# HOLTEK

**Health Measurement OTP MCU**

# BH66R2040

Revision: V1.00    Date: October 29, 2025

## Features

### CPU Features

- Operating Voltage
  - $f_{SYS}$=8MHz: 1.8V~5.5V
- Up to 0.5μs instruction cycle with 8MHz system clock at $V_{DD}$=5V
- Power down and wake-up functions to reduce power consumption
- Oscillator Types
  - Internal High Speed 8MHz RC – HIRC
  - Internal Low Speed 32kHz RC – LIRC
- Fully integrated internal oscillators require no external components
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 12-level subroutine nesting
- Bit manipulation instruction

### Peripheral Features

- OTP Program Memory: 4K×16
- Data Memory: 512×8
- OTP ROM parameter program – ORPP
- Watchdog Timer function
- 11 bidirectional I/O lines
- 2 external interrupt lines shared with I/O pins
- One Multiple Timer Module for time measure, compare match output or PWM output function
- Dual Time Base functions for generation of fixed time interrupt signals
- Health Measure AFE
  - Multi-channel 24-bit resolution Delta Sigma A/D Converters
  - Internal LDO
  - Internal Reference Voltage Generator
  - Three 12-bit D/A Converters
  - Three internal Operational Amplifiers
- Serial Interface Module – SIM for SPI or I²C interface
- Fully-duplex / Half-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- Low voltage reset function
- Low voltage detect function
- Package type: 32-pin QFN

## General Description

The device is an OTP Program Memory 8-bit high performance RISC architecture microcontroller device with Health Measure AFE module that is applicable to multi-function health measurement applications.

For memory features, the device is supplied with One-Time Programmable, OTP memory. Other memory includes an area of RAM Data Memory.

Analog features include a multi-channel 24-bit Delta Sigma A/D converter, three 12-bit D/A converters and three internal operational amplifiers . Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated SPI, I²C and UART interface functions, three popular interfaces which provide designers with a means of easy communcation with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detect coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

A full choice of internal high and low oscillator functions is provided including two fully integrated system oscillators which require no external components for their 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.

With regard to health measurement applications, the device has integrated many of the functions required by these products. These include functions such as Internal Reference Voltage generator, Internal LDO, 24-bit Delta Sigma A/D converts, 12-bit D/A converts, operational amplifiers and etc. The inclusion of flexible I/O programming features, Time Base functions along with many other features enhance the versatility of the device to suit for multi-function health measurement applications such as glucose meter, blood pressure, thermopile sensor and so on.

## Block Diagram



: Pin- Shared Node  ∗: Including SPI & I²C
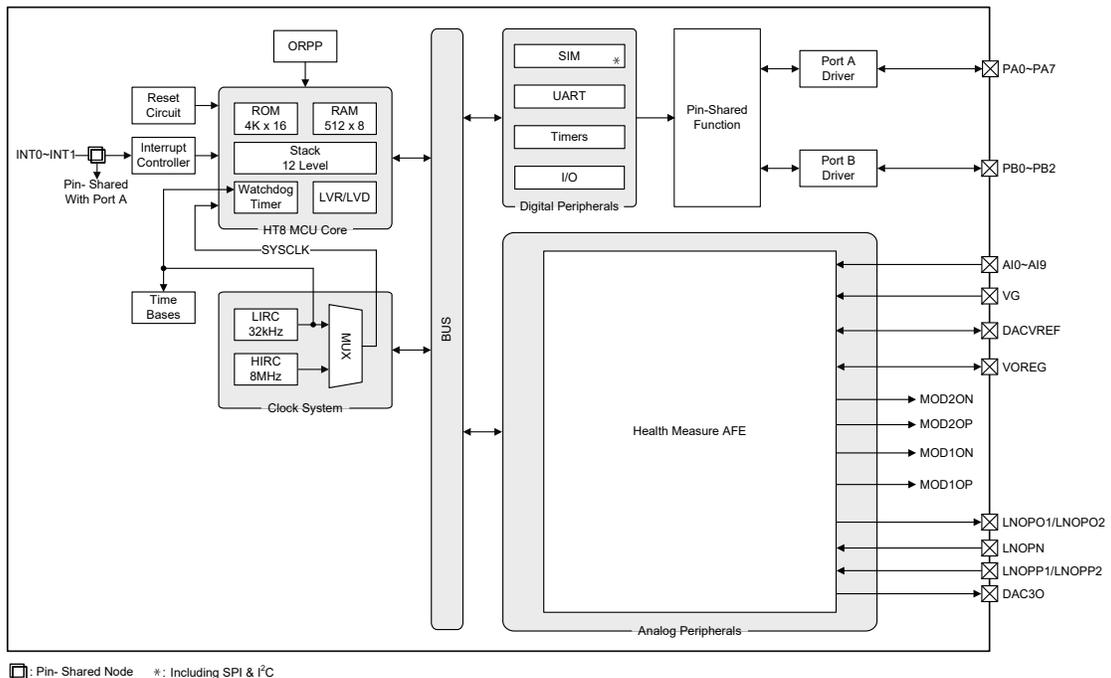
## Pin Assignment
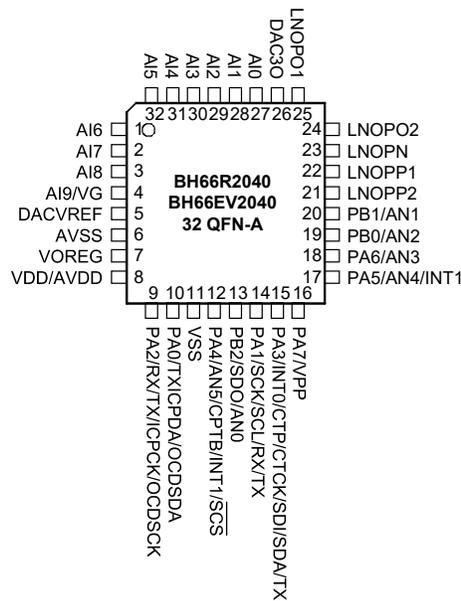


Note: 1. If the pin-shared pin functions have multiple outputs, the desired pin-shared function is determined by the corresponding software control bits.

2. The OCDSDA and OCDSCK pins are used as the OCDS dedicated pins and as such only available for the BH66EV2040 device (Flash type) which is the OCDS EV chip for the BH66R2040 device (OTP type).

3. The VPP pin is the High Voltage input for OTP programming pin and only available for the BH66R2040 device.

## Pin Descriptions

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/TX/ICPDA/OCDSDA | PA0 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | TX | PAS0 | — | CMOS | UART serial data output |
| | ICPDA | — | ST | CMOS | ICP data/address pin |
| | OCDSDA | — | ST | CMOS | OCDS data/address pin |
| PA1/SCK/SCL/RX/TX | PA1 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | SCK | PAS0 | ST | CMOS | SPI serial clock |
| | SCL | PAS0 | ST | NMOS | I²C clock line |
| | RX/TX | IFS PAS0 | ST | CMOS | UART serial data input in full-duplex communication or UART |
| PA2/RX/TX/ICPCK/ OCDSCK | PA2 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | RX/TX | IFS PAS0 | ST | CMOS | UART serial data input in full-duplex communication or UART |
| | ICPCK | — | ST | — | ICP clock pin |
| | OCDSCK | — | ST | — | OCDS clock pin |

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| PA3/INT0/CTP/CTCK/SDI/ SDA/TX | PA3 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | INT0 | PAS0 INTEG INTC0 | ST | — | External interrupt 0 |
| | CTP | PAS0 | — | CMOS | CTM output |
| | CTCK | PAS0 | ST | — | CTM clock input |
| | SDI | PAS0 | ST | — | SPI serial data input |
| | SDA | PAS0 | ST | NMOS | I²C data line |
| | TX | PAS0 | — | CMOS | UART serial data output |
| PA4/AN5/CTPB/INT1/SCS | PA4 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | AN5 | PAS1 | AN | — | A/D converter external input channel |
| | CTPB | PAS1 | — | CMOS | CTM inverted output |
| | INT1 | IFS INTC0 INTEG PAS1 | ST | — | External interrupt 1 |
| | SCS | PAS1 | ST | CMOS | SPI slave select pin |
| PA5/AN4/INT1 | PA5 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | AN4 | PAS1 | AN | — | A/D converter external input channel |
| | INT1 | IFS INTC0 INTEG PAS1 | ST | — | External interrupt 1 |
| PA6/AN3 | PA6 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | AN3 | PAS1 | AN | — | A/D converter external input channel |
| PA7/VPP | PA7 | — | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up. This I/O is pin-shared with VPP and could result in increased current consumption if it is set to output high. |
| | VPP | — | PWR | — | High Voltage input for OTP programming pin, not available for EV chip |
| PB0/AN2 | PB0 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-high |
| | AN2 | PBS0 | AN | — | A/D converter external input channel |
| PB1/AN1 | PB1 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-high |
| | AN1 | PBS0 | AN | — | A/D converter external input channel |
| PB2/SDO/AN0 | PB2 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-high |
| | SDO | PBS0 | — | CMOS | SPI serial data output |
| | AN0 | PBS0 | AN | — | A/D converter external input channel |
| AI0~AI8 | AI0~AI8 | — | AN | AN | AFE analog I/O |
| AI9/VG | AI9 | — | AN | AN | AFE analog I/O |
| | VG | — | AN | — | Virtual ground |
| DACVREF | DACVREF | — | AN | — | D/A Converter reference voltage |

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|---|---|---|---|---|
| VOREG | VOREG | — | — | AN | LDO regulator output |
| | | — | AN | — | Positive power supply for D/A converter, OPA |
| LNOPO1 | LNOPO1 | — | — | AN | Low noise OPA output |
| LNOPO2 | LNOPO2 | — | — | AN | Low noise OPA output |
| LNOPP1 | LNOPP1 | — | AN | — | Low noise OPA positive input |
| LNOPP2 | LNOPP2 | — | AN | — | Low noise OPA positive input |
| LNOPN | LNOPN | — | AN | — | Low noise OPA negative input |
| DAC3O | DAC3O | — | — | AN | DAC3 output |
| VDD/AVDD | VDD | — | PWR | — | Digital positive power supply |
| | AVDD | — | PWR | — | Analog positive power supply |
| VSS/AVSS | VSS | — | PWR | — | Digital negative power supply |
| | AVSS | — | PWR | — | Analog negative power supply |

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

## Absolute Maximum Ratings

Supply Voltage ...................................................................................................... $V_{SS}$-0.3V to 6.0V

Input Voltage ............................................................................................... $V_{SS}$-0.3V to $V_{DD}$+0.3V

Storage Temperature.................................................................................................. -60°C to 150°C

Operating Temperature.............................................................................................. -40°C to 85°C

$I_{OH}$ Total .............................................................................................................................. -80mA

$I_{OL}$ Total ............................................................................................................................... 80mA

Total Power Dissipation .......................................................................................................... 500mW

Note: These are stress ratings only. Stresses exceeding the range specified under "Absolute Maximum Ratings" may cause substantial damage to the device. Functional operation of the 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 | $f_{SYS}$=8MHz | 1.8 | — | 5.5 | V |
| | Operating Voltage – LIRC | $f_{SYS}$=32kHz | 1.8 | — | 5.5 | V |

### Operating Current Characteristics

Ta=-40°C~85°C

| Symbol | Operating Mode | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $I_{DD}$ | SLOW Mode – LIRC | 1.8V | $f_{SYS}$=32kHz | — | 8 | 16 | µA |
| | | 3V | | — | 10 | 20 | |
| | | 5V | | — | 30 | 50 | |
| | FAST Mode – HIRC | 1.8V | $f_{SYS}$=8MHz | — | 0.6 | 1.0 | mA |
| | | 3V | | — | 0.8 | 1.2 | |
| | | 5V | | — | 1.6 | 2.4 | |

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

1. Any digital inputs are setup in a non floating condition and the I/O pin-shared with VPP is not setup in an output high condition.

2. All measurements are taken under conditions of no load and with all peripherals in an off state.

3. There are no DC current paths.

4. All Operating Current values are measured using a continuous NOP instruction program loop.

### Standby Current Characteristics

Ta=25°C, unless otherwise specified

| Symbol | Standby Mode | Test Conditions | | Min. | Typ. | Max. | Max. @85°C | Unit |
|---|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Temp. | | | | | |
| $I_{STB}$ | SLEEP Mode | 1.8V | WDT off | — | 0.45 | 0.80 | 1.90 | µA |
| | | 3V | | — | 0.45 | 0.90 | 2.60 | |
| | | 5V | | — | 0.5 | 2.0 | 5.0 | |
| | | 1.8V | WDT on | — | 1.2 | 2.4 | 2.9 | µA |
| | | 3V | | — | 1.5 | 3.0 | 3.6 | |
| | | 5V | | — | 3 | 5 | 6 | |
| | IDLE0 Mode– LIRC | 1.8V | $f_{SUB}$ on | — | 2.4 | 4.0 | 4.8 | µA |
| | | 3V | | — | 3 | 5 | 6 | |
| | | 5V | | — | 5 | 10 | 12 | |
| | IDLE1 Mode – HIRC | 1.8V | $f_{SUB}$ on, $f_{SYS}$=8MHz | — | 288 | 400 | 480 | mA |
| | | 3V | | — | 360 | 500 | 600 | |
| | | 5V | | — | 600 | 800 | 960 | |

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

1. Any digital inputs are setup in a non floating condition and the I/O pin-shared with VPP is not setup in an output high condition.

2. All measurements are taken under conditions of no load and with all peripherals in an off state.

3. There are no DC current paths.

4. All Standby Current values are taken after a HALT instruction execution thus stopping all instruction execution.

## 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.

### Internal High Speed 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 5V.

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|--------|-----------|--------|--------|------|------|------|------|
| | | $V_{DD}$ | Conditions | | | | |
| $f_{HIRC}$ | 8MHz Writer Trimmed HIRC Frequency | 3V/5V | 25°C | -1% | 8 | +1% | MHz |
| | | 3V/5V | -40°C~85°C | -2% | 8 | +2% | |
| | | 2.2V~5.5V | 25°C | -2.5% | 8 | +2.5% | |
| | | 2.2V~5.5V | -40°C~85°C | -3% | 8 | +3% | |

Note: 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. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.2V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

### Internal Low Speed Oscillator Characteristics – LIRC

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|--------|-----------|--------|--------|------|------|------|------|
| | | $V_{DD}$ | Temp. | | | | |
| $f_{LIRC}$ | LIRC Frequency | 1.8V~5.5V | 25°C | -20% | 32 | +20% | kHz |
| | | 1.8V~5.5V | -40°C~85°C | -50% | 32 | +60% | |
| $t_{START}$ | LIRC Start Up Time | — | -40°C~85°C | — | — | 500 | µs |

### Operating Frequency Characteristic Curves

### System Start Up Time Characteristics

Ta=-40°C~85°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $t_{SST}$ | System Start-up Time (Wake-up from Conditions where $f_{SYS}$ is off) | — | $f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$ | — | 16 | — | $t_{SYS}$ |
| | | — | $f_{SYS}=f_{SUB}=f_{LIRC}$ | — | 2 | — | $t_{SYS}$ |
| | System Start-up Time (Wake-up from Conditions where $f_{SYS}$ is on) | — | $f_{SYS}=f_H\sim f_H/64$, $f_H=f_{HIRC}$ | — | 2 | — | $t_{SYS}$ |
| | | | $f_{SYS}=f_{SUB}$ or $f_{LIRC}$ | — | 2 | — | $t_{SYS}$ |
| | 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) | — | $RR_{POR}=5V/ms$ | 10 | 16 | 24 | ms |
| | System Reset Delay Time (LVRC/WDTC/RSTC Software Reset) | — | — | | | | |
| | System Reset Delay Time (Reset Source from WDT Overflow Reset) | — | — | 10 | 16 | 24 | |
| $t_{SRESET}$ | Minimum Software Reset Width to Reset | — | — | 45 | 90 | 120 | µs |

Note: 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 $t_{HIRC}$ etc. 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=-40°C~85°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 |
| $V_{IH}$ | Input High Voltage for I/O Ports | 5V | — | 3.5 | — | 5.0 | V |
| | | — | — | $0.8V_{DD}$ | — | $V_{DD}$ | V |
| $I_{OL}$ | Sink Current for I/O Ports | 3V | $V_{OL}=0.1V_{DD}$ | 16 | 32 | — | mA |
| | | 5V | | 32 | 65 | — | mA |
| $I_{OH}$ | Source Current for I/O Ports | 3V | $V_{OH}=0.9V_{DD}$ | -4 | -8 | — | mA |
| | | 5V | | -8 | -16 | — | mA |
| $R_{PH}$ | 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 | $V_{IN}=V_{DD}$ or $V_{IN}=V_{SS}$ | — | — | ±1 | |
| $t_{TCK}$ | CTCK Input Pin Minimum Pulse Width | — | — | 0.3 | — | — | µs |
| $t_{INT}$ | External Interrupt 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 pin 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, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| **OTP Program Memory** | | | | | | | |
| $V_{DD}$ | $V_{DD}$ for Read – ORPP | — | — | 1.8 | 5.0 | 5.5 | V |
| | $V_{DD}$ for Write – ORPP | — | — | 4.5 | 5.0 | 5.5 | V |
| $V_{PP}$ | $V_{PP}$ for Write – ORPP | — | — | 8.25 | 8.50 | 8.75 | V |
| $t_{WR}$ | Write Cycle Time – ORPP | — | — | — | 300 | 450 | µs |
| $E_P$ | Cell Endurance | — | — | 1 | — | — | W |
| $t_{RETD}$ | ROM Data Retention Time | — | Ta=25°C | — | 40 | — | Year |
| **Flash Program Memory – for BH66EV2040 only** | | | | | | | |
| $V_{DD}$ | $V_{DD}$ for Read & Write | — | — | 1.8 | — | 5.5 | V |
| $t_{WR}$ | Write Cycle Time | — | — | — | 2.2 | 2.7 | ms |
| $t_{ACTV}$ | ROM Activation Time – Wake-up from Power Down Mode | — | — | 1 | — | 2 | $t_{LIRC}$ |
| **RAM Data Memory** | | | | | | | |
| $V_{DR}$ | RAM Data Retention Voltage | — | — | 1.0 | — | — | V |

Note: 1. The ROM activation time tACTV should be added when calculating the total system start-up time of a wake-up from the power down mode.

2. "W" means Write times.

## LVD/LVR Electrical Characteristics

Ta=-40°C~85°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{DD}$ | Operating Voltage | — | — | 1.8 | — | 5.5 | V |
| $V_{LVR}$ | Low Voltage Reset Voltage | — | LVR enable, voltage select 1.7V | -5% | 1.7 | +5% | V |
| | | | LVR enable, voltage select 1.9V | -5% | 1.9 | +5% | |
| | | | LVR enable, voltage select 2.55V | -3% | 2.55 | +3% | |
| | | | LVR enable, voltage select 3.15V | -3% | 3.15 | +3% | |
| | | | LVR enable, voltage select 3.8V | -3% | 3.8 | +3% | |
| $V_{LVD}$ | Low Voltage Detector Voltage | — | LVD enable, voltage select 1.8V | -5% | 1.8 | +5% | V |
| | | | LVD enable, voltage select 2.0V | | 2.0 | | |
| | | | 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 | | |
| $I_{LVRLVD}$ | Operating Current | 3V | LVD enable, LVR enable, $V_{LVR}$=1.9V, $V_{LVD}$=2.4V | — | — | 10 | µA |
| | | 5V | | — | 10 | 15 | µA |
| $t_{LVDS}$ | LVDO Stable Time | — | For LVR enable, LVD off → on | — | — | 18 | µs |
| | | | For LVR disable, LVD off → on | — | — | 150 | |
| $t_{LVR}$ | Minimum Low Voltage Width to Reset | — | TLVR[1:0]=00B | 120 | 240 | 480 | µs |
| | | | TLVR[1:0]=01B | 0.5 | 1.0 | 2.0 | ms |
| | | | TLVR[1:0]=10B | 1 | 2 | 4 | |
| | | | TLVR[1:0]=11B | 2 | 4 | 8 | |

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|--------|-----------|-----------------|---|------|------|------|------|
| | | $V_{DD}$ | Conditions | | | | |
| $t_{LVD}$ | Minimum Low Voltage Width to Interrupt | — | TLVD[1:0]=00B/11B | 60 | 140 | 220 | µs |
| | | | TLVD[1:0]=01B | 90 | 200 | 340 | |
| | | | TLVD[1:0]=10B | 150 | 320 | 580 | |
| $I_{LVR}$ | Additional current for LVR enable | 5V | LVD disable | — | — | 14 | µA |
| $I_{LVD}$ | Additional current for LVD enable | 5V | LVR disable | — | — | 14 | µA |

## Analog Front End Circuit Characteristics

### LDO Electrical Characteristics

$V_{DD}=V_{IN}$, Ta=25°C, unless otherwise specified
LDO 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 | — | Ta=0°C~45°C | 2.4 | — | 5.5 | V |
| $I_Q$ | LDO Quiescent Current | — | LDOVS[1:0]=00B, $V_{IN}$=3.6V, no load | — | 600 | 720 | 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 | | 2.2 | | |
| | | | LDOVS[1:0]=11B, $V_{IN}$=3.6V, $I_{LOAD}$=0.1mA | | 2.3 | | |
| $\Delta V_{LOAD}$ | LDO Load Regulation[1] | — | LDOVS[1:0]=00B, $V_{IN}=V_{OUT\_LDO}$+0.25V, 0mA≤$I_{LOAD}$≤10mA | — | 0.105 | 0.210 | %/mA |
| $V_{DROP\_LDO}$ | LDO Dropout Voltage [2] | — | LDOVS[1:0]=00B, $V_{IN}$=3.6V, $I_{LOAD}$=10mA, $\triangle V_{OUT\_LDO}$=2% | — | — | 220 | mV |
| | | | LDOVS[1:0]=01B, $V_{IN}$=3.6V, $I_{LOAD}$=10mA, $\triangle V_{OUT\_LDO}$=2% | — | — | 200 | |
| | | | LDOVS[1:0]=10B, $V_{IN}$=3.6V, $I_{LOAD}$=10mA, $\triangle V_{OUT\_LDO}$=2% | — | — | 240 | |
| | | | LDOVS[1:0]=11B, $V_{IN}$=3.6V, $I_{LOAD}$=10mA, $\triangle V_{OUT\_LDO}$=2% | — | — | 230 | |
| $TC_{LDO}$ | LDO Temperature Coefficient | — | Ta=-40°C~85°C, LDOVS[1:0]=00B, $V_{IN}$=3.6V, $I_{LOAD}$=100µA | — | — | ±250 | ppm/°C |
| $\Delta V_{LINE\_LDO}$ | LDO Line Regulation | — | LDOVS[1:0]=00B, 2.65V≤$V_{IN}$≤5.5V, $I_{LOAD}$=100µA | — | — | 0.7 | %/V |
| | | | LDOVS[1:0]=00B, 2.65V≤$V_{IN}$≤3.6V, $I_{LOAD}$=100µA | — | — | 0.2 | %/V |

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 $V_{IN}$.

### Operational Amplifier Electrical Characteristics

Ta=25°C, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{DD}$ | Operating Voltage | — | — | 2.2 | — | 5.5 | V |
| $I_{OPA}$ | Additional Current for OPA1 | 3V | — | — | 200 | 420 | µA |
| | Additional Current for OPA2 | 3V | — | — | — | 1300 | µA |
| $A_{OL}$ | OPA Open Loop Gain | 3V | — | 80 | 100 | — | dB |
| Ro | Output Resistance | 2.4V~3.6V | Ta=0°C~50°C, $R_{LOAD}$=50kΩ, 0.2V<$V_{OP}$<$V_{DD}$-1.4V (Voltage follower configuration) | — | — | 260 | Ω |
| $I_{OS}$ | Input Offset Current | 2.4V~3.6V | Ta=0°C~50°C | -5 | — | 5 | nA |
| TC | Temperature Coefficient of Offset Voltage | 3V | Ta=0°C~50°C | — | — | ±20 | µV/°C |
| GBW | Gain Bandwidth for OPA1 | 3V | $R_L$=1MΩ, $C_L$=60pF, $V_{IN}$=$V_{CM}$/2 | — | 2 | — | MHz |
| | Gain Bandwidth for OPA2 | 3V | $R_L$=1MΩ, $C_L$=60pF, $V_{IN}$=$V_{CM}$/2 | — | 7 | — | MHz |
| $V_{OS}$ | Input Offset Voltage | 3V | Without calibration (OPAxOF[4:0]=10000B) | -15 | — | 15 | mV |
| $V_{CM}$ | OPA Common Mode Voltage Range | 3V | — | 0.1 | — | $V_{DD}$-1.4 | V |
| $V_{OR}$ | OPA Maximum Output Voltage Range | 3V | — | $V_{SS}$+0.1 | — | $V_{DD}$-0.1 | V |

### Internal Reference Voltage Characteristics

Ta=25°C, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{IREF}$ | Internal Reference Voltage | 3V | IREFEN=1, PVREF=10000000B | -3% | 2.0 | +3% | V |
| $I_{IREF}$ | Additional Current for Internal Reference Voltage | 2.5V~3.3V | IREFEN=1 | — | 650 | 1000 | µA |
| $T_{CIRE}F$ | Temperature Coefficient of Internal Reference Voltage | 3V | Ta=10°C~40°C | — | ±40 | — | ppm/°C |

### Analog Switch Electrical Characteristics

Ta=25°C, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $R_{OP}$ | On Resistance for the Pins expect VG Pin | 3V | Ta=10°C~40°C | — | — | 1300 | Ω |
| $R_{VG}$ | On Resistance for VG Pin | 3V | Ta=10°C~40°C | — | — | 20 | Ω |
| $I_{LEAK}$ | Leakage Current for VG Pin | 3V | Ta=10°C~40°C | -5 | ±0.5 | 5 | nA |

### LNOPA Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $I_{OPA}$ | Additional Current for OPA Enable | — | No load | — | 350 | 550 | µA |
| $V_{OS}$ | Input Offset Voltage | — | — | -2 | — | +2 | mV |
| $V_{CM\_OPA}$ | Common Mode Voltage Range | — | — | $V_{SS}$+0.15 | — | $V_{OREG}$-1.4 | V |
| PSRR | Power Supply Rejection Ratio | — | — | 55 | 90 | — | dB |
| CMRR | Common Mode Rejection Ratio | — | — | 55 | 90 | — | dB |

### 12-bit D/A Converter Electrical Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{DD}$ | Operating Voltage | — | — | 2.4 | — | 3.6 | V |
| DNL | Differential Non-linearity | 3V | DACVRS[1:0]=00B | — | — | ±8 | LSB |
| INL | Integral Non-linearity | 3V | DACVRS[1:0]=00B | — | — | ±20 | LSB |
| $V_{DACO}$ | Output Voltage Range (DAC1, DAC2) | — | — | 0.00 | — | 1.00 | $V_{DACVREF}$ |
| | Output Voltage Range (DAC3) | — | — | $V_{SS}$+0.3 | — | $V_{OREG}$-1.4 | V |
| $V_{DACO\_RIPPLE}$ | Output Voltage Ripple | 3V | DACVRS[1:0]=10B | -0.6 | — | +0.6 | mV |

### A/D Converter Electrical Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $AV_{DD}$ | Supply Voltage for A/D, PGA | — | — | 2.4 | — | 3.3 | V |
| $I_{ADC}$ | Additional Current for ADC Enable | — | FLMS[2:0]=010B, 000B $V_{IN}$=($V_{CM\_PGA\_MAX}$+$V_{CM\_PGA\_MIN}$)/2 | — | 600 | 750 | µA |
| $I_{ADSTB}$ | Standby Current | — | MCU enter SLEEP mode, no load | — | — | 1 | µA |
| $N_R$ | Resolution | — | — | — | — | 24 | Bit |
| INL | Integral Non-linearity | — | $AV_{DD}$=3.3V, $V_{REF}$=$V_{OREG}$/2=1.65V, △SI=±450mV, PGAGN=1 | — | ±50 | ±200 | ppm |
| NFB | Noise Free Bits | — | $f_{MCLK}$=4MHz, FLMS[2:0]=000B, $AV_{DD}$=3.3V, $V_{REF}$=$V_{OREG}$/2=1.65V, PGAGN=128, OSR=16384 | — | 16 | — | Bit |
| ENOB | Effective Number of Bits | — | $f_{MCLK}$=4MHz, FLMS[2:0]=000B, $AV_{DD}$=3.3V, $V_{REF}$=$V_{OREG}$/2=1.65V, ADC gain=1, PGA gain=128, OSR=16384 | — | 18.7 | — | 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, R_FILSEL=1, R_CKCHOP=1 | 4 | — | 1042 | Hz |
| | | — | $f_{MCLK}$=4MHz, FLMS[2:0]=010B, R_FILSEL=1, R_CKCHOP=1 | 10 | — | 2604 | Hz |
| $V_{REFP}$ | Reference Input Voltage | — | — | $V_{REFN}$+0.8 | — | $AV_{DD}$ | V |
| $V_{REFN}$ | | — | — | 0 | — | $V_{REFP}$-0.8 | V |
| $V_{REF}$ | | — | $V_{REF}$=($V_{REFP}$-$V_{REFN}$)×VREFGN | 0.80 | — | 1.75 | V |

### PGA Electrical Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $V_{CM\_PGA}$ | Common Mode Voltage Range | — | — | 0.40 | — | $AV_{DD}$-0.95 | V |
| $\Delta D_I$ | Differential input voltage range | — | Gain=PGAGN×ADGN | -$V_{REF}$/Gain | — | +$V_{REF}$/Gain | V |

## I²C Electrical Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|---|
| | | $V_{DD}$ | Conditions | | | | |
| $f_{I2C}$ | I²C Standard Mode (100kHz) $f_{SYS}$ Frequency (Note) | — | No clock debounce | 2 | — | — | MHz |
| | | | 2 system clock debounce | 4 | — | — | |
| | | | 4 system clock debounce | 4 | — | — | |
| | I²C Fast Mode (400kHz) $f_{SYS}$ Frequency (Note) | — | No clock debounce | 4 | — | — | MHz |
| | | | 2 system clock debounce | 8 | — | — | |
| | | | 4 system clock debounce | 8 | — | — | |
| $f_{SCL}$ | SCL Clock Frequency | 3V/5V | Standard mode | — | — | 100 | kHz |
| | | | Fast mode | — | — | 400 | |
| $t_{SCL(H)}$ | SCL Clock High Time | 3V/5V | Standard mode | 3.5 | — | — | µs |
| | | | Fast mode | 0.9 | — | — | |
| $t_{SCL(L)}$ | SCL Clock Low Time | 3V/5V | Standard mode | 3.5 | — | — | µs |
| | | | Fast mode | 0.9 | — | — | |
| $t_{FALL}$ | SCL and SDA Fall Time | 3V/5V | Standard mode | — | — | 1.3 | µs |
| | | | Fast mode | — | — | 0.34 | |
| $t_{RISE}$ | SCL and SDA Rise Time | 3V/5V | Standard mode | — | — | 1.3 | µs |
| | | | Fast mode | — | — | 0.34 | |
| $t_{SU(SDA)}$ | SDA Data Setup Time | 3V/5V | Standard mode | 0.25 | — | — | µs |
| | | | Fast mode | 0.1 | — | — | |
| $t_{H(SDA)}$ | SDA Data Hold Time | 3V/5V | — | 0.1 | — | — | µs |
| $t_{VD(SDA)}$ | SDA Data Valid Time | 3V/5V | — | — | — | 0.6 | µs |
| $t_{SU(STA)}$ | Start Condition Setup Time | 3V/5V | Standard mode | 3.5 | — | — | µs |
| | | | Fast mode | 0.6 | — | — | |
| $t_{H(STA)}$ | Start Condition Hold Time | 3V/5V | Standard mode | 4.0 | — | — | µs |
| | | | Fast mode ($f_{SYS} \geq$ 8MHz) | 0.6 | — | — | |
| | | | Fast mode ($f_{SYS} <$ 8MHz) | 0.8 | — | — | |
| $t_{SU(STO)}$ | Stop Condition Setup Time | 3V/5V | Standard mode | 3.5 | — | — | µs |
| | | | Fast mode | 0.6 | — | — | |

Note: Using the debounce function can make the transmission more stable and reduce the probability of communication failure due to interference.



**I²C Timing Diagram**
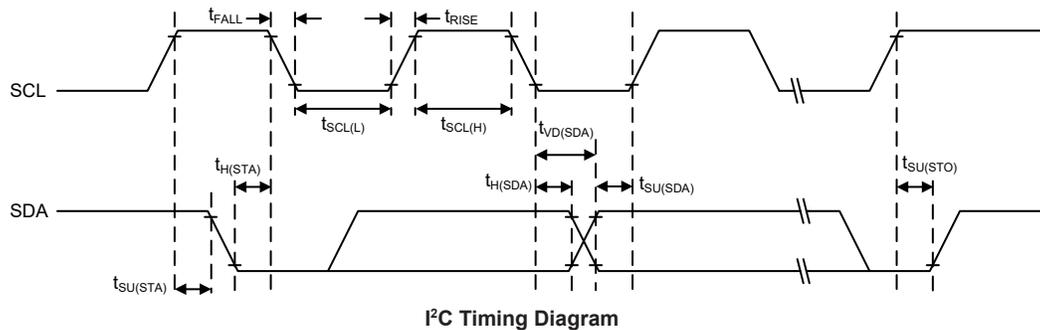
## Power-on Reset Characteristics

Ta=-40°C~85°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 |
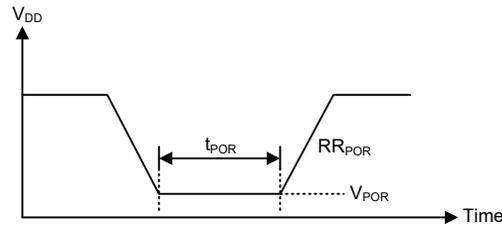


## System Architecture

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 enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one or two cycles for most of the standard or extended instructions respectively. The exceptions to these are branch or call instructions which need one more cycle. 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 affordable, high-volume production for controller applications.

### Clocking and Pipelining

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

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.
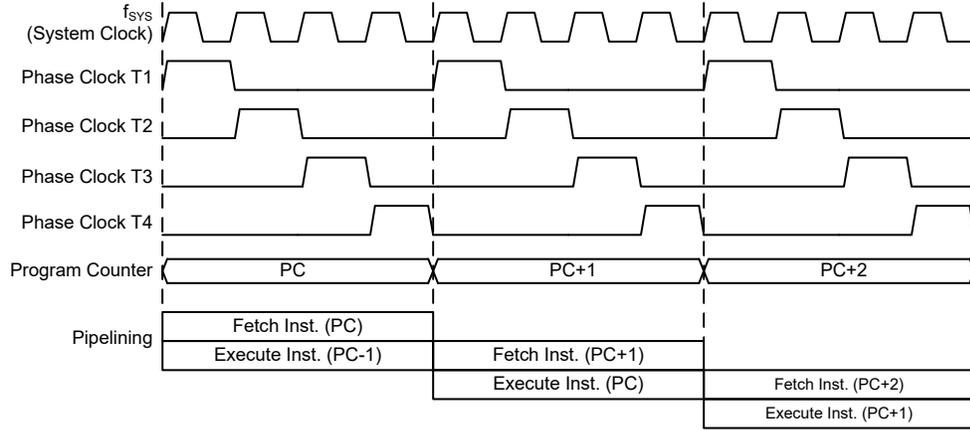
**System Clocking and Pipelining**
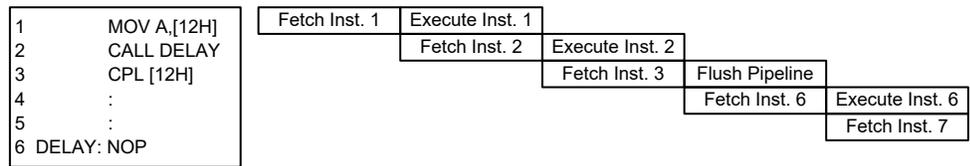


**Instruction Fetching**

## 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 | |
|---|---|
| **High Byte** | **Low Byte (PCL)** |
| PC12~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 12 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, 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.
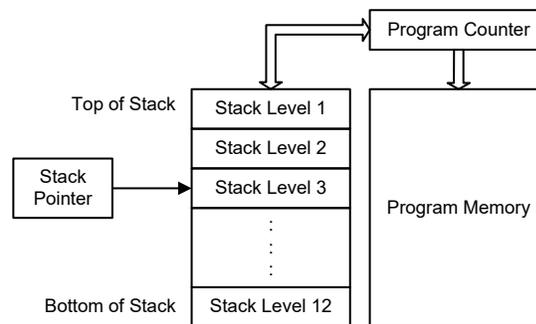


## Arithmetic and Logic Unit – ALU

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,
  LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA

- Logic operations:

  AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
  LAND, LOR, LXOR, LANDM, LORM, LXORM, LCPL, LCPLA

- Rotation:

  RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,
  LRRA, LRR, LRRCA, LRRC, LRLA, LRL, LRLCA, LRLC

- Increment and Decrement:

  INCA, INC, DECA, DEC,
  LINCA, LINC, LDECA, LDEC

- Branch decision:

  JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI,
  LSZ, LSZA, LSNZ, LSIZ, LSDZ, LSIZA, LSDZA

# OTP Program Memory

The Program Memory is the location where the user code or program is stored. The device is supplied with One-Time Programmable, OTP memory where users can program their application code into the device.

## Structure

The Program Memory has a capacity of 4K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer registers.
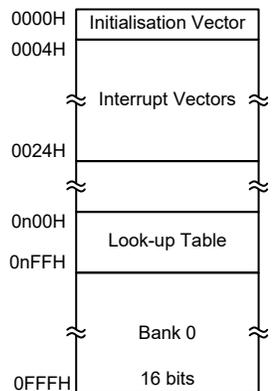


**Program Memory Structure**

## Special Vectors

Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 0000H 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 registers, 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]" instruction when the memory [m] is located in sector 0. If the memory [m] is located in other sectors except sector 0, the data can be retrieved from the program memory using the corresponding extended table read instruction "LTABRD [m]". 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.

The accompanying diagram illustrates the addressing data flow of the look-up table.
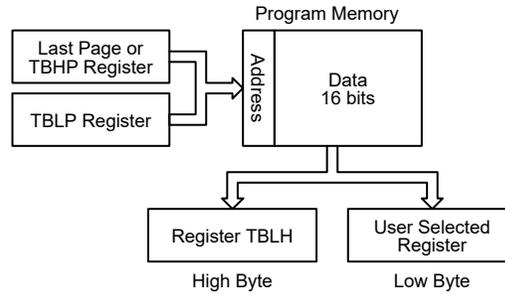
**Table Program Example**

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 "0F00H" which refers to the start address of the last page within the 4K words Program Memory of the device. The table pointer low byte register is set 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 "0F06H" or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by the TBLP and TBHP registers if the "TABRD [m]" or "LTABRD [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]" or "LTABRD [m]" instruction is executed.

Because the TBLH register is a read/write register and can 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,0FH         ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1    ; transfers value in table referenced by table pointer, data at
                  ; program memory address "0F06H" 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 "0F05H" transferred to tempreg2 and TBLH
                  ; in this example the data "1AH" is transferred to tempreg1 and
                  ; data "0FH" to register tempreg2 the value "00H" will be
                  ; transferred to the high byte register TBLH
:
:
org 0F00H         ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
```

### In Circuit Programming – ICP

The provision of OTP type Program Memory, users can program their application One-Time into the device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 5-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with an un-programmed microcontroller, and then programming the program at a later stage.

| Holtek Writer Pins | MCU Programming Pins | Pin Description |
|---|---|---|
| ICPDA | PA0 | Programming Serial Data/Address |
| ICPCK | PA2 | Programming Clock |
| VPP | VPP | Programming OTP ROM power supply (8.5V) |
| VDD | VDD | Power Supply. A 0.1µF capacitor is required to be connected between VDD and VSS for programming. |
| VSS | VSS | Ground |

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

During the programming process, the 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.
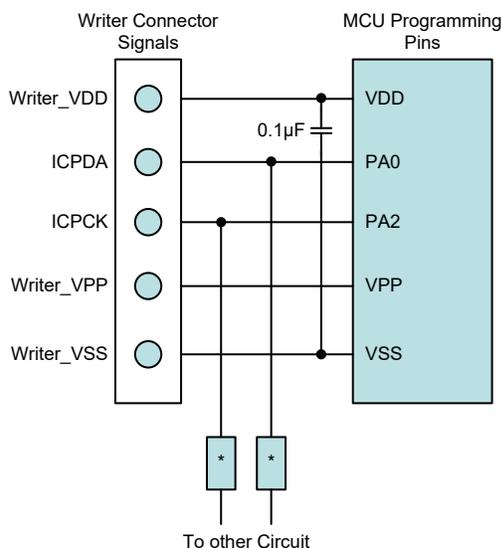


Note: 1. A 0.1µF capacitor is required to be connected between VDD and VSS for ICP programming, and as close to these pins as possible.
2. * may be resistor or capacitor. The resistance of * must be greater than 1kΩ or the capacitance of * must be less than 1nF.

### On-Chip Debug Support – OCDS

There is an EV chip named BH66EV2040 which can emulate the BH66R2040 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 and the package type. 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. 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 | MCU 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 |

## OTP ROM Parameter Program – ORPP

This device contains an ORPP function. The provision of the ORPP function offers users the convenience of OTP Memory programming features. Note that the Write operation only writes data to the last page of OTP Program Memory, and the data can only be written once and cannot be erased.

Before the write operation is implemented, the VPP pin must be connected to an 8.5V power and after the write operation is completed, the high voltage power should be removed from the VPP pin. If the VPP function is pin-shared with an I/O port, the corresponding I/O port cannot be set as an output when it is used as the VPP function.

### ORPP Registers

Three registers control the overall operation of the internal ORPP function. These are data registers ODL and ODH, and a control register OCR.

| Registers Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OCR | — | — | — | — | WREN | WR | — | — |
| ODL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| ODH | D15 | D14 | D13 | D12 | D11 | D10 | D9 | D8 |

**ORPP Register List**

#### • ODL 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    **D7~D0**: ORPP program memory data bit 7 ~ bit 0

#### • ODH Register

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

Bit 7~0    **D15~D8**: ORPP program memory data bit 15 ~ bit 8

• **OCR Register**

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

Bit 7~4      Unimplemented, read as "0"

Bit 3      **WREN**: ORPP Write Enable
          0: Disable
          1: Enable
          This is the ORPP Write Enable Bit which must be set high before write operations are carried out. This bit will be automatically reset to zero by the hardware after the write cycle has finished. Clearing this bit to zero will inhibit ORPP write operations.

Bit 2      **WR**: ORPP Write Control
          0: Write cycle has finished
          1: Activate a write cycle
          This is the ORPP 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 1~0      Unimplemented, read as "0"

Note: 1. The WREN and WR cannot be set high at the same time in one instruction.

  2. Note that the CPU will be stopped when a write operation is successfully activated.

  3. Ensure that the $f_{SUB}$ clock is stable before executing the write operation.

  4. Ensure that the write operation is totally complete before executing other operations.

### ORPP Writing Data to the OTP Program Memory

For ORPP write operation the data to be written should be placed in the ODH and ODL registers and the desired write address should first be placed in the TBLP register. To write data to the OTP Program Memory, the write enable bit, WREN, in the OCR register must first be set high to enable the write function. After this, the WR bit in the OCR register must be immediately set high to initiate a write cycle. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set again after a valid write activation procedure has completed. Note that the CPU will be stopped when a write operation is successfully activated. When the write cycle terminates, the CPU will resume executing the application program. And the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the OTP Program Memory.

### ORPP Reading Data from the OTP Program Memory

For ORPP read operation the desired address should first be placed in the TBLP register. Then the data can be retrieved from the program memory using the "TABRDL [m]" instruction. 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.

### Programming Considerations

Care must be taken that data is not inadvertently written to the OTP Program Memory. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. 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 set high again after a write activation procedure has completed. Note that the device should not enter the IDLE or SLEEP mode until the ORPP write operation is totally complete. Otherwise, the ORPP write operation will fail.

### Programming Examples

#### ORPP Reading Data from the OTP Program Memory

```
Tempreg1 db?             ; temporary register
MOV A, 03H
MOV TBLP, A              ; set read address 03H
TABRDL Tempreg1          ; transfers value in table (last page) referenced by table
                         ; pointer, data at program memory address "0F03H"
                         ; transferred to tempreg1 and TBLH
```

#### ORPP Writing Data to the OTP Program Memory

```
MOV  A, ORPP_ADRES       ; user defined address
MOV  TBLP, A
MOV  A, HORPP_DATA_L     ; user defined data
MOV  ODL, A
MOV  A, ORPP_DATA_H
MOV  ODH, A
MOV  A, 00H
MOV  OCR, A
CLR  EMI
SET  WREN                ; set WREN bit, enable write operation
SET  WR                  ; start Write Cycle – set WR bit – executed immediately
                         ; after setting WREN bit
SET  EMI
BACK:
SZ   WR                  ; check for write cycle end
JMP  BACK
NOP
```

# Data Memory

The Data Memory is an 8-bit wide RAM internal memory and is the location where temporary information is stored.

Categorized into two types, the first of these is an area of RAM where special function registers are located. These registers have fixed locations and are necessary for correct operation of the device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is reserved for general purpose use. All locations within this area are read and write accessible under program control.

There is another area of the Data Memory reserved for the Sine Pattern function. The addresses of the Sine Pattern Memory area overlap those in the Special Purpose Data Memory area.

## Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide RAM. Each of the Data Memory Sector is categorized into two types, the special Purpose Data Memory and the General Purpose Data Memory. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH. The Sine Wave Pattern Memory is located from 00H to 7FH in Sector 3.

Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value if using the indirect addressing method.

| Special Purpose Data Memory | General Purpose Data Memory | | Sine Wave Pattern Memory | |
|---|---|---|---|---|
| Located Sectors | Capacity | Sector: Address | Capacity | Sector: Address |
| 0 | 512×8 | 0: 80H~FFH<br>1: 80H~FFH<br>2: 80H~FFH<br>3: 80H~FFH | 128×8 | 3: 00H~7FH |

**Data Memory Summary**
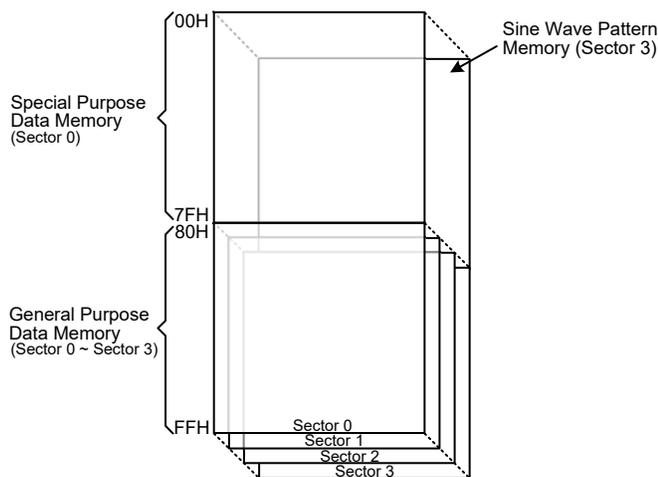


**Data Memory Structure**

### Data Memory Addressing

For the device that supports the extended instructions, there is no Bank Pointer for Data Memory addressing. For Data Memory the desired Sector is pointed by the MP1H or MP2H register and the certain Data Memory address in the selected sector is specified by the MP1L or MP2L register when using indirect addressing access.

Direct Addressing can be used in all sectors using the extended instructions which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address "m" in the extended instructions has 10 valid bits for this device, the high byte indicates a sector and the low byte indicates a specific address.

### 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 programming 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".

| Sector 0 | | Sector 0 | |
| --- | --- | --- | --- |
| 00H | IAR0 | 40H | OCR |
| 01H | MP0 | 41H | ODL |
| 02H | IAR1 | 42H | ODH |
| 03H | MP1L | 43H | PWRC |
| 04H | MP1H | 44H | PGAC0 |
| 05H | ACC | 45H | PGAC1 |
| 06H | PCL | 46H | PGACS |
| 07H | TBLP | 47H | ADRL |
| 08H | TBLH | 48H | ADRM |
| 09H | TBHP | 49H | ADRH |
| 0AH | STATUS | 4AH | ADCR0 |
| 0BH | | 4BH | ADCR1 |
| 0CH | IAR2 | 4CH | ADCS |
| 0DH | MP2L | 4DH | ░░░░░░░ |
| 0EH | MP2H | 4EH | CHOP |
| 0FH | RSTFC | 4FH | IREFC |
| 10H | SCC | 50H | PVREF |
| 11H | HIRCC | 51H | OPA1C |
| 12H | | 52H | AFEDA1C |
| 13H | | 53H | AFEDA1L |
| 14H | PA | 54H | AFEDA1H |
| 15H | PAC | 55H | AFEDA2C |
| 16H | PAPU | 56H | AFEDA2L |
| 17H | PAWU | 57H | AFEDA2H |
| 18H | LVRC | 58H | SGC |
| 19H | LVDC | 59H | SGN |
| 1AH | TLVRC | 5AH | SGDN |
| 1BH | MFI | 5BH | OPA3C |
| 1CH | | 5CH | OPA2C |
| 1DH | | 5DH | ASWAI0 |
| 1EH | | 5EH | ASWAI1 |
| 1FH | WDTC | 5FH | ASWAI2 |
| 20H | INTEG | 60H | ASWAI3 |
| 21H | INTC0 | 61H | ASWAI4 |
| 22H | INTC1 | 62H | ASWAI5 |
| 23H | INTC2 | 63H | ASWAI6 |
| 24H | | 64H | ASWAI7 |
| 25H | PB | 65H | ASWAI8 |
| 26H | PBC | 66H | ASWAI9 |
| 27H | PBPU | 67H | ASWDAC |
| 28H | | 68H | AFET0 |
| 29H | | 69H | AFET1 |
| 2AH | | 6AH | LNOPC |
| 2BH | PSCR | 6BH | LNDA3C |
| 2CH | TB0C | 6CH | LNDA3L |
| 2DH | TB1C | 6DH | LNDA3H |
| 2EH | RSTC | 6EH | |
| 2FH | | 6FH | |
| 30H | USR | 70H | CTMC0 |
| 31H | UCR1 | 71H | CTMC1 |
| 32H | UCR2 | 72H | CTMDL |
| 33H | UCR3 | 73H | CTMDH |
| 34H | BRDH | 74H | CTMAL |
| 35H | BRDL | 75H | CTMAH |
| 36H | UFCR | 76H | |
| 37H | TXR_RXR | 77H | |
| 38H | RxCNT | 78H | IFS |
| 39H | SIMC0 | 79H | |
| 3AH | SIMC1 | 7AH | PAS0 |
| 3BH | SIMD | 7BH | PAS1 |
| 3CH | SIMA/SIMC2 | 7CH | PBS0 |
| 3DH | SIMTOC | 7DH | |
| 3EH | | 7EH | |
| 3FH | | 7FH | ░░░░░░░ |

▢ : Unused, read as 00H          ▨ : Reserved, cannot be changed

**Special Purpose Data Memory Structure**

## Special Function Register Description

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

### Indirect Addressing Registers – IAR0, IAR1, IAR2

The Indirect Addressing Registers, IAR0, IAR1 and IAR2, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0, IAR1 and IAR2 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0, MP1L/MP1H or MP2L/MP2H. Acting as a pair, IAR0 and MP0 can together access data only from Sector 0 while the IAR1 register together with MP1L/MP1H register pair and IAR2 register together with MP2L/MP2H register pair can access data from any Data Memory sector. 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, MP1H/MP1L, MP2H/MP2L

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

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

### Indirect Addressing Program Example

#### Example 1

```
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:
```

**Example 2**

```
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,01h                ; setup the memory sector
    mov mp1h,a
    mov a,offset adres1      ; Accumulator loaded with first RAM address
    mov mp1l,a               ; setup memory pointer with first RAM address
loop:
    clr IAR1                 ; clear the data at address defined by MP1L
    inc mp1l                 ; increment memory pointer MP1L
    sdz block                ; check if last memory location has been cleared
    jmp loop
continue:
    :
```

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

## Accumulator – ACC

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

## Program Counter Low Byte 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. The TBLP and TBHP registers are the table pointer pair and indicates the location where the table data is located. Their value must be setup before any table read instructions 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), SC flag, CZ flag, 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, C, SC and CZ flags generally reflect the status of the latest operations.

- C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- AC is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
- Z is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.
- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
- PDF is cleared by a system power-up or executing the "CLR WDT" instruction. PDF is set by executing the "HALT" instruction.
- TO is cleared by a system power-up or executing the "CLR WDT" or "HALT" instruction. TO is set by a WDT time-out.
- SC is the result of the "XOR" operation which is performed by the OV flag and the MSB of the current instruction operation result.
- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.

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

- **STATUS Register**

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

"x": unknown

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

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

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

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

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

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

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

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

## Oscillators

Various oscillator types 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 by using some 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 oscillator provides 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 Oscillator | HIRC | 8MHz |
| 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 8MHz RC oscillator, HIRC. The low speed oscillator is the internal 32kHz RC oscillator, LIRC. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.

**System Clock Configurations**
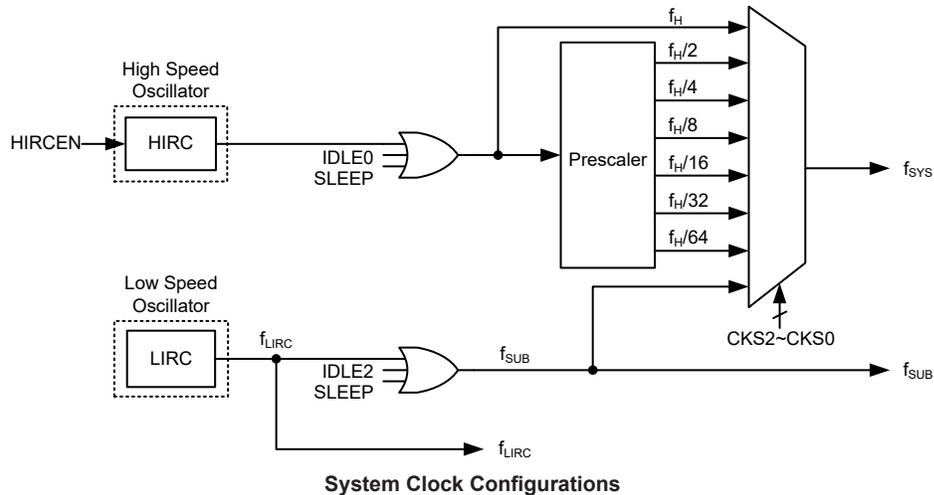
### Internal High Speed RC Oscillator – HIRC

The high speed internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 8MHz. 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, requiring no external components for its implementation.

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

The main system clock, can come from 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 is sourced from the HIRC oscillator. The low speed system clock source is sourced from the LIRC oscillator. The other choice, which is a divided version of the high speed system oscillator has a range of $f_H/2$~$f_H/64$.
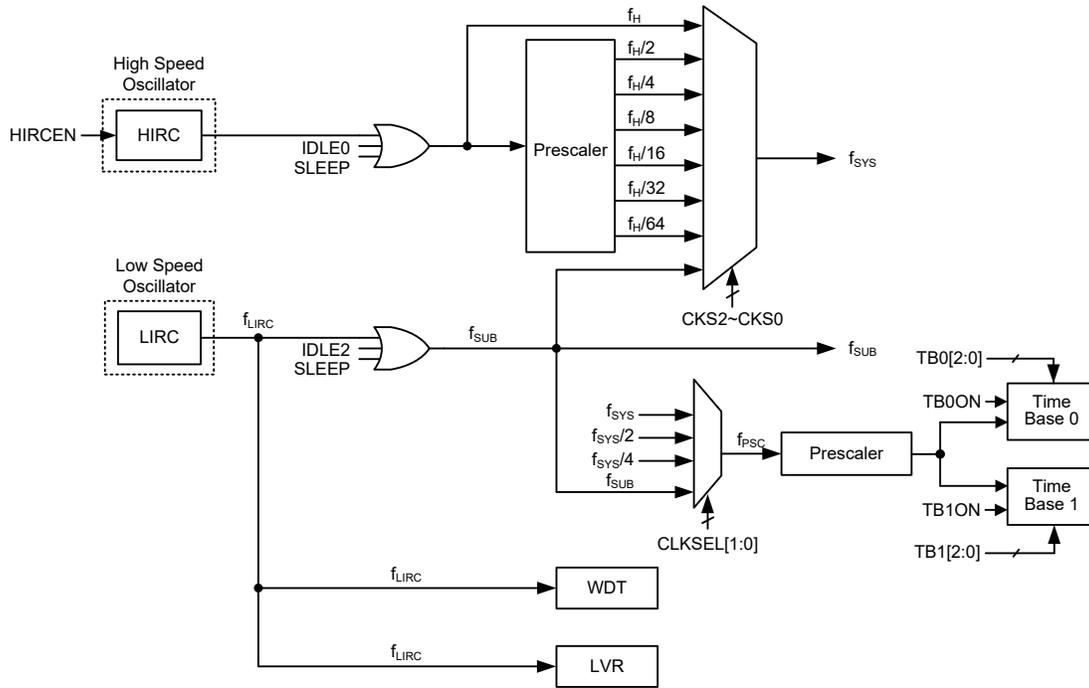
**Device Clock Configurations**

Note: When the system clock source $f_{SYS}$ is switched to $f_{SUB}$ from $f_H$, the high speed oscillation can be stopped to conserve the power or continue to oscillate to provide the clock source, $f_H \sim f_H/64$, for peripheral circuits 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 | $f_H \sim f_H/64$ | 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

This is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source will come from the high speed oscillator HIRC. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 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 $f_{SUB}$, which is derived from the LIRC oscillator.

### SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the FHIDEN and FSIDEN bits in the SCC register are both low. In the SLEEP mode the CPU will be stopped. The $f_{SUB}$ clock provided to the peripheral function will also be stopped. However the $f_{LIRC}$ clock will continue to operate if the WDT function is enabled by the WDTC register.

### IDLE0 Mode

The IDLE0 Mode is entered when a 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 on to drive some peripheral functions.

### IDLE1 Mode

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

### IDLE2 Mode

The IDLE2 Mode is entered when a HALT instruction is executed and when the FHIDEN bit in the SCC register is high and the FSIDEN bit in the SCC register is low. In the IDLE2 Mode the CPU will be switched off but the high speed oscillator will be 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 | — | — | — | — | — | — | HIRCF | HIRCEN |

**System Operating Mode Control Register 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 a "HALT" instruction.

Bit 0 **FSIDEN**: Low Frequency oscillator control when CPU is switched off
    0: Disable
    1: Enable

This bit is used to control whether the low speed oscillator is activated or stopped when the CPU is switched off by executing a "HALT" instruction.

Note: A certain delay is required before the relevant clock is successfully switched to the target clock source after any clock switching setup using the CKS2~CKS0 bits. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock source.

Clock switching delay time=$4 \times t_{SYS}+[0 \sim (1.5 \times t_{Curr.}+0.5 \times t_{Tar.})]$, where $t_{Curr.}$ indicates the current clock period, $t_{Tar.}$ Indicates the target clock period and $t_{SYS}$ indicates the current sytem clock period.

• **HIRCC Register**

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

Bit 7~2 Unimplemented, read as "0"

Bit 1 **HIRCF**: HIRC oscillator stable flag
    0: Unstable
    1: 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.
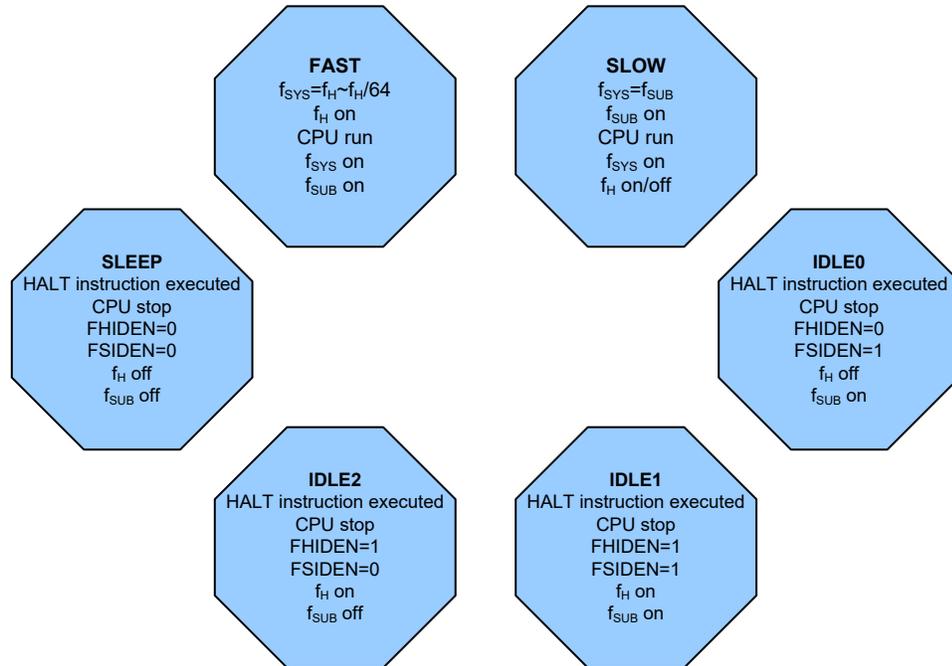
Bit 0 **HIRCEN**: HIRC oscillator enable control
    0: Disable
    1: Enable

## 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 a HALT instruction is executed, whether the device enters the IDLE Mode or the SLEEP Mode is determined by the condition of the FHIDEN and FSIDEN bits in the SCC register.

**FAST**
$f_{SYS}=f_H\sim f_H/64$
$f_H$ on
CPU run
$f_{SYS}$ on
$f_{SUB}$ on

**SLOW**
$f_{SYS}=f_{SUB}$
$f_{SUB}$ on
CPU run
$f_{SYS}$ on
$f_H$ on/off

**SLEEP**
HALT instruction executed
CPU stop
FHIDEN=0
FSIDEN=0
$f_H$ off
$f_{SUB}$ off

**IDLE0**
HALT instruction executed
CPU stop
FHIDEN=0
FSIDEN=1
$f_H$ off
$f_{SUB}$ on

**IDLE2**
HALT instruction executed
CPU stop
FHIDEN=1
FSIDEN=0
$f_H$ on
$f_{SUB}$ off

**IDLE1**
HALT instruction executed
CPU stop
FHIDEN=1
FSIDEN=1
$f_H$ on
$f_{SUB}$ on

### 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 setting 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.
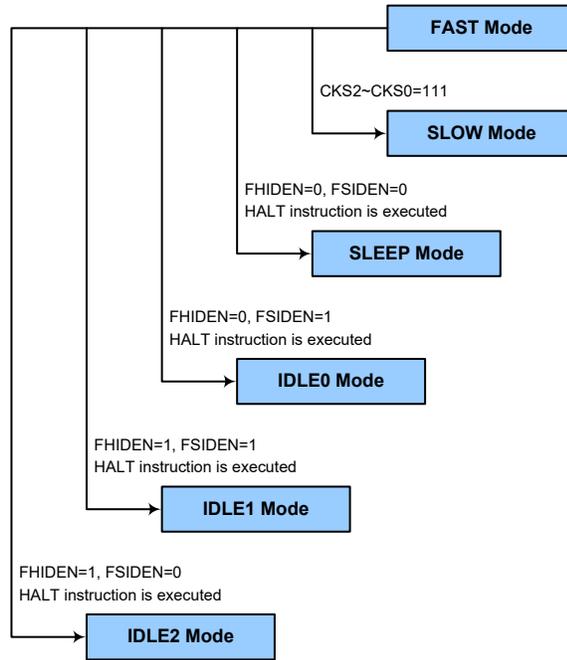
### SLOW Mode to FAST Mode Switching

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 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.
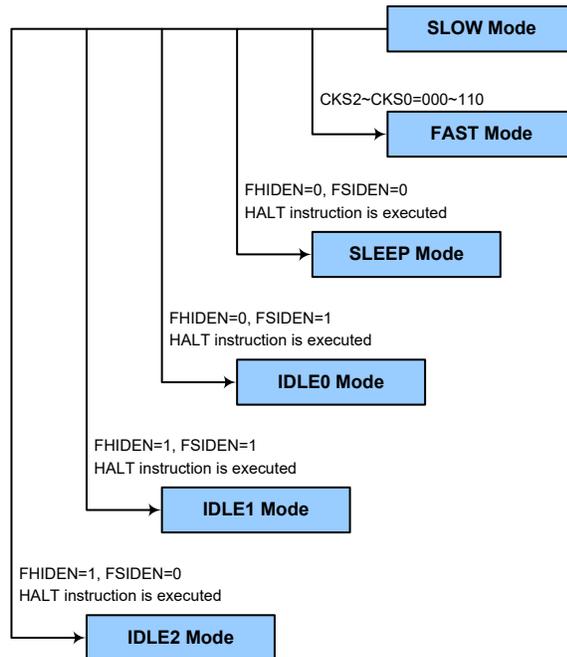
### Entering the SLEEP Mode

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 WDT timeout 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 WDT timeout 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 WDT timeout 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 WDT timeout 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. In addition, the I/O pin-shared with VPP must not be set to output high, as this could result in increased current consumption.

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 been 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 IDLE or SLEEP 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 Port A pin wake-up occurs, the program will resume execution at the instruction following the "HALT" instruction.If the system is woken up by an interrupt, then two possible situations may occur. The first is where the related interrupt is disabled or the interrupt is enabled but the stack is full, in which case the program will resume execution at the instruction following the "HALT" instruction. In this situation, the interrupt which woke up the device will not

be immediately serviced, but will rather be serviced later when the related interrupt is finally enabled or when a stack level becomes free. The other situation is where the related interrupt is enabled and the stack is not full, in which case the regular interrupt response takes place. If an interrupt request flag is set high before entering the SLEEP or IDLE Mode, the wake-up function of the related interrupt will be disabled.

## Watchdog Timer

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

### Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal 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 Watchdog Timer enable/disable and the MCU reset 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 enable 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
         000B: $2^8/f_{LIRC}$
         001B: $2^{10}/f_{LIRC}$
         010B: $2^{12}/f_{LIRC}$
         011B: $2^{14}/f_{LIRC}$
         100B: $2^{15}/f_{LIRC}$
         101B: $2^{16}/f_{LIRC}$
         110B: $2^{17}/f_{LIRC}$
         111B: $2^{18}/f_{LIRC}$

         These three bits determine the division ratio of the watchdog timer source clock, which in turn determines the time-out 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
Refer to the Internal Reset Control section.

Bit 2    **LVRF**: LVR function reset flag
Refer to the Low Voltage Reset section.

Bit 1    **LRF**: LVRC register software reset flag
Refer to the Low Voltage Reset section.

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. This bit can only be cleared to zero by application program.

## Watchdog Timer Operation

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

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

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

**Watchdog Timer**
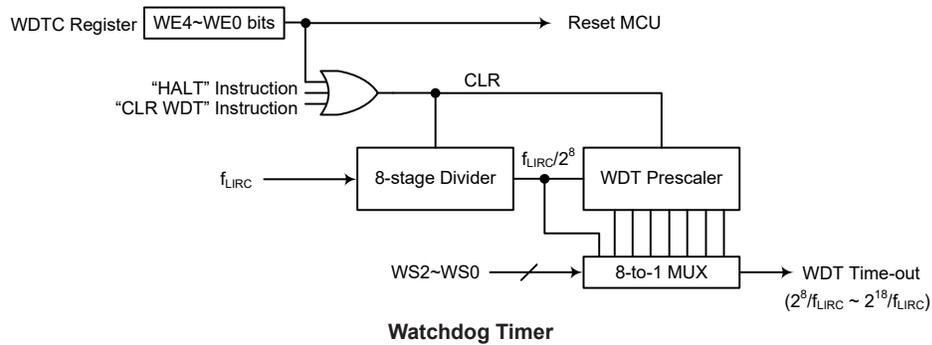
## Reset and Initialisation

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**
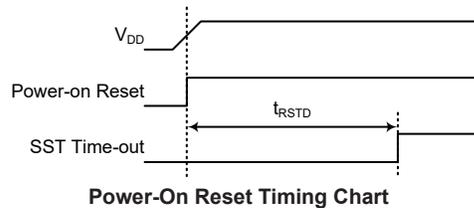
#### Internal Reset Control

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: MCU reset (reset will be active after $t_{SRESET}$ time for de-bounce time).

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. All resets will reset this register to POR value except the WDT time-out hardware warm reset.

- **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. This bit can only be cleared to zero by application program.

Bit 2    **LVRF**: LVR function reset flag
Refer to the Low Voltage Reset section.

Bit 1    **LRF**: LVRC register software reset flag
Refer to the Low Voltage Reset section.

Bit 0    **WRF**: WDT control register software reset flag
Refer to the Watchdog Timer Control Register section.

## 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 when the value falls below a certain predefined level.

The LVR function can be enabled or disabled by the LVRC control register. If the LVRC control register is configured to enable the LVR function, the LVR function will be always enabled except in the SLEEP or IDLE mode. If the supply voltage of the device drops to within a range of $0.9V \sim V_{LVR}$ such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set high. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between $0.9V \sim V_{LVR}$ must exist for a time greater than that specified by $t_{LVR}$ in the LVD/LVR Electrical 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 $t_{LVR}$ value can be selected by the TLVR1~TLVR0 bits in the TLVRC register.

The actual VLVR value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by the environmental noise or software setting, 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 high. After power on the register will have the value of 01100110B. Note that the LVR function will be automatically disabled when the device enters the IDLE/SLEEP mode.



**Low Voltage Reset Timing Chart**
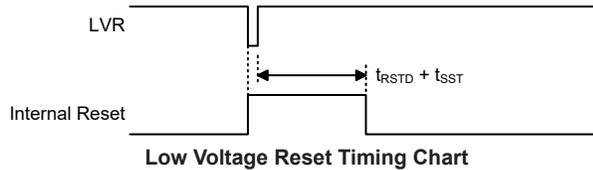
- **LVRC Register**

| 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 | 1 | 0 | 0 | 1 | 1 | 0 |

Bit 7~0　　**LVS7~LVS0**: LVR voltage select
　　　　　　01100110B: 1.7V
　　　　　　01010101B: 1.9V
　　　　　　00110011B: 2.55V
　　　　　　10011001B: 3.15V
　　　　　　10101010B: 3.8V
　　　　　　11110000B: LVR disable
　　　　　　Other values: Generates an MCU reset

When an actual low voltage condition as specified above occurs, an MCU reset will be generated. The reset operation will be activated after the low voltage condition keeps more than a $t_{LVR}$ time. The actual $t_{LVR}$ value can be selected by the TLVR1~TLVR0 bits in the TLVRC register. In this situation the register contents will remain the same after such a reset occurs.

Any register value, other than the five defined LVR values and 11110000B 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.

- **TLVRC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | TLVR1 | TLVR0 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 1 |

Bit 7~2　　Unimplemented, read as "0"

Bit 1~0　　**TLVR1~TLVR0**: Minimum low voltage width to reset time ($t_{LVR}$)
　　　　　　00: (7~8)×$t_{LIRC}$
　　　　　　01: (31~32)×$t_{LIRC}$
　　　　　　10: (63~64)×$t_{LIRC}$
　　　　　　11: (127~128)×$t_{LIRC}$

• **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
           Refer to the Internal Reset Control section.

Bit 2      **LVRF**: LVR function reset flag
           0: Not occurred
           1: Occurred
           This bit is set to 1 when a specific low voltage reset condition occurs. This bit can only
           be cleared to zero by application program.

Bit 1      **LRF**: LVRC register software reset flag
           0: Not occurred
           1: Occurred
           This bit is set high 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
           zero by application program.

Bit 0      **WRF**: WDT control register software reset flag
           Refer to the Watchdog Timer Control Register section.

## Watchdog Time-out Reset during Normal Operation

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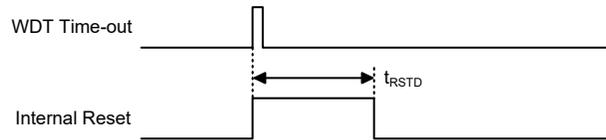


**WDT Time-out Reset during Normal Operation Timing Chart**

## 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 and PDF flags will be set to "1". Refer to the System Start
Up Time Characteristics for $t_{SST}$ details.
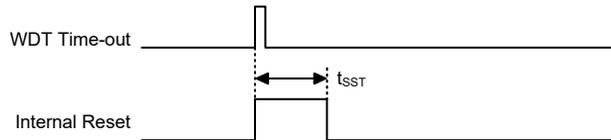


**WDT Time-out Reset during SLEEP or IDLE Mode Timing Chart**

## 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": 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 | Cleared after reset, WDT begins counting |
| Timer Modules | Timer Modules 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 the microcontroller internal registers.

| Register | Power-On Reset | LVR Reset (Normal Operation) | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|----------|----------------|------------------------------|--------------------------------|---------------------------|
| IAR0 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP0 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| IAR1 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP1L | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP1H | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ACC | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| PCL | 0000 0000 | 0000 0000 | 0000 0000 | 0000 0000 |
| TBLP | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBLH | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| TBHP | xxxx xxxx | uuuu uuuu | uuuu uuuu | uuuu uuuu |
| STATUS | xx00 xxxx | uuuu uuuu | uu1u uuuu | uu11 uuuu |
| IAR2 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP2L | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP2H | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| RSTFC | ---- 0x00 | ---- u1uu | ---- uuuu | ---- uuuu |
| SCC | 000- 0000 | 000- 0000 | 000- 0000 | uuu- uuuu |
| HIRCC | ---- --01 | ---- --01 | ---- --01 | ---- --uu |
| PA | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAC | 1111 1111 | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAPU | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PAWU | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| LVRC | 0110 0110 | uuuu uuuu | 0110 0110 | uuuu uuuu |
| LVDC | 0000 -000 | 0000 -000 | 0000 -000 | uuuu -uuu |
| TLVRC | ---- --01 | ---- --01 | ---- --01 | ---- --uu |
| MFI | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| WDTC | 0101 0011 | 0101 0011 | 0101 0011 | uuuu uuuu |
| INTEG | ---- 0000 | ---- 0000 | ---- 0000 | uuuu uuuu |

| Register | Power-On Reset | LVR Reset (Normal Operation) | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|---|---|---|---|---|
| INTC0 | - 0 0 0 0 0 0 0 | - 0 0 0 0 0 0 0 | - 0 0 0 0 0 0 0 | - u u u u u u u |
| INTC1 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | u u u u u u u u |
| INTC2 | - - 0 0 - - 0 0 | - - 0 0 - - 0 0 | - - 0 0 - - 0 0 | - - u u - - u u |
| PB | - - - - - 1 1 1 | - - - - - 1 1 1 | - - - - - 1 1 1 | - - - - - u u u |
| PBC | - - - - - 1 1 1 | - - - - - 1 1 1 | - - - - - 1 1 1 | - - - - - u u u |
| PBPU | - - - - - 0 0 0 | - - - - - 0 0 0 | - - - - - 0 0 0 | - - - - - u u u |
| PSCR | - - - - - 0 0 0 | - - - - - 0 0 0 | - - - - - 0 0 0 | - - - - - u u u |
| TB0C | 0 - - - - 0 0 0 | 0 - - - - 0 0 0 | 0 - - - - 0 0 0 | u - - - - u u u |
| TB1C | 0 - - - - 0 0 0 | 0 - - - - 0 0 0 | 0 - - - - 0 0 0 | u - - - - u u u |
| RSTC | 0 1 0 1 0 1 0 1 | 0 1 0 1 0 1 0 1 | 0 1 0 1 0 1 0 1 | u u u u u u u u |
| USR | 0 0 0 0 1 0 1 1 | 0 0 0 0 1 0 1 1 | 0 0 0 0 1 0 1 1 | u u u u u u u u |
| UCR1 | 0 0 0 0 0 0 x 0 | 0 0 0 0 0 0 x 0 | 0 0 0 0 0 0 x 0 | u u u u u u u u |
| UCR2 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | u u u u u u u u |
| UCR3 | - - - - - - - 0 | - - - - - - - 0 | - - - - - - - 0 | - - - - - - - u |
| BRDH | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | u u u u u u u u |
| BRDL | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | u u u u u u u u |
| UFCR | - - 0 0 0 0 0 0 | - - 0 0 0 0 0 0 | - - 0 0 0 0 0 0 | - - u u u u u u |
| TXR_RXR | x x x x x x x x | x x x x x x x x | x x x x x x x x | u u u u u u u u |
| RxCNT | - - - - - 0 0 0 | - - - - - 0 0 0 | - - - - - 0 0 0 | - - - - - u u u |
| SIMC0 | 1 1 1 - 0 0 0 0 | 1 1 1 - 0 0 0 0 | 1 1 1 - 0 0 0 0 | u u u - u u u u |
| SIMC1 | 1 0 0 0 0 0 0 1 | 1 0 0 0 0 0 0 1 | 1 0 0 0 0 0 0 1 | u u u u u u u u |
| SIMD | x x x x x x x x | x x x x x x x x | x x x x x x x x | u u u u u u u u |
| SIMA/SIMC2 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | u u u u u u u u |
| SIMTOC | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | u u u u u u u u |
| OCR | - - - - 0 0 - - | - - - - 0 0 - - | - - - - 0 0 - - | - - - - u u - - |
| ODL | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | u u u u u u u u |
| ODH | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | u u u u u u u u |
| PWRC | 0 - - - - 0 0 0 | 0 - - - - 0 0 0 | 0 - - - - 0 0 0 | u - - - - u u u |
| PGAC0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | u u u u u u u u |
| PGAC1 | 0 0 0 0 0 0 0 - | 0 0 0 0 0 0 0 - | 0 0 0 0 0 0 0 - | u u u u u u u - |
| PGACS | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | u u u u u u u u |
| ADRL | x x x x x x x x | x x x x x x x x | x x x x x x x x | u u u u u u u u |
| ADRM | x x x x x x x x | x x x x x x x x | x x x x x x x x | u u u u u u u u |
| ADRH | x x x x x x x x | x x x x x x x x | x x x x x x x x | u u u u u u u u |
| ADCR0 | - 0 0 0 0 0 0 | - 0 0 0 0 0 0 | - 0 0 0 0 0 0 | - u u u u u u u |
| ADCR1 | 0 - 0 0 0 1 0 0 | 0 - 0 0 0 1 0 0 | 0 - 0 0 0 1 0 0 | u - u u u u u u |
| ADCS | - - - 0 0 0 0 0 | - - - 0 0 0 0 0 | - - - 0 0 0 0 0 | - - - u u u u u |
| CHOP | 1 0 1 0 0 0 1 1 | 1 0 1 0 0 0 1 1 | 1 0 1 0 0 0 1 1 | u u u u u u u u |
| IREFC | 0 - - 0 - - 0 0 | 0 - - 0 - - 0 0 | 0 - - 0 - - 0 0 | u - - u - - u u |
| PVREF | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | u u u u u u u u |
| OPA1C | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | u u u u u u u u |
| AFEDA1C | - - - - - - 0 0 | - - - - - - 0 0 | - - - - - - 0 0 | - - - - - - u u |
| AFEDA1L | 0 0 0 0 - - - - | 0 0 0 0 - - - - | 0 0 0 0 - - - - | u u u u - - - - |
| AFEDA1H | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | u u u u u u u u |
| AFEDA2C | - - - - - - 0 0 | - - - - - - 0 0 | - - - - - - 0 0 | - - - - - - u u |
| AFEDA2L | 0 0 0 0 - - - - | 0 0 0 0 - - - - | 0 0 0 0 - - - - | u u u u - - - - |
| AFEDA2H | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | 0 0 0 0 0 0 0 0 | u u u u u u u u |

| Register | Power-On Reset | LVR Reset (Normal Operation) | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|---|---|---|---|---|
| SGC | 0 - - - - - - 0 | 0 - - - - - - 0 | 0 - - - - - - 0 | u - - - - - - u |
| SGN | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SGDN | --00 0000 | --00 0000 | --00 0000 | --uu uuuu |
| BOPAC0 | 000- 0000 | 000- 0000 | 000- 0000 | uuu- uuuu |
| BOPAC1 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ASWAI0 | 0000 000- | 0000 000- | 0000 000- | uuuu uuu- |
| ASWAI1 | 0000 000- | 0000 000- | 0000 000- | uuuu uuu- |
| ASWAI2 | 0000 000- | 0000 000- | 0000 000- | uuuu uuu- |
| ASWAI3 | 0000 000- | 0000 000- | 0000 000- | uuuu uuu- |
| ASWAI4 | 0000 000- | 0000 000- | 0000 000- | uuuu uuu- |
| ASWAI5 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ASWAI6 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ASWAI7 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| ASWAI8 | --00 0000 | --00 0000 | --00 0000 | --uu uuuu |
| ASWAI9 | --00 0000 | --00 0000 | --00 0000 | --uu uuuu |
| ASWDAC | 0--- 0000 | 0--- 0000 | 0--- 0000 | u--- uuuu |
| AFET0 | 0001 --11 | 0001 --11 | 0001 --11 | uuuu --uu |
| AFET1 | 1111 0110 | 1111 0110 | 1111 0110 | uuuu uuuu |
| LNOPC | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| LNDA3C | 0--- --00 | 0--- --00 | 0--- --00 | u--- --uu |
| LNDA3L | 0000 ---- | 0000 ---- | 0000 ---- | uuuu ---- |
| LNDA3H | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMC0 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMC1 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMDL | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMDH | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| CTMAL | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| CTMAH | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| IFS | ---- --00 | ---- --00 | ---- --00 | ---- --uu |
| PAS0 | 0000 0000 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PAS1 | --00 0000 | --00 0000 | --00 0000 | --uu uuuu |
| PBS0 | --00 0000 | --00 0000 | --00 0000 | --uu uuuu |

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

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PA | 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 | — | — | — | — | — | PB2 | PB1 | PB0 |
| PBC | — | — | — | — | — | PBC2 | PBC1 | PBC0 |
| PBPU | — | — | — | — | — | PBPU2 | PBPU1 | PBPU0 |

"—": 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 the relevant pull-high control registers 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 a digital 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" is the Port name which can be A or B. 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 register only when the pin is selected as a general purpose input and the MCU enters the IDLE or SLEEP 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 pin Wake-up function control
      0: Disable
      1: Enable

## I/O Port Control Registers

Each I/O port has its own control register which controls 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.

**• PxCn 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" is the Port name which can be A or B. However, the actual available bits for each I/O Port may be different.

## 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, and Input Function Selection register, labeled as IFS, 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. For most pin-shared functions, 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. However, a special point must be noted for some digital input pins, such as INTn, CTCK, etc., which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control bit fields. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they must also be setup as input by setting the corresponding bit in the I/O port control register. 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 | — | — | PAS15 | PAS14 | PAS13 | PAS12 | PAS11 | PAS10 |
| PBS0 | — | — | PBS05 | PBS04 | PBS03 | PBS02 | PBS01 | PBS00 |
| IFS | — | — | — | — | — | — | RXTXPS | INT1PS |

**Pin-shared Function Selection Register 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/INT0/CTCK
　　　　　　01: CTP
　　　　　　10: SDI/SDA
　　　　　　11: TX

Bit 5~4      **PAS05~PAS04**: PA2 pin-shared function selection
　　　　　　00: PA2
　　　　　　01: PA2
　　　　　　10: PA2
　　　　　　11: RX/TX

Bit 3~2      **PAS03~PAS02**: PA1 pin-shared function selection
　　　　　　00: PA1
　　　　　　01: PA1
　　　　　　10: SCK/SCL
　　　　　　11: RX/TX

Bit 1~0      **PAS01~PAS00**: PA0 pin-shared function selection
　　　　　　00: PA0
　　　　　　01: PA0
　　　　　　10: PA0
　　　　　　11:TX

• **PAS1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | PAS15 | PAS14 | PAS13 | PAS12 | PAS11 | PAS10 |
| 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    **PAS15~PAS14**: PA6 pin-shared function selection
        00: PA6
        01: PA6
        10: PA6
        11: AN3

Bit 3~2    **PAS13~PAS12**: PA5 pin-shared function selection
        00: PA5/INT1
        01: PA5/INT1
        10: PA5/INT1
        11: AN4

Bit 1~0    **PAS11~PAS10**: PA4 pin-shared function selection
        00: PA4/INT1
        01: CTPB
        10: $\overline{\text{SCS}}$
        11: AN5

• **PBS0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | PBS05 | PBS04 | PBS03 | PBS02 | PBS01 | PBS00 |
| 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    **PBS05~PBS04**: PB2 pin-shared function selection
        00: PB2
        01: SDO
        10: PB2
        11: AN0

Bit 3~2    **PBS03~PBS02**: PB1 pin-shared function selection
        00: PB1
        01: PB1
        10: PB1
        11: AN1

Bit 1~0    **PBS01~PBS00**: PB0 pin-shared function selection
        00: PB0
        01: PB0
        10: PB0
        11: AN2

• **IFS Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | RXTXPS | INT1PS |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

Bit 7~2　　Unimplemented, read as "0"

Bit 1　　　**RXTXPS**: RX/TX input source pin selection
　　　　　　　0: PA1
　　　　　　　1: PA2

Bit 0　　　**INT1PS**: INT1 input source pin selection
　　　　　　　0: PA4
　　　　　　　1: PA5

### I/O Pin Structures

The accompanying diagram illustrates the internal structures 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 logic function I/O pins. The wide range of pin-shared structures does not permit all types to be shown.



**Logic Function Input/Output Structure**
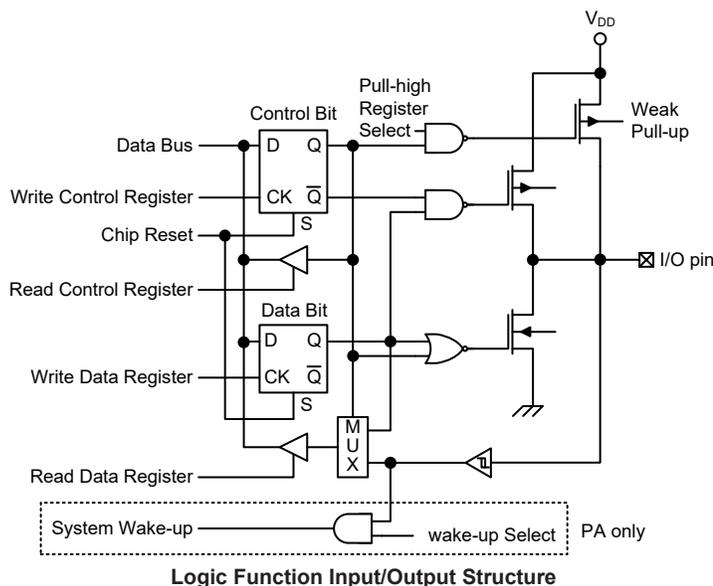
### Programming Considerations

Within the user program, one of the things first 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 functions. When the device is in the SLEEP or IDLE Mode, various methods are available to wake the device up. One of these is a high to low transition of any of the Port A pins. Single or multiple pins on Port A can be setup to have this function.

## Timer Modules – TM

One of the most fundamental functions in any microcontroller device is the ability to control and measure time. To implement time related functions the device includes a Timer Module, abbreviated to the name TM. The TM is 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.

The general features of the Compact Type TM are described here with more detailed information provided in the individual Compact Type TM section.

### Introduction

The device contains one Compact Type TM. The main features of the CTM are summarised in the accompanying table.

| TM Function | CTM |
|---|---|
| Timer/Counter | √ |
| Input Capture | — |
| Compare Match Output | √ |
| PWM Output | √ |
| Single Pulse Output | — |
| PWM Alignment | Edge |
| PWM Adjustment Period & Duty | Duty or Period |

**CTM Function Summary**

### TM Operation

The Compact Type 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 the 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, 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 Type 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 TM has two output pins with the label CTP and CTPB. The CTPB pin outputs the inverted signal of the CTP. 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 function must first be setup using relevant pin-shared function selection register described in the Pin-shared Function section.

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

**CTM External Pins**
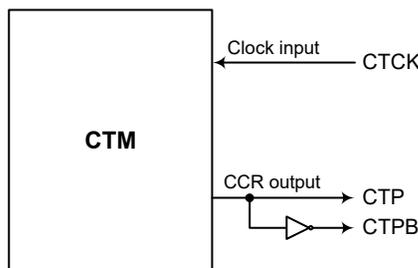


**CTM Function Pin Control Block Diagram**

## Programming Considerations

The TM Counter Registers and the Compare CCRA registers, all have a low and high byte structure. The high bytes can be directly accessed, but as the low bytes can only be accessed via an internal 8-bit buffer, reading or writing to these register pairs must be carried out in a specific way. The important point to note is that data transfer to and from the 8-bit buffer and its related low byte only takes place when a write or read operation to its corresponding high byte is executed.

As the CCRA registers are implemented in the way shown in the following diagram and accessing this register pair is carried out in a specific way described above, it is recommended to use the "MOV" instruction to access the CCRA low byte register, named CTMAL, in the following access procedures. Accessing the CCRA low byte register without following these access procedures will result in unpredictable values.
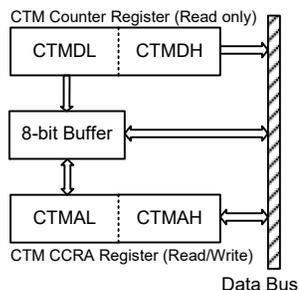
The following steps show the read and write procedures:

- 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 CCRA
    - Step 1. Read data from the High Byte CTMDH and 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 and CTMAL
        - This step reads data from the 8-bit buffer.

## Compact Type TM – CTM

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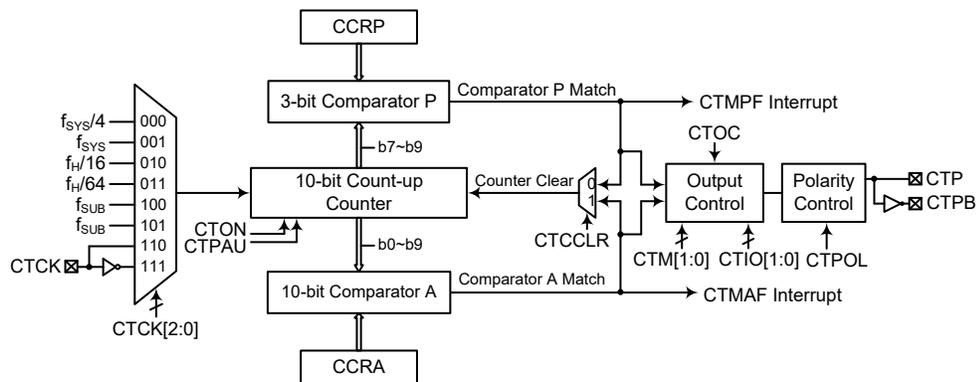


Note: 1. As the CTM external pins are pin-shared with other functions, the relevant pin-shared control bits should be properly configured before using these pins. The CTCK pin, if used, must also be set as an input by setting the corresponding bits in the port control register.

2. The CTPB is the inverted signal of the CTP.

**10-bit Compact Type TM Block Diagram**

### Compact Type TM Operation

At its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with CCRP and CCRA registers. The CCRP is 3-bit wide whose value is compared with the highest three bits in the counter while the CCRA is 10-bit wide 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 the output pins. All operating setup conditions are selected using relevant internal registers.

### Compact Type TM Register Description

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

| 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 |

**10-bit 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 "Operating Modes and System Clocks" 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 CTM external pin 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 external 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 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    **D7~D0**: 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    **D9~D8**: 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    **D7~D0**: 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    **D9~D8**: 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 occurs 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 the 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 it 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.

Counter Value

Counter overflow

CCRP=0

CCRP > 0
Counter cleared by CCRP value

0x3FF

CCRP > 0

Resume

Counter
Restart

CCRP

CCRA

Pause

Stop

Time

CTON

CTPAU

CTPOL

CCRP Int.
Flag CTMPF

CCRA Int.
Flag CTMAF

CTM O/P Pin

Output pin set to
initial Level Low if
CTOC=0

Output Toggle with
CTMAF flag

Here CTIO [1:0] = 11
Toggle Output select

Note CTIO [1:0] = 10 Active
High Output select

Output not affected by CTMAF
flag. Remains High until reset
by CTON bit

Output Pin
Reset to Initial value

Output controlled by other
pin-shared function

Output Inverts
when CTPOL is high

CTCCLR = 0; CTM [1:0] = 00

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

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

2. The CTM output pin is controlled only by the CTMAF flag

3. The output pin reset to initial state by a CTON bit rising edge
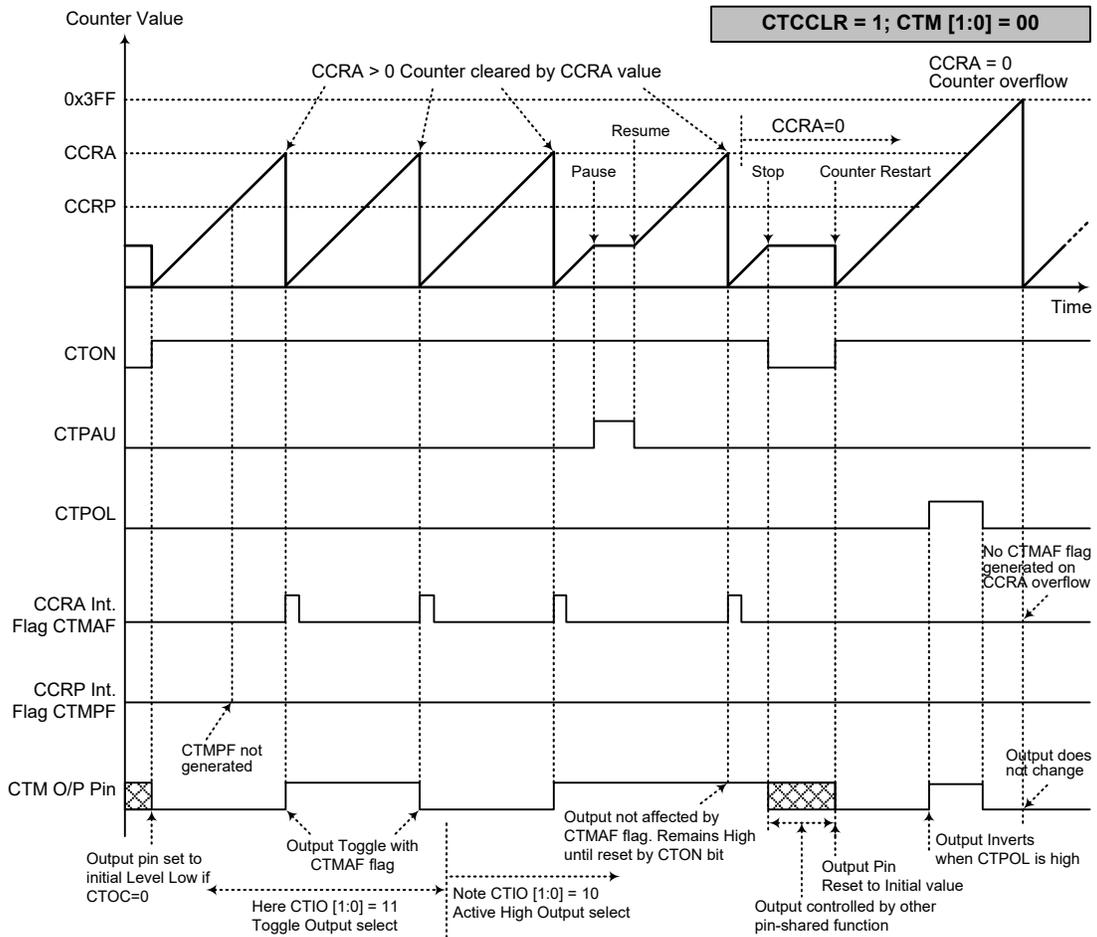
**Compare Match Output Mode – CTCCLR=1**

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

    2. The CTM output pin is controlled only by the CTMAF flag

    3. The output pin reset to initial state by a CTON bit 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.

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

| CCRP | 1~7 | 0 |
|---|---|---|
| Period | CCRP×128 | 1024 |
| Duty | CCRA | |

If $f_{SYS}$=8MHz, CTM clock source is $f_{SYS}$/4, CCRP=2 and CCRA=128,

The CTM PWM output frequency=($f_{SYS}$/4)/(2×128)=$f_{SYS}$/1024=8kHz, duty=128/(2×128)=50%.

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

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

| CCRP | 1~7 | 0 |
|---|---|---|
| Period | CCRA | |
| Duty | CCRP×128 | 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**
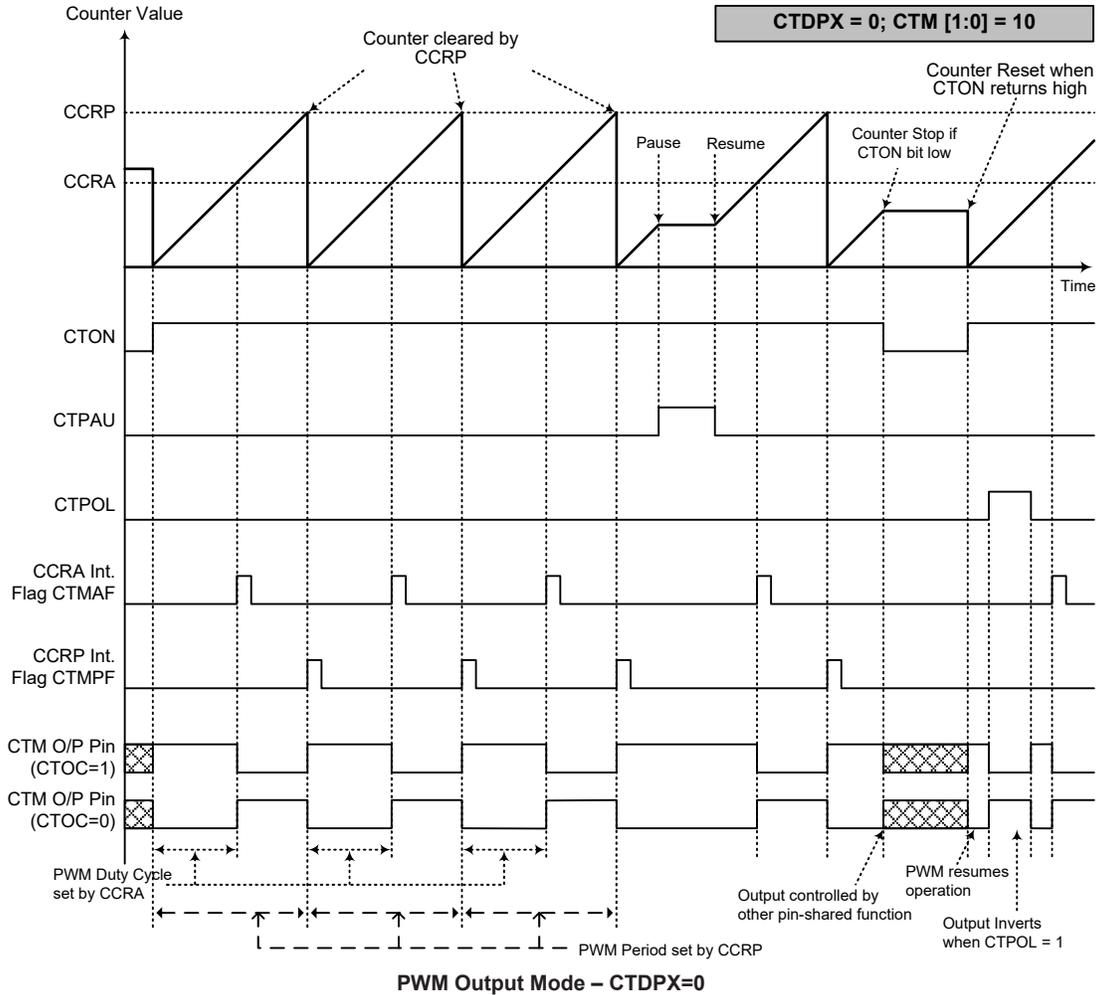
Note: 1. Here CTDPX=0 – Counter cleared by CCRP

2. A counter clear sets the 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**
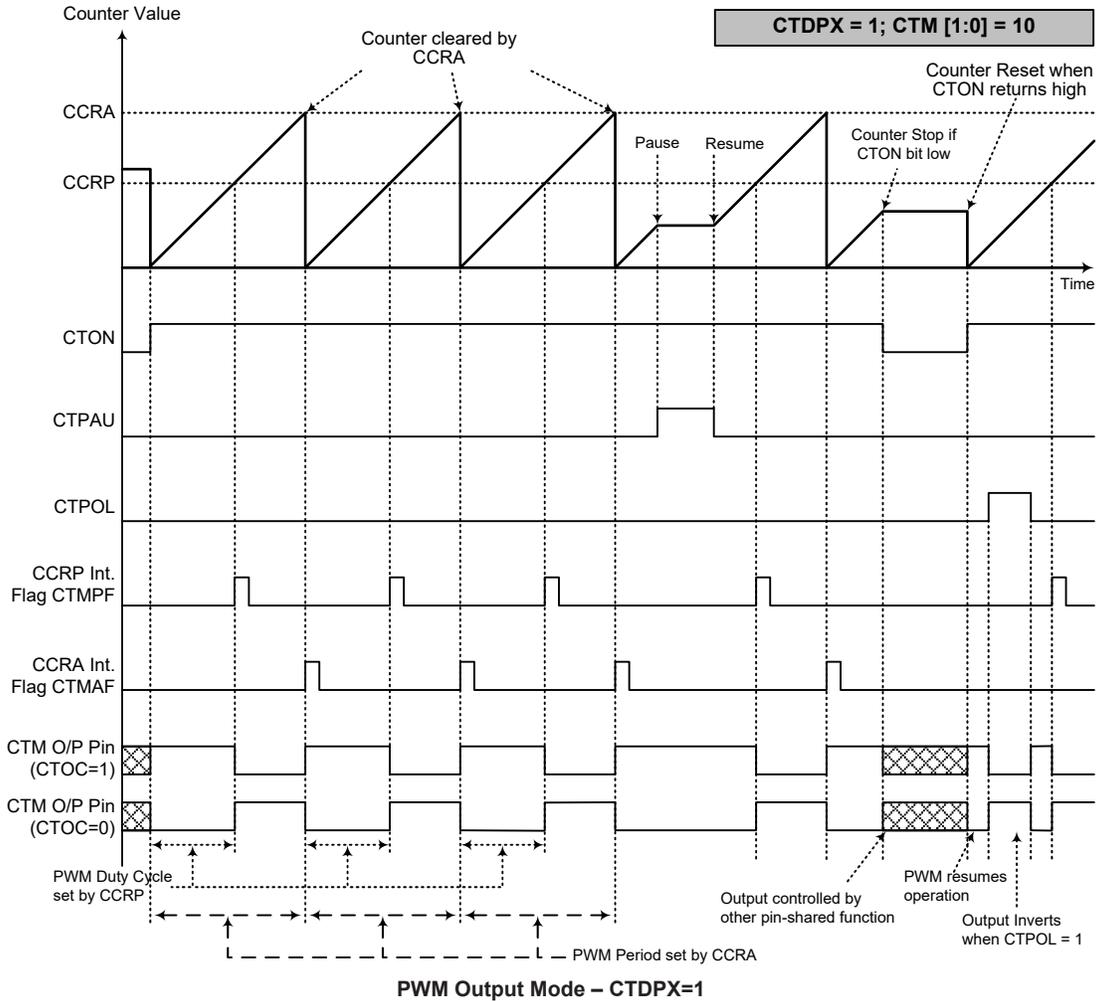
Note: 1. Here CTDPX=1 – Counter cleared by CCRA

2. A counter clear sets the 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

## Internal Power Supply

This device contains an LDO for power supply regulation. The accompanying block diagram illustrates the basic function operation. The internal LDO can provide a fixed voltage for OPA, D/A converter or external components. There are four LDO voltage levels, 2.4V, 2.6V, 2.2V or 2.3V, determined by the LDOVS1~LDOVS0 bits in the PWRC register. The LDO function can be controlled by the LDOEN bit and can be powered off to reduce overall power consumption. The $V_{OREG}$ can provide the power supply for internal reference voltage generator, D/A converter, operational amplifier and bio-impedance analysis function
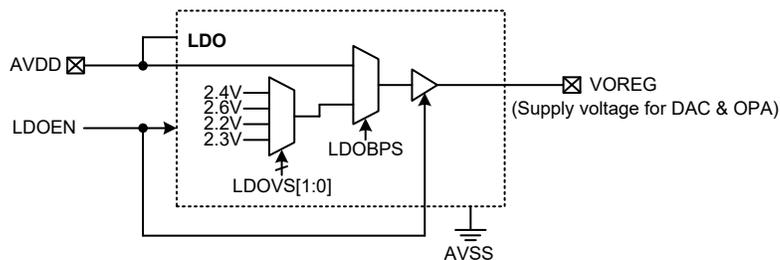


**Internal Power Supply Block Diagram**

• **PWRC Register**

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

Bit 7        **LDOEN**: LDO function control
    0: Disable
    1: Enable
    If the LDO is disabled, there will be no power consumption and the LDO output pin will remain at a low level using a weak internal pull-low resistor.

Bit 6~3     Unimplemented, read as "0"

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

Bit 1~0     **LDOVS1~LDOVS0**: LDO output voltage selection
    00: 2.4V
    01: 2.6V
    10: 2.2V
    11: 2.3V

## Internal Reference Voltage Generator

The device includes an internal reference voltage generator to provide an accurate reference voltage $V_{IREF}$. This reference voltage can be used as the internal 12-bit D/A Converter or 24-bit Delta Sigma A/D Converter reference voltage. Refer to the Internal Reference Voltage Characteristics section for more information. This function is power supplied by the $V_{OREG}$.
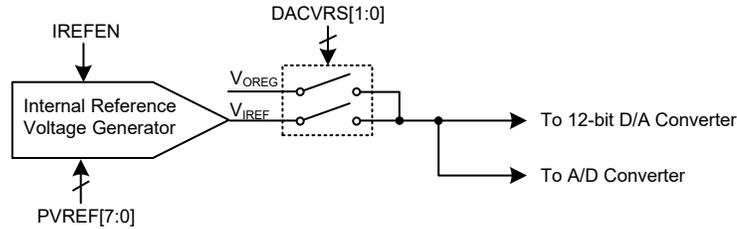


Note: The internal reference voltage generator is controlled by the IREFEN and DACVRS[1:0] bits.

| IREFEN | DACVRS1 | DACVRS0 | Internal Reference Voltage Generator |
|--------|---------|---------|--------------------------------------|
| 0 | x | x | OFF |
| 1 | 0 | 0 | ON |
| 1 | 0 | 1 | ON |
| 1 | 1 | 0 | OFF |
| 1 | 1 | 1 | OFF |

"x": Don't care

### Internal Reference Voltage Register Description

The internal reference voltage is controlled by two registers. The IREFC register is used for the enable/disable control while the PVREF register is used for fine tuning the internal reference voltage value.

| Register Name | Bit | | | | | | | |
|---------------|-----|-----|-----|--------|-----|-----|---------|---------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IREFC | OP1DO | — | — | IREFEN | — | — | DACVRS1 | DACVRS0 |
| PVREF | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

**D/A Converter Register List**

• **IREFC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|--------|-----|-----|---------|---------|
| Name | OP1DO | — | — | IREFEN | — | — | DACVRS1 | DACVRS0 |
| R/W | R | — | — | R/W | — | — | R/W | R/W |
| POR | 0 | — | — | 0 | — | — | 0 | 0 |

Bit 7        **OP1DO**: OPA1 digital output; positive logic
Refer to the Operational Amplifier section.

Bit 6~5      Unimplemented, read as "0"

Bit 4        **IREFEN**: Internal Reference Voltage Generator control
0: Disable
1: Enable
This bit controls the internal reference voltage generator on/off. When this bit is set high, the internal reference voltage $V_{IREF}$ can be generated and used as the reference voltage. When this bit is cleared to zero, the internal reference voltage generator will enter the power down mode and the output will be in a floating state. If the internal referenc voltage is not used by any circuits, the IREFEN bit should be cleared to zero to reduce power consumption.

Bit 3~2     Unimplemented, read as "0"

Bit 1~0     **DACVRS1~DACVRS0**: 12-bit D/A Converter reference voltage $V_{DACVREF}$ selection
            00/01: $V_{IREF}$
            10: $V_{OREG}$
            11: Floating

- **PVREF 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     **D7~D0**: Internal Reference Voltage Generator fine tune control
            Setting the register can fine tune the internal reference voltage with a range of -60mV~ +60mV (based on PVREF=80H). When the PVREF register value is increased by one, the internal reference voltage will decrease around 500µV, vice verse.

# D/A Converter – DAC

This device includes two 12-bit D/A Converters, which can provide a certain voltage from 0 to $V_{DACVREF}$ to the positive input of the internal operational amplifiers. The D/A converter output voltage, $V_{DACnO}$, can be connected to the A/D converter input for measurement. This function is power supplied by the $V_{OREG}$.



**D/A Conterver Structure (n=1~2, x=0~7)**
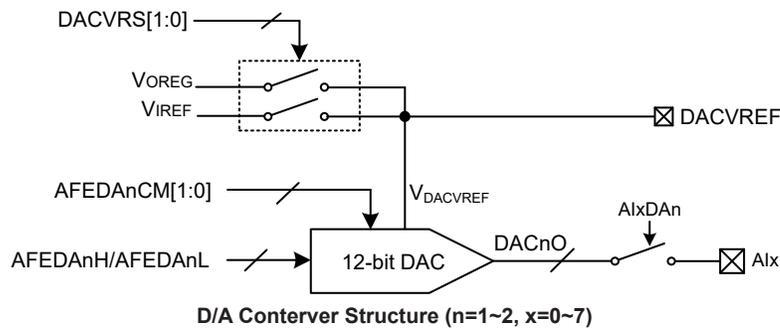
## D/A Converter Register Description

The D/A converter overall function is controlled by a series registers. The DACVRS1~DACVRS0 bits in the IREFC register are used to select the D/A converter reference voltage. The AFEDAnC register is used for D/A converter function enable/disable control. A 12-bit value of the register pair, AFEDAnH and AFEDAnL, is used for the D/A converter output control. Other registers are used for switch state control.

| Register Name | Bit | | | | | | | |
|---------------|-----|-----|-----|-----|-----|-----|-----|-----|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IREFC | OP1DO | — | — | IREFEN | — | — | DACVRS1 | DACVRS0 |
| AFEDAnC | — | — | — | — | — | — | AFEDAnCM1 | AFEDAnCM0 |
| AFEDAnL | D3 | D2 | D1 | D0 | — | — | — | — |
| AFEDAnH | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 |
| ASWAIx (x=0~4) | AIxDA1 | AIxDA2 | AIxSINO | AIxOP3P | AIxOP3N | AIxOP2N | AIxOP1N | — |
| ASWAIx (x=5~7) | AIxDA1 | AIxDA2 | AIxOP3P | AIxOP3N | AIxOP2P | AIxOP2O | AIxOP1P | AIxOP1O |

**Internal Reference Voltage Register List (n=1~2)**

- **IREFC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | OP1DO | — | — | IREFEN | — | — | DACVRS1 | DACVRS0 |
| R/W | R | — | — | R/W | — | — | R/W | R/W |
| POR | 0 | — | — | 0 | — | — | 0 | 0 |

Bit 7       **OP1DO**: OPA1 digital output; positive logic

Refer to the Operational Amplifier section.

Bit 6~5     Unimplemented, read as "0"

Bit 4       **IREFEN**: Internal Reference Voltage Generator control

Refer to the Internal Reference Voltage Generator section.

Bit 3~2     Unimplemented, read as "0"

Bit 1~0     **DACVRS1~DACVRS0**: 12-bit D/A Converter reference voltage $V_{DACVREF}$ selection

00/01: $V_{IREF}$

10: $V_{OREG}$

11: Floating

- **AFEDAnC Register (n=1~2)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | — | — | — | — | — | — | AFEDAnCM1 | AFEDAnCM0 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

Bit 7~2     Unimplemented, read as "0"

Bit 1~0     **AFEDAnCM1~AFEDAnCM0**: 12-bit D/A Converter mode control

00: DAC Disable, output in a high impedance state

01/11: DAC Enable

10: DAC Disable, output in a ground state

- **AFEDAnH & AFEDAnL Registers (n=1~2)**

| Register | AFEDAnH | | | | | | | | AFEDAnL | | | | | | | |
|----------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | D11 | D10 | D9 | D8 | 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 | R/W | R/W | R/W | R/W | — | — | — | — |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | — | — | — |

"—": Unimplemented, read as "0"

**D11~D0**: 12-bit D/A Converter output control bits

The bit 7 ~ bit 0 in the AFEDAnH register combine with the bit 7 ~ bit 4 in the AFEDAnL register to form a 12-bit DAC value of 0~4095.

DAC Output Voltage ($V_{DACnO}$)=$V_{DACVREF}$×(DAC value/4096)

Note: It's necessary to firstly write into the AFEDAnL register followed by writing into the AFEDAnH register.

- **ASWAIx Register (x=0~4)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | AIxDA1 | AIxDA2 | AIxSINO | AIxOP3P | AIxOP3N | AIxOP2N | AIxOP1N | — |
| 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       **AIxDA1**: DAC1O to AIx switch control bit

0: Off

1: On

Bit 6        **AIxDA2**: DAC2O to AIx switch control bit

      0: Off

      1: On

Bit 5~1      Refer to other sections

Bit 0        Unimplemented, read as "0"

Note: If more than one switch is set on to connect signals to a specific AIx pin at the same time, the AIx pin will be shorted. Ensure that these bits are propoerly configured to meet the requirements. For example, when the AI0DA1 and AI0DA2 bits are set high at the same time, the AI0 pin will be shorted.

- **ASWAIx Register (x=5~7)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | AIxDA1 | AIxDA2 | AIxOP3P | AIxOP3N | AIxOP2P | AIxOP2O | AIxOP1P | AIxOP1O |
| 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        **AIxDA1**: DAC1O to AIx switch control bit

      0: Off

      1: On

Bit 6        **AIxDA2**: DAC2O to AIx switch control bit

      0: Off

      1: On

Bit 5~0      Refer to other sections

Note: If more than one switch is set on to connect signals to a specific AIx pin at the same time, the AIx pin will be shorted. Ensure that these bits are propoerly configured to meet the requirements. For example, when the AI5DA1 and AI5DA2 bits are set high at the same time, the AI5 pin will be shorted.

## Operational Amplifier – OPA

This device includes two operational amplifiers for measure applications, known as OPA1 and OPA2. The 12-bit D/A Converters offer a voltage of $0\sim V_{DACVREF}$ to the positive input of the OPAs. The operational amplifiers provide two operating modes by controlling the switches S0~S2 and S4~S6 respectively. By proper setup, the OPA output voltage can be connected to the A/D converter internal input channel for measurement. This function is power supplied by the $V_{OREG}$.
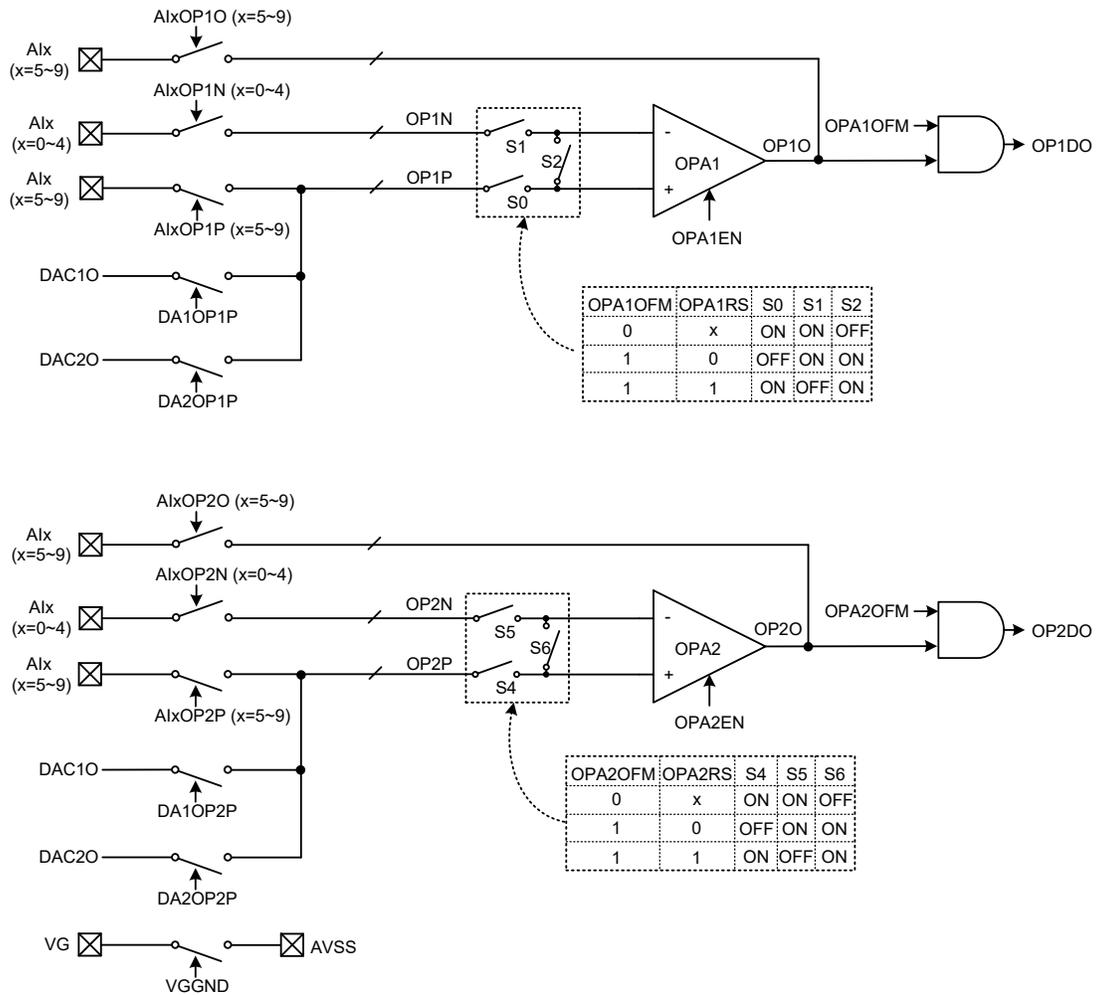


**Operational Amplifier Structure**

### OPA Register Description

The overall operation of the internal Operational Amplifiers is controlled by several registers. The OPA1C and OPA2C registers together with the OPnDO bit in the IREFC and OPA3C registers are used for the OPAs enable/disable control and input voltage offset calibration settings. Other registers are used for switches state control.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| IREFC | OP1DO | — | — | IREFEN | — | — | DACVRS1 | DACVRS0 |
| OPA1C | OPA1OFM | OPA1RS | OPA1EN | OPA1OF4 | OPA1OF3 | OPA1OF2 | OPA1OF1 | OPA1OF0 |
| OPA2C | OPA2OFM | OPA2RS | OPA2EN | OPA2OF4 | OPA2OF3 | OPA2OF2 | OPA2OF1 | OPA2OF0 |
| OPA3C | OPA3EN | IQ_FWR | OP2DO | — | OP3G3 | OP3G2 | OP3G1 | OP3G0 |
| ASWAIx (x=0~4) | AIxDA1 | AIxDA2 | AIxSINO | AIxOP3P | AIxOP3N | AIxOP2N | AIxOP1N | — |
| ASWAIx (x=5~7) | AIxDA1 | AIxDA2 | AIxOP3P | AIxOP3N | AIxOP2P | AIxOP2O | AIxOP1P | AIxOP1O |
| ASWAIx (x=8~9) | — | — | AIxOP3P | AIxOP3N | AIxOP2P | AIxOP2O | AIxOP1P | AIxOP1O |
| ASWDAC | VGGND | — | — | — | DA1OP1P | DA1OP2P | DA2OP1P | DA2OP2P |

**OPA Register List**

• **IREFC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | OP1DO | — | — | IREFEN | — | — | DACVRS1 | DACVRS0 |
| R/W | R | — | — | R/W | — | — | R/W | R/W |
| POR | 0 | — | — | 0 | — | — | 0 | 0 |

Bit 7 **OP1DO**: OPA1 digital output; positive logic

The OP1DO bit is cleared to zero when the OPA1 function is disabled.

When the OPA1OFM bit is set high, the OP1DO value is defined as OPA1 output status. Refer to the offset calibration procedure.

Bit 6~5 Unimplemented, read as "0"

Bit 4 **IREFEN**: Internal Reference Voltage Generator control

Refer to the Internal Reference Voltage Generator section.

Bit 3~2 Unimplemented, read as "0"

Bit 1~0 **DACVRS1~DACVRS0**: 12-bit D/A Converter reference voltage $V_{DACVREF}$ selection

Refer to the D/A Converter section.

• **OPA1C Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | OPA1OFM | OPA1RS | OPA1EN | OPA1OF4 | OPA1OF3 | OPA1OF2 | OPA1OF1 | OPA1OF0 |
| 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 **OPA1OFM**: OPA1 normal operation or input offset voltage calibration mode selection

　　0: Normal operation
　　1: Offset calibration mode

When the input offset calibration mode is selected, the reference voltage input should be derived from the positive input.

Bit 6 **OPA1RS**: OPA1 input offset voltage calibration reference selection

　　0: Input reference voltage comes from OP1N
　　1: Input reference voltage comes from OP1P

Note: The OPA1 input offset voltage calibration can be executed only when the OPA1RS bit is set high.

Bit 5 **OPA1EN**: OPA1 enable/disable control

　　0: Disable
　　1: Enable

Bit 4~0 **OPA1OF4~OPA1OF0**: OPA1 input offset voltage calibration control

• **OPA2C Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | OPA2OFM | OPA2RS | OPA2EN | OPA2OF4 | OPA2OF3 | OPA2OF2 | OPA2OF1 | OPA2OF0 |
| 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　　　**OPA2OFM**: OPA2 normal operation or input offset voltage calibration mode selection
　　　　　　0: Normal operation
　　　　　　1: Offset calibration mode
　　　　　When the input offset calibration mode is selected, the reference voltage input should be derived from the positive input.

Bit 6　　　**OPA2RS**: OPA2 input offset voltage calibration reference selection
　　　　　　0: Input reference voltage comes from OP2N
　　　　　　1: Input reference voltage comes from OP2P
　　　　　Note: The OPA2 input offset voltage calibration can be executed only when the OPA2RS bit is set high.

Bit 5　　　**OPA2EN**: OPA2 enable/disable control
　　　　　　0: Disable
　　　　　　1: Enable

Bit 4~0　　**OPA2OF4~OPA2OF0**: OPA2 input offset voltage calibration control

• **OPA3C Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | OPA3EN | IQ_FWR | OP2DO | — | OP3G3 | OP3G2 | OP3G1 | OP3G0 |
| R/W | R/W | R/W | R | — | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | — | 0 | 0 | 0 | 0 |

Bit 7　　　**OPA3EN**: OPA3 enable/disable control
　　　　　Refer to the Bio-impedance Analysis Function section.

Bit 6　　　**IQ_FWR**: IQ/FWR mode selection
　　　　　Refer to the Bio-impedance Analysis Function section.

Bit 5　　　**OP2DO**: OPA2 digital output; positive logic
　　　　　The OP2DO bit is cleared to zero when the OPA2 function is disabled.
　　　　　When the OPA2OFM bit is set high, the OP2DO value is defined as OPA2 output status. Refer to the offset calibration procedure.

Bit 4　　　Unimplemented, read as "0"

Bit 3~0　　**OP3G3~OP3G0**: OPA3 gain control bit
　　　　　Refer to the Bio-impedance Analysis Function section.

• **ASWAIx Register (x=0~4)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | AIxDA1 | AIxDA2 | AIxSINO | AIxOP3P | AIxOP3N | AIxOP2N | AIxOP1N | — |
| 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 7~3　　Refer to other sections

Bit 2　　　**AIxOP2N**: OP2N to AIx switch control bit
　　　　　　0: Off
　　　　　　1: On

Bit 1　　　**AIxOP1N**: OP1N to AIx switch control bit
　　　　　　0: Off
　　　　　　1: On

Bit 0        Unimplemented, read as "0"

Note: If more than one switch is set on to connect signals to a specific AIx pin at the same time, the AIx pin will be shorted. Ensure that these bits are propoerly configured to meet the requirements.

• **ASWAIx Register (x=5~7)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | AIxDA1 | AIxDA2 | AIxOP3P | AIxOP3N | AIxOP2P | AIxOP2O | AIxOP1P | AIxOP1O |
| 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 7~4      Refer to other sections

Bit 3        **AIxOP2P**: OP2P to AIx switch control bit
             0: Off
             1: On

Bit 2        **AIxOP2O**: OP2O to AIx switch control bit
             0: Off
             1: On

Bit 1        **AIxOP1P**: OP1P to AIx switch control bit
             0: Off
             1: On

Bit 0        **AIxOP1O**: OP1O to AIx switch control bit
             0: Off
             1: On

Note: If more than one switch is set on to connect signals to a specific AIx pin at the same time, the AIx pin will be shorted. Ensure that these bits are propoerly configured to meet the requirements.

• **ASWAIx Register (x=8~9)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | — | — | AIxOP3P | AIxOP3N | AIxOP2P | AIxOP2O | AIxOP1P | AIxOP1O |
| 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      Refer to other sections

Bit 3        **AIxOP2P**: OP2P to AIx switch control bit
             0: Off
             1: On

Bit 2        **AIxOP2O**: OP2O to AIx switch control bit
             0: Off
             1: On

Bit 1        **AIxOP1P**: OP1P to AIx switch control bit
             0: Off
             1: On

Bit 0        **AIxOP1O**: OP1O to AIx switch control bit
             0: Off
             1: On

Note: If more than one switch is set on to connect signals to a specific AIx pin at the same time, the AIx pin will be shorted. Ensure that these bits are propoerly configured to meet the requirements.

• **ASWDAC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|-----|--------|--------|--------|--------|
| Name | VGGND | — | — | — | DA1OP1P | DA1OP2P | DA2OP1P | DA2OP2P |
| R/W | R/W | — | — | — | R/W | R/W | R/W | R/W |
| POR | 0 | — | — | — | 0 | 0 | 0 | 0 |

Bit 7      **VGGND**: VG to ground switch control bit
      0: Off
      1: On

Bit 6~4      Unimplemented, read as "0"

Bit 3      **DA1OP1P**: DAC1O to OP1P switch control bit
      0: Off
      1: On

Bit 2      **DA1OP2P**: DAC1O to OP2P switch control bit
      0: Off
      1: On

Bit 1      **DA2OP1P**: DAC2O to OP1P switch control bit
      0: Off
      1: On

Bit 0      **DA2OP2P**: DAC2O to OP2P switch control bit
      0: Off
      1: On

Note: Only one of the DA1OP1P and DA2OP1P bits can be on each time.

Only one of the DA1OP2P and DA2OP2P bits can be on each time.

## Input Offset Calibration

The OPAnOFM bit in the OPA1C and OPA2C register is used to select the Operational Amplifier n operating mode, normal operation or input offset calibration mode. The input voltage offset calibration can be executed only when the OPAnRS bit is set to 1. For operational amplifier input offset calibration, the procedures are summarized as the following.

• Step 1: Set OPAnOFM=1 and OPAnRS=1, the Operational Amplifier n is now under the offset calibration mode, both of S0 and S2 switches or S4 and S6 switches on. To make sure $V_{OS}$ as minimize as possible after calibration, the input reference voltage in calibration should be the same as input DC operating voltage in normal mode operation.

• Step 2: Set OPAnOF[4:0]=00000, then read the OPnDO bit.

• Step 3: Let OPAnOF[4:0]=OPAnOF[4:0]+1, then read the OPnDO bit.

If the OPnDO bit state does not change, then repeat Step 3 until the OPnDO bit state changes.

If the OPnDO bit state changes, record the OPAnOF[4:0] bits value as $V_{OS1}$ and then go to Step 4.

• Step 4: Set OPAnOF[4:0]=11111, then read OPnDO bit.

• Step 5: Let OPAnOF[4:0]=OPAnOF[4:0]-1, then read the OPnDO bit.

If the OPnDO bit state does not change, then repeat Step 5 until the OPnDO bit state changes.

If the OPnDO bit state changes, record the OPAnOF[4:0] bits value as $V_{OS2}$ and then go to Step 6.

• Step 6: Restore the Operational Amplifier n input offset calibration value into the OPAnOF[4:0] bits. The offset calibration procedure is now finished.

Where $V_{OS}=(V_{OS1}+V_{OS2})/2$; if $(V_{OS1}+V_{OS2})/2$ is not integral, discard the decimal.

## Bio-impedance Analysis Function

The bio-impedance analysis (BIA) circuit consists of a sine generator, an amplifier and a filter. It is high quality, flexibility and high integration for bio-impedance measurement. This function is supplied by the $V_{OREG}$.



**Bio-impedance Analysis Structure**
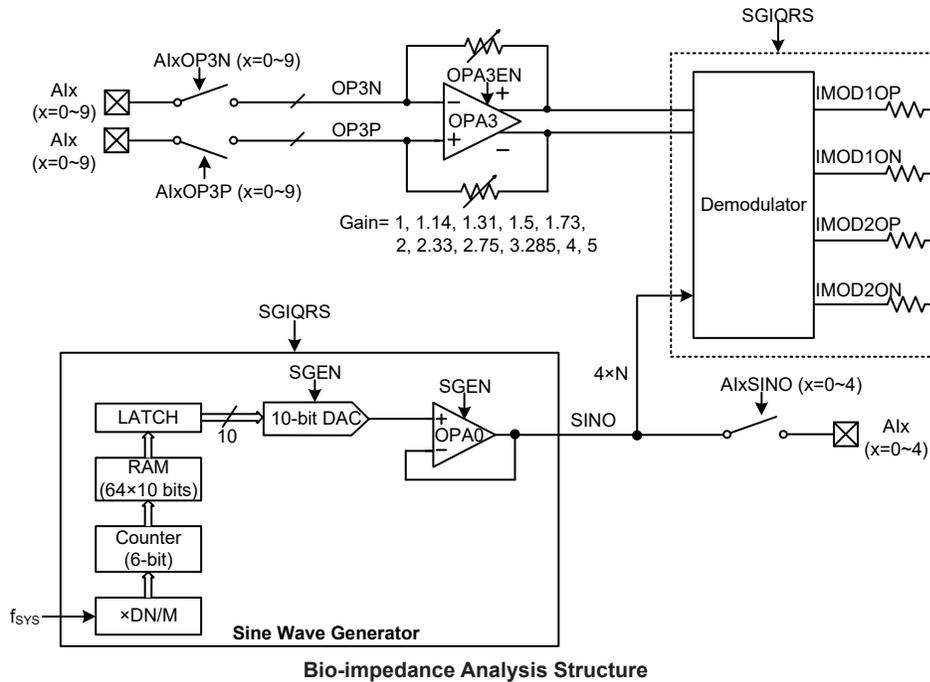
### Sine Wave Generator

The sine generator consists of a frequency divider, a 6-bit counter, RAM, a 10-bit D/A converter and the operational amplifier 0. It offers a sine wave ranging from 5kHz to 500kHz and a 64×10 bits RAM for sine wave pattern which is set by software.The frequency divider will multiply by DN/M to generate a clock to counter. For related details refer to following formula.

- System clock/M=the frequency of sine wave
- System clock×(DN/M)=the count rate of counter
- M must be the multiple of N and 8
- M=N×DN
- DN: the data number of a sine wave cycle (DN ≤ 64)

Refer to following table for details.

| System Frequency | 4MHz | | | 8MHz | | |
|---|---|---|---|---|---|---|
| Sine Wave Frequency (kHz) | 500 | 50 | 5 | 500 | 50 | 5 |
| M | 8 | 80 | 800 | 16 | 160 | 1600 |
| N | 1 | 2 | 20 | 1 | 4 | 25 |
| DN | 8 | 40 | 40 | 16 | 40 | 64 |

Note: The sine wave generator circuit consists of a 10-bit D/A converter and a smoothing filter whose frequency at -3dB is equal to 489kHz. When the output frequency is 500kHz, the output wave amplitude will decrease and therefore it is impossible to achieve a full range of $V_{OREG}$~0.
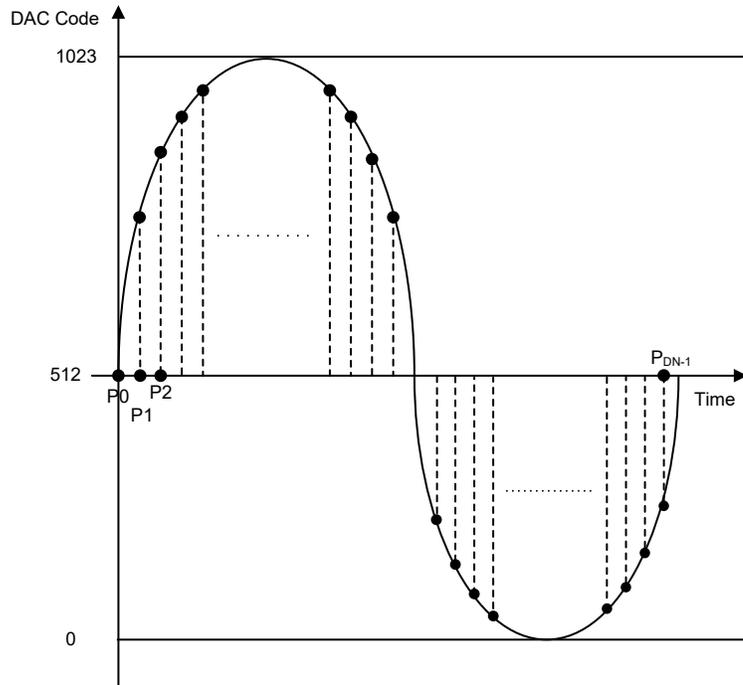
Users only need to prepare a whole sine wave pattern $P_0 \sim P_{DN-1}$ and store them in RAM Sector 3 (00H~7FH). The bits 7~0 of the sine wave patterns are stored in even addresses and the bits 9~8 of the sine wave patterns are stored in odd addresses. Once the sine wave generator is enabled, the CPU cannot write/read any data to/from this RAM and the sine wave generator will read RAM data to the 10-bit D/A converter.

The controller reads the whole sine wave patterns from RAM and generates a sine waveform to the AIx pin. Refer to following figure.
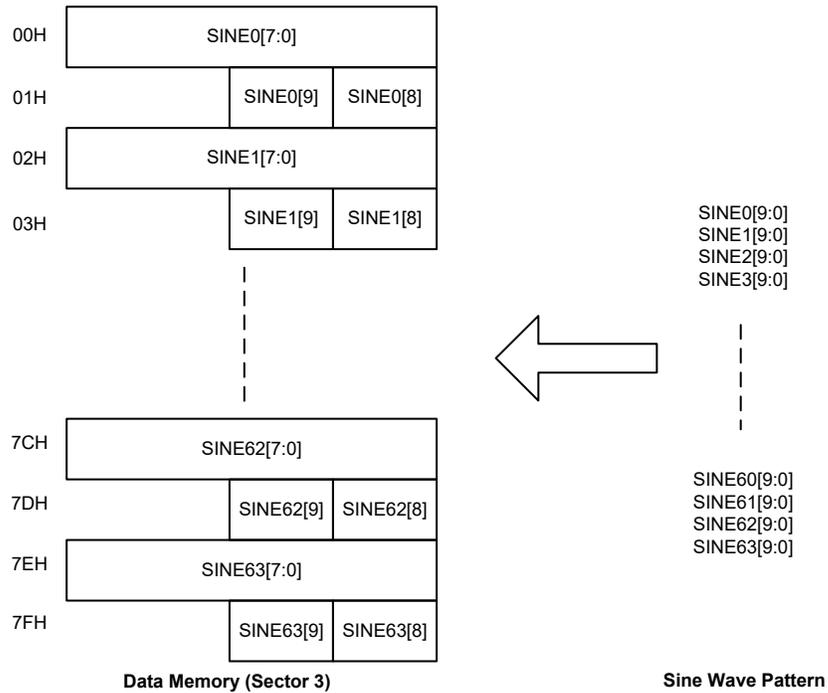
## Bio-impedance Analysis Register Description

There are a series of registers control the overall operation of the Bio-impedance Analysis Function.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SGC | SGEN | — | — | — | — | — | — | SGIQRS |
| SGN | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SGDN | — | — | D5 | D4 | D3 | D2 | D1 | D0 |
| OPA3C | OPA3EN | IQ_FWR | OP2DO | — | OP3G3 | OP3G2 | OP3G1 | OP3G0 |
| ASWAIx (x=0~4) | AIxDA1 | AIxDA2 | AIxSINO | AIxOP3P | AIxOP3N | AIxOP2N | AIxOP1N | — |
| ASWAIx (x=5~7) | AIxDA1 | AIxDA2 | AIxOP3P | AIxOP3N | AIxOP2P | AIxOP2O | AIxOP1P | AIxOP1O |
| ASWAIx (x=8~9) | — | — | AIxOP3P | AIxOP3N | AIxOP2P | AIxOP2O | AIxOP1P | AIxOP1O |

Bio-impedance Analysis Register List

### • SGC Register

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

Bit 7       **SGEN**: Sine Wave Generator control

0: Disable

1: Enable

When this bit is equal to "0", the OPA0 and 10-bit D/A converter will be in power down mode.

Bit 6~1     Unimplemented, read as "0"

Bit 0       **SGIQRS**: Sine wave generator and IQ circuit reset

0: Normal

1: Reset

The bit is normally low but if set high and then cleared low again, the sine wave generator and IQ circuit will be reset to restart from the first data.

### • SGN 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     **D7~D0**: The multiplicator of system clock frequency (N)

The multiplicator of system clock frequency (N)=D[7:0]+1

- **SGDN Register**

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

Bit 7~6 Unimplemented, read as "0"

Bit 5~0 **D5~D0**: Data number of sample

A whole sine wave cycle numerical value is stored in Data memory Sector 3. DN is equal to D[5:0]+1. When in the IQ mode, the DN value must be a multiple of 8 for normal operations.

- **OPA3C Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | OPA3EN | IQ_FWR | OP2DO | — | OP3G3 | OP3G2 | OP3G1 | OP3G0 |
| R/W | R/W | R/W | R | — | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | — | 0 | 0 | 0 | 0 |

Bit 7 **OPA3EN**: OPA3 enable/disable control

0: Disable

1: Enable

When this bit is equal to "0", OPA3 will be in power down mode.

Bit 6 **IQ_FWR**: IQ/FWR mode selection

0: IQ mode

1: FWR mode

Bit 5 **OP2DO**: OPA2 digital output; positive logic

Refer to the Operational Amplifier section.

Bit 4 Unimplemented, read as "0"

Bit 3~0 **OP3G3~OP3G0**: OPA3 gain control bit

0001: 1.14

0010: 1.31

0011: 1.5

0100: 1.73

0101: 2

0110: 2.33

0111: 2.75

1000: 3.285

1001: 4

1010: 5

Others: 1

- **ASWAIx Register (x=0~4)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | AIxDA1 | AIxDA2 | AIxSINO | AIxOP3P | AIxOP3N | AIxOP2N | AIxOP1N | — |
| 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 7~6 Refer to other sections

Bit 5 **AIxSINO**: SINO to AIx switch control bit

0: Off

1: On

Bit 4 **AIxOP3P**: OP3P to AIx switch control bit

0: Off

1: On

Bit 3          **AIxOP3N**: OP3N to AIx switch control bit
　　　　　0: Off
　　　　　1: On

Bit 2~1       Refer to other sections

Bit 0          Unimplemented, read as "0"

Note: If more than one switch is set on to connect signals to a specific AIx pin at the same time, the AIx pin will be shorted. Ensure that these bits are propoerly configured to meet the requirements.

• **ASWAIx Register (x=5~7)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | AIxDA1 | AIxDA2 | AIxOP3P | AIxOP3N | AIxOP2P | AIxOP2O | AIxOP1P | AIxOP1O |
| 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       Refer to other sections

Bit 5          **AIxOP3P**: OP3P to AIx switch control bit
　　　　　0: Off
　　　　　1: On

Bit 4          **AIxOP3N**: OP3N to AIx switch control bit
　　　　　0: Off
　　　　　1: On

Bit 3~0       Refer to other sections

Note: If more than one switch is set on to connect signals to a specific AIx pin at the same time, the AIx pin will be shorted. Ensure that these bits are propoerly configured to meet the requirements.

• **ASWAIx Register (x=8~9)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | — | — | AIxOP3P | AIxOP3N | AIxOP2P | AIxOP2O | AIxOP1P | AIxOP1O |
| 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          **AIxOP3P**: OP3P to AIx switch control bit
　　　　　0: Off
　　　　　1: On

Bit 4          **AIxOP3N**: OP3N to AIx switch control bit
　　　　　0: Off
　　　　　1: On

Bit 3~0       Refer to other sections

Note: If more than one switch is set on to connect signals to a specific AIx pin at the same time, the AIx pin will be shorted. Ensure that these bits are propoerly configured to meet the requirements.

# Analog to Digital Converter – ADC

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 Converter Overview

This device contains a high accuracy multi-channel 24-bit Delta Sigma analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 24-bit digital value.

In addition, PGA gain control, A/D converter gain control and A/D converter reference voltage gain control determine the amplification gain for A/D converter 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 A/D converter basic operational function. The A/D converter external input channel can be arranged as 26 single-ended A/D converter input channels or 13 differential input channels. The input signal can be amplified by PGA before entering the 24-bit Delta Sigma A/D converter. The A/D converter module 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. With high accuracy and performance, the device is very suitable for differential output sensor applications such as weight measurement scales and other related products.
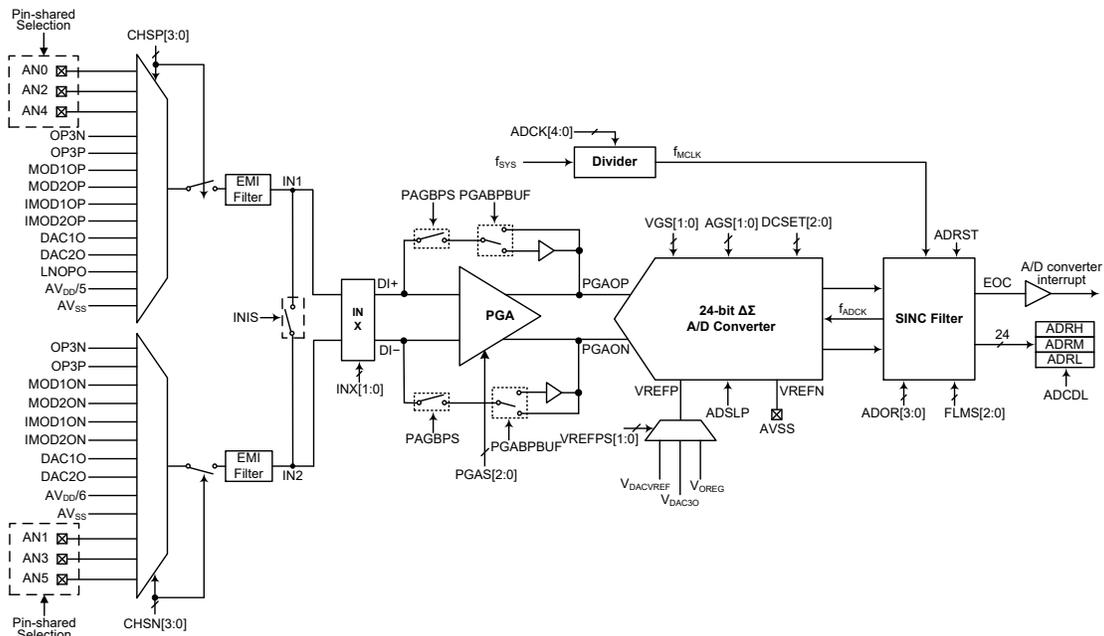


Note: The PGA and A/D converter are power supplied by the $AV_{DD}$ and $AV_{SS}$. The SINC Filter is power supplied by the $V_{DD}$ and $V_{SS}$.

**A/D Converter Structure**

## A/D Converter Register Description

Overall operation of the A/D converter is controlled using several registers. Three read only registers exist to store the A/D converter data 24-bit value. A control register named LNOPC is used to control the low noise operational amplifier. Three registers are used to control the D/A converter

operation. The remaining registers are control registers which setup the gain selections and control function of the A/D converter.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PGAC0 | PGABPS | VGS1 | VGS0 | AGS1 | AGS0 | PGAS2 | PGAS1 | PGAS0 |
| PGAC1 | PGABPBUF | INIS | INX1 | INX0 | DCSET2 | DCSET1 | DCSET0 | — |
| PGACS | CHSN3 | CHSN2 | CHSN1 | CHSN0 | CHSP3 | 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 | — | FLMS2 | FLMS1 | FLMS0 | ADOR3 | ADOR2 | ADOR1 | ADOR0 |
| ADCR1 | VREFPS1 | VREFPS0 | ADCDL | EOC | ADRST | ADOFF | ADSLP | EDGE_SEL |
| ADCS | — | — | — | ADCK4 | ADCK3 | ADCK2 | ADCK1 | ADCK0 |
| CHOP | D7 | D6 | D5 | D4 | D3 | D2 | R_FILSEL | R_CKCHOP |
| LNOPC | LNOPEN | LNOPS0 | LNOPPS0 | LNOPPS2 | LNOPPS1 | LNOPNS1 | LNOPOS2 | LNOPOS1 |
| LNDA3C | LNDA3VRS | — | — | — | — | — | LNDA3CM1 | LNDA3CM0 |
| LNDA3L | D3 | D2 | D1 | D0 | — | — | — | — |
| LNDA3H | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 |

**A/D Converter Register List**

### Programmable Gain Amplifier Registers – PGAC0, PGAC1, PGACS

There are three registers related to the programmable gain control, PGAC0, PGAC1 and PGACS. The PGAC0 resister is used to select the PGA gain, A/D Converter gain, A/D Converter reference voltage gain and PGA bypass control. The PGAC1 register is used to define the input connection, PGA enable/disable control and PGA bypass gain with/without buffer control. In addition, the PGACS register is used to select the PGA inputs. Therefore, the input channels have to be determined by the CHSP3~CHSP0 and CHSN3~CHSN0 bits to determine which analog channel signals, OPA inputs, DAC outputs 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 | PGABPS | VGS1 | VGS0 | AGS1 | AGS0 | PGAS2 | PGAS1 | PGAS0 |
| 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　　　　**PGABPS**: PGA DI+/DI- differential channel input bypass selection
　　　　　　　　0: Not bypass
　　　　　　　　1: Bypass

Bit 6~5　　　**VGS1~VGS0**: VREFP/VREFN 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/PAGON differential input signal gain selection
　　　　　　　　00: ADGN=1
　　　　　　　　01: ADGN=2
　　　　　　　　10: ADGN=4
　　　　　　　　11: Reserved

Bit 2~0      **PGAS2~PGAS0**: PGA 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 | PGABPBUF | INIS | INX1 | INX0 | DCSET2 | DCSET1 | DCSET0 | — |
| 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      **PGABPBUF**: DI+/DI- differential channel input bypass gain with buffer
         0: Without buffer
         1: With buffer

         This bit is used to set the signal bypass gain with buffer or without buffer when the PGABPS bit is set high to enable the bypass circuit.

Bit 6      **INIS**: Selected inputs, IN1/IN2, internal connection control
         0: Not connected
         1: Connected

Bit 5~4      **INX1~INX0**: Selected inputs, IN1/IN2, and the PGA differential input ends, DI+/DI- connection control bit
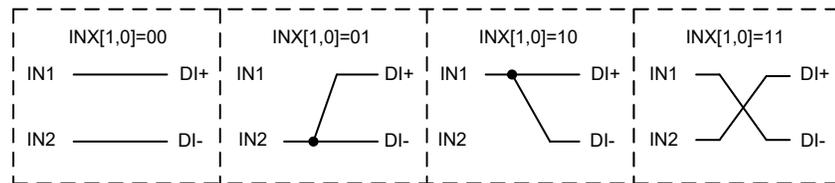


Bit 3~1      **DCSET2~DCSET0**: Differential input signal PGAOP/PGAON offset selection
         000: DCSET=+0V
         001: DCSET=+0.25×ΔVR_I
         010: DCSET=+0.5×ΔVR_I
         011: DCSET=+0.75×ΔVR_I
         100: DCSET=+0V
         101: DCSET=-0.25×ΔVR_I
         110: DCSET=-0.5×ΔVR_I
         111: DCSET=-0.75×ΔVR_I

         The voltage, ΔVR_I, is the differential reference voltage after amplification, 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 | CHSN3 | CHSN2 | CHSN1 | CHSN0 | CHSP3 | CHSP2 | CHSP1 | CHSP0 |
| 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 7~4     **CHSN3~CHSN0**: Negative input end IN2 selection
            0000: OP3N
            0001: OP3P
            0010: MOD1ON
            0011: MOD2ON
            0100: IMOD1ON
            0101: IMOD2ON
            0110: DAC1O
            0111: DAC2O
            1000: $AV_{DD}/6$
            1001: $AV_{SS}$
            1010~1011: Reserved
            1100: AN1
            1101: AN3
            1110: AN5
            1111: Reserved
            These bits are used to select the negative input, IN2.

Bit 3~0     **CHSP3~CHSP0**: Positive input end IN1 selection
            0000: OP3N
            0001: OP3P
            0010: MOD1OP
            0011: MOD2OP
            0100: IMOD1OP
            0101: IMOD2OP
            0110: DAC1O
            0111: DAC2O
            1000: $AV_{DD}/5$
            1001: $AV_{SS}$
            1010: LNOPO
            1011: Reserved
            1100: AN0
            1101: AN2
            1110: AN4
            1111: Reserved
            These bits are used to select the positive input, IN1.

**Low Noise Operational Amplifier Control Register – LNOPC**

There is a fully integrated low noise operational amplifier in the device. This LNOPA can be used for signal amplification according to specific user requirements. The LNOPA can be disabled or enabled entirely under software control using LNOPC register.
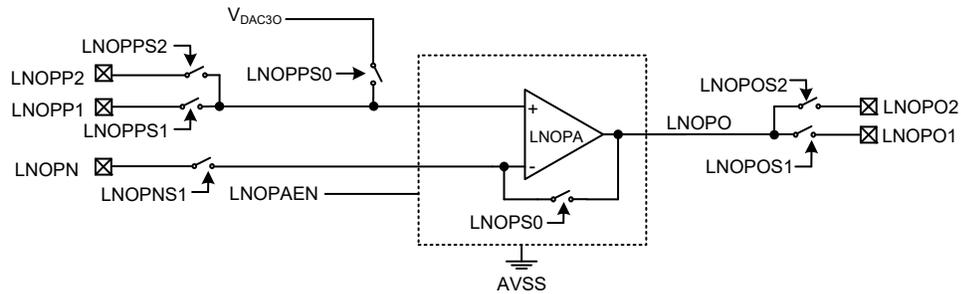


**Low Noise Operational Amplifier Structure**

• **LNOPC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | LNOPEN | LNOPS0 | LNOPPS0 | LNOPPS2 | LNOPPS1 | LNOPNS1 | LNOPOS2 | LNOPOS1 |
| 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 **LNOPEN**: LNOPA enable control
  0: Disable
  1: Enable

Bit 6 **LNOPS0**: Switch control bit for LNOPA output and negative input configuration
  0: Off
  1: On

Bit 5 **LNOPPS0**: Switch control bit for DAC3 output to LNOPA positive input configuration
  0: Off
  1: On

Bit 4 **LNOPPS2**: Switch control bit for LNOPP2 configuration
  0: Off
  1: On

Bit 3 **LNOPPS1**: Switch control bit for LNOPP1 configuration
  0: Off
  1: On

Bit 2 **LNOPNS1**: Switch control bit for LNOPN configuration
  0: Off
  1: On

Bit 1 **LNOPOS2**: Switch control bit for LNOPO2 configuration
  0: Off
  1: On

Bit 0 **LNOPOS1**: Switch control bit for LNOPO1 configuration
  0: Off
  1: On

**D/A Converter Registers – LNDA3C, LNDA3L, LNDA3H**

There is a fully integrated a 12-bit D/A converter, DAC3, in the device. The DAC3 offer a voltage of $0\sim V_{DACVREF}$ to the positive input of the LNOPA and to the A/D converter VREFP reference input. There are three registers related to the D/A converter operation. One control register is used for D/A converter mode control and reference voltage selection, two data registers are used for D/A converter output control.



**D/A Converter Structure**
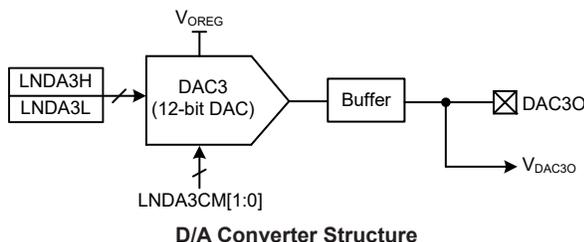
**• LNDA3C Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | LNDA3VRS | — | — | — | — | — | LNDA3CM1 | LNDA3CM0 |
| R/W | R/W | — | — | — | — | — | R/W | R/W |
| POR | 0 | — | — | — | — | — | 0 | 0 |

Bit 7        **LNDA3VRS**: Reserved bit

Bit 6~2      Unimplemented, read as "0"

Bit 1~0      **LNDA3CM1~ LNDA3CM0**: 12-bit DAC3 mode control
              00: DAC3 disable, output in a high-impendance state
              01: DAC3 enable
              10: DAC3 disable, output in a ground state
              11: DAC3 enable

**• LNDA3H & LNDA3L Registers**

| Register | LNDA3H | | | | | | | | LNDA3L | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Name | D11 | D10 | D9 | D8 | 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 | R/W | R/W | R/W | R/W | — | — | — | — |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | — | — | — | — |

"—": Unimplemented, read as "0"

**D11~D0**: DAC3 output control code

The bit 7 ~ bit 0 in the LNDA3H register combine with the bit 7 ~ bit 4 in the LNDA3L register to form a 12-bit D/A value from 0~4095.

The D/A output is $V_{DACVREF} \times$ (D/A value/4096)

Note: It's necessary to firstly write into the LNDA3L register followed by writing into the LNDA3H register.

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

The 24-bit Delta Sigma A/D converter 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, D23~D0.

• **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    **D7~D0**: 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    **D15~D8**: 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    **D23~D16**: 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, several control registers known as ADCR0, ADCR1 and ADCS are provided. These 8-bit registers define functions such as the A/D converter clock source, A/D conversion oversampling data rate as well as controlling the power-up function and monitoring the A/D converter end of conversion status.

• **ADCR0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | — | FLMS2 | FLMS1 | FLMS0 | ADOR3 | ADOR2 | ADOR1 | ADOR0 |
| 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 6~4      **FLMS2~FLMS0**: A/D converter clock $f_{ADCK}$ divider and sampled data doubling
function 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 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 a low latency conversion mode if the CHOP is equal to 1, which means that the sampled data doubling function is disabled.

Bit 3~0 **ADOR3~ADOR0**: A/D conversion oversampling rate selection
0000: Oversampling rate OSR=32768
0001: Oversampling rate OSR=16384
0010: Oversampling rate OSR=8192
0011: Oversampling rate OSR=4096
0100: Oversampling rate OSR=2048
0101: Oversampling rate OSR=1024
0110: Oversampling rate OSR=512
0111: Oversampling rate OSR=256
1000: Oversampling rate OSR=128
Others: Reserved

• **ADCR1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | VREFPS1 | VREFPS0 | ADCDL | EOC | ADRST | ADOFF | ADSLP | EDGE_SEL |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

Bit 7~6 **VREFPS1~VREFPS0**: A/D converter VREFP reference voltage selection
00: $V_{DACVREF}$
11: $V_{OREG}$
10: $V_{DAC3O}$
11: $V_{DACVREF}$

Bit 5 **ADCDL**: A/D converted data latch function 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 will not be updated by any subsequent conversion 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 an interrupt and the EOC flag bit will not change. It is recommended that this bit should be set high before reading the converted data from the ADRL, ADRM and ADRH registers. After the converted data has been read out, the bit can then be cleared to zero 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 4 **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 3 **ADRST**: A/D converter software reset control
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 2 **ADOFF**: A/D converter module power on/off control

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 1 **ADSLP**: A/D converter sleep mode enable control

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.

Bit 0 **EDGE_SEL**: A/D converter latch clock selection

0: Normal
1: Reversed

• **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 $f_{MCLK}$ setup

00000~11110: $f_{MCLK}=f_{SYS}/2/(ADCK[4:0]+1)$
11111: $f_{MCLK}=f_{SYS}$

**SINC Filter Register – CHOP**

There is a register related to the SINC Filter control, CHOP. This register is used for the output digital filter selection and the chopper function control.

• **CHOP Register**

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

Bit 7~2 **D7~D2**: Reserved bits, cannot be changed

Bit 1 **R_FILSEL**: Output digital filter selection

0: SINC2 filter (R_CKCHOP should be logic low), Data latency=3 output data clocks
1: SINC3 filter (R_CKCHOP should be logic high), Data latency=4 output data clocks

Bit 0 **R_CKCHOP**: System chopper function selection

0: Enable (R_FILSEL should be logic low)
1: Disable (R_FILSEL should be logic high)

### A/D Converter Rate Definition

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

Data Rate for SINC2

$$=f_{ADCK}/(CHOP \times OSR)$$

$$=(f_{MCLK}/N)/(CHOP \times OSR)$$

$$=f_{MCLK}/(N \times CHOP \times OSR)$$

Data Rate for SINC3

$$=f_{ADCK}/OSR$$

$$=(f_{MCLK}/N)/OSR$$

$$=f_{MCLK}/(N \times OSR)$$

$f_{ADCK}$: A/D converter clock frequency, derived from $f_{MCLK}/N$.

$f_{MCLK}$: A/D converter clock source, derived from $f_{SYS}$ or $f_{SYS}/2/(ADCK[4:0]+1)$ using the ADCK[4:0] bits.

N: A constant divide factor equal to 12 or 30 determined by the FLMS[2:0] bits.

CHOP: Sampling data amount doubling function control equal to 1 or 2 determined by the FLMS bits.

OSR: Oversampling rate determined by the ADOR[3:0] bits.

For example, if a data rate of 8Hz is desired, an $f_{MCLK}$ clock source with a frequency of 4MHz can be selected. Then set the FLMS[2:0] bits to "000" to obtain an "N" equal to 30 and "CHOP" equal to 2.

For the SINC2 filter, set the ADOR[3:0] bits to "0010" to select an oversampling rate equal to 8192. Therefore, the Data Rate=4MHz/(30×2×8192)=8Hz.

For the SINC3 filter, set the ADOR[3:0] bits to "0001" to select an oversampling rate equal to 16384. Therefore, the Data Rate=4MHz/(30×16384)=8Hz.

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

### A/D Converter Operation

The A/D Converter provides four operating modes, which are the Normal mode, Power down mode, Sleep mode and Reset mode, controlled by the IREFEN bit in the IREFC register and the ADOFF, ADSLP and ADRST bits in the ADCR1 register. The following table illustrates the operating mode selection.

| IREFEN | ADOFF | ADSLP | ADRST | Operating Mode | Description |
|---|---|---|---|---|---|
| 0 | 1 | x | x | Power down mode | Internal reference voltage off, PGA off, ADC off, SINC filter off |
| 1 | 1 | x | x | Power down mode | Internal reference voltage on, PGA off, ADC off, SINC filter off |
| 0 | 0 | 1 | x | Sleep mode (External voltage must be supplied on DACVREF pin) | Internal reference voltage off, PGA on, ADC off, SINC filter on |
| 0 | 0 | 0 | 0 | Normal mode (External voltage must be supplied on DACVREF pin) | Internal reference voltage off, PGA on, ADC on, SINC filter on |
| 0 | 0 | 0 | 1 | Reset mode (External voltage must be supplied on DACVREF pin) | Internal reference voltage off, PGA on, ADC on, SINC filter reset |
| 1 | 0 | 1 | x | Sleep mode | Internal reference voltage on, PGA on, ADC off, SINC filter on |

| IREFEN | ADOFF | ADSLP | ADRST | Operating Mode | Description |
|--------|-------|-------|-------|----------------|-------------|
| 1 | 0 | 0 | 0 | Normal mode | Internal reference voltage on, PGA on, ADC on, SINC filter on |
| 1 | 0 | 0 | 1 | Reset mode | Internal reference voltage on, PGA on, ADC on, SINC filter reset |

"x" means unknown

**A/D Operating Mode Summary**

To enable the A/D Converter, the first step is to disable the A/D converter power down and sleep mode by clearing the ADOFF and ADSLP bits to make sure the A/D converter is powered on. The ADRST bit in the ADCR1 register is used to start and reset the A/D converter after power on. When the microcontroller changes the ADRST bit from low to high and then low again, an analog to digital conversion in the SINC filter will be initiated. After this setup is completed, 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 high by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D converter interrupt request flag will be set in the interrupt control register, and if the A/D converter and global interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D converter internal interrupt signal will direct the program flow to the associated A/D converter internal interrupt address for processing. If the A/D converter internal interrupt is disabled, the microcontroller can poll the EOC bit in the ADCR1 register to check whether it has been set to "1" as an alternative method of detecting the end of an A/D conversion cycle. The A/D converted data will be updated continuously by the new converted data. If the A/D converted data latch function is enabled, the latest converted data will be latched and the following newly 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 A/D Converter.

The differential reference voltage supply to the A/D Converter can be supplied from VREFP and VREFN. The desired VREFP selection is made using the VREFPS[1:0] bits in the ADCR1 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

  Select the PGA, A/D converter gains and reference voltage gain by the PGAC0 register.

- Step 2

  Select the input connection and bypass control by the PGAC1 register.

- Step 3

  Select the required A/D conversion clock source by correctly programming bits ADCK4~ADCK0 in the ADCS register.

- Step 4

  Select output data rate by configuring the ADOR3~ADOR0 and FLMS2~FLMS0 bits in the ADCR0 register.

- Step 5

  Select which channel is to be connected to the internal PGA by correctly programming the CHSP3~CHSP0 and CHSN3~CHSN0 bits which are contained in the PGACS register.

- Step 6

Release the power down mode and sleep mode by clearing the ADOFF and ADSLP bits in ADCR1 register.

- Step 7

Reset the A/D by setting the ADRST in the ADCR1 register to high, and then release the reset status by clearing this bit to zero.

- Step 8

If the A/D converter interrupt is to be used, the related 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 in advance.

- Step 9

To check when the analog to digital conversion process is completed, the EOC bit in the ADCR1 register can be polled. The conversion process is completed when this bit goes high. When this occurs the A/D converter data registers ADRL, ADRM and ADRH can be read to obtain the conversion value. As an alternative method, if the A/D converter interrupt is enabled and the stack is not full, the program can wait for an A/D converter 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 converter internal circuitry can be switched off to reduce power consumption, by setting the ADOFF bit high. 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 Converter Transfer Function

The 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 using 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 $\Delta VR\_I$, which is the amplified value of the differential reference input voltage $\Delta VR\pm$ selected by the VREFPS[1:0] bits in ADCR1 register, this gives a single bit analog input value of $\Delta VR\_I$ divided by 8388608.

$1\ LSB = \Delta VR\_I / 8388608$

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

$\Delta SI\_I = (PGAGN \times ADGN \times \Delta DI\pm) + DCSET$

$\Delta VR\_I = VREFGN \times \Delta VR\pm$

$ADC\_Conversion\_Data = (\Delta SI\_I / \Delta VR\_I) \times K$

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

Note: 1. The PGAGN, ADGN and VREFGN values are decided by the PGAS[2:0], AGS[1:0] and VGS[1:0] control bits.

2. $\Delta SI\_I$: Differential Input Signal after amplification and offset adjustment.

3. PGAGN: Programmable Gain Amplifier gain

4. ADGN: A/D Converter gain

5. VREFGN: Reference voltage gain

6. $\Delta DI\pm$: 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 A/D converted value is 8388607 and the minimum value is -8388608. Therefore, there is a middle value of 0. The ADC_Conversion_Data equation illustrates this range of converted data variation.

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

**A/D Conversion Data Range**

The above A/D conversion data table illustrates the range of A/D conversion data.

The following diagram shows the relationship between the DC input value and the A/D converted data which is presented using Two's Complement.
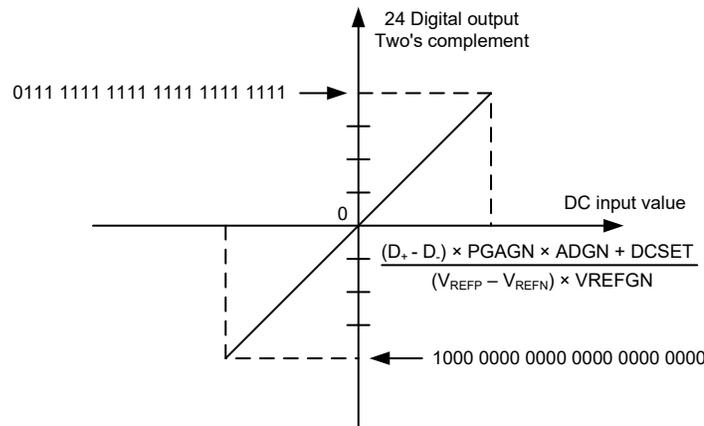


## A/D Converted Data

The A/D converted data is related to the input voltage and the PGA selections. The format of the A/D Converter 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", this represents a "positive" input. If the MSB is "1", this represents a "negative" input. The maximum value is 8388607 and the minimum value is -8388608. If the input signal is greater than the maximum value, the converted data is limited to 8388607, and if the input signal is less than the minimum value, the converted data is limited to -8388608.

## A/D Converted Data to Voltage

The converted data can be recovered using the following equations:

If MSB = 0 – Positive Converted data

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

If the MSB = 1 – Negative Converted data

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

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

## Serial Interface Module – SIM

The device contains a Serial Interface Module, which includes both the four-line SPI interface or two-line I²C 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 I²C based hardware such as sensors, Flash or EEPROM memory, etc. The SIM interface pins are pin-shared with other I/O pins and 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 I²C 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 pins 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, the device provided 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, SCK 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 I²C 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. After the desired SPI configuration has been set it can be disabled or enabled using the SIMEN bit in the SIMC0 register. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single $\overline{SCS}$ 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.
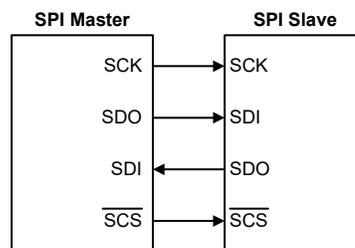


**SPI Master/Slave Connection**

The SPI function in this device offers the following features:

- Full duplex synchronous data transfer
- Both Master and Slave modes
- LSB first or MSB first data transmission modes

- Transmission complete flag
- Rising or falling active clock edge

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 Block Diagram**
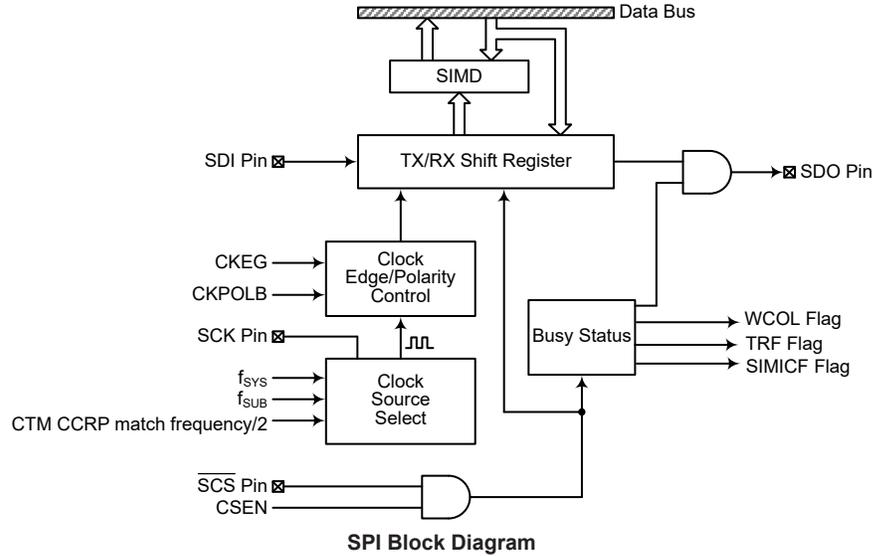
### SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two 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 Register 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 SIMC1 register is not used by the SPI function, only by the I²C function. Register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency. Register SIMC2 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: Non SIM function

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

The SIMDEB1~SIMDEB0 bits are only used in the I²C mode and the detailed definition is described in the I²C 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 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 slave mode Incomplete Transfer Flag
    0: SIM SPI slave mode incomplete condition not occurred
    1: SIM SPI slave mode incomplete condition 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 whether a data collision has occurred or not. 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. This 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 transfer is completed

The TRF bit is the Transmit/Receive Complete flag and is set to 1 automatically when an SPI data transfer is completed, but must cleared 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 complete, the TRF flag will be set automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SIMD register will be transmitted and any data on the SDI pin will be shifted into the SIMD register. The master should output a $\overline{\text{SCS}}$ 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 SCK signal depending upon the configurations of the CKPOLB bit and CKEG bit. The accompanying timing diagram shows the relationship between the slave data and SCK signal for various configurations of the CKPOLB and CKEG bits.

The SPI Master mode will continue to function if the SPI clock is running.
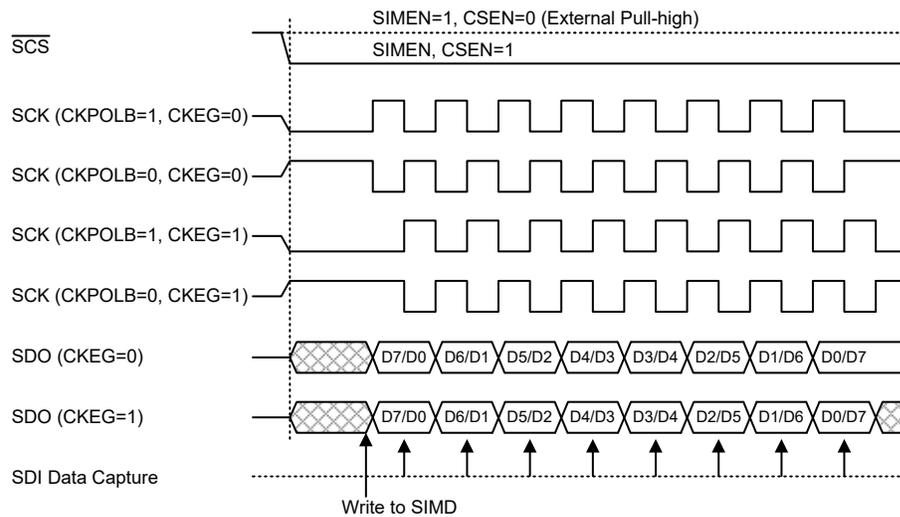
**SPI Master Mode Timing**

**SPI Slave Mode Timing – CKEG=0**

Write to SIMD
(SDO changes as soon as writing occurs; SDO is floating if $\overline{SCS}$=1)
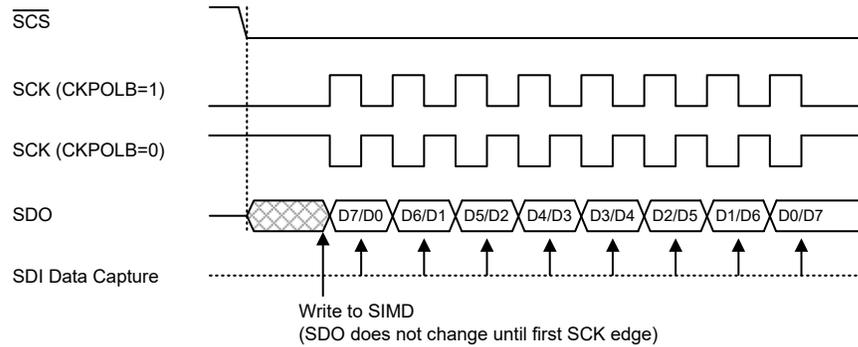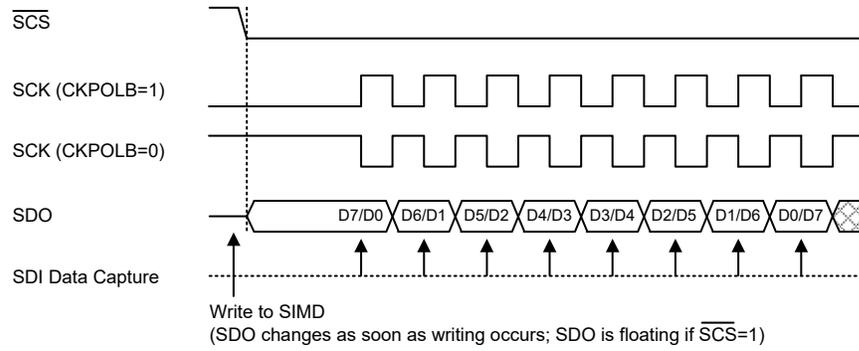
Note: For SPI slave mode, if SIMEN=1 and CSEN=0, SPI is always
enabled and ignores the $\overline{SCS}$ level.

**SPI Slave Mode Timing – CKEG=1**
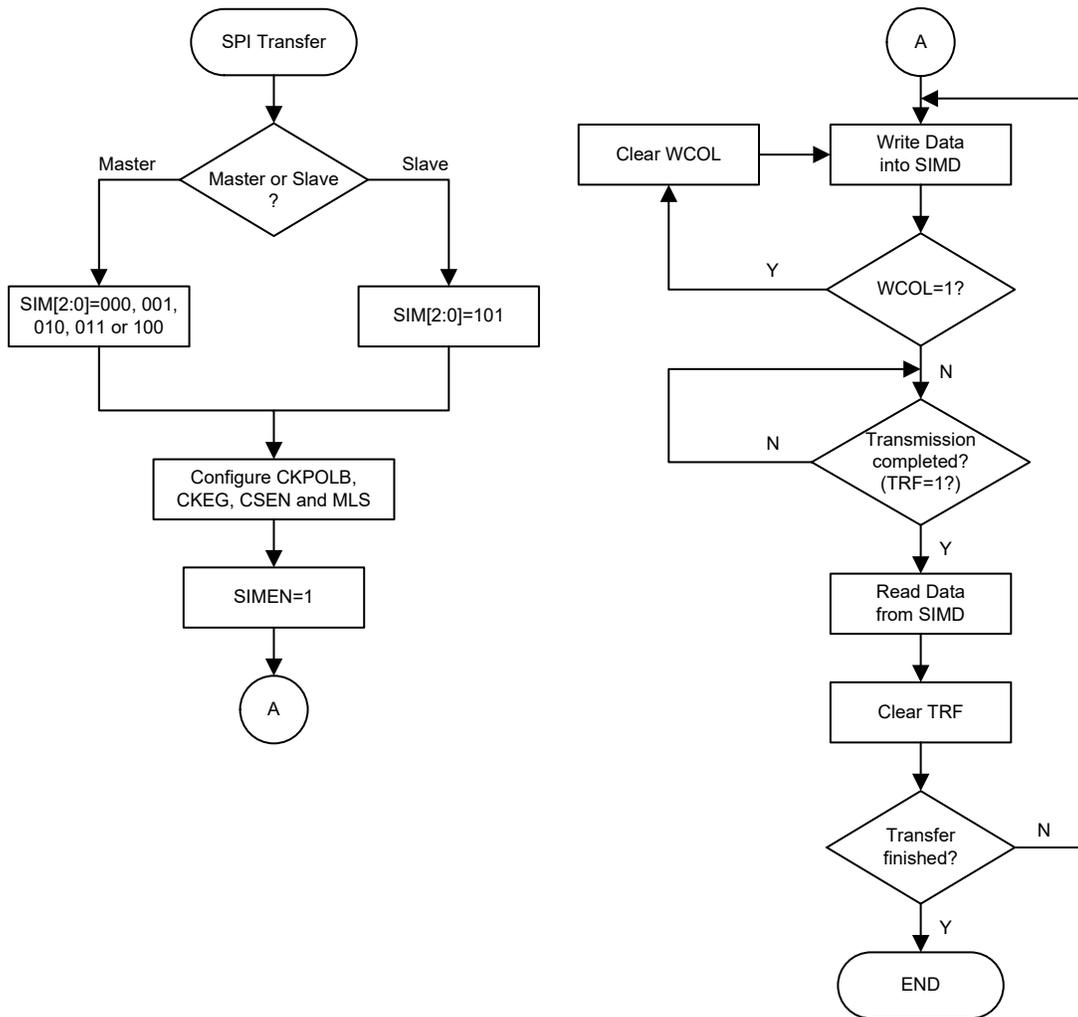


**SPI Transfer Control Flow Chart**

**SPI Bus Enable/Disable**

To enable the SPI bus, set CSEN=1 and $\overline{\text{SCS}}$=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, the SCK, SDI, SDO and $\overline{\text{SCS}}$ 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 $\overline{\text{SCS}}$ 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 $\overline{\text{SCS}}$ 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 the $\overline{\text{SCS}}$, 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 SDO 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 an SIM 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 an SIM 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.
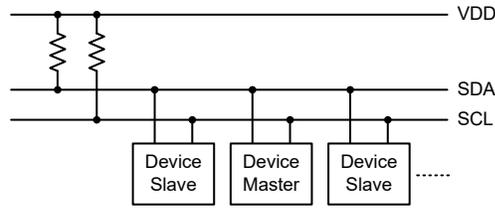
**I²C Master/Slave Bus Connection**

## 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 function could be controlled by its corresponding pull-high control register.
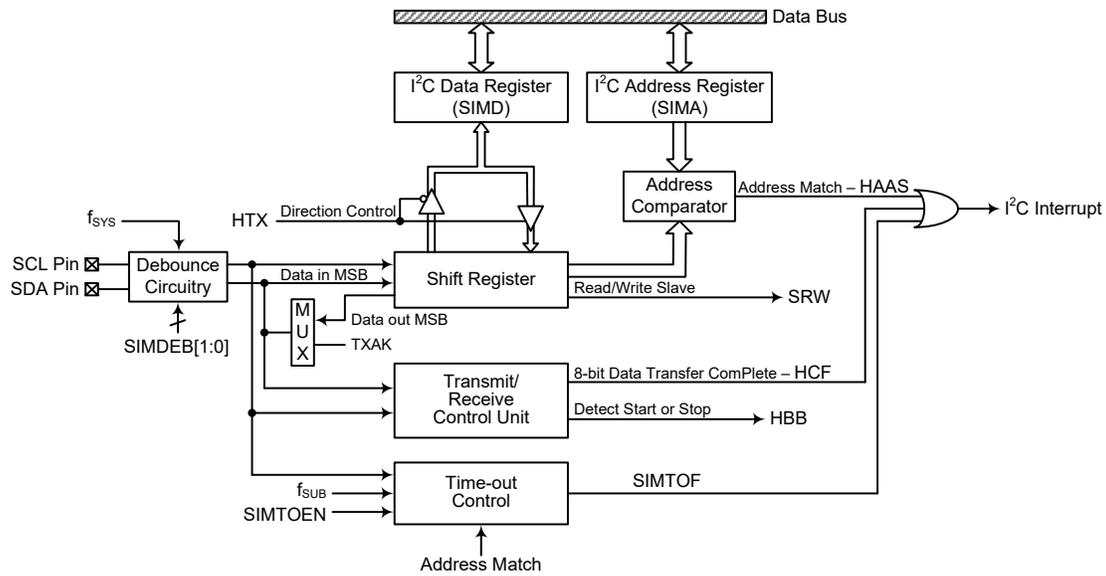


**I²C Interface Block Diagram**
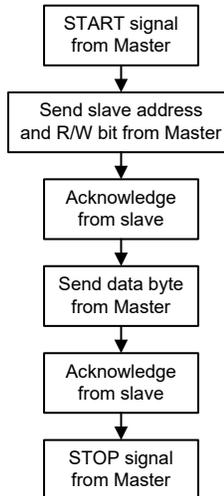
**I²C Interface Operation**

The SIMDEB1 and SIMDEB0 bits determine the debounce time of the I²C interface. This uses the system 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 | $f_{SYS} > 2MHz$ | $f_{SYS} > 4MHz$ |
| 2 system clock debounce | $f_{SYS} > 4MHz$ | $f_{SYS} > 8MHz$ |
| 4 system clock debounce | $f_{SYS} > 4MHz$ | $f_{SYS} > 8MHz$ |

**I²C Minimum $f_{SYS}$ Frequency Requirement**

### 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 |
| SIMA | SIMA6 | SIMA5 | SIMA4 | SIMA3 | SIMA2 | SIMA1 | SIMA0 | D0 |
| SIMD | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SIMTOC | SIMTOEN | SIMTOF | SIMTOS5 | SIMTOS4 | SIMTOS3 | SIMTOS2 | SIMTOS1 | SIMTOS0 |

**I²C Register 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 implemented.

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 | SIMA6 | SIMA5 | SIMA4 | SIMA3 | SIMA2 | SIMA1 | SIMA0 | 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    **SIMA6~SIMA0**: I²C slave address
           SIMA6~SIMA0 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 register SIMC0 is used to control the enable/disable function and to select the I2C slave mode and debounce time. 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: Non SIM function
           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. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. 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
The SIMICF bit is only used in the SPI mode and the detailed definition is described in the SPI 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 completion flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated. Below is an example of the flow of a two-byte I²C data transfer. First, I²C slave device receives a start signal from I²C master and then HCF bit is automatically cleared to zero. Second, I²C slave device finishes receiving the 1st data byte and then HCF bit is automatically set high. Third, user read the 1st data byte from SIMD register by the application program and then HCF bit is automatically cleared to zero. Fourth, I²C slave device finishes receiving the 2nd data byte and then HCF bit is automatically set to one and so on. Finally, I²C slave device receives a stop signal from I²C master and then HCF bit is automatically set high.

Bit 6    **HAAS**: I²C Bus data transfer completion 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 cleared to "0" when the bus is free which will occur when a STOP signal is detected.

Bit 4     **HTX**: I²C slave device transmitter/receiver selection
     0: Slave device is the receiver
     1: Slave device is the transmitter

Bit 3     **TXAK**: I²C bus transmit acknowledge flag
     0: Slave sends acknowledge flag
     1: Slave does not send acknowledge flag

The TXAK flag is the transmit acknowledge flag. After the slave device has received 8 bits of data, this flag will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set the 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 – must be cleared by the application program after wake-up

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 receives 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 is 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 SIM 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 either an address match or the completion of an 8-bit data transfer or 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 bits to "110" and SIMEN bit to "1" in the SIMC0 register 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.

```
                    ┌─────────────┐
                    │    Start    │
                    └──────┬──────┘
                           │
                 ┌─────────▼─────────┐
                 │ SET SIM[2:0]=110  │
                 │ SET SIMEN         │
                 └─────────┬─────────┘
                           │
                 ┌─────────▼─────────┐
                 │   Write Slave     │
                 │  Address to SIMA  │
                 └─────────┬─────────┘
                           │
           No      ┌───────▼───────┐      Yes
      ┌────────────┤   I²C Bus     ├────────────┐
      │            │  Interrupt?   │            │
      │            └───────────────┘            │
┌─────▼──────────────────┐         ┌────────────▼─────────┐
│      CLR SIME          │         │      SET SIME        │
│ Poll SIMF to decide    │         │   Wait for Interrupt │
│when to go to I²C Bus ISR│         │                      │
└─────┬──────────────────┘         └────────────┬─────────┘
      │                                         │
┌─────▼──────────┐                    ┌─────────▼─────────┐
│Go to Main Program│                   │Go to Main Program │
└────────────────┘                    └───────────────────┘
```
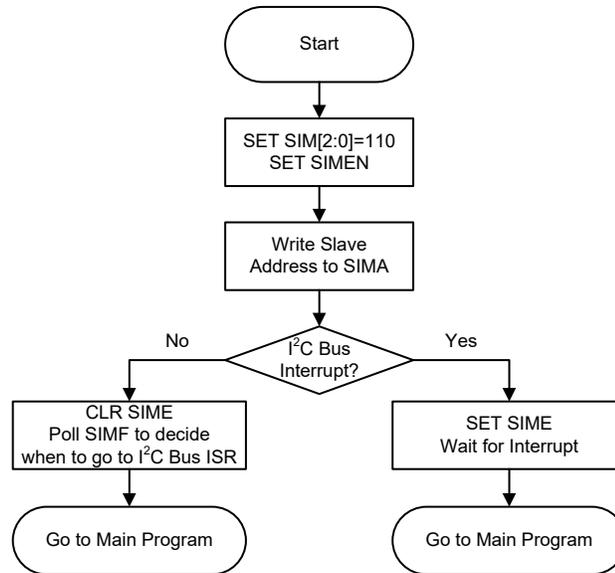
**I²C Bus Initialisation Flow Chart**

### 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 SIM 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 SIM 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 either a matching slave address, the completion of a data byte transfer or 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 cleared 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.
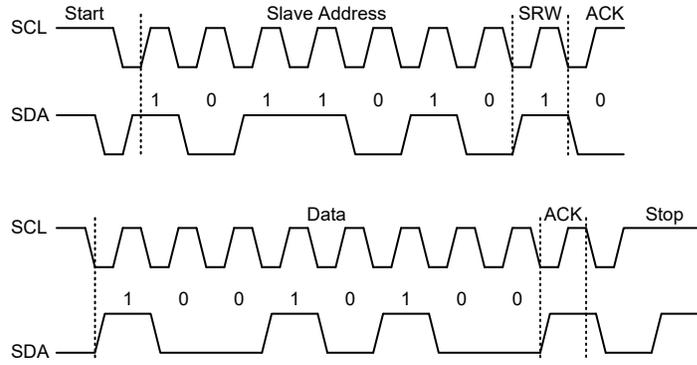
S=Start (1 bit)
SA=Slave Address (7 bits)
SR=SRW bit (1 bit)
M=Slave device send acknowledge bit (1 bit)
D=Data (8 bits)
A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)
P=Stop (1 bit)

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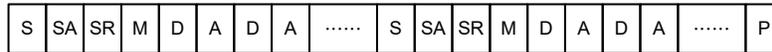
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

**I²C Communication Timing Diagram**
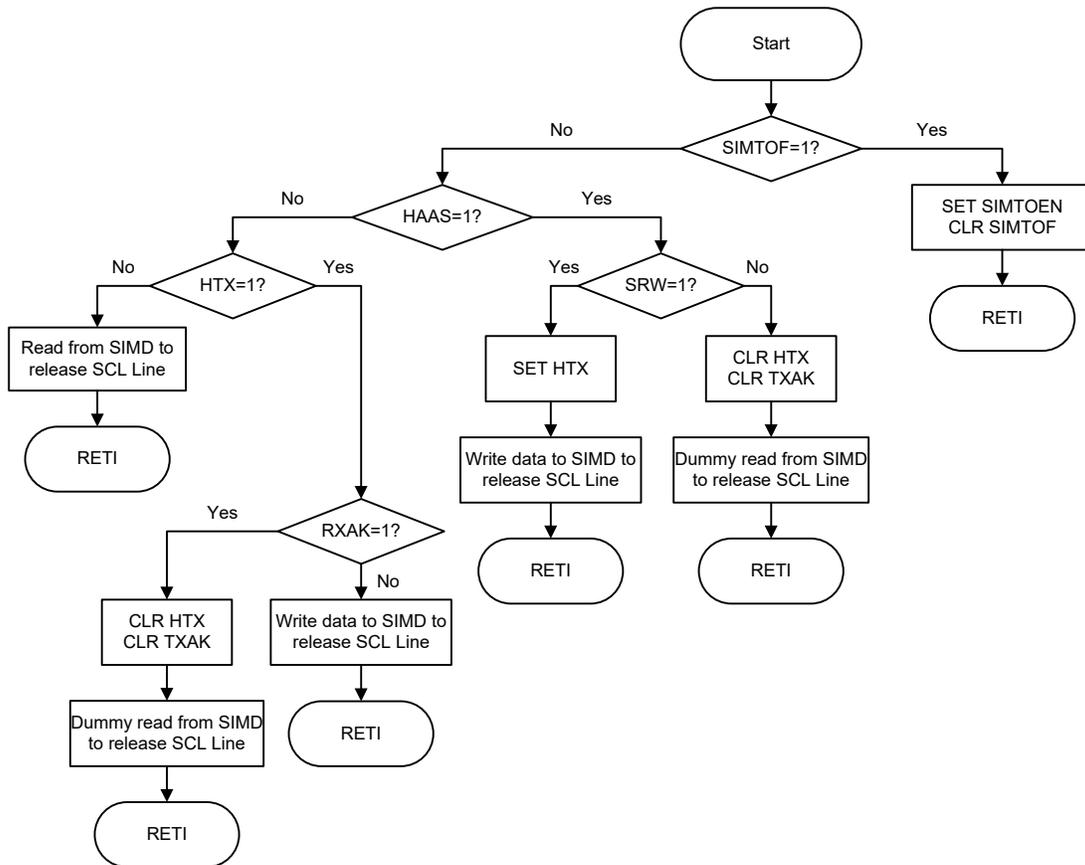


**I²C Bus ISR Flow Chart**

### I²C Time-out Control

In order to reduce the I²C lockup problem due to reception of erroneous clock sources, a time-out function is provided. If the clock source connected to the I²C bus 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 to count 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 period specified by the SIMTOC register, then a time-out condition will occur. The time-out function will stop when an I²C "STOP" condition occurs.
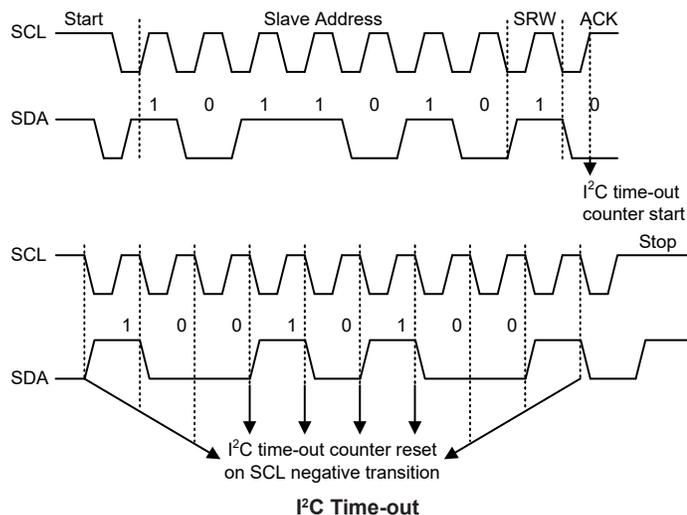


**I²C Time-out**

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 SIM 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 period selections which can be selected using the SIMTOS5~SIMTOS0 bits in the SIMTOC register. The time-out duration is calculated by the formula: $((1\sim64)\times(32/f_{SUB}))$. 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**: SIM I²C Time-out control
        0: Disable
        1: Enable

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

Bit 5~0    **SIMTOS5~SIMTOS0**: SIM I²C Time-out period selection

I²C Time-out clock source is $f_{SUB}/32$.

I²C Time-out period is equal to $(SIMTOS[5:0]+1)\times(32/f_{SUB})$.

## UART Interface

The device contains an integrated full-duplex or half-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data with eight or nine data bits per transmission as well as being able to detect errors when the data is overwritten or incorrectly framed. The UART function possesses its own internal interrupt which can be used to indicate when a reception occurs or when a transmission terminates.

The integrated UART function contains the following features:

- Full-duplex half-duplex (single wire mode), asynchronous communication

- 8 or 9 bits character length

- Even, odd, mark, space or no parity options

- One or two stop bits configurable for receiver

- Two stop bits for transmitter

- Baud rate generator with 16-bit prescaler

- Parity, framing, noise and overrun error detection

- Support for interrupt on address detect (last character bit=1)

- Separately enabled transmitter and receiver

- 4-byte Deep FIFO Receive Data Buffer

- 1-byte Deep FIFO Transmit Data Buffer

- RX/TX pin wake-up function

- Transmit and receive interrupts

- Interrupts can be triggered by the following conditions:

  - Transmitter Empty

  - Transmitter Idle

  - Receiver reaching FIFO trigger level

  - Receiver Overrun

  - Address Mode Detect
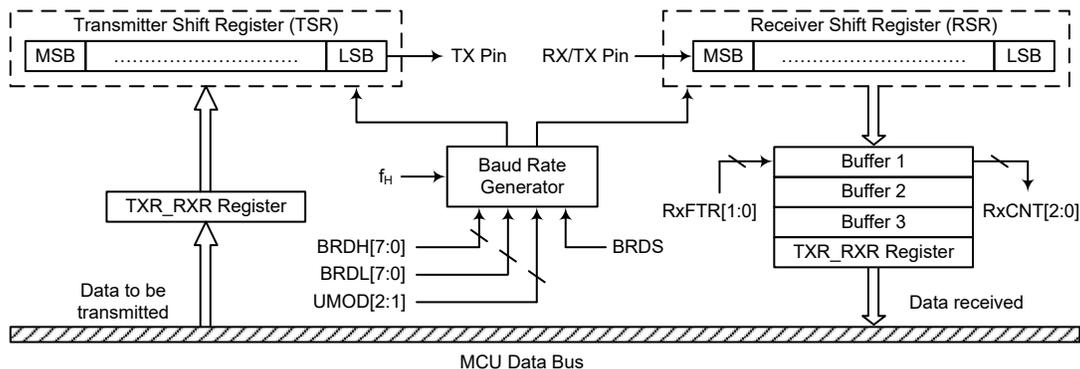


**UART Data Transfer Block Diagram – SWM=0**
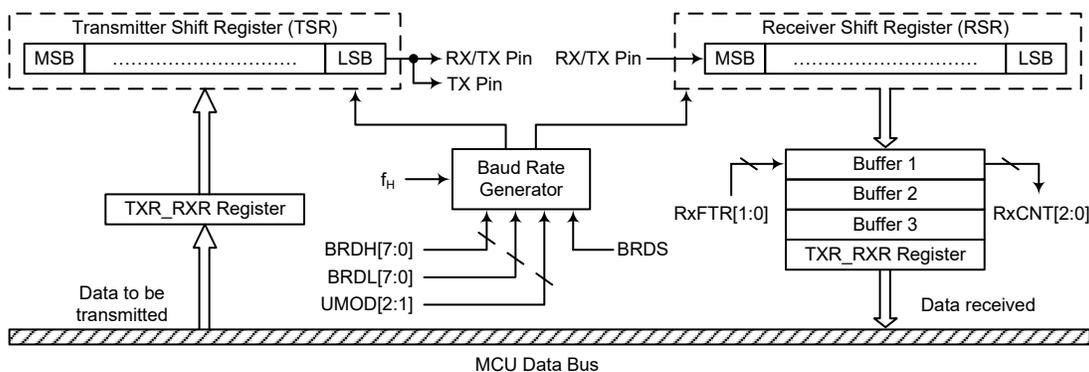
**UART Data Transfer Block Diagram – SWM=1**

### UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as TX and RX/TX, which are pin-shared with I/O or other pin functions. The TX and RX/TX pin function should first be selected by the corresponding pin-shared function selection register before the UART function is used. Along with the UARTEN bit, the TXEN and RXEN bits, if set, will configure these pins to transmitter output and receiver input conditions. At this time the internal pull-high resistor related to the transmitter output pin will be disabled, while the internal pull-high resistor related to the receiver input pin is controlled by the corresponding I/O pull-high function control bit. When the TX or RX/TX pin function is disabled by clearing the UARTEN, TXEN or RXEN bit, the TX or RX/TX pin will be set to a floating state. At this time whether the internal pull-high resistor is connected to the TX or RX/TX pin or not is determined by the corresponding I/O pull-high function control bit.

### UART Single Wire Mode

The UART function also supports the Single Wire Mode communication which is selected using the SWM bit in the UCR3 register. When the SWM bit is set high, the UART function will be in the single wire mode. In the single wire mode, a single RX/TX pin can be used to transmit and receive data depending upon the corresponding control bits. When the RXEN bit is set high, the RX/TX pin is used as a receiver pin. When the RXEN bit is cleared to zero and the TXEN bit is set high, the RX/TX pin will act as a transmitter pin.

It is recommended not to set both the RXEN and TXEN bits high in the single wire mode. If both the RXEN and TXEN bits are set high, the RXEN bit will have the priority and the UART will act as a receiver.

It is important to note that the functional description in this UART chapter, which is described from the full-duplex communication standpoint, also applies to the half-duplex (single wire mode) communication except the pin usage. In the single wire mode, the TX pin mentioned in this chapter should be replaced by the RX/TX pin to understand the whole UART single wire mode function.

In the single wire mode, the data can also be transmitted on the TX pin in a transmission operation with proper software configurations. Therefore, the data will be output on the RX/TX and TX pins.

### UART Data Transfer Scheme

The UART Data Transfer Block Diagrams show the overall data transfer structure arrangement for the UART. The actual data to be transmitted from the MCU is first transferred to the TXR_RXR register by the application program. The data will then be transferred to the Transmit Shift Register from where it will be shifted out, LSB first, onto the TX pin at a rate controlled by the Baud Rate

Generator. Only the TXR_RXR register is mapped onto the MCU Data Memory, the Transmit Shift Register is not mapped and is therefore inaccessible to the application program.

Data to be received by the UART is accepted on the external RX/TX pin, from where it is shifted in, LSB first, to the Receiver Shift Register at a rate controlled by the Baud Rate Generator. When the shift register is full, the data will then be transferred from the shift register to the internal TXR_RXR register, where it is buffered and can be manipulated by the application program. Only the TXR_RXR register is mapped onto the MCU Data Memory, the Receiver Shift Register is not mapped and is therefore inaccessible to the application program.

It should be noted that the actual register for data transmission and reception only exists as a single shared register, TXR_RXR, in the Data Memory.

### UART Status and Control Registers

There are nine control registers associated with the UART function. The SWM bit in the UCR3 register is used to enable/disable the UART Single Wire Mode. The USR, UCR1, UCR2, UFCR and RxCNT registers control the overall function of the UART, while the BRDH and BRDL registers control the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR_RXR data register.

| Register Name | Bit | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| USR | PERR | NF | FERR | OERR | RIDLE | RXIF | TIDLE | TXIF |
| UCR1 | UARTEN | BNO | PREN | PRT1 | PRT0 | TXBRK | RX8 | TX8 |
| UCR2 | TXEN | RXEN | STOPS | ADDEN | WAKE | RIE | TIIE | TEIE |
| UCR3 | — | — | — | — | — | — | — | SWM |
| TXR_RXR | TXRX7 | TXRX6 | TXRX5 | TXRX4 | TXRX3 | TXRX2 | TXRX1 | TXRX0 |
| BRDH | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| BRDL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| UFCR | — | — | UMOD2 | UMOD1 | UMOD0 | BRDS | RxFTR1 | RxFTR0 |
| RxCNT | — | — | — | — | — | D2 | D1 | D0 |

**UART Register List**

• **USR Register**

The USR register is the status register for the UART, which can be read by the program to determine the present status of the UART. All flags within the USR register are read only. Further explanation on each of the flags is given below:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Name | PERR | NF | FERR | OERR | RIDLE | RXIF | TIDLE | TXIF |
| R/W | R | R | R | R | R | R | R | R |
| POR | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

Bit 7      **PERR**: Parity error flag
        0: No parity error is detected
        1: Parity error is detected
        The PERR flag is the parity error flag. When this read only flag is "0", it indicates a parity error has not been detected. When the flag is "1", it indicates that the parity of the received word is incorrect. This error flag is applicable only if the parity is enabled and the parity type (odd, even, mark or space) is selected. The flag can also be cleared to zero by a software sequence which involves a read to the status register USR followed by an access to the TXR_RXR data register.

—

Bit 6    **NF**: Noise flag

    0: No noise is detected

    1: Noise is detected

The NF flag is the noise flag. When this read only flag is "0", it indicates no noise condition. When the flag is "1", it indicates that the UART has detected noise on the receiver input. The NF flag is set during the same cycle as the RXIF flag but will not be set in the case of as overrun. The NF flag can be cleared to zero by a software sequence which will involve a read to the status register USR followed by an access to the TXR_RXR data register.

Bit 5    **FERR**: Framing error flag

    0: No framing error is detected

    1: Framing error is detected

The FERR flag is the framing error flag. When this read only flag is "0", it indicates that there is no framing error. When the flag is "1", it indicates that a framing error has been detected for the current character. The flag can also be cleared to zero by a software sequence which will involve a read to the status register USR followed by an access to the TXR_RXR data register.

Bit 4    **OERR**: Overrun error flag

    0: No overrun error is detected

    1: Overrun error is detected

The OERR flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is "0", it indicates that there is no overrun error. When the flag is "1", it indicates that an overrun error occurs which will inhibit further transfers to the TXR_RXR receive data register. The flag is cleared to zero by a software sequence, which is a read to the status register USR followed by an access to the TXR_RXR data register.

Bit 3    **RIDLE**: Receiver status

    0: Data reception is in progress (Data being received)

    1: No data reception is in progress (Receiver is idle)

The RIDLE flag is the receiver status flag. When this read only flag is "0", it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is "1", it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is "1" indicating that the UART receiver is idle and the RX/TX pin stays in logic high condition.

Bit 2    **RXIF**: Receive TXR_RXR data register status

    0: TXR_RXR data register is empty

    1: TXR_RXR data register has available data and reaches receiver FIFO trigger level

The RXIF flag is the receive data register status flag. When this read only flag is "0", it indicates that the TXR_RXR read data register is empty. When the flag is "1", it indicates that the TXR_RXR read data register contains new data and reaches the Receiver FIFO trigger level. When the contents of the shift register are transferred to the TXR_RXR register, and Receiver FIFO trigger level is reached, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag will eventually be cleared to zero when the USR register is read with RXIF set, followed by a read from the TXR_RXR register, and if the TXR_RXR register has no more new data available.

Bit 1    **TIDLE**: Transmission idle

    0: Data transmission is in progress (Data being transmitted)

    1: No data transmission is in progress (Transmitter is idle)

The TIDLE flag is known as the transmission complete flag. When this read only flag is "0", it indicates that a transmission is in progress. This flag will be set high when the TXIF flag is "1" and when there is no transmit data or break character being transmitted. When TIDLE is equal to "1", the TX pin becomes idle with the pin state

in logic high condition. The TIDLE flag is cleared to zero by reading the USR register with TIDLE set and then writing to the TXR_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.

Bit 0          **TXIF**: Transmit TXR_RXR data register status
    0: Character is not transferred to the transmit shift register
    1: Character has transferred to the transmit shift register (TXR_RXR data register is empty)

The TXIF flag is the transmit data register empty flag. When this read only flag is "0", it indicates that the character is not transferred to the transmitter shift register. When the flag is "1", it indicates that the transmitter shift register has received a character from the TXR_RXR data register. The TXIF flag is cleared to zero by reading the UART status register (USR) with TXIF set and then writing to the TXR_RXR data register. Note that when the TXEN bit is set, the TXIF flag bit will also be set since the transmit data register is not yet full.

• **UCR1 Register**

The UCR1 register together with the UCR2 register are the two UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length, etc. Further explanation on each of the bits is given below:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | UARTEN | BNO | PREN | PRT1 | PRT0 | TXBRK | RX8 | TX8 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R | W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | x | 0 |

"x": unknown

Bit 7          **UARTEN**: UART function enable control
    0: Disable UART. TX and RX/TX pins are in a floating state
    1: Enable UART. TX and RX/TX pins can function as UART pins

The UARTEN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the RX/TX pin as well as the TX pin will be in a floating state. When the bit is equal to "1", the UART will be enabled and the TX and RX/TX pins will function as defined by the SWM mode selection bit together with the TXEN and RXEN enable control bits.

When the UART is disabled, it will empty the buffer so any character remaining in the buffer will be discarded. In addition, the value of the baud rate counter will be reset. If the UART is disabled, all error and status flags will be reset. Also the TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF bits as well as the RxCNT register will be cleared to zero, while the TIDLE, TXIF and RIDLE bits will be set high. Other control bits in UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UART is active and the UARTEN bit is cleared to zero, all pending transmissions and receptions will be terminated and the module will be reset as defined above. When the UART is re-enabled, it will restart in the same configuration.

Bit 6          **BNO**: Number of data transfer bits selection
    0: 8-bit data transfer
    1: 9-bit data transfer

This bit is used to select the data length format, which can have a choice of either 8-bit or 9-bit format. When this bit is equal to "1", a 9-bit data length format will be selected. If the bit is equal to "0", then an 8-bit data length format will be selected. If 9-bit data length format is selected, then bits RX8 and TX8 will be used to store the 9th bit of the received and transmitted data respectively.

Note that the 9th bit of data if BNO=1, or the 8th bit of data if BNO=0, which is used as the parity bit, does not transfer to RX8 or TXRX7 (i.e. TXR_RXR.7) respectively when the parity function is enabled.

Bit 5     **PREN**: Parity function enable control

0: Parity function is disabled

1: Parity function is enabled

This is the parity enable bit. When this bit is equal to "1", the parity function will be enabled. If the bit is equal to "0", then the parity function will be disabled.

Bit 4~3   **PRT1~PRT0**: Parity type selection bits

00: Even parity for parity generator

01: Odd parity for parity generator

10: Mark parity for parity generator

11: Space parity for parity generator

These bits are the parity type selection bits. When these bits are equal to 00b, even parity type will be selected. If these bits are equal to 01b, then odd parity type will be selected. If these bits are equal to 10b, then a 1 (Mark) in the parity bit location will be selected. If these bits are equal to 11b, then a 0 (Space) in the parity bit location will be selected.

Bit 2     **TXBRK**: Transmit break character

0: No break character is transmitted

1: Break characters transmit

The TXBRK bit is the Transmit Break Character bit. When this bit is "0", there are no break characters and the TX pin operates normally. When the bit is "1", there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to "1", after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.

Bit 1     **RX8**: Receive data bit 8 for 9-bit data transfer format (read only)

This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

Bit 0     **TX8**: Transmit data bit 8 for 9-bit data transfer format (write only)

This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

### • UCR2 Register

The UCR2 register is the second of the UART control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UART Transmitter and Receiver as well as enabling the various UART interrupt sources. The register also serves to control the receiver STOP bit number selection, receiver wake-up enable and the address detect enable. Further explanation on each of the bits is given below:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | TXEN | RXEN | STOPS | ADDEN | WAKE | RIE | TIIE | TEIE |
| 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     **TXEN**: UART Transmitter enabled control

0: UART transmitter is disabled

1: UART transmitter is enabled

The bit named TXEN is the Transmitter Enable Bit. When this bit is equal to "0", the transmitter will be disabled with any pending data transmissions being aborted. In addition, the buffers will be reset. In this situation the TX pin will be in a floating state.

If the TXEN bit is equal to "1" and the UARTEN bit is also equal to "1", the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TX pin will be in a floating state.

Bit 6       **RXEN**: UART Receiver enabled control
              0: UART receiver is disabled
              1: UART receiver is enabled

The bit named RXEN is the Receiver Enable Bit. When this bit is equal to "0", the receiver will be disabled with any pending data receptions being aborted. In addition, the receive buffers will be reset. In this situation the RX/TX pin will be in a floating state. If the RXEN bit is equal to "1" and the UARTEN bit is also equal to "1", the receiver will be enabled and the RX/TX pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX/TX pin will be in a floating state.

Bit 5       **STOPS**: Number of Stop bits selection for receiver
              0: One stop bit format is used
              1: Two stop bits format is used

This bit determines if one or two stop bits are to be used for receiver. When this bit is equal to "1", two stop bits are used. If this bit is equal to "0", then only one stop bit is used. Two stop bits are used for transmitter.

Bit 4       **ADDEN**: Address detect function enable control
              0: Address detect function is disabled
              1: Address detect function is enabled

The bit named ADDEN is the address detect function enable control bit. When this bit is equal to "1", the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to TXRX7 if BNO=0 or the 9th bit, which corresponds to RX8 if BNO=1, has a value of "1", then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNO. If the address bit known as the 8th or 9th bit of the received word is "0" with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.

Bit 3       **WAKE**: RX/TX pin wake-up UART function enable control
              0: RX/TX pin wake-up UART function is disabled
              1: RX/TX pin wake-up UART function is enabled

This bit is used to control the wake-up UART function when a falling edge on the RX/TX pin occurs. Note that this bit is only available when the UART clock ($f_H$) is switched off. There will be no RX/TX pin wake-up UART function if the UART clock ($f_H$) exists. If the WAKE bit is set to 1 as the UART clock ($f_H$) is switched off, a UART wake-up request will be initiated when a falling edge on the RX/TX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX/TX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock ($f_H$) via the application program. Otherwise, the UART function cannot resume even if there is a falling edge on the RX/TX pin when the WAKE bit is cleared to 0.

Bit 2       **RIE**: Receiver interrupt enable control
              0: Receiver related interrupt is disabled
              1: Receiver related interrupt is enabled

This bit enables or disables the receiver interrupt. If this bit is equal to "1" and when the receiver overrun flag OERR or receive data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.

Bit 1       **TIIE**: Transmitter Idle interrupt enable control
              0: Transmitter idle interrupt is disabled
              1: Transmitter idle interrupt is enabled

This bit enables or disables the transmitter idle interrupt. If this bit is equal to "1" and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.

Bit 0　　　　**TEIE**: Transmitter Empty interrupt enable control

　　　　　　　0: Transmitter empty interrupt is disabled

　　　　　　　1: Transmitter empty interrupt is enabled

　　　　　　This bit enables or disables the transmitter empty interrupt. If this bit is equal to "1" and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to "0", the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

• **UCR3 Register**

The UCR3 register is used to enable the UART Single Wire Mode communication. As the name suggests in the single wire mode the UART communication can be implemented in one single line, RX/TX, together with the control of the RXEN and TXEN bits in the UCR2 register.

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

Bit 7~1　　Unimplemented, read as "0"

Bit 0　　　**SWM**: Single Wire Mode enable control

　　　　　　0: Disable, the RX/TX pin is used as UART receiver function only

　　　　　　1: Enable, the RX/TX pin can be used as UART receiver or transmitter function controlled by the RXEN and TXEN bits

　　　　　Note that when the Single Wire Mode is enabled, if both the RXEN and TXEN bits are high, the RX/TX pin will only be used as UART receiver input.

• **TXR_RXR Register**

The TXR_RXR register is the data register which is used to store the data to be transmitted on the TX pin or being received from the RX/TX pin.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | TXRX7 | TXRX6 | TXRX5 | TXRX4 | TXRX3 | TXRX2 | TXRX1 | TXRX0 |
| 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　　**TXRX7~TXRX0**: UART Transmit/Receive Data bit 7~bit 0

• **BRDH 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　　**D7~D0**: Baud rate divider high byte

　　　　　The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.

　　　　　Baud Rate=$f_H$/(BRD+UMOD/8)

　　　　　BRD=16~65535 or 8~65535 depending on BRDS

　　　　　Note: 1. BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.

　　　　　　　　2. The BRDL must be written first and then BRDH, otherwise errors may occur.

　　　　　　　　3. The BRDH register should not be modified during data transmission process.

• **BRDL 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　　**D7~D0**: Baud rate divider low byte

The baud rate divider BRD (BRDH/BRDL) defines the UART clock divider ratio.

Baud Rate=$f_H$/(BRD+UMOD/8)

BRD=16~65535 or 8~65535 depending on BRDS

Note: 1. BRD value should not be set to less than 16 when BRDS=0 or less than 8 when BRDS=1, otherwise errors may occur.

　　　2. The BRDL must be written first and then BRDH, otherwise errors may occur.

　　　3. The BRDL register should not be modified during data transmission process.

• **UFCR Register**

The UFCR register is the FIFO control register which is used for UART modulation control, BRD range selection and trigger level selection for RXIF and interrupt.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | — | — | UMOD2 | UMOD1 | UMOD0 | BRDS | RxFTR1 | RxFTR0 |
| 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　　**UMOD2~UMOD0**: UART Modulation Control bits

The modulation control bits are used to correct the baud rate of the received or transmitted UART signal. These bits determine if the extra UART clock cycle should be added in a UART bit time. The UMOD2~UMOD0 value will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases a UART clock cycle.

Bit 2　　**BRDS**: BRD range selection

　　0: BRD range is from 16 to 65535
　　1: BRD range is from 8 to 65535

The BRDS bit is used to control the sampling point in a UART bit time. If the BRDS bit is cleared to zero, the sampling point will be BRD/2, BRD/2+1×$f_H$, and BRD/2+2×$f_H$ in a UART bit time. If the BRDS bit is set high, the sampling point will be BRD/2-1×$f_H$, BRD/2, and BRD/2+2×$f_H$ in a UART bit time.

Note that the BRDS bit should not be modified during data transmission process.

Bit 1~0　　**RxFTR1~RxFTR0**: Receiver FIFO trigger level (bytes)

　　00: 4 bytes in RX FIFO
　　01: 1 or more bytes in RX FIFO
　　10: 2 or more bytes in RX FIFO
　　11: 3 or more bytes in RX FIFO

For the receiver these bits define the number of received data bytes in the RX FIFO that will trigger the RXIF bit being set high, an interrupt will also be generated if the RIE bit is enabled. To prevent OERR from being set high, the receiver FIFO trigger level can be set to 2 bytes, avoiding an overrun state that cannot be processed by the program in time when more than 4 data bytes are received. After the reset the receiver FIFO is empty.

• **RxCNT Register**

The RxCNT register is the counter used to indicate the number of received data bytes in the RX FIFO which have not been read by the MCU. This register is read only.

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

Bit 7~3     Unimplemented, read as "0"

Bit 2~0     **D2~D0**: RX FIFO counter

The RxCNT register is the counter used to indicate the number of received data bytes in the RX FIFO which is not read by the MCU. When RX FIFO receives one byte of data, the RxCNT will increase by one; when the MCU reads one byte of data from RX FIFO, the RxCNTn will decrease by one. If there are 4 bytes of data in the RX FIFO, the 5th data will be saved in the shift register. If there is 6th data, the 6th data will be saved in the shift register. But the RxCNT remains the value of 4. The RxCNT will be cleared when reset occurs or UARTEN=1. This register is read only.

## Baud Rate Generator

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 16-bit timer, the period of which is determined by two factors. The first of these is the value placed in the BRDH and BRDL baud rate registers and the second is the UART modulation control bits UMOD2~UMOD0. To prevent accumulated error of the receiver baud rate frequency, it is recommended to use two stop bits for resynchronization after each byte is received. If a baud rate BR is required with UART clock $f_H$.

$f_H$/BR=Integer Part + Fractional Part

The integer part is loaded into BRD (BRDH/BRDL). The fractional part is multiplied by 8 and rounded, then loaded into UMOD bit field as following:

BRD=TRUNC($f_H$/BR)

UMOD=ROUND[MOD($f_H$/BR)×8]

Therefore, the actual baud rate is as following:

Baud Rate=$f_H$/[BRD+(UMOD/8)]

### Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, determine the BRDH/BRDL register value, the actual baud rate and the error value for a desired baud rate of 230400.

From the above formula, the BRD=TRUNC($f_H$/BR)=TRUNC(17.36111)=17

The UMOD=ROUND[MOD($f_H$/BR)×8]=ROUND(0.36111×8)=ROUND(2.88888)=3

The actual Baud Rate=$f_H$/[BRD+(UMOD/8)]=230215.83

Therefore the error is equal to (230215.83-230400)/230400=-0.08%

### Modulation Control Example

To get the best-fitting bit sequence for UART modulation control bits UMOD2~UMOD0, the following algorithm can be used: Firstly, the fractional part of the theoretical division factor is multiplied by 8. Then the product will be rounded and the UMOD2~UMOD0 bits will be filled with the rounded value. The UMOD2~UMOD0 value will be added to internal accumulator for every UART bit time. Until a carry to bit 3, the corresponding UART bit time increases by a UART clock cycle. The following is an example using the fraction 0.36111 previously calculated: UMOD[2:0]= ROUND(0.36111×8)=011b

| Fraction Addition | Carry to Bit 3 | UART Bit Time Sequence | Extra UART Clock Cycle |
|---|---|---|---|
| 0000b + 0011b=0011b | No | Start bit | No |
| 0011b + 0011b=0110b | No | D0 | No |
| 0110b + 0011b=1001b | Yes | D1 | Yes |
| 1001b + 0011b=1100b | No | D2 | No |
| 1100b + 0011b=1111b | No | D3 | No |
| 1111b + 0011b=0010b | Yes | D4 | Yes |
| 0010b + 0011b=0101b | No | D5 | No |
| 0101b + 0011b=1000b | Yes | D6 | Yes |
| 1000b + 0011b=1011b | No | D7 | No |
| 1011b + 0011b=1110b | No | Parity bit | No |
| 1110b + 0011b=0001b | Yes | Stop bit | Yes |

**Baud Rate Correction Example**

The following figure presents an example using a baud rate of 230400 generated with UART clock $f_H$. The data format for the following figure is: eight data bits, parity enabled, no address bit, two stop bits.

The following figure shows three different frames:

- The upper frame is the correct one, with a bit-length of 17.36 $f_H$ cycles (4000000/230400=17.36).

- The middle frame uses a rough estimate, with 17 $f_H$ cycles for the bit length.

- The lower frame shows a corrected frame using the best fit for the UART modulation control bits UMOD2~UMOD0.
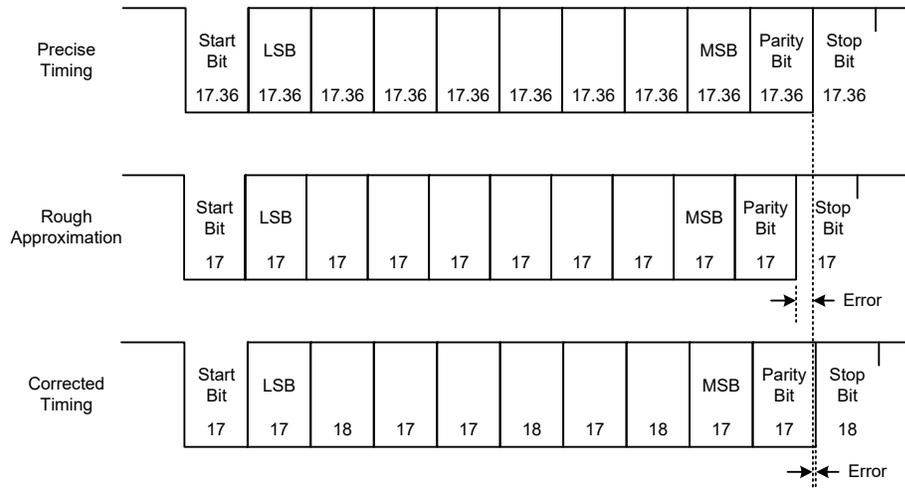


**UART Setup and Control**

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd, mark, space or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits along with the parity are setup by programming the BNO, PRT1~PRT0 and PREN bits. The transmitter always uses two stop bits while the receiver uses one or two stop bits which is determined by the STOPS bit. The baud rate used to transmit and receive data is setup using the internal 16-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

### Enabling/Disabling the UART Interface

The basic on/off function of the internal UART function is controlled using the UARTEN bit in the UCR1 register. If the UARTEN, TXEN and RXEN bits are set, then these two UART pins will act as normal TX output pin and RX/TX input pin respectively. If no data is being transmitted on the TX pin, then it will default to a logic high value.

Clearing the UARTEN bit will disable the TX and RX/TX pins and allow these two pins to be used as normal I/O or other pin-shared functional pins. When the UART function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UART will also reset the error and status flags with bits TXEN, RXEN, TXBRK, RXIF, OERR, FERR, PERR and NF as well as register RxCNT being cleared while bits TIDLE, TXIF and RIDLE will be set. The remaining control bits in the UCR1, UCR2, UCR3, UFCR, BRDH and BRDL registers will remain unaffected. If the UARTEN bit in the UCR1 register is cleared while the UART is active, then all pending transmissions and receptions will be immediately suspended and the UART will be reset to a condition as defined above. If the UART is then subsequently re-enabled, it will restart again in the same configuration.

### Data, Parity and Stop Bit Selection

The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UCR1 and UCR2 registers. The BNO bit controls the number of data bits which can be set to either 8 or 9, the PRT1~PRT0 bits control the choice of odd, even, mark or space parity, the PREN bit controls the parity on/off function and the STOPS bit decides whether one or two stop bits are to be used for the receiver, while the transmitter always uses two stop bits. The following table shows various formats for data transmission. The address bit, which is the MSB of the data byte, identifies the frame as an address character or data if the address detect function is enabled. The number of stop bits, which can be either one or two, is independent of the data length and is only configurable for the receiver. The transmitter uses two stop bits.

| Start Bit | Data Bits | Address Bit | Parity Bit | Stop Bit |
|---|---|---|---|---|
| **Example of 8-bit Data Formats** | | | | |
| 1 | 8 | 0 | 0 | 1 or 2 |
| 1 | 7 | 0 | 1 | 1 or 2 |
| 1 | 7 | 1 | 0 | 1 or 2 |
| **Example of 9-bit Data Formats** | | | | |
| 1 | 9 | 0 | 0 | 1 or 2 |
| 1 | 8 | 0 | 1 | 1 or 2 |
| 1 | 8 | 1 | 0 | 1 or 2 |

**Transmitter Receiver Data Format**

The following diagram shows the transmit and receive waveforms for both 8-bit and 9-bit data formats.



**8-bit Data Format**

**9-bit Data Format**
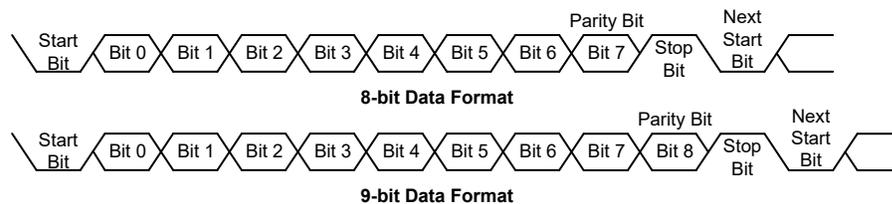
### UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the BNO bit in the UCR1 register. When BNO bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the TX8 bit in the UCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR, whose data is obtained from the transmit data register, which is known as the TXR_RXR register. The data to be transmitted is loaded into this TXR_RXR register by the application program. The TSR register is not written to with new data until the stop bit from the previous transmission has been sent out. As soon as this stop bit has been transmitted, the TSR can then be loaded with new data from the TXR_RXR register, if it is available. It should be noted that the TSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the TXEN bit is set, but the data will not be transmitted until the TXR_RXR register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the TXR_RXR register, after which the TXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the TXR_RXR register will result in an immediate transfer to the TSR. If during a transmission the TXEN bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX output pin can then be configured as the I/O or other pin-shared function by configuring the corresponding pin-shared control bits.

### Transmitting Data

When the UART is transmitting data, the data is shifted on the TX pin from the shift register, with the least significant bit first. In the transmit mode, the TXR_RXR register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the TX8 bit in the UCR1 register. The steps to initiate a data transfer can be summarized as follows:

• Make the correct selection of the BNO, PRT1~PRT0 and PREN bits to define the required word length and parity type. Two stop bits are used for the transmitter

• Setup the BRDH and BRDL registers and the UMOD2~UMOD0 bits to select the desired baud rate.

• Set the TXEN bit to ensure that the TX pin is used as a UART transmitter pin.

• Access the USR register and write the data that is to be transmitted into the TXR_RXR register. Note that this step will clear the TXIF bit.

This sequence of events can now be repeated to send additional data.

It should be noted that when TXIF=0, data will be inhibited from being written to the TXR_RXR register. Clearing the TXIF flag is always achieved using the following software sequence:

1. A USR register access

2. A TXR_RXR register write execution

The read-only TXIF flag is set by the UART hardware and if set indicates that the TXR_RXR register is empty and that other data can now be written into the TXR_RXR register without overwriting the previous data. If the TEIE bit is set, then the TXIF flag will generate an interrupt.

During a data transmission, a write instruction to the TXR_RXR register will place the data into the TXR_RXR register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the TXR_RXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIF bit being immediately set. When a frame transmission is complete, which happens

after stop bits are sent or after the break frame, the TIDLE bit will be set. To clear the TIDLE bit the following software sequence is used:

1. A USR register access

2. A TXR_RXR register write execution

Note that both the TXIF and TIDLE bits are cleared by the same software sequence.

### Transmitting Break

If the TXBRK bit is set high and the state keeps for a time of greater than $[(BRD+1)×t_H]$ while TIDLE=1, then break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by 13×N '0' bits and stop bits, where N=1, 2, etc. If a break character is to be transmitted then the TXBRK bit must be first set by the application program, and then cleared to generate the stop bits. Transmitting a break character will not generate a transmit interrupt. Note that a break condition length is at least 13 bits long. If the TXBRK bit is continually kept at a logic high level, then the transmitter circuitry will transmit continuous break characters. After the application program has cleared the TXBRK Sbit, the transmitter will finish transmitting the last break character and subsequently send out two stop bits. The automatic logic highs at the end of the last break character will ensure that the start bit of the next frame is recognized.

## UART Receiver

The UART is capable of receiving word lengths of either 8 or 9 bits. If the BNO bit is set, the word length will be set to 9 bits with the MSB being stored in the RX8 bit of the UCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX/TX pin is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RX/TX pin is sampled for the stop bit, the received data in RSR is transferred to the receive data register, if the register is empty. The data which is received on the external RX/TX pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX/TX pin. It should be noted that the RSR register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations.

### Receiving Data

When the UART receiver is receiving data, the data is serially shifted in on the external RX/TX pin, LSB first. In the read mode, the TXR_RXR register forms a buffer between the internal bus and the receiver shift register. The TXR_RXR register is a four-byte deep FIFO data buffer, where four bytes can be held in the FIFO while a fifth byte can continue to be received. Note that the application program must ensure that the data is read from TXR_RXR before the fifth byte has been completely shifted in, otherwise this fifth byte will be discarded and an overrun error OERR will be subsequently indicated. For continuous multi-byte data transmission, it is strongly recommended that the receiver uses two stop bits to avoid a receiving error caused by the accumulated error of the receiver baud rate frequency.

The steps to initiate a data transfer can be summarized as follows:

• Make the correct selection of BNO, PRT1~PRT0, PREN and STOPS bits to define the word length, parity type and number of stop bits.

• Setup the BRDH and BRDL registers and the UMOD2~UMOD0 bits to select the desired baud rate.

• Set the RXEN bit to ensure that the RX/TX pin is used as a UART receiver pin.

At this point the receiver will be enabled which will begin to look for a start bit.

When a character is received the following sequence of events will occur:

- The RXIF bit in the USR register will be set when the TXR_RXR register has data available, the number of the available data bytes can be checked by polling the RxCNT register content.
- When the contents of the shift register have been transferred to the TXR_RXR register and Receiver FIFO trigger level is reached, then if the RIE bit is set, an interrupt will be generated.
- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

The RXIF bit can be cleared using the following software sequence:

1. A USR register access
2. A TXR_RXR register read execution

### Receiving Break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the BNO bit plus one or two stop bits. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by BNO plus one or two stop bits. The RXIF bit is set, FERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the RIDLE bit is set. A break is regarded as a character that contains only zeros with the FERR flag set. If a long break signal has been detected, the receiver will regard it as a data frame including a start bit, data bits and the invalid stop bit and the FERR flag will be set. The receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that the break condition on the line is the next start bit. The break character will be loaded into the buffer and no further data will be received until one or two stop bits are received. It should be noted that the RIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:

- The framing error flag, FERR, will be set.
- The receive data register, TXR_RXR, will be cleared.
- The OERR, NF, PERR, RIDLE or RXIF flags will possibly be set.

### Idle Status

When the receiver is reading data, which means it will be in between the detection of a start bit and the reading of a stop bit, the receiver status flag in the USR register, otherwise known as the RIDLE flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the RIDLE flag will have a high value, which indicates the receiver is in an idle condition.

### Receiver Interrupt

The read only receive interrupt flag RXIF in the USR register is set by an edge generated by the receiver. An interrupt is generated if RIE=1, when a word is transferred from the Receive Shift Register, RSR, to the Receive Data Register, TXR_RXR. An overrun error can also generate an interrupt if RIE=1.

When a subroutine will be called with an execution time longer than the time for UART to receive five data bytes, if the UART received data could not be read in time during the subroutine execution, clear the RXEN bit to zero in advance to suspend data reception. If the UART interrupt could not be served in time to process the overrun error during the subroutine execution, ensure that both EMI and RXEN bits are disabled during this period, and then enable EMI and RXEN again after the subroutine execution has been completed to continue the UART data reception.

### Managing Receiver Errors

Several types of reception errors can occur within the UART module, the following section describes the various types and how they are managed by the UART.

#### Overrun Error – OERR

The TXR_RXR register is composed of a four byte deep FIFO data buffer, where four bytes can be held in the FIFO register, while a fifth byte can continue to be received. Before this fifth byte has been entirely shifted in, the data should be read from the TXR_RXR register. If this is not done, the overrun error flag OERR will be consequently indicated.

In the event of an overrun error occurring, the following will happen:

- The OERR flag in the USR register will be set.
- The TXR_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIE bit is set.

When the OERR flag is set to "1", it is necessary to read five data bytes from the four-byte deep receiver FIFO and the shift register immediately to avoid unexpected errors, such as the UART is unable to receive data. If such an error occurs, clear the RXEN bit to "0" then set it to "1" again to continue data reception.

The OERR flag can be cleared by an access to the USR register followed by a read to the TXR_RXR register.

#### Noise Error – NF

Over-sampling is used for data recovery to identify valid incoming data and noise. If noise is detected within a frame the following will occur:

- The read only noise flag, NF, in the USR register will be set on the rising edge of the RXIF bit.
- Data will be transferred from the Shift register to the TXR_RXR register.
- No interrupt will be generated. However this bit rises at the same time as the RXIF bit which itself generates an interrupt.

Note that the NF flag is reset by a USR register read operation followed by a TXR_RXR register read operation.

#### Framing Error – FERR

The read only framing error flag, FERR, in the USR register, is set if a zero is detected instead of stop bits. If two stop bits are selected, both stop bits must be high; otherwise the FERR flag will be set. The FERR flag and the received data will be recorded in the USR and TXR_RXR registers respectively, and the flag is cleared in any reset.

#### Parity Error – PERR

The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN=1, and if the parity type, odd, even, mark or space, is selected. The read only PERR flag and the received data will be recorded in the USR and TXR_RXR registers respectively. It is cleared on any reset, it should be noted that the flags, FERR and PERR, in the USR register should first be read by the application program before reading the data word.

### UART Interrupt Structure

Several individual UART conditions can generate a UART interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX/TX pin wake-up. When any of these conditions are created, if the global interrupt enable bit and its corresponding interrupt control bit are enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding USR register flags which will generate a UART interrupt if its associated interrupt enable control bit in the UCR2 register is set. The two transmitter interrupt conditions have their own corresponding enable control bits, while the two receiver interrupt conditions have a shared enable control bit. These enable bits can be used to mask out individual UART interrupt sources.

The address detect condition, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt when an address detect condition occurs if its function is enabled by setting the ADDEN bit in the UCR2 register. An RX/TX pin wake-up, which is also a UART interrupt source, does not have an associated flag, but will generate a UART interrupt if the UART clock ($f_H$) source is switched off and the WAKE and RIE bits in the UCR2 register are set when a falling edge on the RX/TX pin occurs.

Note that the USR register flags are read only and cannot be cleared or set by the application program, neither will they be cleared when the program jumps to the corresponding interrupt servicing routine, as is the case for some of the other interrupts. The flags will be cleared automatically when certain actions are taken by the UART, the details of which are given in the UART register section. The overall UART interrupt can be disabled or enabled by the related interrupt enable control bits in the interrupt control registers of the microcontroller to decide whether the interrupt requested by the UART module is masked out or allowed.
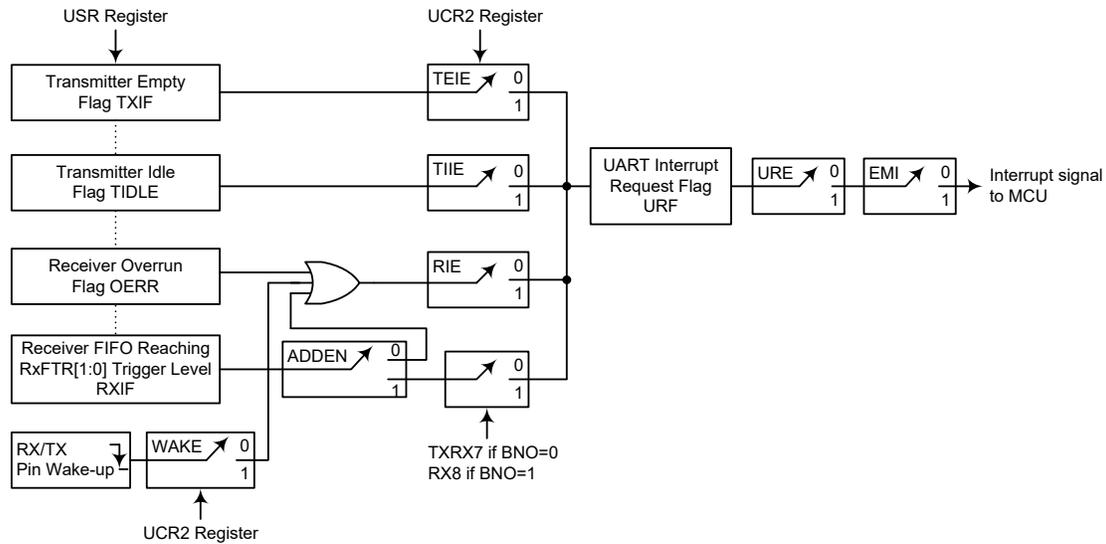


**UART Interrupt Structure**

### Address Detect Mode

Setting the Address Detect Mode bit, ADDEN, in the UCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF flag. If the ADDEN bit is enabled, then when data is available, an interrupt will only be generated, if the highest received bit has a high value. Note that the URE and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO=1 or the 8th bit if BNO=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the RXIF flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit PREN to zero.

| ADDEN | 9th Bit if BNO=1, 8th Bit if BNO=0 | UART Interrupt Generated |
|---|---|---|
| 0 | 0 | √ |
| | 1 | √ |
| 1 | 0 | × |
| | 1 | √ |

ADDEN Bit Function

## UART Power Down and Wake-up

When the UART clock ($f_H$) is off, the UART will cease to function, all clock sources to the module are shutdown. If the UART clock ($f_H$) is off while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU enters the IDLE or SLEEP mode while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP mode, note that the USR, UCR1, UCR2, UCR3, UFCR, RxCNT, TXR_RXR as well as the BRDH and BRDL registers will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART function contains a receiver RX/TX pin wake-up function, which is enabled or disabled by the WAKE bit in the UCR2 register. If this bit, along with the UART enable bit, UARTEN, the receiver enable bit, RXEN and the receiver interrupt bit, RIE, are all set when the UART clock ($f_H$) is off, then a falling edge on the RX/TX pin will trigger an RX/TX pin wake-up UART interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX/TX pin will be ignored.

For a UART wake-up interrupt to occur, in addition to the bits for the wake-up being set, the global interrupt enable bit, EMI, and the UART interrupt enable bit, URE, must be set. If the EMI and URE bits are not set then only a wake up event will occur and no interrupt will be generated. Note also that as it takes certain system clock cycles after a wake-up before normal microcontroller resumes, the UART interrupt will not be generated until after this time has elapsed.

## 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, $V_{DD}$, 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, VLVD2~VLVD0, are used to select one of six fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the $V_{DD}$ voltage is above the preset low voltage value. The LVDEN bit is used to control the overall on/off function of the low voltage detector. Setting the bit high will enable the low voltage detector. Clearing the bit to zero will switch off the internal low voltage detector circuits. As the low voltage detector will consume a certain amount of power, it may be desirable to switch off the circuit when not in use, an important consideration in power sensitive battery powered applications.

- **LVDC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|---|---|---|---|---|---|---|---|
| Name | TLVD1 | TLVD0 | LVDO | LVDEN | — | VLVD2 | VLVD1 | VLVD0 |
| R/W | R/W | R/W | R | R/W | — | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | — | 0 | 0 | 0 |

Bit 7~6      **TLVD1~TLVD0**: Minimum low voltage width to interrupt time ($t_{LVD}$)
         00: $(1\sim2)\times t_{LIRC}$
         01: $(3\sim4)\times t_{LIRC}$
         10: $(7\sim8)\times t_{LIRC}$
         11: $(1\sim2)\times t_{LIRC}$

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      Unimplemented, read as "0"

Bit 2~0      **VLVD2~VLVD0**: LVD Voltage selection
         000: 1.8V
         001: 2.0V
         010: 2.4V
         011: 2.7V
         100: 3.0V
         101: 3.3V
         110: 3.6V
         111: 4.0V

### LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, $V_{DD}$, with a pre-specified voltage level stored in the LVDC register. This has a range of between 1.8V 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. 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 $V_{DD}$ voltage may rise and fall rather slowly, at the voltage nears that of $V_{LVD}$, there may be multiple bit LVDO transitions.



**LVD Operation**
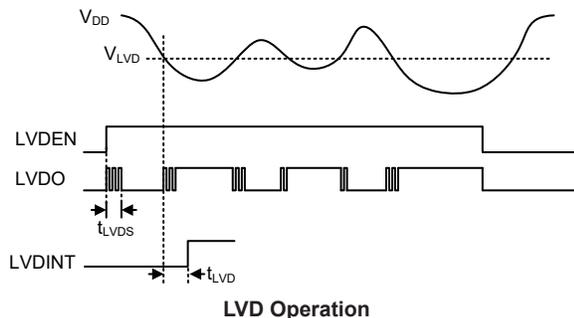
The Low Voltage Detector also has its own interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of $t_{LVD}$ after the LVDO bit has been set high by a low voltage condition, i.e., $V_{DD}$ falls below the preset LVD voltage. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated. This will cause the device to wake-up from the 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 IDLE Mode.

## Interrupts

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

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.

**Interrupt Structure**
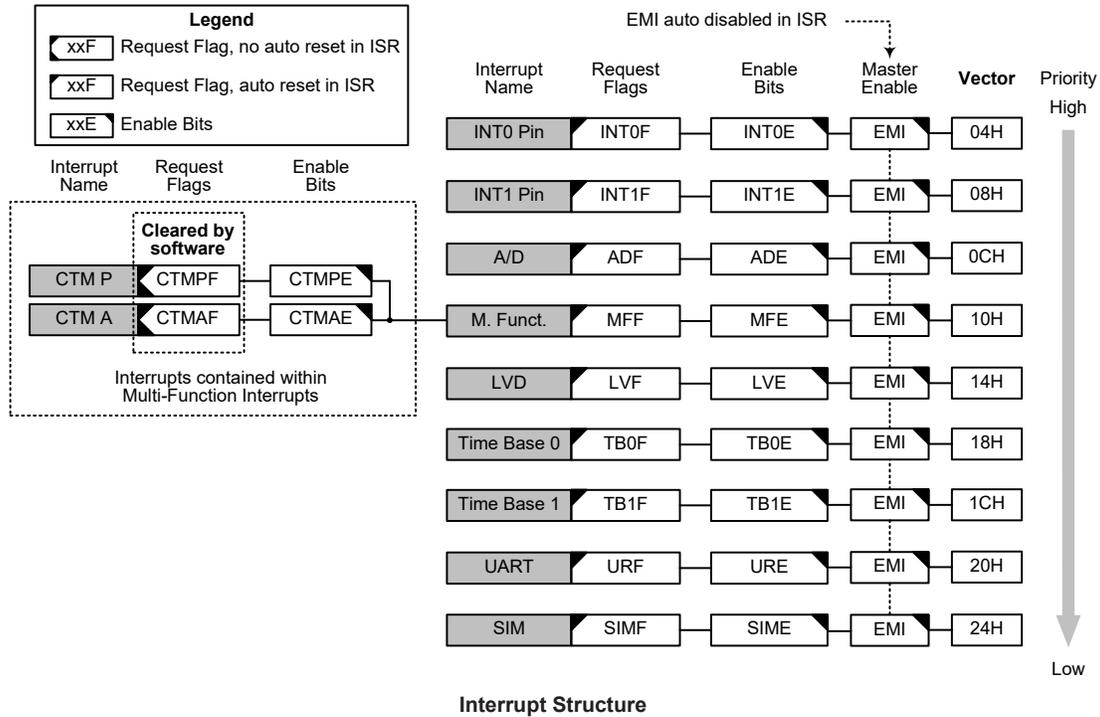
## Interrupt Registers

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

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

| Function | Enable Bit | Request Flag | Notes |
|---|---|---|---|
| Global | EMI | — | — |
| INTn Pins | INTnE | INTnF | n=0~1 |
| A/D Converter | ADE | ADF | — |
| Multi-function | MFE | MFF | — |
| Time Base | TBnE | TBnF | n=0~1 |
| UART Interface | URE | URF | — |
| SIM Interface | SIME | SIMF | — |
| LVD | LVE | LVF | — |
| CTM | CTMPE | CTMPF | — |
| | CTMAE | CTMAF | |

**Interrupt Register Bit Naming Conventions**

| Register | Bit | | | | | | | |
|----------|-----|---|---|---|---|---|---|---|
| Name | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTEG | — | — | — | — | INT1S1 | INT1S0 | INT0S1 | INT0S0 |
| INTC0 | — | ADF | INT1F | INT0F | ADE | INT1E | INT0E | EMI |
| INTC1 | TB1F | TB0F | LVF | MFF | TB1E | TB0E | LVE | MFE |
| INTC2 | — | — | SIMF | URF | — | — | SIME | URE |
| MFI | — | — | CTMAF | CTMPF | — | — | CTMAE | CTMPE |

**Interrupt Register List**

• **INTEG Register**

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

Bit 7~4        Unimplemented, read as "0"

Bit 3~2        **INT1S1~INT1S0**: 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 | TB1F | TB0F | LVF | MFF | TB1E | TB0E | LVE | MFE |
| 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        **TB1F**: Time Base 1 interrupt request flag
      0: No request
      1: Interrupt request

Bit 6        **TB0F**: Time Base 0 interrupt request flag
      0: No request
      1: Interrupt request

Bit 5        **LVF**: LVD interrupt request flag
      0: No request
      1: Interrupt request

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

Bit 3        **TB1E**: Time Base 1 interrupt control
      0: Disable
      1: Enable

Bit 2        **TB0E**: Time Base 0 interrupt control
      0: Disable
      1: Enable

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

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

• **INTC2 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | — | — | SIMF | URF | — | — | SIME | URE |
| R/W | — | — | R/W | R/W | — | — | R/W | R/W |
| POR | — | — | 0 | 0 | — | — | 0 | 0 |

Bit 7~6     Unimplemented, read as "0"

Bit 5        **SIMF**: SIM interrupt request flag
      0: No request
      1: Interrupt request

Bit 4        **URF**: UART interrupt request flag
      0: No request
      1: Interrupt request

Bit 3~2     Unimplemented, read as "0"

Bit 1        **SIME**: SIM interrupt control
      0: Disable
      1: Enable

Bit 0        **URE**: UART interrupt control
             0: Disable
             1: Enable

- **MFI 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        Unimplemented, read as "0"

Bit 5        **CTMAF**: CTM Comparator A match interrupt request flag
             0: No request
             1: Interrupt request
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 4        **CTMPF**: CTM Comparator P match interrupt request flag
             0: No request
             1: Interrupt request
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

Bit 3        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

## 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.

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 Interrupt Structure 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.

### External Interrupts

The external interrupts are controlled by signal transitions on the pins INT0~INT1. An external interrupt request will take place when the external interrupt request flags, INT0F~INT1F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI, respective external interrupt enable bit, INT0E~INT1E, and relevant Multi-function Interrupt enable bit, MFE, 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 or the relevant Multi-function 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.

### 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 interrupt vector address, the global interrupt enable bit, EMI, and A/D Interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D Converter Interrupt vector, will take place. When the interrupt is serviced, the A/D Converter Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

### Multi-function Interrupts

Within the device there is one Multi-function interrupt. Unlike the other independent interrupts, this interrupt has no independent source, but rather is formed from other existing interrupt sources, namely the CTM interrupt.

A Multi-function interrupt request will take place when the Multi-function interrupt request flag MFF is set. The Multi-function interrupt flag will be set when any of their included functions generate an interrupt request flag. When the Multi-function interrupt is enabled and the stack is not full, and one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine

call to the relevant Multi-function interrupt vector 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 request 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.

### Timer Module Interrupts

The Timer Module Interrupts are contained within the Multi-function Interrupt. 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 Compact Type TM interrupts are contained within the Multi-function Interrupts. For the TM 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, MFE, 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 MFF 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.

### LVD Interrupt

An LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage . To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, Low Voltage Interrupt enable bit, LVE, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a a subroutine call to the LVD Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. When the Low Voltage Interrupt is serviced, the LVD Interrupt flag, LVF, 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 signals 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 cleared, the EMI bit will also be automatically 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 $f_{SYS}$, $f_{SYS}/4$, $f_{SYS}/2$ 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 cource that generate $f_{PSC}$, which in turn controls the Time Base interrupt period, is selected using the CLKSEL[1:0] bits in the PSCR register.

**Time Base Interrupts**
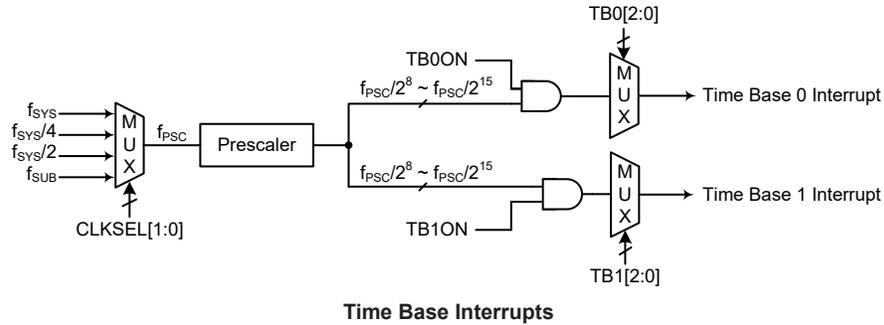
• **PSCR Register**

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

Bit 7~3    Unimplemented, read as "0"

Bit 2    **PSCEN**: Prescaler clock enable control
    0: Disable
    1: Enable
    This PSCEN bit is the Prescaler clock enable or disable control bit. When the Prescale clock is disabled, it can reduce extra power consumption.

Bit 1~0    **CLKSEL1~CLKSEL0**: Prescaler clock source $f_{PSC}$ selection
    00: $f_{SYS}$
    01: $f_{SYS}/4$
    10: $f_{SYS}/2$
    11: $f_{SUB}$

• **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 7~3    **TB0ON**: Time Base 0 Enable Control
    0: Disable
    1: Enable

Bit 6~3    Unimplemented, read as "0"

Bit 2~0    **TB02~TB00**: Time Base 0 time-out period selection
    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 Enable Control
         0: Disable
         1: Enable

Bit 6~3      Unimplemented, read as "0"

Bit 2~0      **TB12~TB10**: Time Base 1 time-out period selection
         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: $2^{15}/f_{PSC}$

## Serial Interface Module Interrupt

The Serial Interface Module Interrupt is also known as the SIM interrupt. A SIM Interrupt request 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, an I²C slave address match or I²C bus time-out occurs. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and SIM 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 corresponding SIM Interrupt vector, will take place. When the interrupt is serviced, the SIMF flag will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts.

## UART Interrupts

The UART Interrupt is controlled by several UART transfer conditions. When one of these conditions occurs, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RX/TX pin wake-up. To allow the program to branch to their interrupt vector addresses, the global interrupt enable bit, EMI, and UART Interrupt enable bit, URE, must first be set. When the interrupt is enabled, the stack is not full and any of the conditions described above occurs, a subroutine call to the corresponding UART Interrupt vector, will take place. When the interrupt is serviced, the UART Interrupt flag, URF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the USR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART section.

### 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 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, MFF, 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.

## Application Circuits
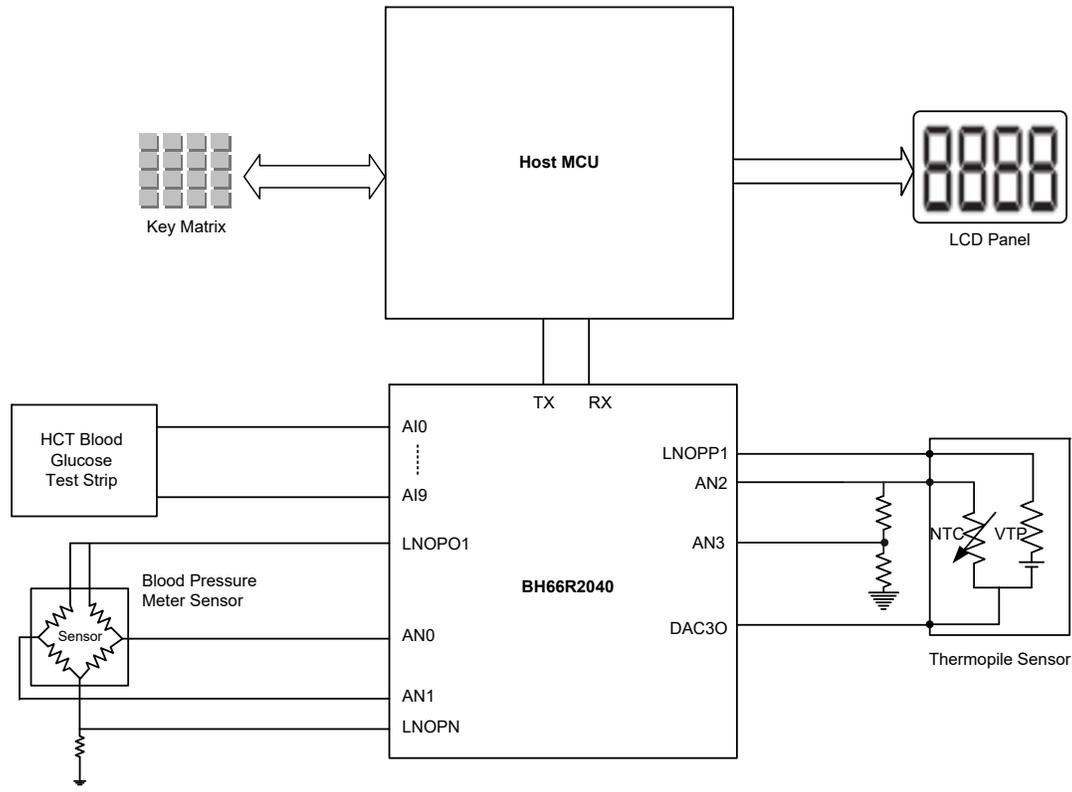
## Instruction Set

### 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 several 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 such as INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.

## Logical and Rotate Operation

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

## Branches and Control Transfer

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

## Bit Operations

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

## Table Read Operations

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

## Other Operations

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

## Instruction Set Summary

The instructions related to the data memory access in the following table can be used when the desired data memory is located in Data Memory sector 0.

### 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, SC |
| ADDM A,[m] | Add ACC to Data Memory | 1[Note] | Z, C, AC, OV, SC |
| ADD A,x | Add immediate data to ACC | 1 | Z, C, AC, OV, SC |
| ADC A,[m] | Add Data Memory to ACC with Carry | 1 | Z, C, AC, OV, SC |
| ADCM A,[m] | Add ACC to Data memory with Carry | 1[Note] | Z, C, AC, OV, SC |
| SUB A,x | Subtract immediate data from the ACC | 1 | Z, C, AC, OV, SC, CZ |
| SUB A,[m] | Subtract Data Memory from ACC | 1 | Z, C, AC, OV, SC, CZ |
| SUBM A,[m] | Subtract Data Memory from ACC with result in Data Memory | 1[Note] | Z, C, AC, OV, SC, CZ |
| SBC A,x | Subtract immediate data from ACC with Carry | 1 | Z, C, AC, OV, SC, CZ |
| SBC A,[m] | Subtract Data Memory from ACC with Carry | 1 | Z, C, AC, OV, SC, CZ |
| SBCM A,[m] | Subtract Data Memory from ACC with Carry, result in Data Memory | 1[Note] | Z, C, AC, OV, SC, CZ |
| 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 | 1[Note] | None |
| MOV A,x | Move immediate data to ACC | 1 | None |
| **Bit Operation** | | | |
| CLR [m].i | Clear bit of Data Memory | 1[Note] | None |
| SET [m].i | Set bit of Data Memory | 1[Note] | None |
| **Branch Operation** | | | |
| JMP addr | Jump unconditionally | 2 | None |
| SZ [m] | Skip if Data Memory is zero | 1[Note] | None |
| SZA [m] | Skip if Data Memory is zero with data movement to ACC | 1[Note] | None |
| SZ [m].i | Skip if bit i of Data Memory is zero | 1[Note] | None |
| SNZ [m] | Skip if Data Memory is not zero | 1[Note] | None |
| SNZ [m].i | Skip if bit i of Data Memory is not zero | 1[Note] | None |
| SIZ [m] | Skip if increment Data Memory is zero | 1[Note] | None |
| SDZ [m] | Skip if decrement Data Memory is zero | 1[Note] | None |
| SIZA [m] | Skip if increment Data Memory is zero with result in ACC | 1[Note] | None |
| SDZA [m] | Skip if decrement Data Memory is zero with result in ACC | 1[Note] | None |
| CALL addr | Subroutine call | 2 | None |
| RET | Return from subroutine | 2 | None |
| RET A,x | Return from subroutine and load immediate data to ACC | 2 | None |
| RETI | Return from interrupt | 2 | None |
| **Table Read Operation** | | | |
| TABRD [m] | Read table (specific page) to TBLH and Data Memory | 2[Note] | None |
| TABRDL [m] | Read table (last page) to TBLH and Data Memory | 2[Note] | None |
| ITABRD [m] | Increment table pointer TBLP first and Read table (specific page) to TBLH and Data Memory | 2[Note] | None |
| ITABRDL [m] | Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory | 2[Note] | None |
| **Miscellaneous** | | | |
| NOP | No operation | 1 | None |
| CLR [m] | Clear Data Memory | 1[Note] | None |
| SET [m] | Set Data Memory | 1[Note] | None |
| CLR WDT | Clear Watchdog Timer | 1 | TO, PDF |
| SWAP [m] | Swap nibbles of Data Memory | 1[Note] | None |
| SWAPA [m] | Swap nibbles of Data Memory with result in ACC | 1 | None |
| HALT | Enter power down mode | 1 | TO, PDF |

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

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

### Extended Instruction Set

The extended instructions are used to support the full range address access for the data memory. When the accessed data memory is located in any data memory sector except sector 0, the extended instruction can be used to directly access the data memory instead of using the indirect addressing access. This can not only reduce the use of Flash memory space but also improve the CPU execution efficiency.

| Mnemonic | Description | Cycles | Flag Affected |
|---|---|---|---|
| **Arithmetic** | | | |
| LADD A,[m] | Add Data Memory to ACC | 2 | Z, C, AC, OV, SC |
| LADDM A,[m] | Add ACC to Data Memory | 2[Note] | Z, C, AC, OV, SC |
| LADC A,[m] | Add Data Memory to ACC with Carry | 2 | Z, C, AC, OV, SC |
| LADCM A,[m] | Add ACC to Data memory with Carry | 2[Note] | Z, C, AC, OV, SC |
| LSUB A,[m] | Subtract Data Memory from ACC | 2 | Z, C, AC, OV, SC, CZ |
| LSUBM A,[m] | Subtract Data Memory from ACC with result in Data Memory | 2[Note] | Z, C, AC, OV, SC, CZ |
| LSBC A,[m] | Subtract Data Memory from ACC with Carry | 2 | Z, C, AC, OV, SC, CZ |
| LSBCM A,[m] | Subtract Data Memory from ACC with Carry, result in Data Memory | 2[Note] | Z, C, AC, OV, SC, CZ |
| LDAA [m] | Decimal adjust ACC for Addition with result in Data Memory | 2[Note] | C |
| **Logic Operation** | | | |
| LAND A,[m] | Logical AND Data Memory to ACC | 2 | Z |
| LOR A,[m] | Logical OR Data Memory to ACC | 2 | Z |
| LXOR A,[m] | Logical XOR Data Memory to ACC | 2 | Z |
| LANDM A,[m] | Logical AND ACC to Data Memory | 2[Note] | Z |
| LORM A,[m] | Logical OR ACC to Data Memory | 2[Note] | Z |
| LXORM A,[m] | Logical XOR ACC to Data Memory | 2[Note] | Z |
| LCPL [m] | Complement Data Memory | 2[Note] | Z |
| LCPLA [m] | Complement Data Memory with result in ACC | 2 | Z |
| **Increment & Decrement** | | | |
| LINCA [m] | Increment Data Memory with result in ACC | 2 | Z |
| LINC [m] | Increment Data Memory | 2[Note] | Z |
| LDECA [m] | Decrement Data Memory with result in ACC | 2 | Z |
| LDEC [m] | Decrement Data Memory | 2[Note] | Z |
| **Rotate** | | | |
| LRRA [m] | Rotate Data Memory right with result in ACC | 2 | None |
| LRR [m] | Rotate Data Memory right | 2[Note] | None |
| LRRCA [m] | Rotate Data Memory right through Carry with result in ACC | 2 | C |
| LRRC [m] | Rotate Data Memory right through Carry | 2[Note] | C |
| LRLA [m] | Rotate Data Memory left with result in ACC | 2 | None |
| LRL [m] | Rotate Data Memory left | 2[Note] | None |
| LRLCA [m] | Rotate Data Memory left through Carry with result in ACC | 2 | C |
| LRLC [m] | Rotate Data Memory left through Carry | 2[Note] | C |
| **Data Move** | | | |
| LMOV A,[m] | Move Data Memory to ACC | 2 | None |
| LMOV [m],A | Move ACC to Data Memory | 2[Note] | None |
| **Bit Operation** | | | |
| LCLR [m].i | Clear bit of Data Memory | 2[Note] | None |
| LSET [m].i | Set bit of Data Memory | 2[Note] | None |

| Mnemonic | Description | Cycles | Flag Affected |
|---|---|---|---|
| **Branch** | | | |
| LSZ [m] | Skip if Data Memory is zero | 2[Note] | None |
| LSZA [m] | Skip if Data Memory is zero with data movement to ACC | 2[Note] | None |
| LSNZ [m] | Skip if Data Memory is not zero | 2[Note] | None |
| LSZ [m].i | Skip if bit i of Data Memory is zero | 2[Note] | None |
| LSNZ [m].i | Skip if bit i of Data Memory is not zero | 2[Note] | None |
| LSIZ [m] | Skip if increment Data Memory is zero | 2[Note] | None |
| LSDZ [m] | Skip if decrement Data Memory is zero | 2[Note] | None |
| LSIZA [m] | Skip if increment Data Memory is zero with result in ACC | 2[Note] | None |
| LSDZA [m] | Skip if decrement Data Memory is zero with result in ACC | 2[Note] | None |
| **Table Read** | | | |
| LTABRD [m] | Read table (specific page) to TBLH and Data Memory | 3[Note] | None |
| LTABRDL [m] | Read table (last page) to TBLH and Data Memory | 3[Note] | None |
| LITABRD [m] | Increment table pointer TBLP first and Read table (specific page) to TBLH and Data Memory | 3[Note] | None |
| LITABRDL [m] | Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory | 3[Note] | None |
| **Miscellaneous** | | | |
| LCLR [m] | Clear Data Memory | 2[Note] | None |
| LSET [m] | Set Data Memory | 2[Note] | None |
| LSWAP [m] | Swap nibbles of Data Memory | 2[Note] | None |
| LSWAPA [m] | Swap nibbles of Data Memory with result in ACC | 2 | None |

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

2. Any extended instruction which changes the contents of the PCL register will also require three 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    $ACC \leftarrow ACC + [m] + C$

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


**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    $[m] \leftarrow ACC + [m] + C$

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


**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    $ACC \leftarrow ACC + [m]$

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


**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    $ACC \leftarrow ACC + x$

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


**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    $[m] \leftarrow ACC + [m]$

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


**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    $ACC \leftarrow ACC \ "AND" \ [m]$

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    $ACC \leftarrow ACC \ "AND" \ x$

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<br>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<br>TO ← 0<br>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] ← [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 ← [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<br>[m] ← ACC + 06H or<br>[m] ← ACC + 60H or<br>[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)<br>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)<br>[m].0 ← C<br>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    $ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$
$ACC.0 \leftarrow C$
$C \leftarrow [m].7$

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    $[m].i \leftarrow [m].(i+1); (i=0\sim6)$
$[m].7 \leftarrow [m].0$

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    $ACC.i \leftarrow [m].(i+1); (i=0\sim6)$
$ACC.7 \leftarrow [m].0$

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    $[m].i \leftarrow [m].(i+1); (i=0\sim6)$
$[m].7 \leftarrow C$
$C \leftarrow [m].0$

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    $ACC.i \leftarrow [m].(i+1); (i=0\sim6)$
$ACC.7 \leftarrow C$
$C \leftarrow [m].0$

Affected flag(s)    C

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

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

Operation    $ACC \leftarrow ACC - [m] - C$

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

**SBC A, x**          Subtract immediate data from ACC with Carry

| | |
|---|---|
| Description | The immediate data and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1. |
| Operation | ACC ← ACC – [m] – C |
| Affected flag(s) | OV, Z, AC, C, SC, CZ |

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

| | |
|---|---|
| Description | The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1. |
| Operation | [m] ← ACC − [m] − C |
| Affected flag(s) | OV, Z, AC, C, SC, CZ |

**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<br>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<br>Skip if ACC=0 |
| Affected flag(s) | None |

**SET [m]**         Set Data Memory

| | |
|---|---|
| Description | Each bit of the specified Data Memory is set to 1. |
| Operation | [m] ← FFH |
| Affected flag(s) | None |

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

| | |
|---|---|
| Description | Bit i of the specified Data Memory is set to 1. |
| Operation | [m].i ← 1 |
| Affected flag(s) | None |

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

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

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

Affected flag(s)    None

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

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

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

Affected flag(s)    None

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

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

Operation    Skip if $[m].i \neq 0$

Affected flag(s)    None

**SNZ [m]**    Skip if Data Memory is not 0

Description    The contents of the specified Data Memory are read out and then written back to the specified Data Memory again. If 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 $[m] \neq 0$

Affected flag(s)    None

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

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

Operation    $ACC \leftarrow ACC - [m]$

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

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

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

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

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

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

| **SWAP [m]** | Swap nibbles of Data Memory |
|---|---|
| Description | The low-order and high-order nibbles of the specified Data Memory are interchanged. |
| Operation | $[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$ |
| 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 | $ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$<br>$ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$ |
| 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 back 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 | $ACC \leftarrow [m]$<br>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) to TBLH and Data Memory

Description  The low byte of the program code (specific page) addressed by the table pointer (TBLP and 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


**ITABRD [m]**  Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory Description Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.

Operation  [m] ← program code (low byte)

TBLH ← program code (high byte)

Affected flag(s)  None


**ITABRDL [m]**  Increment table pointer low byte first and read table (last page) to TBLH and Data Memory

Description  Increment table pointer low byte, TBLP, first and then 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

### Extended Instruction Definition

The extended instructions are used to directly access the data stored in any data memory sections.

**LADC 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, SC |

**LADCM 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, SC |

**LADD 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, SC |

**LADDM 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, SC |

**LAND 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 |

**LANDM 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 |

**LCLR [m]**    Clear Data Memory

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

Operation    [m] ← 00H

Affected flag(s)    None


**LCLR [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


**LCPL [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] ← [m]

Affected flag(s)    Z


**LCPLA [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 ← [m]

Affected flag(s)    Z


**LDAA [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


**LDEC [m]**    Decrement Data Memory

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

Operation    [m] ← [m] − 1

Affected flag(s)    Z

**LDECA [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


**LINC [m]**         Increment Data Memory

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

Operation            [m] ← [m] + 1

Affected flag(s)     Z


**LINCA [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


**LMOV 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


**LMOV [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

**LOR 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


**LORM 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

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

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

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

Affected flag(s)  None

**LRLA [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

**LRLC [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

**LRLCA [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 theAccumulator 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

**LRR [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

**LRRA [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


**LRRC [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


**LRRCA [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


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

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

Operation       ACC ← ACC − [m] − C

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


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

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

Operation       [m] ← ACC − [m] − C

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

**LSDZ [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 three cycle instruction. If the result is not 0 the program
                   proceeds with the following instruction.

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

Affected flag(s)   None


**LSDZA [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 three cycle instruction. If the result is
                   not 0, the program proceeds with the following instruction.

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

Affected flag(s)   None


**LSET [m]**        Set Data Memory

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

Operation          $[m] \leftarrow FFH$

Affected flag(s)   None


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

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

Operation          $[m].i \leftarrow 1$

Affected flag(s)   None


**LSIZ [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 three cycle instruction. If the result is not 0 the program
                   proceeds with the following instruction.

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

Affected flag(s)   None

**LSIZA [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 three 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

**LSNZ [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 three cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation     Skip if [m].i ≠ 0

Affected flag(s)     None

**LSNZ [m]**     Skip if Data Memory is not 0

Description     The contents of the specified Data Memory are read out and then written to the specified Data Memory again. If the content 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 three cycle instruction. If the result is 0 the program proceeds with the following instruction.

Operation     Skip if [m] ≠ 0

Affected flag(s)     None

**LSUB 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, SC, CZ

**LSUBM 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, SC, CZ

| **LSWAP [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 |

| **LSWAPA [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<br>ACC.7~ACC.4 ← [m].3~[m].0 |
| Affected flag(s) | None |

| **LSZ [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 three cycle instruction. If the result is not 0 the program proceeds with the following instruction. |
| Operation | Skip if [m]=0 |
| Affected flag(s) | None |

| **LSZA [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 three cycle instruction. If the result is not 0 the program proceeds with the following instruction. |
| Operation | ACC ← [m]<br>Skip if [m]=0 |
| Affected flag(s) | None |

| **LSZ [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 three 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 |

| **LTABRD [m]** | Read table (specific page) to TBLH and Data Memory |
|---|---|
| Description | The low byte of the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH. |
| Operation | [m] ← program code (low byte)<br>TBLH ← program code (high byte) |
| Affected flag(s) | None |

| **LTABRDL [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) <br> TBLH ← program code (high byte) |
| Affected flag(s) | None |

| **LITABRD [m]** | Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory |
|---|---|
| Description | Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH. |
| Operation | [m] ← program code (low byte) <br> TBLH ← program code (high byte) |
| Affected flag(s) | None |

| **LITABRDL [m]** | Increment table pointer low byte first and read table (last page) to TBLH and Data Memory |
|---|---|
| Description | Increment table pointer low byte, TBLP, first and then 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) <br> TBLH ← program code (high byte) |
| Affected flag(s) | None |

| **LXOR 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 |

| **LXORM 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 |

# 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

### SAW Type 32-pin QFN (4mm×4mm×0.75mm) Outline Dimensions
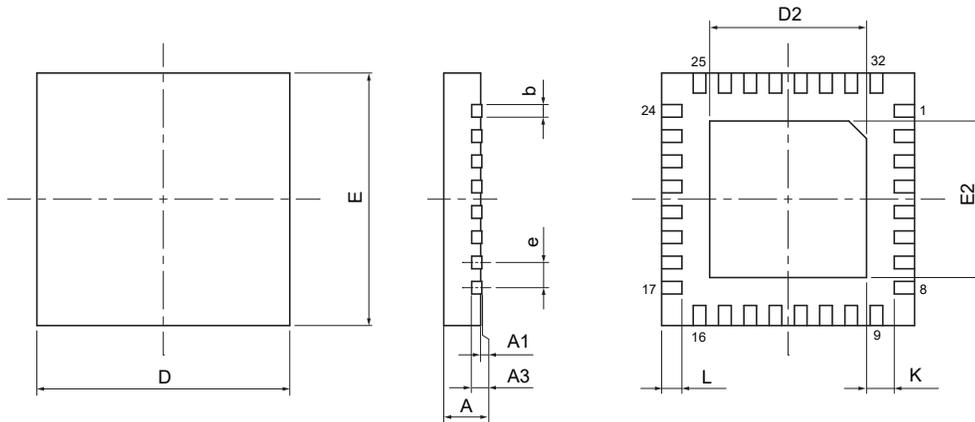
| Symbol | Dimensions in inch | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.028 | 0.030 | 0.031 |
| A1 | 0.000 | 0.001 | 0.002 |
| A3 | 0.008 REF | | |
| b | 0.006 | 0.008 | 0.010 |
| D | 0.157 BSC | | |
| E | 0.157 BSC | | |
| e | 0.016 BSC | | |
| D2 | 0.100 | — | 0.108 |
| E2 | 0.100 | — | 0.108 |
| L | 0.010 | — | 0.018 |
| K | 0.008 | — | — |

| Symbol | Dimensions in mm | | |
|---|---|---|---|
| | Min. | Nom. | Max. |
| A | 0.70 | 0.75 | 0.80 |
| A1 | 0.00 | 0.02 | 0.05 |
| A3 | 0.203 REF | | |
| b | 0.15 | 0.20 | 0.25 |
| D | 4.00 BSC | | |
| E | 4.00 BSC | | |
| e | 0.40 BSC | | |
| D2 | 2.55 | — | 2.75 |
| E2 | 2.55 | — | 2.75 |
| L | 0.25 | — | 0.45 |
| K | 0.20 | — | — |