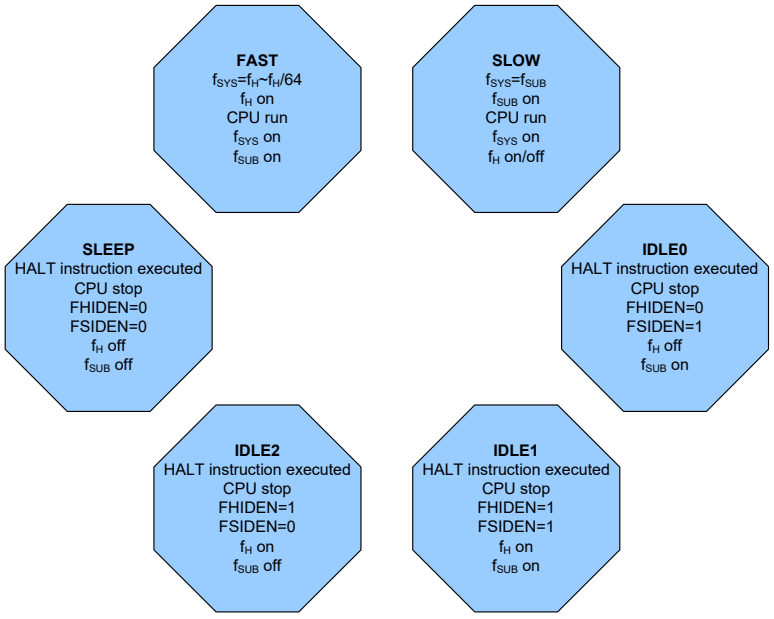
Bit 1 **HIRCF**: HIRC oscillator stable flag
0: HIRC unstable
1: HIRC stable
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is 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.

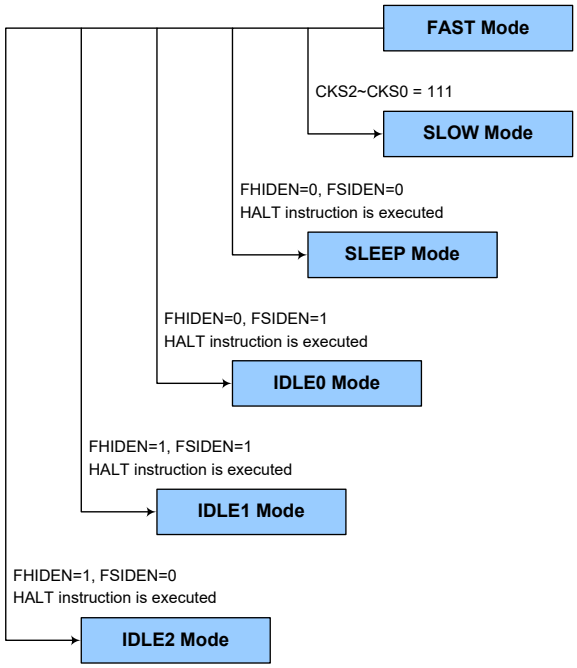
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When an HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the FHIDEN and FSIDEN bits in the SCC register.



### FAST Mode to SLOW Mode Switching

When running in the FAST 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.

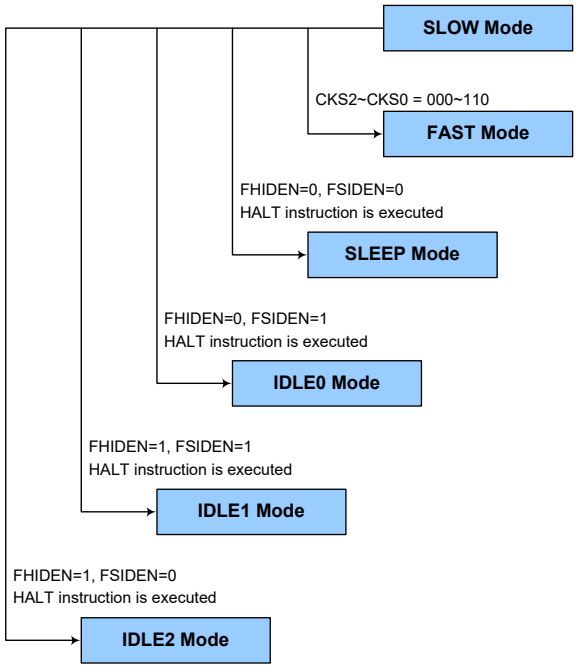
The SLOW Mode is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



### SLOW Mode to FAST Mode Switching

In SLOW mode the system clock is derived from $f_{SUB}$. When system clock is switched back to the FAST 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 and thus switched off, it will take some time to re-oscillate and stabilise when switching to the FAST mode from the SLOW Mode. This is monitored using the HIRCF bit in the HIRCC register. The time duration required for the high speed system oscillator stabilization is specified in the System Start Up Time Characteristics.




### 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 FSIDEN bits in the SCC register equal to "0". In this mode all the clocks and functions will be switched off except the WDT function. 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 the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

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

- The $f_H$ 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 the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### 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 FSIDEN bits 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 $f_{SUB}$ clocks 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 the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### 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 FSIDEN 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 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 will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

## 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 the device which has different package types, as there may be unbonded 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 IDLE2 Mode the high speed oscillator is on, if the peripheral function clock source is derived 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 will take a considerable time for the original system oscillator to restart, stabilise and allow normal operation to resume.

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

When the device executes the "HALT" instruction, it will enter the Power down mode and the PDF flag will be set to 1. The PDF flag will be cleared to 0 if the device experiences a system power-up or executes the clear Watchdog Timer instruction. If the system is woken up by a WDT overflow, a Watchdog Timer reset will be initiated and the TO flag will be set to 1. 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, 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 pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction. If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the "HALT" instruction. In this situation, the interrupt which woke-up the device will not be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

# Watchdog Timer

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

## Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, $f_{LIRC}$ which is sourced from the LIRC oscillator. 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 time-out period as well as the enable/disable and reset MCU operation.

### 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: Disable
01010: Enable
Other values: Reset MCU
When these bits are changed to any other values due to environmental noise the microcontroller will be reset; this reset operation will be activated after a delay time, $t_{SRESET}$ and the WRF bit in the RSTFC register will be set high.

Bit 2~0 **WS2~WS0**: WDT time-out period selection
000: $2^8/f_{LIRC}$
001: $2^{10}/f_{LIRC}$
010: $2^{12}/f_{LIRC}$
011: $2^{14}/f_{LIRC}$
100: $2^{15}/f_{LIRC}$
101: $2^{16}/f_{LIRC}$
110: $2^{17}/f_{LIRC}$
111: $2^{18}/f_{LIRC}$
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 | — | — | — | — | RSTF | LVRF | LRF | WRF |
| R/W | — | — | — | — | R/W | R/W | R/W | R/W |
| POR | — | — | — | — | 0 | x | 0 | 0 |

"x": unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Describe elsewhere.

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

Bit 1 **LRF**: LVR control register software reset flag
Described elsewhere.

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

## Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog instructions. If the program malfunctions for whatever reason, jumps to an unknown location, or enters an endless loop, these clear instructions will not be executed in the correct manner, in which case the Watchdog Timer will overflow and reset the device. There are five bits, WE4~WE0, in the WDTC register to offer the enable/disable control and reset control of the Watchdog Timer. The WDT function will be disabled when the WE4~WE0 bits are set to a value of 10101B while the WDT function will be enabled if the WE4~WE0 bits are equal to 01010B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after a delay time, $t_{SRESET}$. After power on these bits will have a value of 01010B.

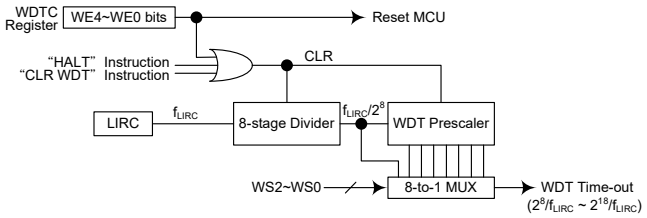
| WE4~WE0 Bits | WDT Function |
|---|---|
| 10101B | Disable |
| 01010B | Enable |
| Any other value | Reset MCU |

**Watchdog Timer Enable/Disable Control**

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

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




**Watchdog Timer**

## Reset and Initialisation

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

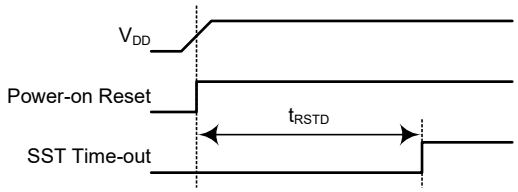
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

### Reset Functions

There are 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.




**Power-on Reset Timing Chart**

### Internal Reset Control

There is an internal reset control register, RSTC, which is used to provide a reset when the device operates abnormally due to the environmental noise interference. If the content of the RSTC register is set to any value other than 01010101B or 10101010B, it will reset the device after a delay time, $t_{SRESET}$. After power on the register will have a value of 01010101B.

| RSTC7~RSTC0 Bits | Reset Function |
|---|---|
| 01010101B | No operation |
| 10101010B | No operation |
| Any other value | Reset MCU |

**Internal Reset Function Control**

- **RSTC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | RSTC7 | RSTC6 | RSTC5 | RSTC4 | RSTC3 | RSTC2 | RSTC1 | RSTC0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |

Bit 7~0 **RSTC7~RSTC0**: Reset function control
01010101: No operation
10101010: No operation
Other values: Reset MCU.
If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time, $t_{SRESET}$ and the RSTF bit in the RSTFC register will be set to 1.

- **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
0: Not occurred
1: Occurred
This bit is set to 1 by the RSTC control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

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

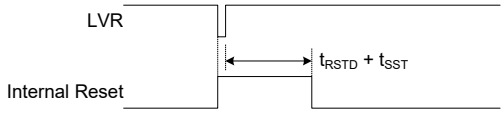
Bit 1 **LRF**: LVR control register software reset flag
Described elsewhere.

Bit 0 **WRF**: WDT control register software reset flag
Described elsewhere.

### Low Voltage Reset – LVR

The microcontroller contains a low voltage reset circuit in order to monitor the supply voltage of the device and provides an MCU reset should the value fall below a certain predefined level.

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 in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between 0.9V~$V_{LVR}$ must exist for a time greater than that specified by $t_{LVR}$ in the LVR/LVD characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual $V_{LVR}$ value can be selected by the LVS bits in the LVRC register. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the device after a delay time, $t_{SRESET}$. When this happens, the LRF bit in the RSTFC register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the SLEEP or IDLE Mode.




**Low Voltage Reset Timing Chart**

- **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR Voltage Select control
01010101: 2.1V
00110011: 2.55V
10011001: 3.15V
10101010: 3.8V
Any other value: Generates MCU reset – register is reset to POR value

When an actual low voltage condition occurs, as specified by one of the four defined LVR voltage values above, an MCU reset will be generated. The reset operation will be activated after the low voltage condition keeps more than a $t_{LVR}$ time. In this situation the register contents will remain the same after such a reset occurs.

Any register value, other than the four defined LVR values above, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time, $t_{SRESET}$. However in this situation the register contents will be reset to the POR value.

- **RSTFC Register**

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

"x": unknown

Bit 7~4 Unimplemented, read as "0"

Bit 3 **RSTF**: Reset control register software reset flag
Describe elsewhere.

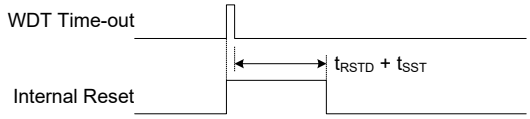
Bit 2 **LVRF**: LVR function reset flag
0: Not occur
1: Occurred
This bit is set to 1 when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.

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

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

### Watchdog Time-out Reset during Normal Operation

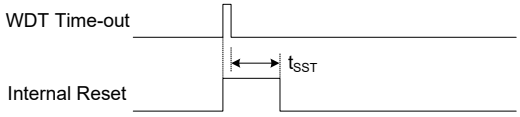
The Watchdog time-out Reset during normal operations in the FAST or SLOW mode is the same as a LVR reset except that the Watchdog time-out flag TO will be set to "1".




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

### Watchdog Time-out Reset during 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 System Start Up Time Characteristics for $t_{SST}$ details.




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

## Reset Initial Conditions

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

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

"u" stands for unchanged

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

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

The different kinds of resets all affect the internal registers of the microcontroller in different ways. To ensure reliable continuation of normal program execution after a reset occurs, it is important to know what condition the microcontroller is in after a particular reset occurs. The following table describes how each type of reset affects each of the microcontroller internal registers.

| Register Name | Power On Reset | LVR Reset (FAST) | LVR Reset (IDLE/SLEEP) | WDT Time-out (FAST) | WDT Time-out (IDLE/SLEEP) |
|---|---|---|---|---|---|
| IAR0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| IAR1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| BP | ---- ---0 | ---- ---0 | ---- ---0 | ---- ---0 | ---- ---u |
| ACC | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| PCL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 |
| TBLP | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBLH | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBHP | ---- -xxx | ---- -uuu | ---- -uuu | ---- -uuu | ---- -uuu |
| STATUS | --00 xxxx | --uu uuuu | --01 uuuu | --1u uuuu | --11 uuuu |
| RSTFC | ---- 0x00 | ---- uuuu | ---- uuuu | ---- uuuu | ---- uuuu |
| SCC | 000- 0000 | 000- 0000 | 000- 0000 | 000- 0000 | uuu- uuuu |
| HIRCC | ---- 0001 | ---- 0001 | ---- 0001 | ---- 0001 | ---- uuuu |
| PA | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAC | 1111 1111 | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAPU | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PAWU | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RSTC | 0101 0101 | 0101 0101 | 0101 0101 | 0101 0101 | uuuu uuuu |
| LVRC | 0101 0101 | 0101 0101 | 0101 0101 | 0101 0101 | uuuu uuuu |
| LVDC | --00 0000 | --00 0000 | --00 0000 | --00 0000 | --uu uuuu |
| MFI0 | --00 --00 | --00 --00 | --00 --00 | --00 --00 | --uu --uu |
| MFI1 | --00 --00 | --00 --00 | --00 --00 | --00 --00 | --uu --uu |
| WDTC | 0101 0011 | 0101 0011 | 0101 0011 | 0101 0011 | uuuu uuuu |
| INTEG | ---- 0000 | ---- 0000 | ---- 0000 | ---- 0000 | ---- uuuu |
| INTC0 | -000 0000 | -000 0000 | -000 0000 | -000 0000 | -uuu uuuu |
| INTC1 | -000 -000 | -000 -000 | -000 -000 | -000 -000 | -uuu -uuu |
| INTC2 | --00 --00 | --00 --00 | --00 --00 | --00 --00 | --uu --uu |
| PB | ---- 1111 | ---- 1111 | ---- 1111 | ---- 1111 | ---- uuuu |
| PBC | ---- 1111 | ---- 1111 | ---- 1111 | ---- 1111 | ---- uuuu |
| PBPU | ---- 0000 | ---- 0000 | ---- 0000 | ---- 0000 | ---- uuuu |
| PC | ---- --11 | ---- --11 | ---- --11 | ---- --11 | ---- --uu |
| PCC | ---- --11 | ---- --11 | ---- --11 | ---- --11 | ---- --uu |

| Register Name | Power On Reset | LVR Reset (FAST) | LVR Reset (IDLE/SLEEP) | WDT Time-out (FAST) | WDT Time-out (IDLE/SLEEP) |
|---|---|---|---|---|---|
| PCPU | ---- --00 | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| PSCR | ---- --00 | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| TB0C | 0--- -000 | 0--- -000 | 0--- -000 | 0--- -000 | u--- -uuu |
| TB1C | 0--- -000 | 0--- -000 | 0--- -000 | 0--- -000 | u--- -uuu |
| PAS0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PAS1 | ---- --00 | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| PBS0 | ---- --00 | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| SIMC0 | 111- 0000 | 111- 0000 | 111- 0000 | 111- 0000 | uuu- uuuu |
| SIMC1 | 1000 0001 | 1000 0001 | 1000 0001 | 1000 0001 | uuuu uuuu |
| SIMD | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| SIMA | 0000 000- | 0000 000- | 0000 000- | 0000 000- | uuuu uuu- |
| SIMC2 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SIMTOC | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| EEC | ---- 0000 | ---- 0000 | ---- 0000 | ---- 0000 | ---- uuuu |
| EEA | ---0 0000 | ---0 0000 | ---0 0000 | ---0 0000 | ---u uuuu |
| EED | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMC0 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMC1 | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMDL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMDH | ---- --00 | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| CTMAL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMAH | ---- --00 | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| ADCS | ---0 0000 | ---0 0000 | ---0 0000 | ---0 0000 | ---u uuuu |
| ADCR0 | 0010 00-0 | 0010 00-0 | 0010 00-0 | 0010 00-0 | uuuu uu-u |
| ADCR1 | 0000 000- | 0000 000- | 0000 000- | 0000 000- | uuuu uuu- |
| PWRC | 00-- -000 | 00-- -000 | 00-- -000 | 00-- -000 | uu-- -uuu |
| PGAC0 | -000 0000 | -000 0000 | -000 0000 | -000 0000 | -uuu uuuu |
| PGAC1 | 1000 000- | 1000 000- | 1000 000- | 1000 000- | uuuu uuu- |
| PGACS | --00 0000 | --00 0000 | --00 0000 | --00 0000 | --uu uuuu |
| ADRL | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| ADRM | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| ADRH | xxxx xxxx | xxxx xxxx | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| DSDAH | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| DSDAL | ---- 0000 | ---- 0000 | ---- 0000 | ---- 0000 | ---- uuuu |
| DSDACC | 00-- ---- | 00-- ---- | 00-- ---- | 00-- ---- | uu-- ---- |
| SLEDC0 | --00 0000 | --00 0000 | --00 0000 | --00 0000 | --uu uuuu |
| SLEDC1 | ---- --00 | ---- --00 | ---- --00 | ---- --00 | ---- --uu |

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

## 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~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 | PA7 | PA6 | PA5 | PA4 | PA3 | PA2 | PA1 | PA0 |
| PAC | PAC7 | PAC6 | PAC5 | PAC4 | PAC3 | PAC2 | PAC1 | PAC0 |
| PAPU | PAPU7 | PAPU6 | PAPU5 | PAPU4 | PAPU3 | PAPU2 | PAPU1 | PAPU0 |
| PAWU | PAWU7 | PAWU6 | PAWU5 | PAWU4 | PAWU3 | PAWU2 | PAWU1 | PAWU0 |
| PB | — | — | — | — | PB3 | PB2 | PB1 | PB0 |
| PBC | — | — | — | — | PBC3 | PBC2 | PBC1 | PBC0 |
| PBPU | — | — | — | — | PBPU3 | PBPU2 | PBPU1 | PBPU0 |
| PC | — | — | — | — | — | — | PC1 | PC0 |
| PCC | — | — | — | — | — | — | PCC1 | PCC0 |
| PCPU | — | — | — | — | — | — | PCPU1 | PCPU0 |

"—": Unimplemented, read as "0"

**I/O Logic Function Registers List**

### 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, namely PAPU~PCPU, and are implemented using weak PMOS transistors.

Note that the pull-high resistor can be controlled by the relevant pull-high control register only when the pin-shared functional pin is selected as an input or NMOS output. Otherwise, the pull-high resistors cannot be enabled.

#### PxPU Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PxPU7 | PxPU6 | PxPU5 | PxPU4 | PxPU3 | PxPU2 | PxPU1 | PxPU0 |
| 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 |

**PxPUn**: I/O Port x Pin pull-high function control
0: Disable
1: Enable

The PxPUn bit is used to control the pin pull-high function. Here the "x" can be A, B, and C. However, the actual available bits for each I/O Port may be different.

## 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.

Note that the wake-up function can be controlled by the wake-up control registers only when the pin-shared functional pin is selected as general purpose input/output and the MCU enters the Power down mode.

### PAWU Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PAWU7 | PAWU6 | PAWU5 | PAWU4 | PAWU3 | PAWU2 | PAWU1 | PAWU0 |
| 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 **PAWU7~PAWU0**: PA7~PA0 wake-up function 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.

### PxC Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PxC7 | PxC6 | PxC5 | PxC4 | PxC3 | PxC2 | PxC1 | PxC0 |
| 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 |

**PxCn**:I/O Port x Pin type selection
0: Output
1: Input

The PxCn bit is used to control the pin type selection. Here the "x"can be A, B, and C. However, the actual available bits for each I/O Port may be different.

## Source Current Selection

The source current of each pin in this device can be configured with different source current which is selected by the corresponding pin source current select bits. These source current bits are available when the corresponding pin is configured as a CMOS output. Otherwise, these select bits have no effect. Users should refer to the Input/Output Characteristics section to obtain the exact value for different applications.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SLEDC0 | — | — | SLEDC05 | SLEDC04 | SLEDC03 | SLEDC02 | SLEDC01 | SLEDC00 |
| SLEDC1 | — | — | — | — | — | — | SLEDC11 | SLEDC10 |

**I/O Port Source Current Selection Registers List**

### SLEDC0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | SLEDC05 | SLEDC04 | SLEDC03 | SLEDC02 | SLEDC01 | SLEDC00 |
| 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 5~4 **SLEDC05~SLEDC04**: PB3~PB0 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 3~2 **SLEDC03~SLEDC02**: PA7~PA4 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

Bit 1~0 **SLEDC01~SLEDC00**: PA3~PA0 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

### SLEDC1 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | SLEDC11 | SLEDC10 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **SLEDC11~SLEDC10**: PC1~PC0 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

## Pin-shared Functions

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

### Pin-shared Function Selection Registers

The limited number of supplied pins in a package can impose restrictions on the amount of functions a certain device can contain. However by allowing the same pins to share several different functions and providing a means of function selection, a wide range of different functions can be incorporated into even relatively small package sizes. The device includes Port "x" Output Function Selection register "n", labeled as PxSn, which can select the desired functions of the multi-function pin-shared pins.

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. To select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. To correctly deselect the pin-shared function, the peripheral function should first be disabled and then the corresponding pin-shared function control register can be modified to select other pin-shared functions.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PAS0 | PAS07 | PAS06 | PAS05 | PAS04 | PAS03 | PAS02 | PAS01 | PAS00 |
| PAS1 | — | — | — | — | — | — | PAS11 | PAS10 |
| PBS0 | — | — | — | — | — | — | PBS01 | PBS00 |

**Pin-shared Function Selection Registers List**

- **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-Shared function selection
00: PA3
01: CTP
10: PA3
11: PA3

Bit 5~4 **PAS05~PAS04**: PA2 Pin-Shared function selection
00: PA2
01: PA2
10: $\overline{\text{SCS}}$
11: PA2

Bit 3~2 **PAS03~PAS02**: PA1 Pin-Shared function selection
00: PA1/INT0
01: SDI/SDA
10: PA1/INT0
11: PA1/INT0

Bit 1~0 **PAS01~PAS00**: PA0 Pin-Shared function selection
00: PA0
01: SDO
10: PA0
11: PA0

- **PAS1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | PAS11 | PAS10 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PAS11~PAS10**: PA4 Pin-Shared function selection
00: PA4
01: CTPB
10: PA4
11: PA4

- **PBS0 Register**

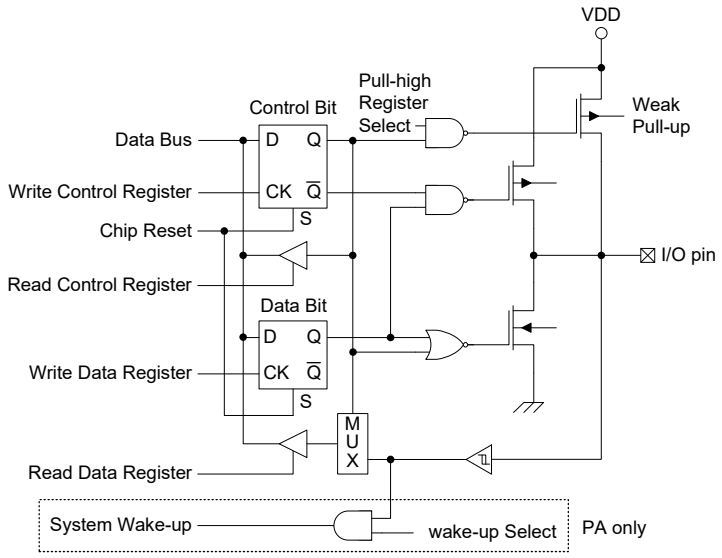
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | PBS01 | PBS00 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

Bit 7~2 Unimplemented, read as "0"

Bit 1~0 **PBS01~PBS00**: PB0 Pin-Shared function selection
00: PB0
01: PB0
10: SCK/SCL
11: LVDIN

## I/O Pin Structures

The accompanying diagram illustrates the internal structure of the I/O logic function. As the exact logical construction of the I/O pin will differ from this drawing, it is supplied as a guide only to assist with the functional understanding of the I/O logic function. The wide range of pin-shared structures does not permit all types to be shown.




**Logic Function Input/Output Structure**

## Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers 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 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.

Port A has the additional capability of providing wake-up function. 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 a Timer Module, abbreviated to the name TM. The TM is a multi-purpose timing unit and serves to provide operations such as Timer/Counter, Compare Match Output as well as being the functional unit for the generation of PWM signals. The TM has two individual interrupts. The addition of input and output pins for the TM ensures that users are provided with timing units with a wide and flexible range of features.

## Introduction

The device contains a TM categorised as Compact Type TM. The main features of CTM is summarised in the accompanying table.

| Function | CTM |
|---|---|
| Timer/Counter | √ |
| Compare Match Output | √ |
| PWM Channels | 1 |
| PWM Alignment | Edge |
| PWM Adjustment Period & Duty | Duty or Period |

## TM Operation

The TM offers 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 CTCK2~CTCK0 bits in the CTM control registers The clock source can be a ratio of the system clock $f_{SYS}$ or the internal high clock $f_H$, the $f_{SUB}$ clock source or the external CTCK pin. The CTCK 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 Compact type TM has two internal interrupts, one for each of the internal comparator A or comparator P, which generate a TM interrupt when a compare match condition occurs. When a TM interrupt is generated it can be used to clear the counter and also to change the state of the TM output pin.

## TM External Pins

The Compact TM has one TM input pin, with the label CTCK. The CTM input pin, CTCK, is essentially a clock source for the CTM and is selected using the CTCK2~CTCK0 bits in the CTMC0 register. This external TM input pin allows an external clock source to drive the internal TM. The CTCK input pin can be chosen to have either a rising or falling active edge.

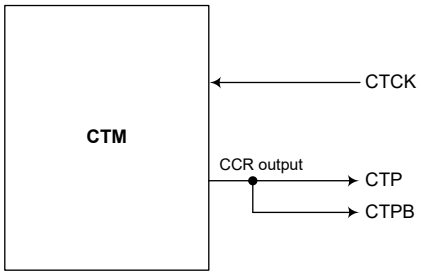
The Compact TM has two output pins with the label CTP and CTPB. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The external CTP and CTPB output pins are also the pins where the TM generates the PWM output waveform. As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be setup using the relevant pin-shared function selection bits described in the Pin-shared Function section.

| CTM | |
|---|---|
| **Input** | **Output** |
| CTCK | CTP, CTPB |

**CTM External Pins**

## TM Input/Output Pin Selection

Selecting to have a TM input/output or whether to retain its other shared function is implemented using the relevant pin-shared function selection registers, with the corresponding selection bits in each pin-shared function register corresponding to a TM input/output pin. Configuring the selection bits correctly will setup the corresponding pin as a TM input/output. The details of the pin-shared function selection are described in the pin-shared function section.

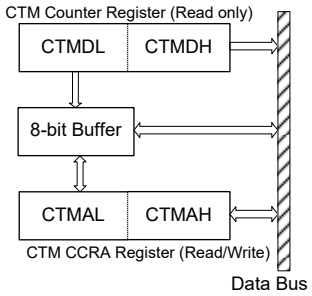



**CTM Function Pin Control Block Diagram**

## Programming Considerations

The TM Counter Registers and the Compare CCRA register, has 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 as described above, it is recommended to use the "MOV" instruction to access the CCRA low byte register, named CTMAL, using the following access procedures. Accessing the CCRA low byte register without following these access procedures will result in unpredictable values.

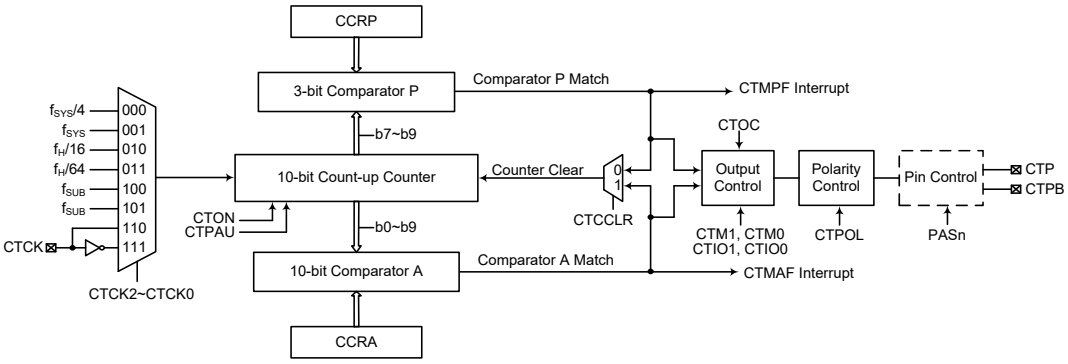



The following steps show the read and write procedures:

- Writing Data to CCRA
  - Step 1. Write data to Low Byte CTMAL
    – Note that here data is only written to the 8-bit buffer.
  - Step 2. Write data to High Byte CTMAH
    – 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 or CCRA
  - Step 1. Read data from the High Byte CTMDH, CTMAH
    – 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 CTMDL, CTMAL
    – This step reads data from the 8-bit buffer.

## Compact Type TM – CTM

Although the simplest form of the three 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.




**Compact Type TM Block Diagram**

### 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 CTON 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 CTM 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 one 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 several 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.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CTMC0 | CTPAU | CTCK2 | CTCK1 | CTCK0 | CTON | CTRP2 | CTRP1 | CTRP0 |
| CTMC1 | CTM1 | CTM0 | CTIO1 | CTIO0 | CTOC | CTPOL | CTDPX | CTCCLR |
| CTMDL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| CTMDH | — | — | — | — | — | — | D9 | D8 |
| CTMAL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| CTMAH | — | — | — | — | — | — | D9 | D8 |

**Compact TM Register List**

### CTMC0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CTPAU | CTCK2 | CTCK1 | CTCK0 | CTON | CTRP2 | CTRP1 | CTRP0 |
| 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 **CTPAU**: CTM 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 CTM 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 **CTCK2~CTCK0**: Select CTM Counter clock
000: $f_{SYS}/4$
001: $f_{SYS}$
010: $f_H/16$
011: $f_H/64$
100: $f_{SUB}$
101: $f_{SUB}$
110: CTCK rising edge clock
111: CTCK falling edge clock
These three bits are used to select the clock source for the CTM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source $f_{SYS}$ is the system clock, while $f_H$ and $f_{SUB}$ are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 **CTON**: CTM Counter On/Off Control
0: Off
1: On
This bit controls the overall on/off function of the CTM. Setting the bit high enables the counter to run, clearing the bit disables the CTM. Clearing this bit to zero will stop the counter from counting and turn off the CTM 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 CTM is in the Compare Match Output Mode or the PWM Output Mode then the CTM output pin will be reset to its initial condition, as specified by the CTOC bit, when the CTON bit changes from low to high.

Bit 2~0 **CTRP2~CTRP0**: CTM CCRP 3-bit register, compared with the CTM Counter bit 9~bit 7 Comparator P Match Period
000: 1024 CTM clocks
001: 128 CTM clocks
010: 256 CTM clocks
011: 384 CTM clocks
100: 512 CTM clocks
101: 640 CTM clocks
110: 768 CTM clocks
111: 896 CTM 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 CTCCLR bit is set to zero. Setting the CTCCLR 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.

### CTMC1 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | CTM1 | CTM0 | CTIO1 | CTIO0 | CTOC | CTPOL | CTDPX | CTCCLR |
| 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 **CTM1~CTM0**: Select CTM Operating Mode
00: Compare Match Output Mode
01: Undefined
10: PWM Output Mode
11: Timer/Counter Mode

These bits setup the required operating mode for the CTM. To ensure reliable operation the CTM should be switched off before any changes are made to the CTM1 and CTM0 bits. In the Timer/Counter Mode, the CTM output pin state is undefined.

Bit 5~4 **CTIO1~CTIO0**: Select CTP output function

Compare Match Output Mode
00: No change
01: Output low
10: Output high
11: Toggle output

PWM Output 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 CTM output pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the CTM is running.

In the Compare Match Output Mode, the CTIO1 and CTIO0 bits determine how the CTM output pin changes state when a compare match occurs from the Comparator A. The CTM 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 CTM output pin should be setup using the CTOC bit in the CTMC1 register. Note that the output level requested by the CTIO1 and CTIO0 bits must be different from the initial value setup using the CTOC bit otherwise no change will occur on the CTM output pin when a compare match occurs. After the CTM output pin changes state it can be reset to its initial level by changing the level of the CTON bit from low to high.

In the PWM Output Mode, the CTIO1 and CTIO0 bits determine how the CTM 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 CTIO1 and CTIO0 bits only after the CTM has been switched off. Unpredictable PWM outputs will occur if the CTIO1 and CTIO0 bits are changed when The CTM is running.

Bit 3 **CTOC**: CTP Output control bit

Compare Match Output Mode
0: Initial low
1: Initial high

PWM Output Mode
0: Active low
1: Active high

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

Bit 2 **CTPOL**: CTP Output polarity Control
0: Non-invert
1: Invert

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

Bit 1 **CTDPX**: CTM 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 **CTCCLR**: Select CTM Counter clear condition
0: CTM Comparatror P match
1: CTM Comparatror A match

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

### CTMDL 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 CTM Counter Low Byte Register bit 7~bit 0
CTM 10-bit Counter bit 7~bit 0

### CTMDH 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 CTM Counter High Byte Register bit 1~bit 0
CTM 10-bit Counter bit 9~bit 8

### CTMAL 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 CTM CCRA Low Byte Register bit 7~bit 0
CTM 10-bit CCRA bit 7~bit 0

### CTMAH 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 CTM CCRA High Byte Register bit 1~bit 0
CTM 10-bit CCRA bit 9~bit 8

## Compact Type TM Operating Modes

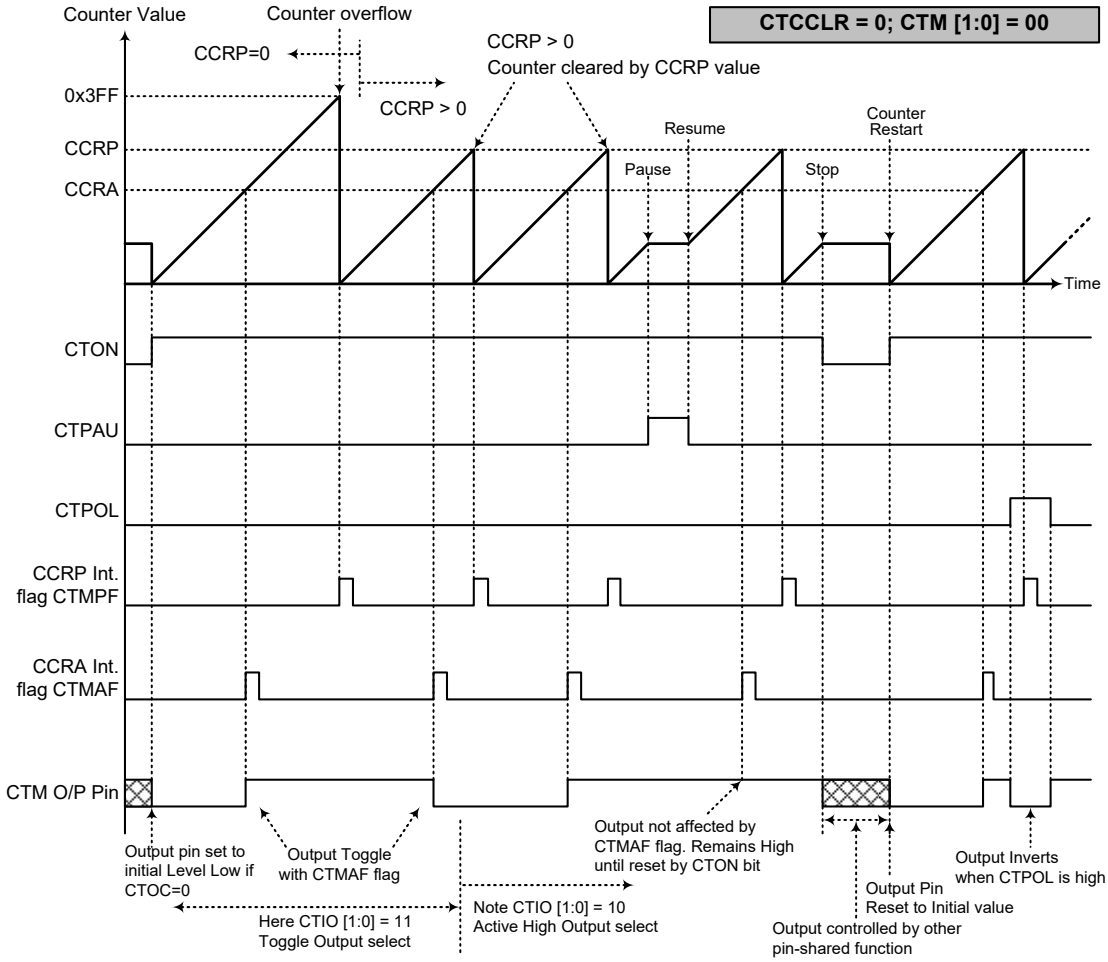
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTM1 and CTM0 bits in the CTMC1 register.

### Compare Match Output Mode

To select this mode, bits CTM1 and CTM0 in the CTMC1 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 CTCCLR 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 CTMAF and CTMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

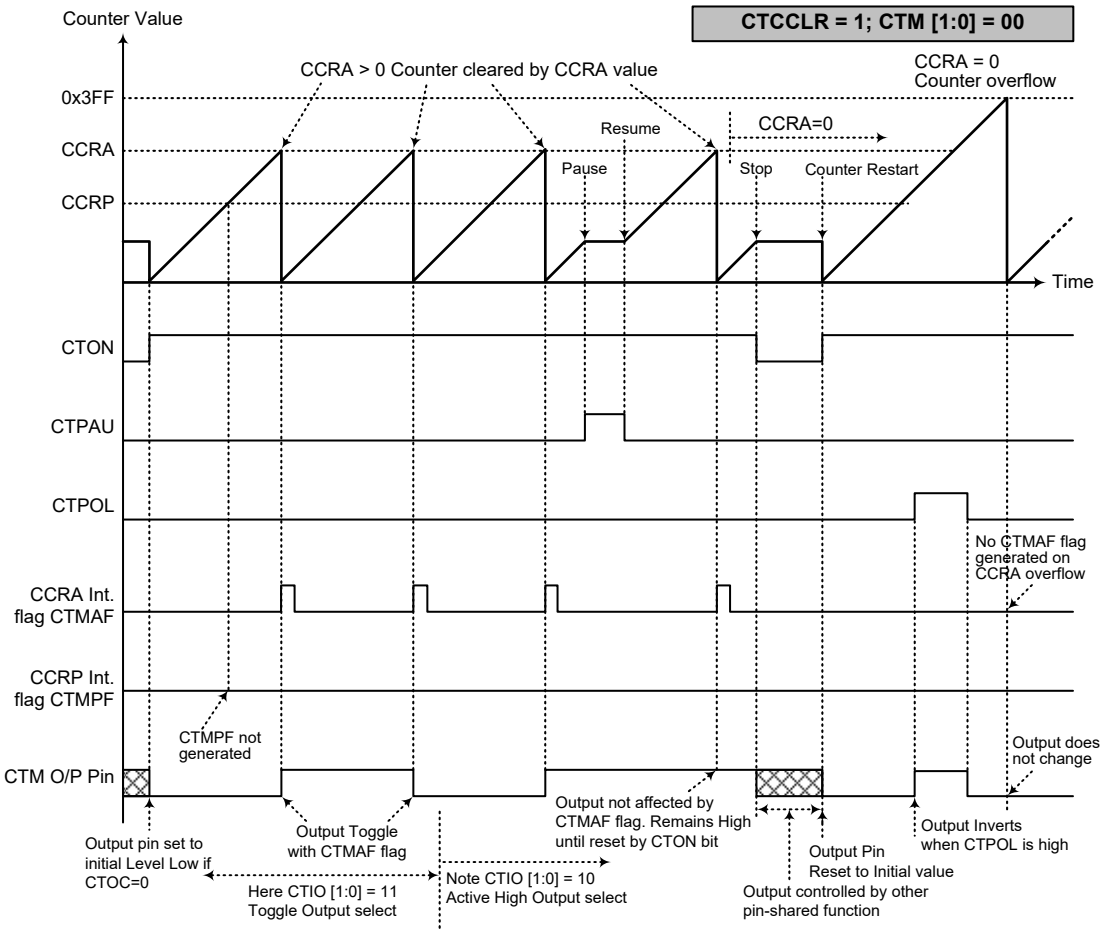
If the CTCCLR bit in the CTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTCCLR is high no CTMPF 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 CTMAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the CTM output pin will change state. The CTM output pin condition however only changes state when a CTMAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the CTM output pin. The way in which the CTM output pin changes state are determined by the condition of the CTIO1 and CTIO0 bits in the CTMC1 register. The CTM output pin can be selected using the CTIO1 and CTIO0 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 CTM output pin, which is setup after the CTON bit changes from low to high, is setup using the CTOC bit. Note that if the CTIO1 and CTIO0 bits are zero then no pin change will take place.




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

Note: 1. With CTCCLR=0, a Comparator P match will clear the counter
2. The CTM output pin controlled only by the CTMAF flag
3. The output pin reset to initial state by a CTON bit rising edge




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

Note: 1. With CTCCLR=1, a Comparator A match will clear the counter
2. The CTM output pin controlled only by the CTMAF flag
3. The output pin reset to initial state by a CTON rising edge
4. The CTMPF flags is not generated when CTCCLR=1

### Timer/Counter Mode

To select this mode, bits CTM1 and CTM0 in the CTMC1 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 CTM 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 CTM 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 CTM1 and CTM0 in the CTMC1 register should be set to 10 respectively. The PWM function within the CTM 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 CTM 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 Output Mode, the CTCCLR 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 CTDPX bit in the CTMC1 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 CTOC bit In the CTMC1 register is used to select the required polarity of the PWM waveform while the two CTIO1 and CTIO0 bits are used to enable the PWM output or to force the CTM output pin to a fixed high or low level. The CTPOL bit is used to reverse the polarity of the PWM output waveform.

- **CTM, PWM Output Mode, Edge-aligned Mode, CTDPX=0**

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

If $f_{SYS}$=8MHz, CTM clock source is $f_{SYS}$/4, CCRP=100b, CCRA =128,

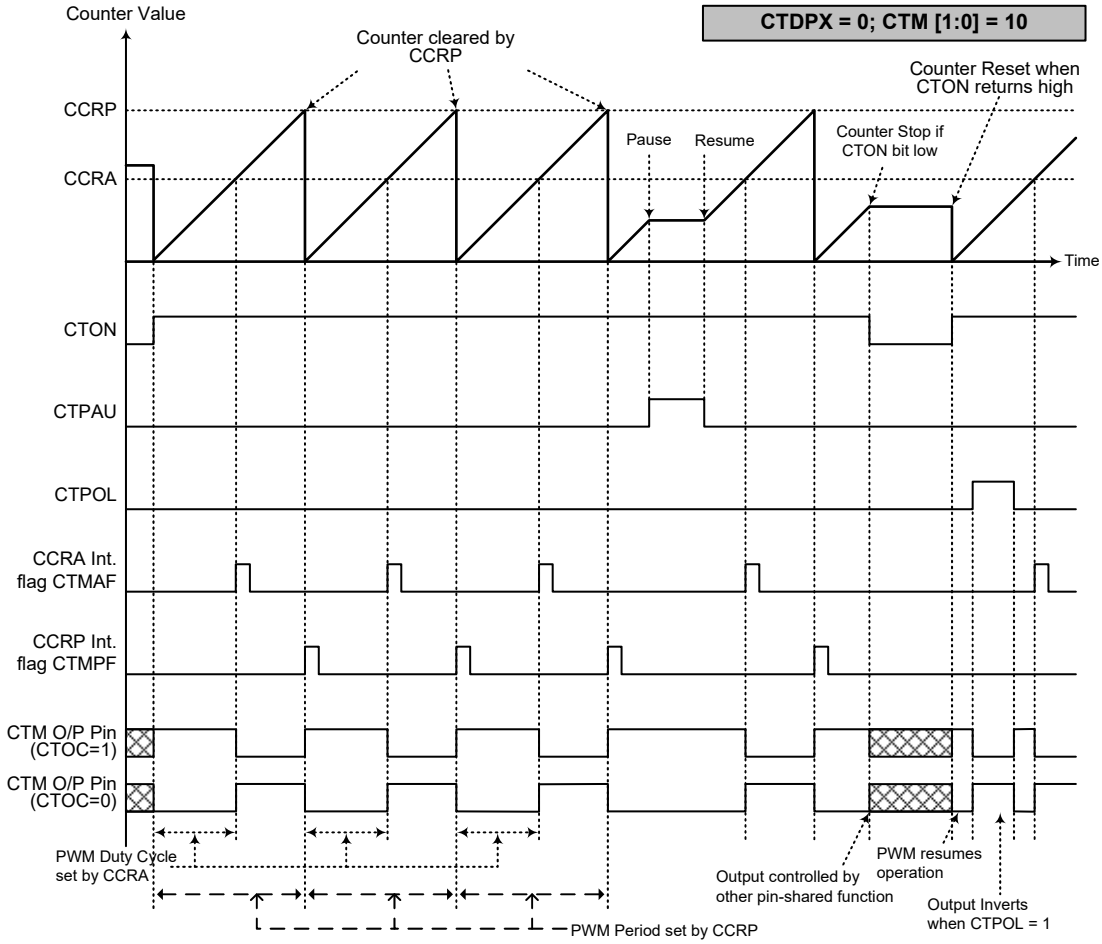
The CTM PWM output frequency=($f_{SYS}$/4) / 512=$f_{SYS}$/2048=3.9063kHz, duty=128/512=25%.

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

- **CTM, PWM Output Mode, Edge-aligned Mode, CTDPX=1**

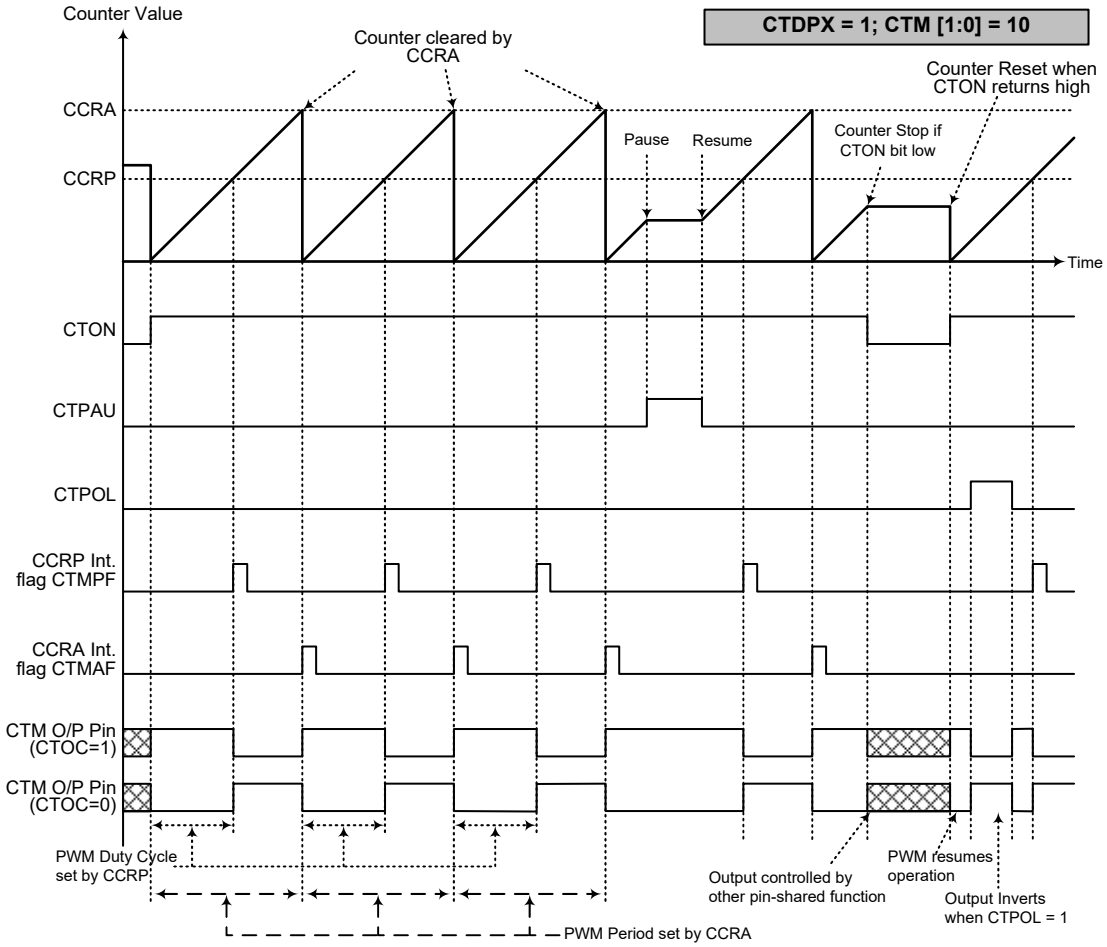
| CCRP | 001b | 010b | 011b | 100b | 101b | 110b | 111b | 000b |
|---|---|---|---|---|---|---|---|---|
| Period | CCRA | | | | | | | |
| Duty | 128 | 256 | 384 | 512 | 640 | 768 | 896 | 1024 |

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




**PWM Output Mode – CTDPX=0**

Note: 1. Here CTDPX=0 - Counter cleared by CCRP
2. A counter clear sets PWM Period
3. The internal PWM function continues running even when CTIO[1:0]=00 or 01
4. The CTCCLR bit has no influence on PWM operation




**PWM Output Mode – CTDPX=1**

Note: 1. Here CTDPX=1 - Counter cleared by CCRA
2. A counter clear sets PWM Period
3. The internal PWM function continues even when CTIO[1:0]=00 or 01
4. The CTCCLR bit has no influence on PWM operation

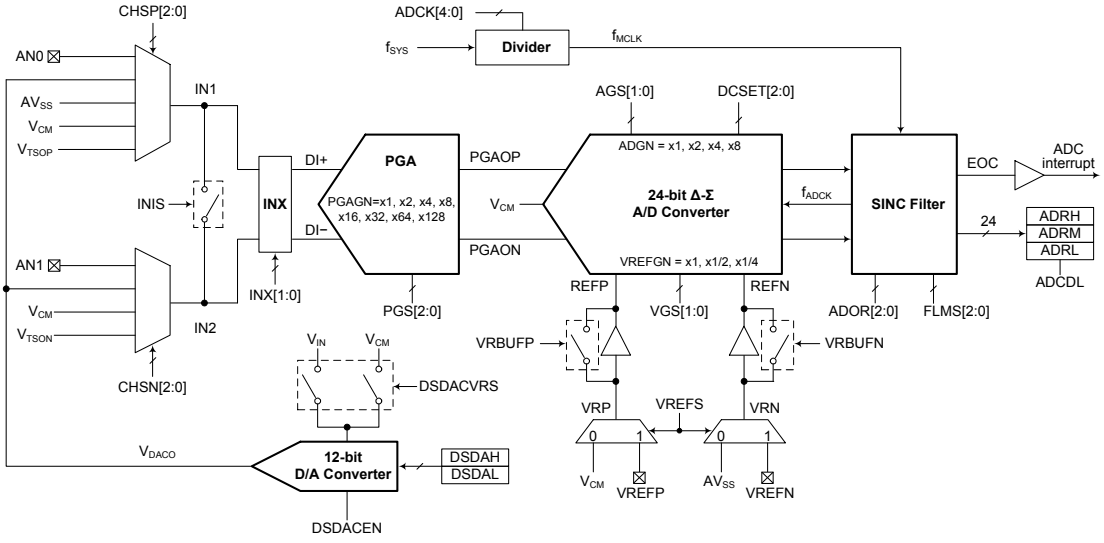
## Analog to Digital Converter

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

This device contains a high accuracy multi-channel 24-bit Delta Sigma A/D 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 24-bit digital value.

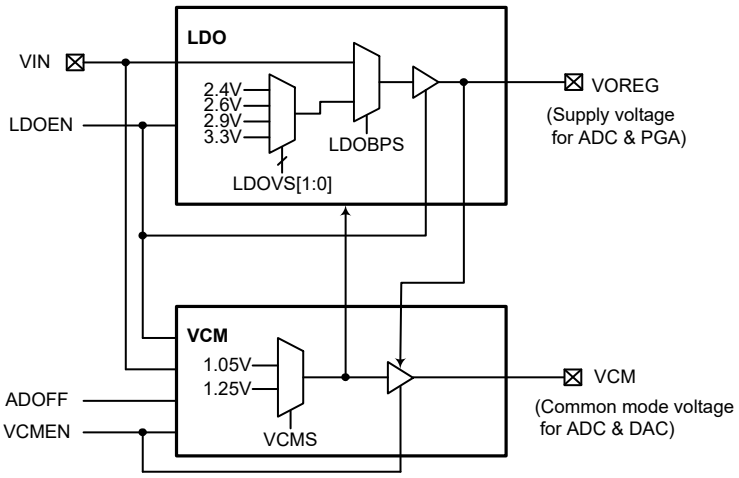
In addition, the PGA gain control, ADC gain control and ADC reference gain control determine the amplification gain for ADC input signal. The designer can select the best gain combination for the desired amplification applied to the input signal. The following block diagram illustrates the ADC basic operational function. The ADC input channel can be arranged as two single-ended A/D input channels or one differential input channel. The input signal can be amplified by PGA before entering the 24-bit Delta Sigma ADC. The Delta Sigma ADC modulator will output one bit converted data to SINC filter which can transform the converted one-bit data to 24 bits and store them into the specific data registers. Additionally, this device also provides a temperature sensor to compensate the A/D converter deviation caused by the temperature. With high accuracy and performance, this device is very suitable for the Weight Scale and similarly related products.




**A/D Converter Structure**

## Internal Power Supply

This device contains an LDO and VCM for the regulated power supply. The accompanying block diagram illustrates the basic functional operation. The internal LDO can provide the fixed voltage for PGA, ADC or the external components; as well the VCM can be used as the reference voltage for ADC or DAC module. There are four LDO voltage levels, 2.4V, 2.6V, 2.9V or 3.3V, decided by LDOVS1~LDOVS0 bits in the PWRC register, as well the VCM has two output voltage levels, 1.05V or 1.25V, selected by the VCMS bit in the PGAC1 register. The LDO and VCM functions can be controlled by the LDOEN and VCMEN bits respectively and can be powered off to reduce the power consumption.




**Internal Power Supply Block Diagram**

| Register Bits | | | Output voltage | |
|---|---|---|---|---|
| ADOFF | LDOEN | VCMEN | VOREG | VCM |
| 1 | 0 | x | Disable | Disable |
| 1 | 1 | x | Enable | Disable |
| 0 | 0 | 0 | Disable | Disable |
| 0 | 1 | 0 | Enable | Disable |
| 0 | 0 | 1 | Disable | Disable<br>(External voltage is not supplied on LDO output pin) |
| | | | | Enable<br>(External voltage must be supplied on LDO output pin) |
| 0 | 1 | 1 | Enable | Enable |

"x" : don't care

**Power Control Table**

**PWRC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | LDOEN | VCMEN | — | — | — | LDOBPS | LDOVS1 | LDOVS0 |
| R/W | R/W | R/W | — | — | — | R/W | R/W | R/W |
| POR | 0 | 0 | — | — | — | 0 | 0 | 0 |

Bit 7 **LDOEN**: LDO function control bit
0: Disable
1: Enable
If the LDO is disabled, there will be no power consumption and LDO output will be in a low level by a weakly pull-low resistor.

Bit 6 **VCMEN**: VCM function control bit
0: Disable
1: Enable
If the VCM is disabled, there will be no power consumption and VCM output pin is floating.

Bit 5~3 Unimplemented, read as "0"

Bit 2 **LDOBPS**: LDO Bypass function control bit
0: Disable
1: Enable

Bit 1~0 **LDOVS1~LDOVS0**: LDO output voltage selection
00: 2.4V
01: 2.6V
10: 2.9V
11: 3.3V

## A/D Data Rate Definition

The Delta Sigma A/D converter data rate can be calculated using the following equation:

$$\text{Data Rate} = \frac{f_{ADCK}}{CHOP \times OSR} = \frac{f_{MCLK}/N}{CHOP \times OSR} = \frac{f_{MCLK}}{N \times CHOP \times OSR}$$

$f_{ADCK}$: A/D clock input, derived from $f_{MCLK}/N$

$f_{MCLK}$: A/D clock source, derived from $f_{SYS}$ or $f_{SYS}/2/(ADCK+1)$ using the ADCK bit field.

N: a constant divided factor that can be equal to 30 or 12 determined by the FLMS bit field.

CHOP: Sampling data amount doubling function control and can be equal to 2 or 1 determined by the FLMS bit field.

OSR: Oversampling rate determined by the ADOR field.

For example, if a data rate of 8Hz is desired, an $f_{MCLK}$ clock source with a frequency of 4MHz A/D converter can be selected. Then set the FLMS field to "000" to obtain an "N" equal to 30 and "CHOP" equal to 2. Finally, set the ADOR field to "001" to select an oversampling rate equal to 8192. Therefore, the Data Rate=4MHz / (30 × 2 × 8192)=8Hz.

Note that the A/D converter has a notch rejection function for an A/C power supply with a frequency of 50Hz or 60Hz when the data rate is equal to 10Hz.

## A/D Converter Register Description

Overall operation of the A/D converter is controlled by using a series of registers. Three read only registers exist to store the A/D converter data 24-bit value. A control register named as PWRC is used to control the required bias and supply voltages for PGA and A/D converter and is described in the "Internal Power Supply" section. Three registers are used for overall control of the D/A converter. The remaining 6 registers are control registers which set up the gain selections and control functions of the A/D converter.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PWRC | LDOEN | VCMEN | — | — | — | LDOBPS | LDOVS1 | LDOVS0 |
| PGAC0 | — | VGS1 | VGS0 | AGS1 | AGS0 | PGS2 | PGS1 | PGS0 |
| PGAC1 | VCMS | INIS | INX1 | INX0 | DCSET2 | DCSET1 | DCSET0 | — |
| PGACS | — | — | CHSN2 | CHSN1 | CHSN0 | CHSP2 | CHSP1 | CHSP0 |
| ADRL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| ADRM | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |
| ADRH | D23 | D22 | D21 | D20 | D19 | D18 | D17 | D16 |
| ADCR0 | ADRST | ADSLP | ADOFF | ADOR2 | ADOR1 | ADOR0 | — | VREFS |
| ADCR1 | FLMS2 | FLMS1 | FLMS0 | VRBUFN | VRBUFP | ADCDL | EOC | — |
| ADCS | — | — | — | ADCK4 | ADCK3 | ADCK2 | ADCK1 | ADCK0 |
| DSDAH | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 |
| DSDAL | — | — | — | — | D3 | D2 | D1 | D0 |
| DSDACC | DSDACEN | DSDACVRS | — | — | — | — | — | — |

**A/D Converter Register List**

### Programmable Gain Amplifier – PGA

There are three registers related to the programmable gain control, PGAC0, PGAC1 and PGACS. The PGAC0 resister is used to select the PGA gain, ADC gain and the ADC reference gain. As well, the PGAC1 register is used to define the input connection, differential input offset voltage adjustment control and the VCM voltage selection. In addition, The PGACS register is used to select the input ends for the PGA. Therefore, the input channels have to be determined by the CHSP2~CHSP0 and CHSN2~CHSN0 bits to determine which analog channel input pins, temperature detector inputs or internal power supply are actually connected to the internal differential A/D converter.

- **PGAC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | VGS1 | VGS0 | AGS1 | AGS0 | PGS2 | PGS1 | PGS0 |
| 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 6~5 **VGS1~VGS0**: REFP/REFN differential reference voltage gain selection
00: VREFGN=1
01: VREFGN=1/2
10: VREFGN=1/4
11: Reserved

Bit 4~3 **AGS1~AGS0**: A/D converter PGAOP/PGAON differential input signal gain selection
00: ADGN=1
01: ADGN=2
10: ADGN=4
11: ADGN=8

Bit 2~0 **PGS2~PGS0**: DI+/DI- differential channel input gain selection
000: PGAGN=1
001: PGAGN=2
010: PGAGN=4
011: PGAGN=8
100: PGAGN=16
101: PGAGN=32
110: PGAGN=64
111: PGAGN=128

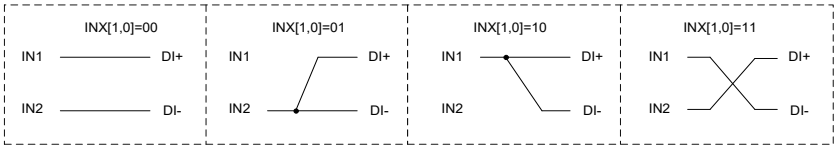
- **PGAC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | VCMS | INIS | INX1 | INX0 | DCSET2 | DCSET1 | DCSET0 | — |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | — |
| POR | 1 | 0 | 0 | 0 | 0 | 0 | 0 | — |

Bit 7 **VCMS**: Analog Common mode voltage selection
0: 1.05V
1: 1.25V

Bit 6 **INIS**: The selected input ends, IN1 and IN2 internal connection control bit
0: Not connected
1: Connected

Bit 5~4 **INX1~INX0**: The selected input ends, IN1/IN2 and the PGA differential input ends, DI+/DI- connection control bits




Bit 3~1 **DCSET2~DCSET0**: Differential input signal PGAOP/PGAON offset selection
000: DCSET=+0V
001: DCSET=+0.25×ΔVR_I
010: DCSET=+0.5×ΔVR_I
011: DCSET=+0.75×ΔVR_I
100: DCSET=+0V
101: DCSET=-0.25×ΔVR_I
110: DCSET=-0.5×ΔVR_I
111: DCSET=-0.75×ΔVR_I
The voltage, ΔVR_I, is the differential reference voltage which is amplified by specific gain selection based on the selected inputs.

Bit 0 Unimplemented, read as "0"

- **PGACS Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | CHSN2 | CHSN1 | CHSN0 | CHSP2 | CHSP1 | CHSP0 |
| 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 5~3 **CHSN2~CHSN0**: Negative input end IN2 selection
000: AN1
001: Reserved
010: Reserved
011: Reserved
100: DACO
101: Reserved
110: $V_{CM}$
111: Temperature sensor output – $V_{TSON}$
These bits are used to select the negative input, IN2. If the IN2 input is selected as a single end input, the $V_{CM}$ voltage must be selected as the positive input on IN1 for single end input applications. It is recommended that when the $V_{TSON}$ signal is selected as the negative input, the $V_{TSOP}$ signal should be selected as the positive input for proper operations.

Bit 2~0 **CHSP2~CHSP0**: Positive input end IN1 selection
000: AN0
001: Reserved
010: Reserved
011: Reserved
100: DACO
101: Unused, connected to AVSS
110: $V_{CM}$
111: Temperature sensor output – $V_{TSOP}$
These bits are used to select the positive input, IN1. If the IN1 input is selected as a single end input, the $V_{CM}$ voltage must be selected as the negative input on IN2 for single end input applications. It is recommended that when the $V_{TSOP}$ signal is selected as the positive input, the $V_{TSON}$ signal should be selected as the negative input for proper operations.

## D/A Converter Registers – DSDAH, DSDAL, DSDACC

There are three registers related to the D/A converter control.

- **DSDAH Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 |
| 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 **D11~D4**: D/A Converter Output Control Code, Only for 12 bits D/A Converter

- **DSDAL Register**

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~0 **D3~D0**: D/A Converter Output Control Code

Note: writing this register only writes to shadow buffer, and until write DSDAH register will also copy the shadow buffer data to DSDAL register.

- **DSDACC Register**

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

Bit 7 **DSDACEN**: D/A Converter Enable or Disable Control Bit
0: Disable
1: Enable

Bit 6 **DSDACVRS**: D/A Converter Reference Voltage Selection
0: D/A converter reference voltage comes from $V_{IN}$
1: D/A converter reference voltage comes from $V_{CM}$

Bit 5~0 Unimplemented, read as "0"

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

This device contains an internal 24-bit Delta Sigma A/D converter, it requires three data registers to store the converted value. These are a high byte register, known as ADRH, a middle byte register, known as ADRM, 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. D0~D23 are the A/D conversion result data bits.

- **ADRL 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 | x | x | x | x | x | x | x | x |

"x": unknown

Bit 7~0 A/D conversion data Register bit 7~bit 0

- **ADRM Register**

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

"x": unknown

Bit 7~0 A/D conversion data Register bit 15~bit 8

- **ADRH Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | D23 | D22 | D21 | D20 | D19 | D18 | D17 | D16 |
| R/W | R | R | R | R | R | R | R | R |
| POR | x | x | x | x | x | x | x | x |

"x": unknown

Bit 7~0 A/D conversion data Register bit 23~bit 16

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

To control the function and operation of the A/D converter, three control registers known as ADCR0, ADCR1 and ADCS are provided. These 8-bit registers define functions such as the selection of which reference source is used to the internal ADC, the ADC clock source , the ADC output data rate as well as controlling the power-up function and monitoring the ADC end of conversion status.

- **ADCR0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | ADRST | ADSLP | ADOFF | ADOR2 | ADOR1 | ADOR0 | — | VREFS |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | — | R/W |
| POR | 0 | 0 | 1 | 0 | 0 | 0 | — | 0 |

Bit 7 **ADRST**: A/D converter software reset control bit
0: Disable
1: Enable
This bit is used to reset the A/D converter internal digital SINC filter. This bit is set low for A/D normal operations. However, if set high, the internal digital SINC filter will be reset and the current A/D converted data will be aborted. A new A/D data conversion process will not be initiated until this bit is set low again.

Bit 6 **ADSLP**: A/D converter sleep mode control bit
0: Normal mode
1: Sleep mode
This bit is used to determine whether the A/D converter enters the sleep mode or not when the A/D converter is powered on by setting the ADOFF bit low. When the A/D converter is powered on and the ADSLP bit is low, the A/D converter will operate normally. However, the A/D converter will enter the sleep mode if the ADSLP bit is set high as the A/D converter has been powered on. The whole A/D converter circuit will be switched off except the PGA and internal Bandgap circuit to reduce the power consumption and $V_{CM}$ start-up stable time.

Bit 5 **ADOFF**: A/D converter module power on/off control bit
0: Power on
1: Power off
This bit controls the power of the A/D converter module. 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.
It is recommended to set the ADOFF bit high before the device enters the IDLE/SLEEP mode for saving power. Setting the ADOFF bit high will power down the A/D converter module regardless of the ADSLP and ADRST bit settings.

Bit 4~2 **ADOR2~ADOR0**: A/D conversion oversampling rate selection
000: Oversampling rate OSR=16384
001: Oversampling rate OSR=8192
010: Oversampling rate OSR=4096
011: Oversampling rate OSR=2048
100: Oversampling rate OSR=1024
101: Oversampling rate OSR=512
110: Oversampling rate OSR=256
111: Oversampling rate OSR=128

Bit 1 Unimplemented, read as "0"

Bit 0 **VREFS**: A/D converter reference voltage pair selection
0: Internal reference voltage pair – $V_{CM}$ & $AV_{SS}$
1: External reference voltage pair – VREFP & VREFN

- **ADCR1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | FLMS2 | FLMS1 | FLMS0 | VRBUFN | VRBUFP | ADCDL | EOC | — |
| 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 7~5 **FLMS2~FLMS0**: A/D converter clock divided ratio selection and sampled data doubling function (CHOP) enable control
000: CHOP=2, $f_{ADCK}=f_{MCLK}$ / 30
010: CHOP=2, $f_{ADCK}=f_{MCLK}$ / 12
100: CHOP=1, $f_{ADCK}=f_{MCLK}$ / 30
110: CHOP=1, $f_{ADCK}=f_{MCLK}$ / 12
Others: Reserved
When the CHOP bit is equal to 2, it means that the sampled data amount will be doubled for the normal conversion mode. However, it can be regarded as the low latency conversion mode if the CHOP bit is equal to 1, which means that the sampled data doubling function is disabled.

Bit 4 **VRBUFN**: A/D converter negative reference voltage input (VRN) buffer control
0: Disable input buffer and enable bypass function
1: Enable input buffer and disable bypass function

Bit 3 **VRBUFP**: A/D converter positive reference voltage input (VRP) buffer control
0: Disable input buffer and enable bypass function
1: Enable input buffer and disable bypass function

Bit 2 **ADCDL**: A/D converted data latch function enable control
0: Disable data latch function
1: Enable data latch function
If the A/D converted data latch function is enabled, the latest converted data value will be latched and not be updated by any subsequent converted results until this function is disabled. Although the converted data is latched into the data registers, the A/D converter circuits remain operational, but will not generate interrupt and EOC will not change. It is recommended that this bit should be set high before reading the converted data in the ADRL, ADRM and ADRH registers. After the converted data has been read out, the bit can then be cleared to low to disable the A/D converter data latch function and allow further conversion values to be stored. In this way, the possibility of obtaining undesired data during A/D converter conversions can be prevented.

Bit 1 **EOC**: End of A/D conversion flag
0: A/D conversion in progress
1: A/D conversion ended
This bit must be cleared by software.

Bit 0 Unimplemented, read as "0"

- **ADCS Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | — | ADCK4 | ADCK3 | ADCK2 | ADCK1 | ADCK0 |
| 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 **ADCK4~ADCK0**: A/D converter clock source $f_{MCLK}$ divided ratio selection
00000~11110: $f_{MCLK}=f_{SYS}/2/(ADCK[4:0]+1)$
11111: $f_{MCLK}=f_{SYS}$

## A/D Operation

The A/D Converter provides four operating modes, which are the Normal mode, Power down mode, Sleep mode and Reset mode, controlled respectively by the ADOFF, ADSLP and ADRST bits in the ADCR0 register. The following table illustrates the operating mode selection.

| LDOEN | ADOFF | ADSLP | ADRST | Operating mode | Description |
|---|---|---|---|---|---|
| 0 | 1 | x | x | Power down mode | Bandgap off, LDO off,<br>$V_{CM}$ generator off, PGA off, ADC off,<br>Temperature sensor off,<br>VRN/VRP buffer off, SINC filter off |
| 1 | 1 | x | x | Power down mode | Bandgap on, LDO on,<br>$V_{CM}$ generator off, PGA off, ADC off,<br>Temperature sensor off,<br>VRN/VRP buffer off, SINC filter off |
| 0 | 0 | 1 | x | Sleep mode<br>(External voltage must be supplied on LDO output pin) | Bandgap on, LDO off,<br>$V_{CM}$ generator off, PGA on, ADC off,<br>Temperature sensor off,<br>VRN/VRP buffer off, SINC filter on |
| 0 | 0 | 0 | 0 | Normal mode<br>(External voltage must be supplied on LDO output pin) | Bandgap on, LDO off,<br>$V_{CM}$ generator on/off(1),<br>PGA on, ADC on,<br>Temperature sensor on/off(2),<br>VRN/VRP buffer on/off(3),<br>SINC filter on |
| 0 | 0 | 0 | 1 | Reset mode<br>(External voltage must be supplied on LDO output pin) | Bandgap on, LDO off,<br>$V_{CM}$ generator on/off(1),<br>PGA on, ADC on,<br>Temperature sensor on/off(2),<br>VRN/VRP buffer on/off(3),<br>SINC filter Reset |
| 1 | 0 | 1 | x | Sleep mode | Bandgap on, LDO on,<br>$V_{CM}$ generator off, PGA on, ADC off,<br>Temperature sensor off,<br>VRN/VRP buffer off, SINC filter on |
| 1 | 0 | 0 | 0 | Normal mode | Bandgap on, LDO on,<br>$V_{CM}$ generator on/off(1),<br>PGA on, ADC on,<br>Temperature sensor on/off(2),<br>VRN/VRP buffer on/off(3),<br>SINC filter on |
| 1 | 0 | 0 | 1 | Reset mode | Bandgap on, LDO on,<br>$V_{CM}$ generator on/off(1),<br>PGA on, ADC on,<br>Temperature sensor on/off(2),<br>VRN/VRP buffer on/off(3),<br>SINC filter Reset |

"x": unknown

Note: 1. The $V_{CM}$ generator can be switched on or off by configuring the VCMEN bit.

2. The Temperature sensor can be switched on or off by configuring the CHSN[2:0] or CHSP[2:0] bits.

3. The VRN buffer can be switched on or off by configuring the VRBUFN bit while the VRP buffer can be switched on or off by configuring the VRBUFP bit

**A/D Operation Mode Selection**

To enable the A/D Converter, the first step is to disable the ADC power down and sleep mode by clearing the ADOFF and ADSLP bits to make sure the A/D Converter is powered up. The ADRST bit in the ADCR0 register is used to start and reset the A/D converter after power on. To set ADRST bit from low to high and then low again, an analog to digital converted data in SINC filter will be initiated. After this setup is complete, the A/D Converter is ready for operation. These three bits are used to control the overall start operation of the internal analog to digital converter.

The EOC bit in the ADCR1 register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set to "1" by the Hardware 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 poll the EOC bit in the ADCR1 register to check whether it has been set "1" as an alternative method of detecting the end of an A/D conversion cycle. The ADC converted data will be updated continuously by the new converted data. If the ADC converted data latch function is enabled, the latest converted data will be latched and the following new converted data will be discarded until this data latch function is disabled.

The clock source for the A/D converter should be typically fixed at a value of 4MHz, which originates from the system clock $f_{SYS}$, and can be chosen to be either $f_{SYS}$ or a subdivided version of $f_{SYS}$. The division ratio value is determined by the ADCK4~ADCK0 bits in the ADCS register to obtain a 4MHz clock source for the ADC.

The differential reference voltage supply to the A/D Converter can be supplied from either the internal power supply, VCM and AVSS, or from an external reference source, VREFP and VREFN. The desired selection is made using the VREFS bit in the ADCR0 register.

## 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
  Enable the power LDO, VCM for PGA and ADC.
- Step 2
  Select the PGA, ADC, reference voltage gains by PGAC0 register
- Step 3
  Select the PGA settings for input connection, VCM voltage level and buffer option by PGAC1 register
- Step 4
  Select the required A/D conversion clock source by correctly programming bits ADCK4~ADCK0 in the ADCS register.
- Step 5
  Select output data rate by configuring the ADOR[2:0] bits in the ADCR0 register and FLMS[2:0] bits in the ADCR1 register.
- Step 6
  Select which channel is to be connected to the internal PGA by correctly programming the CHSP2~CHSP0 and CHSN2~CHSN0 bits which are also contained in the PGACS register.
- Step 7
  Release the power down mode and sleep mode by clearing the ADOFF and ADSLP bits in ADCR0 register.
- Step 8
  Reset the A/D by setting the ADRST to high in the ADCR0 register and clearing this bit to zero to release reset status.
- Step 9
  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 10
  To check when the analog to digital conversion process is complete, the EOC bit in the ADCR1 register can be polled. The conversion process is complete when this bit goes high. When this occurs the A/D data registers ADRL, ADRM 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 EOC bit in the ADCR1 register is used, the interrupt enable step above can be omitted.

## 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.

## A/D Transfer Function

This device contains a 24-bit Delta Sigma A/D converter and its full-scale converted digitized value is from 8388607 to -8388608 in decimal value. The converted data format is formed by a two's complement binary value. The MSB of the converted data is the signed bit. Since the full-scale analog input value is equal to the amplified value of the $V_{CM}$ or differential reference input voltage, ΔVR_I, selected by the VREFS bit in ADCR0 register, this gives a single bit analog input value of ΔVR_I divided by 8388608.

1 LSB=ΔVR_I / 8388608

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

ΔSI_I=(PGAGN × ADGN × ΔDI±) + DCSET

ΔVR_I=VREGN × ΔVR±

ADC_Conversion_Data=(ΔSI_I / ΔVR_I) × K

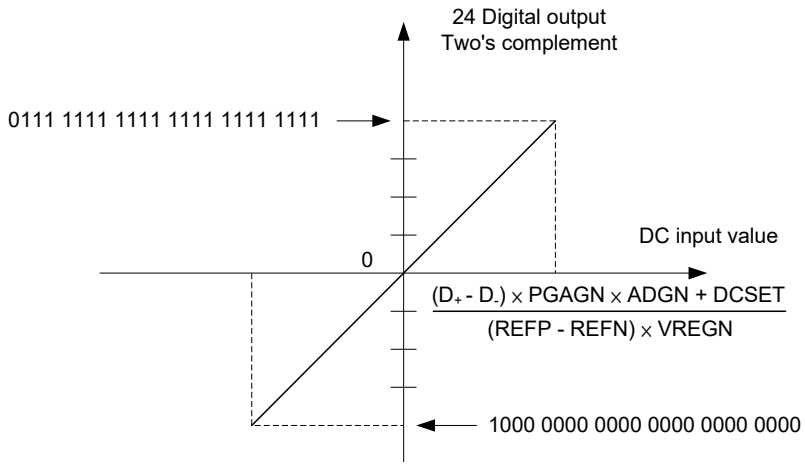
Where K is equal to $2^{23}$

Note: 1. The PGAGN, ADGN, VREGN values are decided by the PGS, AGS, VGS control bits.
2. ΔSI_I: Differential Input Signal after amplification and offset adjustment.
3. PGAGN: Programmable Gain Amplifier gain
4. ADGN: A/D Converter gain
5. VREGN: Reference voltage gain
6. ΔDI±: Differential input signal derived from external channels or internal signals
7. DCSET: Offset voltage
8. ΔVR±: Differential Reference voltage
9. ΔVR_I: Differential Reference input voltage after amplification

Due to the digital system design of the Delta Sigma A/D Converter, the maximum number of the A/D converted value is 8388607 and the minimum value is -8388608, therefore, we can have the middle number 0. The ADC_Conversion_Data equation illustrates this range of converted data variation.

| A/D Conversion Data (2's compliment, Hexadecimal) | Decimal Value |
|---|---|
| 0x7FFFFF | 8388607 |
| 0x800000 | -8388608 |

The above ADC conversion data table illustrates the range of ADC conversion data.
The following diagram shows the relationship between the DC input value and the ADC converted data which is presented by the Two's Complement.



## A/D Converted Data

The A/D converted data is related to the input voltage and the PGA selections. The format of the ADC output is a two's complement binary code. The length of this output code is 24 bits and the MSB is a signed bit. When the MSB is "0", which represents the input is "positive" , on the other hand, as the MSB is "1", it represents the input is "negative". The maximum value is 8388607 and the minimum value is -8388608. If the input signal is over the maximum value, the converted data is limited by the 8388607, and if the input signal is less than the minimum value, the converted data is limited by -8388608.

## A/D Converted Data to Voltage

The designer can recover the converted data by the following equations:

If MSB=0 (Positive Converted data):

$$\text{Input Voltage} = \frac{(\text{Converted data}) \times \text{LSB} - \text{DCSET}}{\text{PGA} \times \text{ADGN}}$$

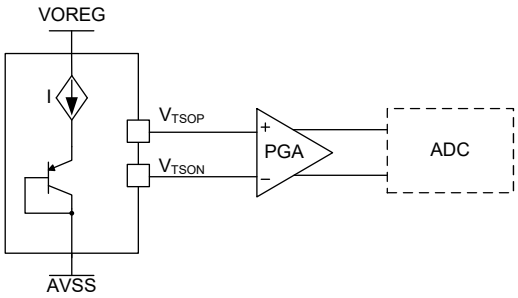
If the MSB=1 (Negative Converted data):

$$\text{Input voltage} = \frac{(\text{Two's\_complement\_of\_Converted\_data}) \times \text{LSB} - \text{DCSET}}{\text{PGA} \times \text{ADGN}}$$

Note: Two's complement=One's complement +1

## Temperature Sensor

This device provides an internal temperature sensor to compensate the device performance. By selecting the PGA input channels to $V_{TSOP}$ and $V_{TSON}$ signals, the A/D Converter can get the temperature information and the designer can do some compensation to the A/D converted data. The following block diagram illustrates the functional operation for the temperature sensor.




**Temperature Sensor Structure**

## Serial Interface Module – SIM

The device contains a Serial Interface Module, which includes both the four line SPI interface and the two line $I^2C$ interface types, to allow an easy method of communication with external peripheral hardware. Having relatively simple communication protocols, these serial interface types allow the microcontroller to interface to external SPI or $I^2C$ based hardware such as sensors, Flash or EEPROM memory, etc. The SIM interface pins are pin-shared with other I/O pins therefore the SIM interface functional pins must first be selected using the corresponding pin-shared function selection bits. As both interface types share the same pins and registers, the choice of whether the SPI or $I^2C$ type is used is made using the SIM operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the SIM pin-shared I/O are selected using pull-high control registers when the SIM function is enabled and the corresponding pins are used as SIM input pins.

### SPI Interface

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

The communication is full duplex and operates as a slave/master type, where the device can be either master or slave. Although the SPI interface specification can control multiple slave devices from a single master, but the device provides only one $\overline{SCS}$ pin. If the master needs to control multiple slave devices from a single master, the master can use I/O pin to select the slave devices.

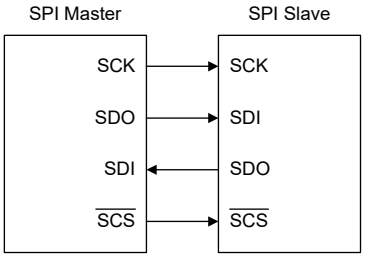
#### SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four line interface with pin names SDI, SDO, SCK and $\overline{SCS}$. Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, the SCK pin is the Serial Clock line and $\overline{SCS}$ is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the $I^2C$ function pins, the SPI interface pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMC0 and SIMC2 registers. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{SCS}$ pin only one slave device can be utilized. The $\overline{SCS}$ pin is controlled by software, set CSEN bit to 1 to enable $\overline{SCS}$ pin function, set CSEN bit to 0 the $\overline{SCS}$ pin will be floating state.

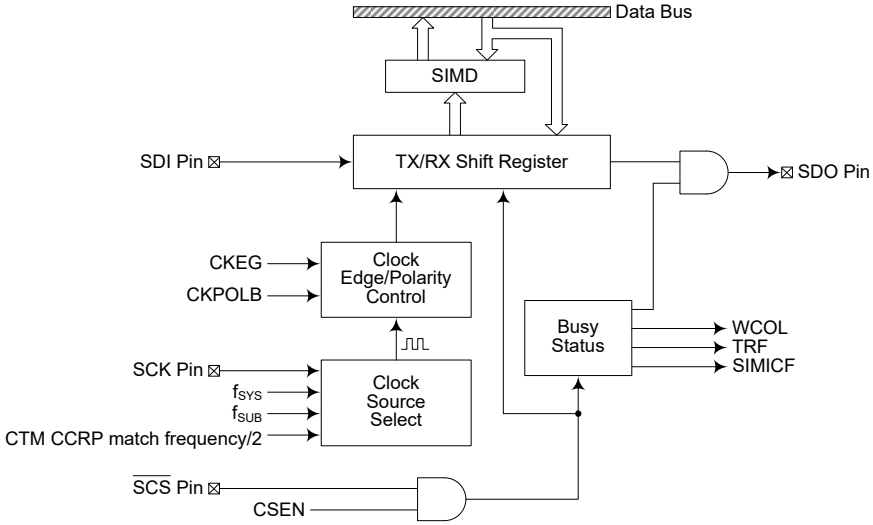
The SPI function in the device offers the following features:

- Full duplex synchronous data transfer
- Both Master and Slave modes
- LSB first or MSB first data transmission modes
- Transmission complete flag
- Rising or falling active clock edge

The status of the SPI interface pins is determined by a number of factors such as whether the device is in the master or slave mode and upon the condition of certain control bits such as CSEN and SIMEN.




**SPI Master/Slave Connection**




**SPI Block Diagram**

## SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two control registers, SIMC0 and SIMC2.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SIMC0 | SIM2 | SIM1 | SIM0 | — | SIMDEB1 | SIMDEB0 | SIMEN | SIMICF |
| SIMC2 | D7 | D6 | CKPOLB | CKEG | MLS | CSEN | WCOL | TRF |
| SIMD | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

**SPI Registers List**

### SPI Data Register

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

- **SIMD 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 | x | x | x | x | x | x | x | x |

"x": unknown

Bit 7~0 **D7~D0**: SIM data register bit 7~bit 0

### SPI Control Registers

There are also two control registers for the SPI interface, SIMC0 and SIMC2. Note that the SIMC2 register also has the name SIMA which is used by the I²C function. The SIMC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SIMC2 register is used for other control functions such as LSB/MSB selection, write collision flag etc.

- **SIMC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SIM2 | SIM1 | SIM0 | — | SIMDEB1 | SIMDEB0 | SIMEN | SIMICF |
| R/W | R/W | R/W | R/W | — | R/W | R/W | R/W | R/W |
| POR | 1 | 1 | 1 | — | 0 | 0 | 0 | 0 |

Bit 7~5 **SIM2~SIM0**: SIM Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is $f_{SUB}$
100: SPI master mode; SPI clock is CTM CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused mode
These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from CTM and $f_{SUB}$. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
These bits are only available when the SIM is configured to operate in the I²C mode. Refer to the I²C register section.

Bit 1 **SIMEN**: SIM Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or $I^2C$ function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an $I^2C$ interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the $I^2C$ control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant $I^2C$ flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: SIM SPI Incomplete Flag
0: SIM SPI incomplete condition is not occurred
1: SIM SPI incomplete condition is occurred
This bit is only available when the SIM is configured to operate in an SPI slave mode. If the SPI operates in the slave mode with the SIMEN and CSEN bits both being set to 1 but the $\overline{SCS}$ line is pulled high by the external master device before the SPI data transfer is completely finished, the SIMICF bit will be set to 1 together with the TRF bit. When this condition occurs, the corresponding interrupt will occur if the interrupt function is enabled. However, the TRF bit will not be set to 1 if the SIMICF bit is set to 1 by software application program.

- **SIMC2 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | D7 | D6 | CKPOLB | CKEG | MLS | CSEN | WCOL | TRF |
| 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 D7~D6: Undefined bits
These bits can be read or written by the application program.

Bit 5 **CKPOLB**: SPI clock line base condition selection
0: The SCK line will be high when the clock is inactive
1: The SCK line will be low when the clock is inactive
The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.

Bit 4 **CKEG**: SPI SCK clock active edge type selection
CKPOLB=0
0: SCK is high base level and data capture at SCK rising edge
1: SCK is high base level and data capture at SCK falling edge
CKPOLB=1
0: SCK is low base level and data capture at SCK falling edge
1: SCK is low base level and data capture at SCK rising edge
The CKEG and CKPOLB bits are used to setup the way that the clock signal outputs and inputs data on the SPI bus. These two bits must be configured before data transfer is executed otherwise an erroneous clock edge may be generated. The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive. The CKEG bit determines active clock edge type which depends upon the condition of CKPOLB bit.

Bit 3 **MLS**: SPI data shift order
0: LSB first
1: MSB first
This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

Bit 2 **CSEN**: SPI $\overline{SCS}$ pin control
0: Disable
1: Enable
The CSEN bit is used as an enable/disable for the $\overline{SCS}$ pin. If this bit is low, then the $\overline{SCS}$ pin will be disabled and placed into a floating condition. If the bit is high the $\overline{SCS}$ pin will be enabled and used as a select pin.

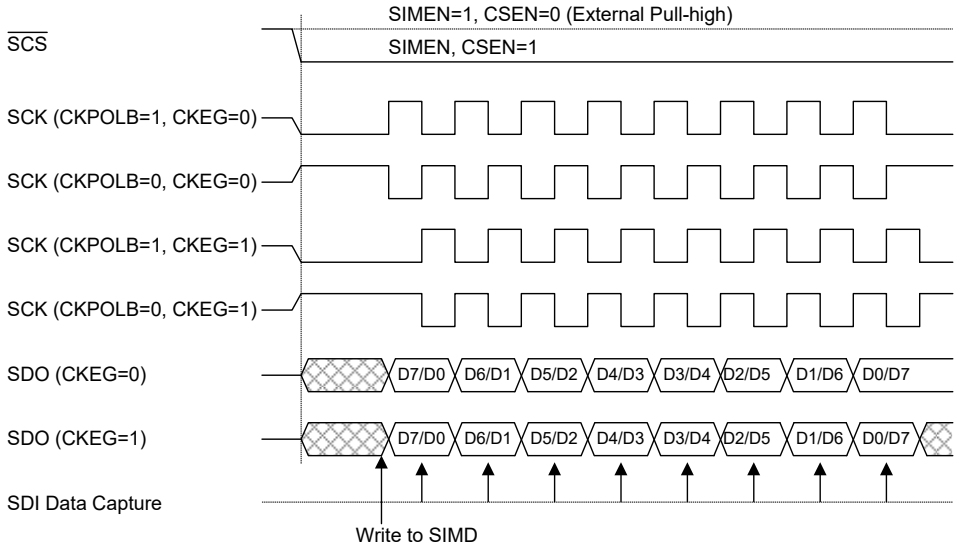
Bit 1 **WCOL**: SPI write collision flag
0: No collision
1: Collision
The WCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared by the application program.

Bit 0 **TRF**: SPI Transmit/Receive complete flag
0: SPI data is being transferred
1: SPI data transmission is completed
The TRF bit is the Transmit/Receive Complete flag and is set "1" automatically when an SPI data transmission is completed, but must set to "0" by the application program. It can be used to generate an interrupt.

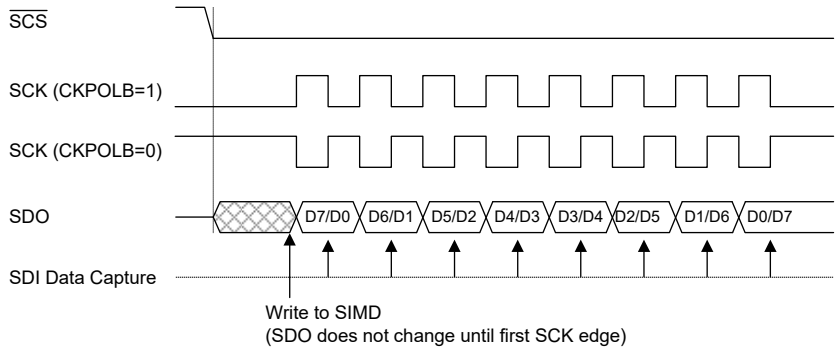
### SPI Communication

After the SPI interface is enabled by setting the SIMEN bit high, then in the Master Mode, when data is written to the SIMD register, transmission/reception will begin simultaneously. When the data transfer is completed, the TRF flag will be set automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SIMD register will be transmitted and any data on the SDI pin will be shifted into the SIMD register. The master should output an $\overline{SCS}$ signal to enable the slave devices before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the $\overline{SCS}$ signal depending upon the configurations of the CKPOLB bit and CKEG bit. The accompanying timing diagram shows the relationship between the slave data and $\overline{SCS}$ signal for various configurations of the CKPOLB and CKEG bits.

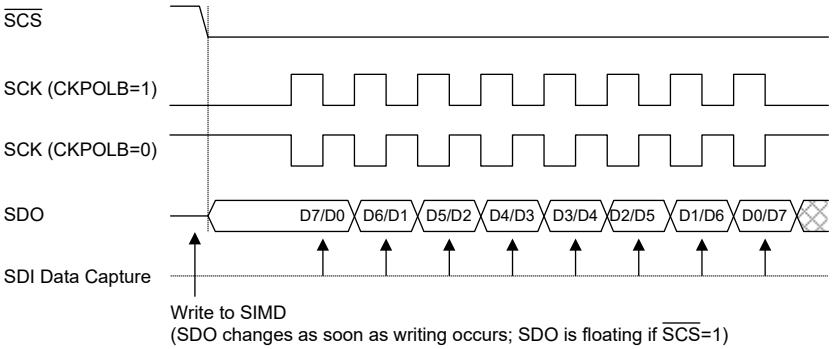
The SPI will continue to function in certain IDLE Modes if the clock source used by the SPI interface is still active.




**SPI Master Mode Timing**

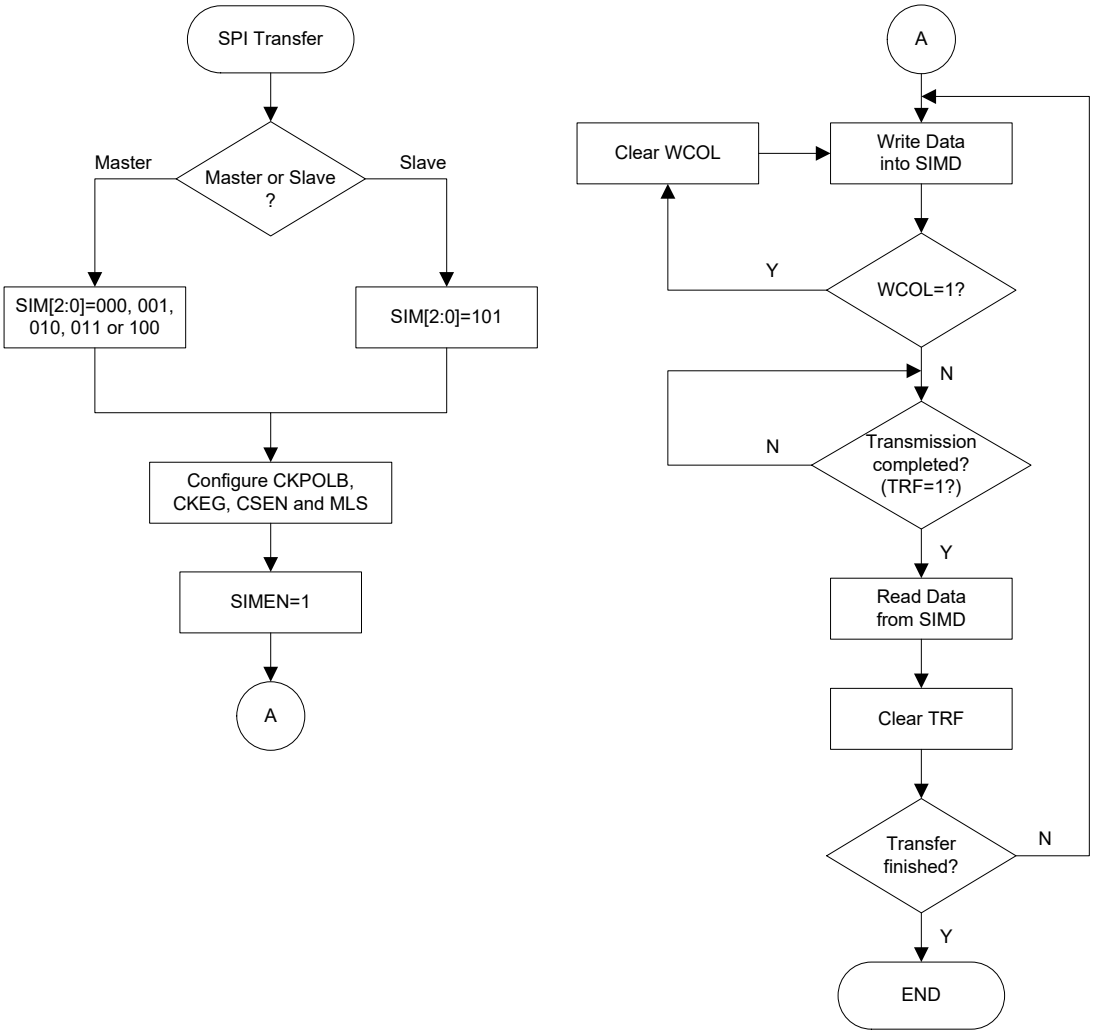



**SPI Slave Mode Timing – CKEG=0**




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

**SPI Slave Mode Timing – CKEG=1**




**SPI Transfer Control Flowchart**

### SPI Bus Enable/Disable

To enable the SPI bus, set CSEN=1 and $\overline{SCS}$=0, then wait for data to be written into the SIMD (TXRX buffer) register. For the Master Mode, after data has been written to the SIMD (TXRX buffer) register, then transmission or reception will start automatically. When all the data has been transferred, the TRF bit should be set. For the Slave Mode, when clock pulses are received on SCK, data in the TXRX buffer will be shifted out or data on SDI will be shifted in.

When the SPI bus is disabled, SCK, SDI, SDO and $\overline{SCS}$ can become I/O pins or other pin-shared functions using the corresponding pin-shared control bits.

### SPI Operation Steps

All communication is carried out using the 4-line interface for either Master or Slave Mode.

The CSEN bit in the SIMC2 register controls the overall function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the $\overline{SCS}$ line to be active, which can then be used to control the SPI interface. If the CSEN bit is low, the SPI interface will be disabled and the $\overline{SCS}$ line will be in a floating condition and can therefore not be used for control of the SPI interface. If the CSEN bit and the SIMEN bit in the SIMC0 are set high, this will place the SDI line in a floating condition and the SDO line high. If in Master Mode the SCK line will be either high or low depending upon the clock polarity selection bit CKPOLB in the SIMC2 register. If in Slave Mode the SCK line will be in a floating condition. If the SIMEN bit is low, then the bus will be disabled and $\overline{SCS}$, SDI, SDO and SCK will all become I/O pins or the other functions using the corresponding pin-shared control bits. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SIMD register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

- **Master Mode**
  - Step 1
    Select the SPI Master mode and clock source using the SIM2~SIM0 bits in the SIMC0 control register.
  - Step 2
    Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Slave devices.
  - Step 3
    Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.
  - Step 4
    For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then use the SCK and $\overline{SCS}$ lines to output the data. After this, go to step 5.
    For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.
  - Step 5
    Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
  - Step 6
    Check the TRF bit or wait for a SPI serial bus interrupt.
  - Step 7
    Read data from the SIMD register.
  - Step 8
    Clear TRF.
  - Step 9
    Go to step 4.

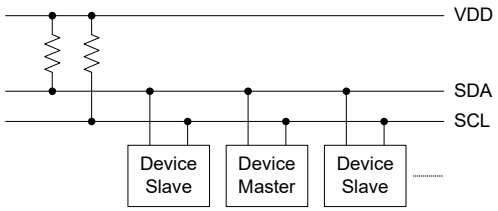
- **Slave Mode**
  - Step 1
    Select the SPI Slave mode using the SIM2~SIM0 bits in the SIMC0 control register
  - Step 2
    Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master devices.
  - Step 3
    Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.
  - Step 4
    For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCK and $\overline{SCS}$ signal. After this, go to step 5.
    For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.
  - Step 5
    Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
  - Step 6
    Check the TRF bit or wait for a SPI serial bus interrupt.
  - Step 7
    Read data from the SIMD register.
  - Step 8
    Clear TRF.
  - Step 9
    Go to step 4.

### Error Detection

The WCOL bit in the SIMC2 register is provided to indicate errors during data transfer. The bit is set by the SPI serial Interface but must be cleared by the application program. This bit indicates that a data collision has occurred which happens if a write to the SIMD register takes place during a data transfer operation and will prevent the write operation from continuing.

## 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.

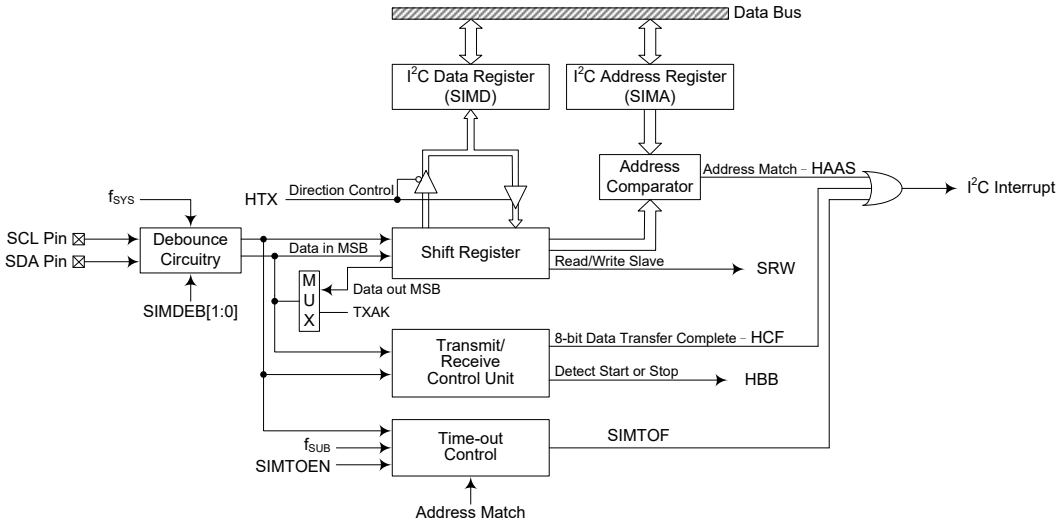



**I²C Master Slave Bus Connection**

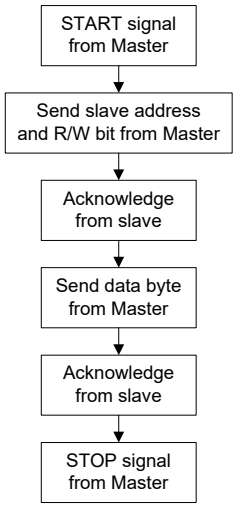
### I²C Interface Operation

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 the 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. The pull-high control function pin-shared with SCL/SDA pin is still applicable even if I²C device is activated and the related internal pull-high register could be controlled by its corresponding pull-high control register.




**I²C Block Diagram**



The SIMDEB1 and SIMDEB0 bits determine the debounce time of the I²C interface. This uses the internal clock to in effect add a debounce time to the external clock to reduce the possibility of glitches on the clock line causing erroneous operation. The debounce time, if selected, can be chosen to be either 2 or 4 system clocks. To achieve the required I²C data transfer speed, there exists a relationship between the system clock, $f_{SYS}$, and the I²C debounce time. For either the I²C Standard or Fast mode operation, users must take care of the selected system clock frequency and the configured debounce time to match the criterion shown in the following table.

| I²C Debounce Time Selection | I²C Standard Mode (100kHz) | I²C Fast Mode (400kHz) |
|---|---|---|
| No Debounce | $f_{SYS}$ > 2 MHz | $f_{SYS}$ > 5 MHz |
| 2 system clock debounce | $f_{SYS}$ > 4 MHz | $f_{SYS}$ > 10 MHz |
| 4 system clock debounce | $f_{SYS}$ > 8 MHz | $f_{SYS}$ > 20 MHz |

**I²C Minimum $f_{SYS}$ Frequency**

## I²C Registers

There are three control registers associated with the I²C bus, SIMC0, SIMC1 and SIMTOC, one address register SIMA and one data register, SIMD.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SIMC0 | SIM2 | SIM1 | SIM0 | — | SIMDEB1 | SIMDEB0 | SIMEN | SIMICF |
| SIMC1 | HCF | HAAS | HBB | HTX | TXAK | SRW | IAMWU | RXAK |
| SIMD | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SIMA | A6 | A5 | A4 | A3 | A2 | A1 | A0 | D0 |
| SIMTOC | SIMTOEN | SIMTOF | SIMTOS5 | SIMTOS4 | SIMTOS3 | SIMTOS2 | SIMTOS1 | SIMTOS0 |

**I²C Registers List**

## I²C Data Register

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

- **SIMD 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 | x | x | x | x | x | x | x | x |

"x": unknown

Bit 7~0 **D7~D0**: SIM data register bit 7~bit 0

### I²C Address Register

The SIMA register is also used by the SPI interface but has the name SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bits 7~1 of the SIMA 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 SIMA register, the slave device will be selected. Note that the SIMA register is the same register address as SIMC2 which is used by the SPI interface.

- **SIMA Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | A6 | A5 | A4 | A3 | A2 | A1 | A0 | 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 7~1 **A6~A0**: I²C slave address
A6~A0 is the I²C slave address bit 6~bit 0.

Bit 0 **D0**: Reserved bit, can be read or written

### I²C Control Registers

There are three control registers for the I²C interface, SIMC0, SIMC1 and SIMTOC. The SIMC0 register is used to control the enable/disable function and to set the data transmission clock frequency. The SIMC1 register contains the relevant flags which are used to indicate the I²C communication status. Another register, SIMTOC, is used to control the I²C time-out function and is described in the corresponding section.

- **SIMC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SIM2 | SIM1 | SIM0 | — | SIMDEB1 | SIMDEB0 | SIMEN | SIMICF |
| R/W | R/W | R/W | R/W | — | R/W | R/W | R/W | R/W |
| POR | 1 | 1 | 1 | — | 0 | 0 | 0 | 0 |

Bit 7~5 **SIM2~SIM0**: SIM Operating Mode Control
000: SPI master mode; SPI clock is $f_{SYS}/4$
001: SPI master mode; SPI clock is $f_{SYS}/16$
010: SPI master mode; SPI clock is $f_{SYS}/64$
011: SPI master mode; SPI clock is $f_{SUB}$
100: SPI master mode; SPI clock is CTM CCRP match frequency/2
101: SPI slave mode
110: I²C slave mode
111: Unused mode
These bits setup the overall operating mode of the SIM function. As well as selecting if the I²C or SPI function, they are used to control the SPI Master/Slave selection and the SPI Master clock frequency. The SPI clock is a function of the system clock but can also be chosen to be sourced from CTM and $f_{SUB}$. If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as "0"

Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
00: No debounce
01: 2 system clock debounce
1x: 4 system clock debounce
These bits are used to select the I²C debounce time when the SIM is configured as the I²C interface function by setting the SIM2~SIM0 bits to "110".

Bit 1 **SIMEN**: SIM Enable Control
0: Disable
1: Enable
The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and $\overline{SCS}$, or SDA and SCL lines will lose their SPI or I²C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I²C interface via the SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I²C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I²C flags such as HCF, HAAS, HBB, SRW and RXAK will be set to their default states.

Bit 0 **SIMICF**: SIM SPI Incomplete Flag
This bit is only available when the SIM is configured to operate in an SPI slave mode. Refer to the SPI register section.

- **SIMC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | HCF | HAAS | HBB | HTX | TXAK | SRW | IAMWU | RXAK |
| R/W | R | R | R | R/W | R/W | R | R/W | R |
| POR | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

Bit 7 **HCF**: I²C Bus data transfer completion flag
0: Data is being transferred
1: Completion of an 8-bit data transfer
The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.

Bit 6 **HAAS**: I²C Bus address match flag
0: Not address match
1: Address match
The HAAS 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 **HBB**: I²C Bus busy flag
0: I²C Bus is not busy
1: I²C Bus is busy
The HBB flag is the I²C busy flag. This flag will be "1" when the I²C 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 **HTX**: I²C slave device is transmitter or receiver selection
0: Slave device is the receiver
1: Slave device is the transmitter

Bit 3 **TXAK**: I²C Bus transmit acknowledge flag
0: Slave send acknowledge flag
1: Slave do not send acknowledge flag
The TXAK 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 TXAK bit to "0" before further data is received.

Bit 2 **SRW**: I²C Slave Read/Write flag
0: Slave device should be in receive mode
1: Slave device should be in transmit mode
The SRW flag is the I²C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I²C bus. When the transmitted address and slave address is match, that is when the HAAS flag is set high, the slave device will check the SRW flag to determine whether it should be in transmit mode or receive mode. If the SRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the SRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.

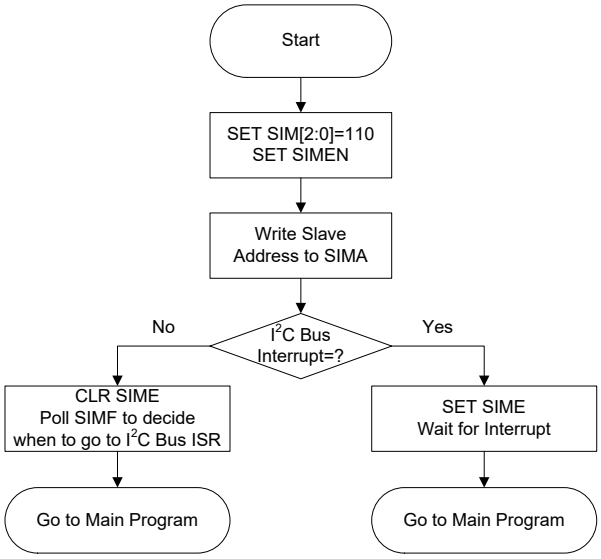
Bit 1 **IAMWU**: I²C Address Match Wake-up control
0: Disable
1: Enable
This bit should be set to 1 to enable the I²C address match wake up from the SLEEP or IDLE Mode. If the IAMWU bit has been set before entering either the SLEEP or IDLE mode to enable the I²C address match wake up, then this bit must be cleared by the application program after wake-up to ensure correction device operation.

Bit 0 **RXAK**: I²C Bus Receive acknowledge flag
0: Slave receive acknowledge flag
1: Slave does not receive acknowledge flag
The RXAK flag is the receiver acknowledge flag. When the RXAK 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 RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK 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.

### 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 HAAS bit in the SIMC1 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 HAAS and SIMTOF bits to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer completion or from the I²C bus time-out occurrence. 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 the SIM2~SIM0 and SIMEN bits in the SIMC0 register to "110" and "1" respectively to enable the I²C bus.
- Step 2
  Write the slave address of the device to the I²C bus address register SIMA.
- Step 3
  Set the SIME interrupt enable bit of the interrupt control register to enable the SIM 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 HBB bit will be set. A START condition occurs when a high to low transition on the SDA line takes place when the SCL line remains high.

### I²C 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 SRW bit of the SIMC1 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 HAAS when the addresses match.

As an I²C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS and SIMTOF bits 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 or from the I²C bus time-out occurrence. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

### I²C Bus Read/Write Signal

The SRW bit in the SIMC1 register defines whether the master 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 SRW 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 SRW 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.

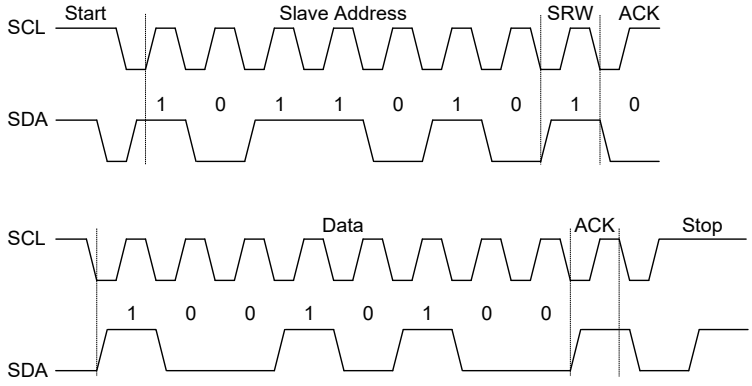
### I²C Bus Slave Address Acknowledge Signal

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 HAAS flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be setup to be a transmitter so the HTX bit in the SIMC1 register should be set to "1". If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMC1 register should be set to "0".

### I²C Bus Data and Acknowledge Signal

The transmitted data is 8-bit 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 SIMD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the SIMD register. If setup as a receiver, the slave device must read the transmitted data from the SIMD register.

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as TXAK, on the 9th clock. The slave device, which is setup as a transmitter will check the RXAK bit in the SIMC1 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.

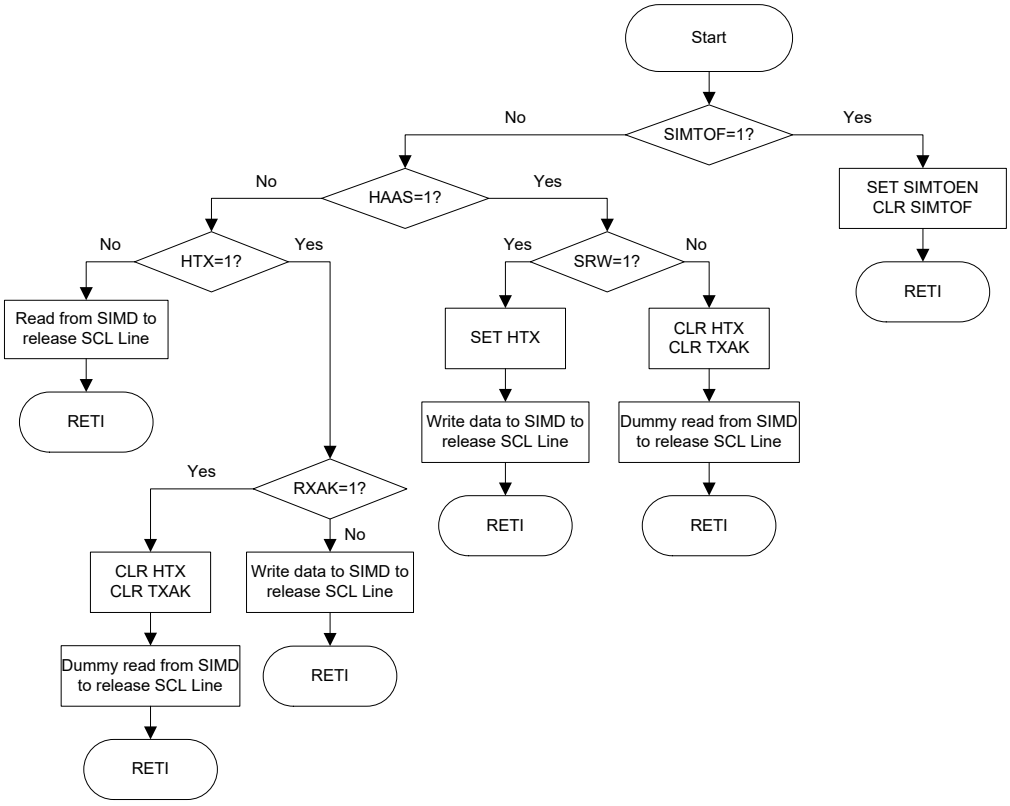



S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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

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

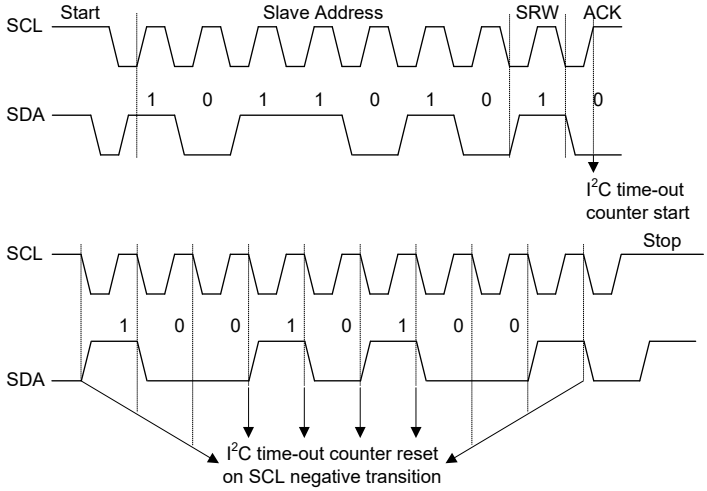
**I²C Communication Timing Diagram**




**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 for a while, then the I²C circuitry and registers will be reset after a certain time-out period. 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 SIMTOC register, then a time-out condition will occur. The time-out function will stop when an I²C "STOP" condition occurs.




**I²C Time-out**

When an I²C time-out counter overflow occurs, the counter will stop and the SIMTOEN bit will be cleared to zero and the SIMTOF 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:

| Registers | After I²C Time-out |
|---|---|
| SIMD, SIMA, SIMC0 | No change |
| SIMC1 | Reset to POR condition |

**I²C Registers after Time-out**

The SIMTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using SIMTOS bit field in the SIMTOC register. The time-out time is given by the formula: ((1~64) × 32) / $f_{SUB}$. This gives a time-out period which ranges from about 1ms to 64ms.

- **SIMTOC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | SIMTOEN | SIMTOF | SIMTOS5 | SIMTOS4 | SIMTOS3 | SIMTOS2 | SIMTOS1 | SIMTOS0 |
| 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 **SIMTOEN**: I²C Time-out control
0: Disable
1: Enable

Bit 6 **SIMTOF**: I²C Time-out flag
0: No time-out occurred
1: Time-out occurred

Bit 5~0 **SIMTOS5~SIMTOS0**: I²C Time-out period selection
I²C time-out clock source is $f_{SUB}/32$.
I²C time-out time is equal to (SIMTOS[5:0]+1) × ($32/f_{SUB}$).

# 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 interrupt functions. The external interrupts are generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by various internal functions such as the TMs, LVD and the A/D converter, etc.

## 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 falls into three categories. The first is the INTC0~INTC2 registers which setup the primary interrupts, the second is the MFI0~MFI1 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=0 or 1 |
| SIM | SIME | SIMF | — |
| Time Base | TBnE | TBnF | n=0 or 1 |
| A/D Converter | ADE | ADF | — |
| Multi-function | MFnE | MFnF | n=0~1 |
| EEPROM | DEE | DEF | — |
| LVD | LVE | LVF | — |
| CTM | CTMPE | CTMPF | — |
| | CTMAE | CTMAF | — |

**Interrupt Register Bit Naming Conventions**

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTEG | — | — | — | — | INT1S1 | INT1S0 | INT0S1 | INT0S0 |
| INTC0 | — | ADF | INT1F | INT0F | ADE | INT1E | INT0E | EMI |
| INTC1 | — | MF1F | SIMF | MF0F | — | MF1E | SIME | MF0E |
| INTC2 | — | — | TB1F | TB0F | — | — | TB1E | TB0E |
| MFI0 | — | — | CTMAF | CTMPF | — | — | CTMAE | CTMPE |
| MFI1 | — | — | DEF | LVF | — | — | DEE | LVE |

**Interrupt Registers List**

### INTEG Register

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

Bit 7~4 Unimplemented, read as "0"

Bit 3~2 **INT1S1~INT1S0**: Interrupt edge control for INT1 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

Bit 1~0 **INT0S1~INT0S0**: Interrupt edge control for INT0 pin
00: Disable
01: Rising edge
10: Falling edge
11: Rising and falling edges

### INTC0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | ADF | INT1F | INT0F | ADE | INT1E | INT0E | EMI |
| R/W | — | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | — | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7 Unimplemented, read as "0"

Bit 6 **ADF**: A/D Converter interrupt request flag
0: No request
1: Interrupt request

Bit 5 **INT1F**: INT1 interrupt request flag
0: No request
1: Interrupt request

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

Bit 3 **ADE**: A/D Converter interrupt control
0: Disable
1: Enable

Bit 2 **INT1E**: INT1 interrupt control
0: Disable
1: Enable

Bit 1 **INT0E**: INT0 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 | — | MF1F | SIMF | MF0F | — | MF1E | SIME | MF0E |
| R/W | — | R/W | R/W | R/W | — | R/W | R/W | R/W |
| POR | — | 0 | 0 | 0 | — | 0 | 0 | 0 |

Bit 7 Unimplemented, read as "0"

Bit 6 **MF1F**: Multi-function interrupt 1 request flag
0: No request
1: Interrupt request

Bit 5 **SIMF**: SIM interrupt 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 Unimplemented, read as "0"

Bit 2 **MF1E**: Multi-function interrupt 1 control
0: Disable
1: Enable

Bit 1 **SIME**: SIM interrupt 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 | — | — | TB1F | TB0F | — | — | TB1E | TB0E |
| R/W | — | — | R/W | R/W | — | — | R/W | R/W |
| POR | — | — | 0 | 0 | — | — | 0 | 0 |

Bit 7~6 Unimplemented, read as "0"

Bit 5 **TB1F**: Time Base 1 request flag
0: No request
1: Interrupt request

Bit 4 **TB0F**: Time Base 0 request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **TB1E**: Time Base 1 interrupt control
0: Disable
1: Enable

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

### MFI0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | CTMAF | CTMPF | — | — | CTMAE | CTMPE |
| R/W | — | — | R/W | R/W | — | — | R/W | R/W |
| POR | — | — | 0 | 0 | — | — | 0 | 0 |

Bit 7~6 Unimplemented, read as "0"

Bit 5 **CTMAF**: CTM Comparator A match interrupt request flag
0: No request
1: Interrupt request

Bit 4 **CTMPF**: CTM Comparator P match interrupt request flag
0: No request
1: Interrupt request

Bit 3~2 Unimplemented, read as "0"

Bit 1 **CTMAE**: CTM Comparator A match interrupt control
0: Disable
1: Enable

Bit 0 **CTMPE**: CTM Comparator P match interrupt control
0: Disable
1: Enable

### MFI1 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

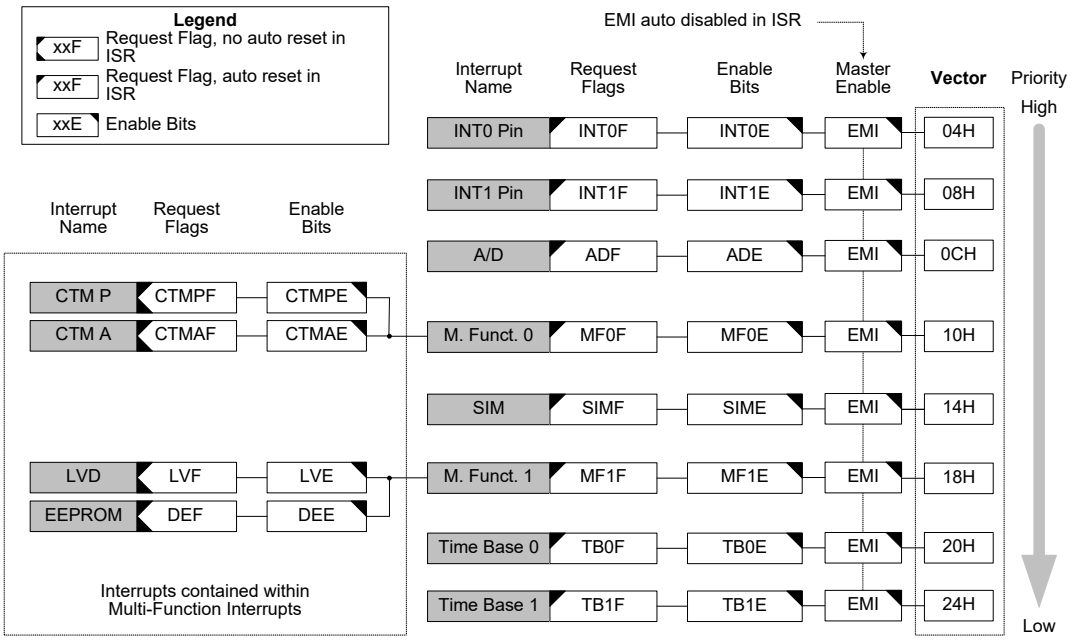
## Interrupt Operation

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 Interrupts

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

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

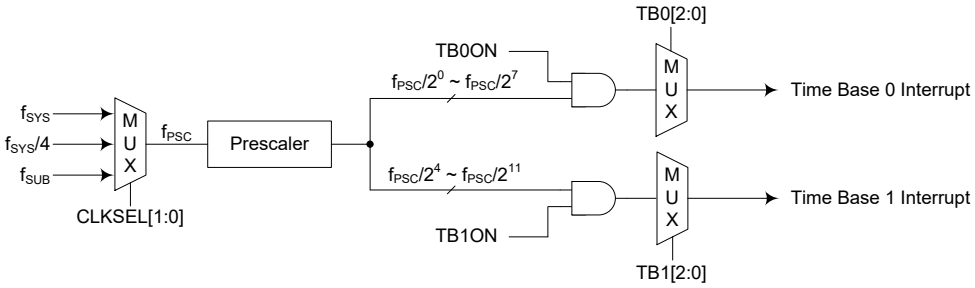
### SIM Interrupt

The Serial Interface Module Interrupt, also known as the SIM interrupt, will take place when the SIM Interrupt request flag, SIMF, is set, which occurs when a byte of data has been received or transmitted by the SIM interface, or an $I^2C$ slave address match occurs, or an $I^2C$ bus time-out occurs. 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, SIME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the respective Interrupt vector, will take place. When the interrupt is serviced, the Serial Interface Interrupt flag, SIMF, 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. Its clock source, $f_{PSC}$, originates from the internal clock source $f_{SYS}$, $f_{SYS}/4$ or $f_{SUB}$ and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL1~CLKSEL0 bits in the PSCR register.




**Time Base Interrupts**

### 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**: Prescaler clock source selection
00: $f_{SYS}$
01: $f_{SYS}/4$
1x: $f_{SUB}$

### TB0C Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TB0ON | — | — | — | — | TB02 | TB01 | TB00 |
| R/W | R/W | — | — | — | — | R/W | R/W | R/W |
| POR | 0 | — | — | — | — | 0 | 0 | 0 |

Bit 7 **TB0ON**: Time Base 0 Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB02~TB00**: Select Time Base 0 Time-out Period
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

### TB1C Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | TB1ON | — | — | — | — | TB12 | TB11 | TB10 |
| R/W | R/W | — | — | — | — | R/W | R/W | R/W |
| POR | 0 | — | — | — | — | 0 | 0 | 0 |

Bit 7 **TB1ON**: Time Base 1 Control
0: Disable
1: Enable

Bit 6~3 Unimplemented, read as "0"

Bit 2~0 **TB12~TB10**: Select Time Base 1 Time-out Period
000: $2^8/f_{PSC}$
001: $2^9/f_{PSC}$
010: $2^{10}/f_{PSC}$
011: $2^{11}/f_{PSC}$
100: $2^{12}/f_{PSC}$
101: $2^{13}/f_{PSC}$
110: $2^{14}/f_{PSC}$
111: $2^{15}/f_{PSC}$

### A/D Converter Interrupt

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 Interrupt vector, will take place. When the A/D Converter Interrupt is serviced, the A/D Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

### Multi-function Interrupts

Within the device there are up to two 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, EEPROM Interrupt and LVD Interrupt.

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

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

### 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, 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 or a low LVDIN input voltage. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, Low Voltage Interrupt enable bit, LVE, and associated Multi-function interrupt enable bit, 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 Multi-function Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the Multi-function interrupt request flag will be also automatically cleared. As the LVF flag will not be automatically cleared, it has to be cleared by the application program.

### TM Interrupts

The Compact Type TM has two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. All of the TM interrupts are contained within the Multi-function Interrupts. There are two interrupt request flags and two enable control bits. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

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

### Interrupt Wake-up Function

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

## Programming Considerations

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

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

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

Every interrupt has the capability of waking up the microcontroller when it is in the 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 enabled the device to monitor the power supply voltage, $V_{DD}$, or the LVDIN input voltage, and provide a warning signal should it fall below a certain level. This function may be especially useful in battery applications where the supply voltage will gradually reduce as the battery ages, as it allows an early warning battery low signal to be generated. The Low Voltage Detector also has the capability of generating an interrupt signal.

### LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, $V_{LVD2}$~$V_{LVD0}$, 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 or the LVDIN input 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 | VBGEN | VLVD2 | VLVD1 | VLVD0 |
| R/W | — | — | R | 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 **LVDO**: LVD Output flag
0: No Low Voltage Detected
1: Low Voltage Detected

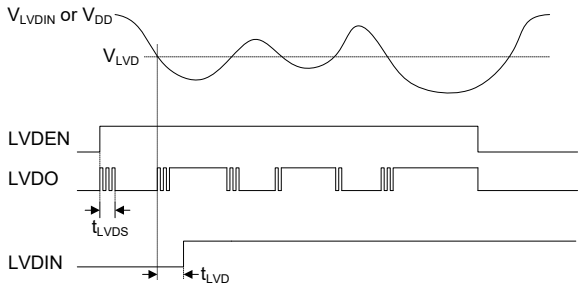
Bit 4 **LVDEN**: Low Voltage Detector Enable control
0: Disable
1: Enable

Bit 3 **VBGEN**: Bandgap Voltage Output Enable control
0: Disable
1: Enable
Note that the Bandgap circuit is enabled when the LVD or LVR function is enabled or when the VBGEN bit is set to 1.

Bit 2~0 **VLVD2~VLVD0**: LVD Voltage selection
000: $V_{LVDIN} \leq 1.04V$
001: 2.2V
010: 2.4V
011: 2.7V
100: 3.0V
101: 3.3V
110: 3.6V
111: 4.0V
When the VLVD bit field is set to 000B, the LVD function operates by comparing the LVD reference voltage with the LVDIN pin input voltage. Otherwise, the LVD function operates by comparing the LVD reference voltage with the power supply voltage when the VLVD bit field is set to any other value except 000B.

## LVD Operation

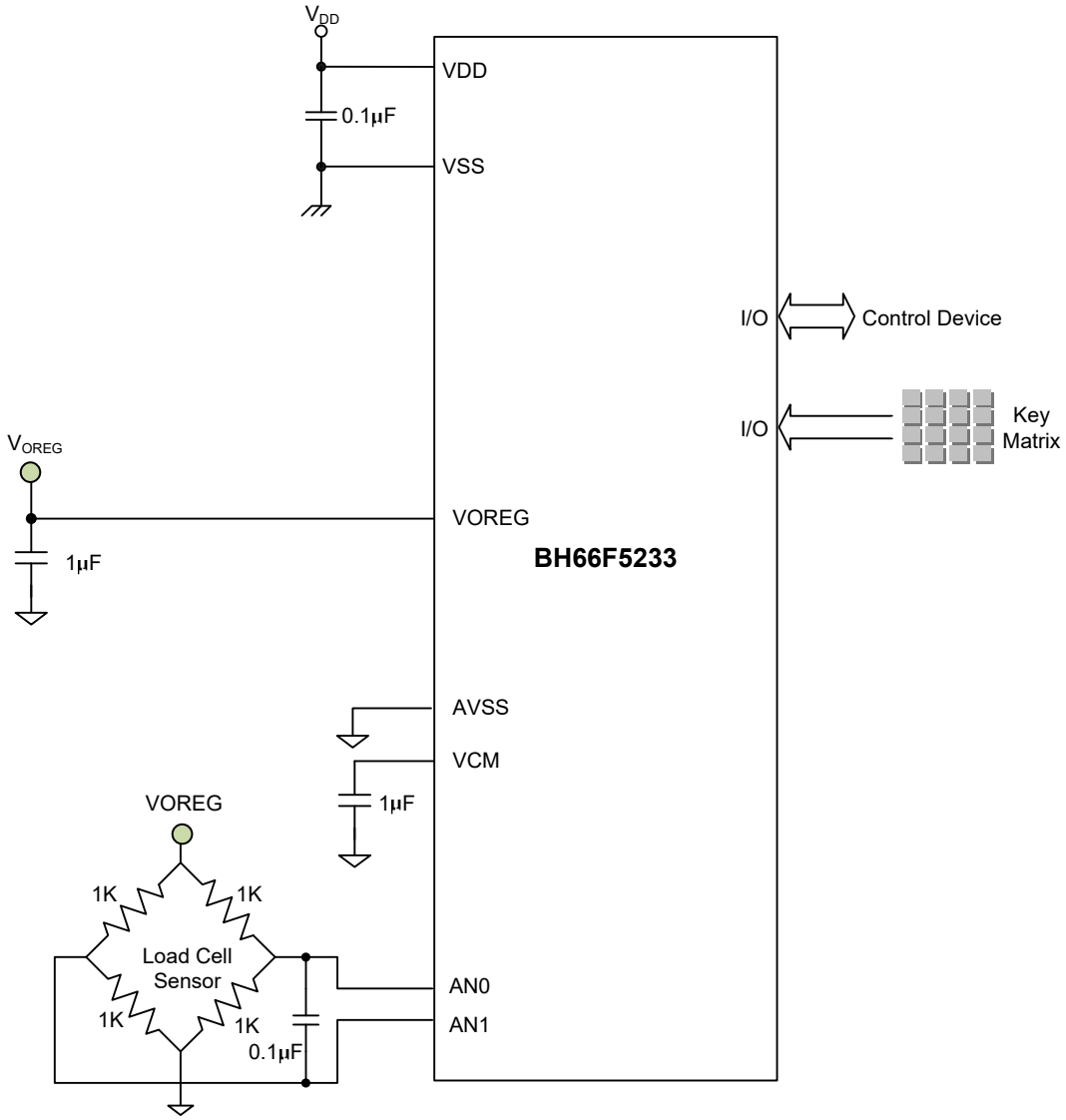
The Low Voltage Detector function operates by comparing the power supply voltage, $V_{DD}$, or the LVDIN input voltage, with a pre-specified voltage level stored in the LVDC register. This has a range of between 1.04V and 4.0V. When the power supply voltage, $V_{DD}$, falls below this pre-determined 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 in the SLEEP mode, the low voltage detector will be disabled even 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 $V_{DD}$ or $V_{LVDIN}$ 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 interrupt 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. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if $V_{DD}$ or $V_{LVDIN}$ falls below the preset LVD voltage. This will cause the device to wake-up from the SLEEP or IDLE Mode, however if the Low Voltage Detector wake up function is not required then the LVF flag should be first set high before the device enters the SLEEP or IDLE Mode.

## Application Circuits



# Instruction Set

## Introduction

Central to the successful operation of any microcontroller is its instruction set, which is a set of program instruction codes that directs the microcontroller to perform certain operations. In the case of Holtek 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 set as a table where data can be directly stored. A set of easy to use instructions provides the means by which this fixed data can be referenced and retrieved from the Program Memory.

### Other Operations

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

## Instruction Set Summary

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

### Table Conventions

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

| Mnemonic | Description | Cycles | Flag Affected |
|---|---|---|---|
| **Arithmetic** | | | |
| ADD A,[m] | Add Data Memory to ACC | 1 | Z, C, AC, OV |
| ADDM A,[m] | Add ACC to Data Memory | 1[Note] | Z, C, AC, OV |
| ADD A,x | Add immediate data to ACC | 1 | Z, C, AC, OV |
| ADC A,[m] | Add Data Memory to ACC with Carry | 1 | Z, C, AC, OV |
| ADCM A,[m] | Add ACC to Data memory with Carry | 1[Note] | Z, C, AC, OV |
| SUB A,x | Subtract immediate data from the ACC | 1 | Z, C, AC, OV |
| SUB A,[m] | Subtract Data Memory from ACC | 1 | Z, C, AC, OV |
| SUBM A,[m] | Subtract Data Memory from ACC with result in Data Memory | 1[Note] | Z, C, AC, OV |
| SBC A,[m] | Subtract Data Memory from ACC with Carry | 1 | Z, C, AC, OV |
| SBCM A,[m] | Subtract Data Memory from ACC with Carry, result in Data Memory | 1[Note] | Z, C, AC, OV |
| DAA [m] | Decimal adjust ACC for Addition with result in Data Memory | 1[Note] | C |
| **Logic Operation** | | | |
| AND A,[m] | Logical AND Data Memory to ACC | 1 | Z |
| OR A,[m] | Logical OR Data Memory to ACC | 1 | Z |
| XOR A,[m] | Logical XOR Data Memory to ACC | 1 | Z |
| ANDM A,[m] | Logical AND ACC to Data Memory | 1[Note] | Z |
| ORM A,[m] | Logical OR ACC to Data Memory | 1[Note] | Z |
| XORM A,[m] | Logical XOR ACC to Data Memory | 1[Note] | Z |
| AND A,x | Logical AND immediate Data to ACC | 1 | Z |
| OR A,x | Logical OR immediate Data to ACC | 1 | Z |
| XOR A,x | Logical XOR immediate Data to ACC | 1 | Z |
| CPL [m] | Complement Data Memory | 1[Note] | Z |
| CPLA [m] | Complement Data Memory with result in ACC | 1 | Z |
| **Increment & Decrement** | | | |
| INCA [m] | Increment Data Memory with result in ACC | 1 | Z |
| INC [m] | Increment Data Memory | 1[Note] | Z |
| DECA [m] | Decrement Data Memory with result in ACC | 1 | Z |
| DEC [m] | Decrement Data Memory | 1[Note] | Z |
| **Rotate** | | | |
| RRA [m] | Rotate Data Memory right with result in ACC | 1 | None |
| RR [m] | Rotate Data Memory right | 1[Note] | None |
| RRCA [m] | Rotate Data Memory right through Carry with result in ACC | 1 | C |
| RRC [m] | Rotate Data Memory right through Carry | 1[Note] | C |
| RLA [m] | Rotate Data Memory left with result in ACC | 1 | None |
| RL [m] | Rotate Data Memory left | 1[Note] | None |
| RLCA [m] | Rotate Data Memory left through Carry with result in ACC | 1 | C |
| RLC [m] | Rotate Data Memory left through Carry | 1[Note] | C |

| 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 Operation** | | | |
| 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 |
| SWAP [m] | Swap nibbles of Data Memory | 1Note | None |
| SWAPA [m] | Swap nibbles of Data Memory with result in ACC | 1 | None |
| HALT | Enter power down mode | 1 | TO, PDF |

Note: 1. For skip instructions, if the result of the comparison involves a skip then two cycles are required, if no skip takes place only one cycle is required.

2. Any instruction which changes the contents of the PCL will also require 2 cycles for execution.

## 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

**CPL [m]** Complement Data Memory

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

Operation [m] ← $\overline{[m]}$

Affected flag(s) Z

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

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

Operation ACC ← $\overline{[m]}$

Affected flag(s) Z

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

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

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

Affected flag(s) C

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

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

| | |
|---|---|
| **HALT** | Enter power down mode |
| Description | This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared. |
| Operation | TO ← 0<br>PDF ← 1 |
| Affected flag(s) | TO, PDF |

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

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

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

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

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

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

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

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

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

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

| | |
|---|---|
| **RET** | Return from subroutine |
| Description | The Program Counter is restored from the stack. Program execution continues at the restored address. |
| Operation | Program Counter ← Stack |
| Affected flag(s) | None |

| | |
|---|---|
| **RET A,x** | Return from subroutine and load immediate data to ACC |
| Description | The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address. |
| Operation | Program Counter ← Stack<br>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<br>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)<br>[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 is 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 The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the contents of the specified Data Memory is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.

Operation Skip if [m]=0

Affected flag(s) None

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

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

Operation ACC ← [m]
Skip if [m]=0

Affected flag(s) None

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

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

Operation Skip if [m].i=0

Affected flag(s) None

**TABRD [m]** Read table (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 |

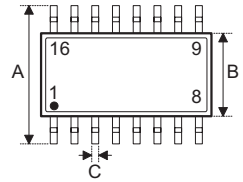
## Package Information

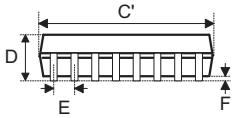
Note that the package information provided here is for consultation purposes only. As this information may be updated at regular intervals users are reminded to consult the Holtek website for the latest version of the Package/Carton Information.

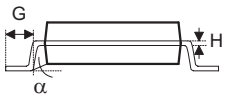
Additional supplementary information with regard to packaging is listed below. Click on the relevant section to be transferred to the relevant website page.

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

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



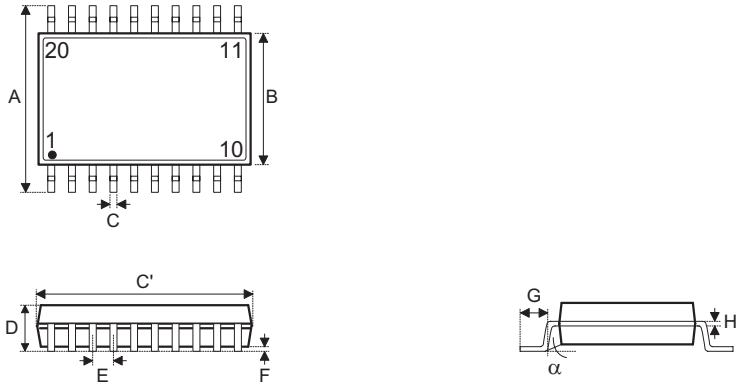




| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | — | 0.236 BSC | — |
| B | — | 0.154 BSC | — |
| C | 0.012 | — | 0.020 |
| C’ | — | 0.390 BSC | — |
| D | — | — | 0.069 |
| E | — | 0.050 BSC | — |
| F | 0.004 | — | 0.010 |
| G | 0.016 | — | 0.050 |
| H | 0.004 | — | 0.010 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | — | 6 BSC | — |
| B | — | 3.9 BSC | — |
| C | 0.31 | — | 0.51 |
| C’ | — | 9.9 BSC | — |
| D | — | — | 1.75 |
| E | — | 1.27 BSC | — |
| F | 0.10 | — | 0.25 |
| G | 0.40 | — | 1.27 |
| H | 0.10 | — | 0.25 |
| α | 0° | — | 8° |

## 20-pin NSOP (150mil) Outline Dimensions




| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.228 | 0.236 | 0.244 |
| B | 0.146 | 0.154 | 0.161 |
| C | 0.009 | — | 0.012 |
| C’ | 0.382 | 0.390 | 0.398 |
| D | — | — | 0.069 |
| E | — | 0.032 BSC | — |
| F | 0.002 | — | 0.009 |
| G | 0.020 | — | 0.031 |
| H | 0.008 | — | 0.010 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 5.80 | 6.00 | 6.20 |
| B | 3.70 | 3.90 | 4.10 |
| C | 0.23 | — | 0.30 |
| C’ | 9.70 | 9.90 | 10.10 |
| D | — | — | 1.75 |
| E | — | 0.80 BSC | — |
| F | 0.05 | — | 0.23 |
| G | 0.50 | — | 0.80 |
| H | 0.21 | — | 0.25 |
| α | 0° | — | 8° |