



**CO/Gas Detector Flash MCU
with 12V Piezoelectric Horn Driver**

BA45F6830

Revision: V1.00 Date: September 22, 2021

www.holtek.com

Table of Contents

| | |
|---|-----------|
| Features | 6 |
| CPU Features | 6 |
| Peripheral Features..... | 6 |
| Piezoelectric Horn Driver Feature | 6 |
| General Description | 7 |
| Block Diagram | 7 |
| Pin Assignment | 8 |
| Pin Description | 9 |
| Absolute Maximum Ratings | 12 |
| MCU Absolute Maximum Ratings | 12 |
| Piezoelectric Horn Driver Absolute Maximum Ratings..... | 12 |
| D.C. Characteristics | 12 |
| Operating Voltage Characteristics..... | 12 |
| Standby Current Characteristics | 12 |
| Operating Current Characteristics..... | 13 |
| A.C. Characteristics | 14 |
| High Speed Internal Oscillator – HIRC – Frequency Accuracy | 14 |
| Low Speed Internal Oscillator Characteristics – LIRC | 14 |
| Operating Frequency Characteristic Curves | 14 |
| System Start Up Time Characteristics | 15 |
| Input/Output Characteristics | 15 |
| Memory Characteristics | 16 |
| LVR/LVD Electrical Characteristics | 16 |
| A/D Converter Electrical Characteristics | 17 |
| Reference Voltage Electrical Characteristics | 17 |
| OPA Electrical Characteristics | 18 |
| LDO Electrical Characteristics | 19 |
| Piezoelectric Horn Driver Electrical Characteristics | 20 |
| Power-on Reset Characteristics | 20 |
| System Architecture | 21 |
| Clocking and Pipelining..... | 21 |
| Program Counter..... | 22 |
| Stack | 22 |
| Arithmetic and Logic Unit – ALU | 23 |
| Flash Program Memory | 24 |
| Structure..... | 24 |
| Special Vectors | 24 |
| Look-up Table..... | 24 |

| | |
|--|-----------|
| Table Program Example | 25 |
| In Circuit Programming – ICP | 26 |
| On-Chip Debug Support – OCDS | 26 |
| Data Memory | 27 |
| Structure..... | 27 |
| Data Memory Addressing..... | 28 |
| General Purpose Data Memory | 28 |
| Special Purpose Data Memory | 28 |
| Special Function Register Description | 30 |
| Indirect Addressing Registers – IAR0, IAR1, IAR2 | 30 |
| Memory Pointers – MP0, MP1L, MP1H, MP2L, MP2H..... | 30 |
| Accumulator – ACC..... | 31 |
| Program Counter Low Byte Register – PCL..... | 32 |
| Look-up Table Registers – TBLP, TBHP, TBLH..... | 32 |
| Status Register – STATUS..... | 32 |
| EEPROM Data Memory..... | 34 |
| EEPROM Data Memory Structure | 34 |
| EEPROM Registers | 34 |
| Reading Data from the EEPROM | 36 |
| Writing Data to the EEPROM..... | 36 |
| Write Protection..... | 36 |
| EEPROM Interrupt | 36 |
| Programming Considerations..... | 37 |
| Oscillators | 38 |
| Oscillator Overview | 38 |
| System Clock Configurations | 38 |
| Internal High Speed RC Oscillator – HIRC | 39 |
| Internal 32kHz Oscillator – LIRC..... | 39 |
| Operating Modes and System Clocks | 39 |
| System Clocks | 39 |
| System Operation Modes..... | 40 |
| Control Registers | 41 |
| Operating Mode Switching..... | 43 |
| Standby Current Considerations | 46 |
| Wake-up | 46 |
| Watchdog Timer..... | 47 |
| Watchdog Timer Clock Source..... | 47 |
| Watchdog Timer Control Register | 47 |
| Watchdog Timer Operation | 48 |
| Reset and Initialisation..... | 49 |
| Reset Functions | 49 |
| Reset Initial Conditions | 50 |

| | |
|---|------------|
| Input/Output Ports | 53 |
| Pull-high Resistors | 53 |
| Port A Wake-up | 54 |
| I/O Port Control Registers | 54 |
| I/O Port Source Current Selection..... | 54 |
| Pin-shared Functions | 55 |
| I/O Pin Structures..... | 59 |
| Programming Considerations..... | 60 |
| Timer Module – TM | 60 |
| Introduction | 60 |
| TM Operation | 60 |
| TM Clock Source..... | 61 |
| TM Interrupts..... | 61 |
| TM External Pins..... | 61 |
| Programming Considerations..... | 62 |
| Periodic Type TM – PTM..... | 63 |
| Periodic TM Operation | 63 |
| Periodic Type TM Register Description | 63 |
| Periodic Type TM Operating Modes | 68 |
| Voltage Regulator – LDO..... | 77 |
| CO/Gas Detector AFE | 78 |
| CO/Gas Detector Registers | 78 |
| Input Offset Calibration | 80 |
| Analog to Digital Converter | 81 |
| A/D Converter Overview | 81 |
| A/D Converter Register Description | 82 |
| A/D Converter Reference Voltage..... | 84 |
| A/D Converter Input Signals..... | 84 |
| A/D Converter Operation | 85 |
| Conversion Rate and Timing Diagram | 86 |
| Summary of A/D Conversion Steps..... | 86 |
| Programming Considerations..... | 87 |
| A/D Conversion Function | 88 |
| A/D Conversion Programming Examples..... | 88 |
| Universal Serial Interface Module – USIM | 90 |
| SPI Interface | 90 |
| I ² C Interface | 98 |
| UART Interface..... | 108 |
| Piezoelectric Horn Driver | 123 |
| External-driving(2-pin) Buzzer Driver Mode Settings | 123 |
| Self-driving(3-pin) Buzzer Driver Mode Settings | 124 |

| | |
|--|------------|
| Low Voltage Detector – LVD | 125 |
| LVD Register | 125 |
| LVD Operation..... | 126 |
| Interrupts | 126 |
| Interrupt Registers..... | 126 |
| Interrupt Operation | 129 |
| External Interrupt..... | 130 |
| A/D Converter Interrupt | 130 |
| EEPROM Interrupt | 131 |
| TM Interrupts | 131 |
| Universal Serial Interface Module Interrupt..... | 131 |
| Time Base Interrupts | 132 |
| LVD Interrupt | 133 |
| Interrupt Wake-up Function..... | 133 |
| Programming Considerations..... | 134 |
| Configuration Options..... | 134 |
| Application Circuits..... | 135 |
| External-driving (2-pin) Buzzer..... | 135 |
| Self-driving (3-pin) Buzzer..... | 136 |
| Instruction Set..... | 137 |
| Introduction | 137 |
| Instruction Timing | 137 |
| Moving and Transferring Data..... | 137 |
| Arithmetic Operations..... | 137 |
| Logical and Rotate Operation | 138 |
| Branches and Control Transfer | 138 |
| Bit Operations | 138 |
| Table Read Operations | 138 |
| Other Operations..... | 138 |
| Instruction Set Summary | 139 |
| Table Conventions..... | 139 |
| Extended Instruction Set..... | 141 |
| Instruction Definition..... | 143 |
| Extended Instruction Definition | 152 |
| Package Information | 159 |
| 24-pin SSOP (150mil) Outline Dimensions | 160 |
| 28-pin SSOP (150mil) Outline Dimensions | 161 |

Features

CPU Features

- Operating voltage
 - ♦ $f_{SYS}=2\text{MHz}$: 2.2V~5.5V
 - ♦ $f_{SYS}=4\text{MHz}$: 2.2V~5.5V
 - ♦ $f_{SYS}=8\text{MHz}$: 2.2V~5.5V
- Up to 0.5 μs instruction cycle with 8MHz system clock at $V_{DD}=5\text{V}$
- Power down and wake-up functions to reduce power consumption
- Oscillator types
 - ♦ Internal High Speed 2/4/8MHz RC – HIRC
 - ♦ Internal Low Speed 32kHz RC – LIRC
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- Fully integrated internal oscillators require no external components
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 6-level subroutine nesting
- Bit manipulation instruction

Peripheral Features

- Flash Program Memory: 2K \times 16
- RAM Data Memory: 128 \times 8
- True EEPROM Memory: 32 \times 8
- Watchdog Timer function
- 14 bidirectional I/O lines
- One pin-shared external interrupt
- One Timer Module for time measurement, input capture, compare match output or PWM output or single pulse output function
- 5 external channel 12-bit resolution A/D converter with internal reference voltage V_{BG}
- CO/Gas detector AFE – OPA
- Universal Serial Interface Module – USIM for SPI, I²C or UART communication
- Dual Time Base functions for generation of fixed time interrupt signals
- Internal LDO: 2.2V/3.0V output
- Low voltage reset function
- Low voltage detect function
- Package types: 24/28-pin SSOP

Piezoelectric Horn Driver Feature

- Input voltage (BDV_{DD}): 2.2V~5.5V
- Supports self-driving(3-pin) or external-driving(2-pin) buzzer
- Integrated 12V boost converter

General Description

The device is a Flash Memory A/D type 8-bit high performance RISC architecture microcontroller, which includes a piezoelectric Horn driver with boost converter, specifically designed for CO/Gas detector applications.

For memory features, the Flash Memory offers users the convenience of multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc.

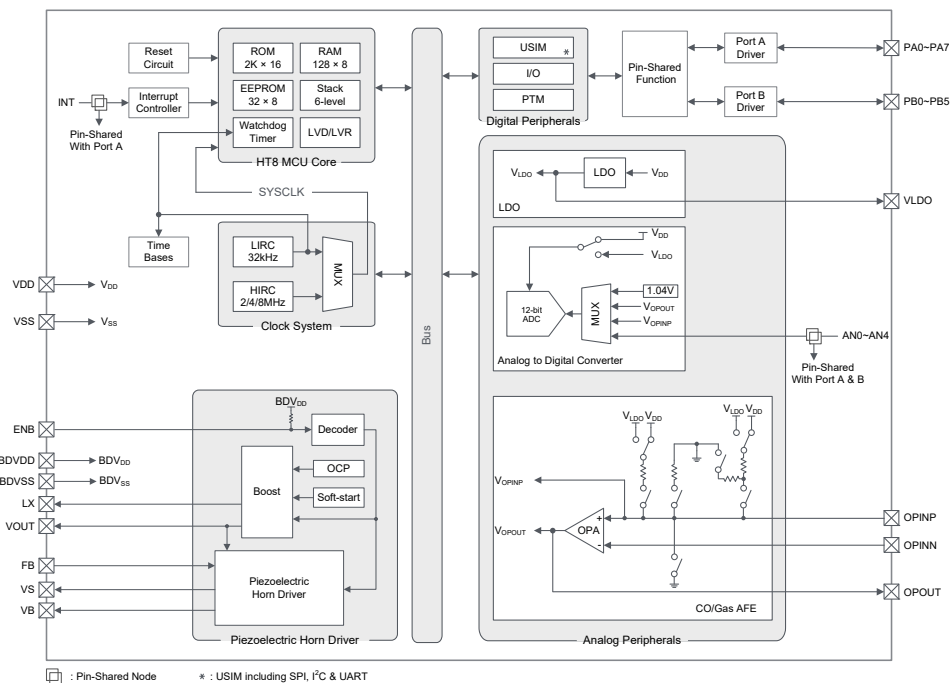
Analog features include a multi-channel 12-bit A/D converter and an OPA function. One extremely flexible Timer Module provides timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated SPI, I²C and UART interface functions, two popular interfaces which provide designers with a means of easy communication with external peripheral hardware. In addition, an internal LDO function provides various power options to the internal modules and external devices. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

A full choice of internal high and low oscillators 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 optimize microcontroller operation and minimize power consumption.

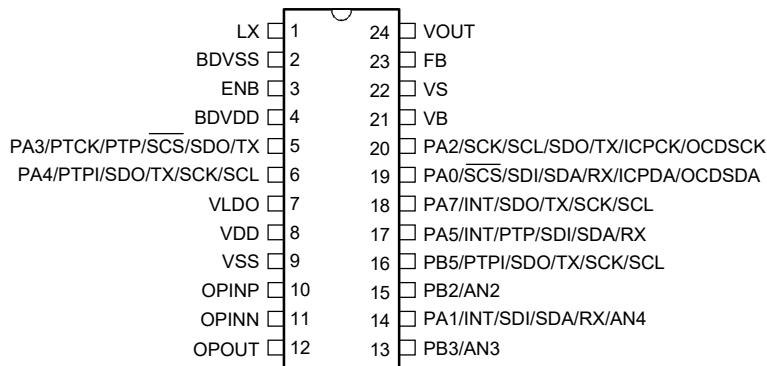
The device contains a piezoelectric horn driver with a 12V boost converter. The piezoelectric horn driver supports 2-pin or 3-pin buzzer.

The inclusion of flexible I/O programming features, Time Base functions along with many other features ensure that the device will find excellent use in CO/Gas detector applications.

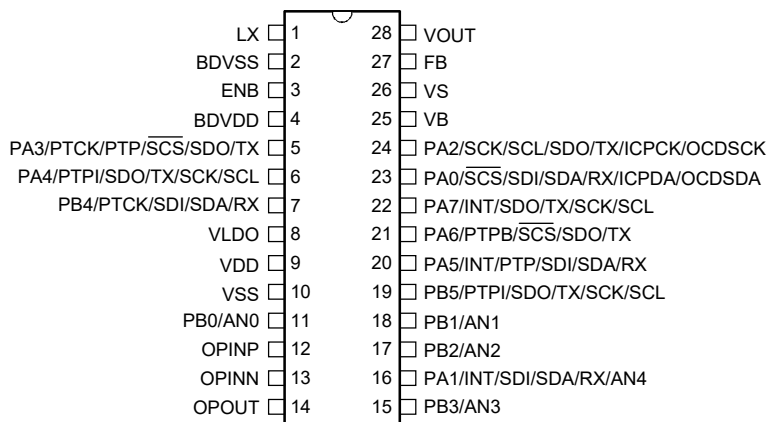
Block Diagram



Pin Assignment



BA45F6830/BA45V6830
24 SSOP-A



BA45F6830/BA45V6830
28 SSOP-A

- Note: 1. If the pin-shared pin functions have multiple outputs simultaneously, the desired pin-shared function is determined by the corresponding software control bits.
2. The OCDSDA and OCDSCK pins are supplied for the OCDS dedicated pins and as such only available for the BA45V6830 device which is the OCDS EV chip for the BA45F6830 device.
3. For the less pin count package type there will be unbounded pins which should be properly configured to avoid unwanted power consumption resulting from floating input conditions. Refer to the “Standby Current Considerations” and “Input/Output Ports” sections.

Pin Description

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet. As the Pin Description table shows the situation for the package with the most pins, not all pins in the table will be available on smaller package sizes.

| Pin Name | Function | OPT | I/T | O/T | Description |
|--|-------------------------|----------------------|-----|------|---|
| PA0/ $\overline{\text{SCS}}$ /SDI/SDA/ RX/ICPDA/OCSDA | PA0 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | $\overline{\text{SCS}}$ | PAS0 IFS1 | ST | CMOS | SPI slave select |
| | SDI | PAS0 IFS0 | ST | — | SPI serial data input |
| | SDA | PAS0 IFS0 | ST | NMOS | I ² C data line |
| | RX | PAS0 IFS0 | ST | — | UART RX serial data input |
| | ICPDA | — | ST | CMOS | ICP address/data |
| | OCSDA | — | ST | CMOS | OCDS data/address pin, for EV chip only |
| PA1/INT/SDI/SDA/ RX/AN4 | PA1 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | INT | PAS0 IFS1 | ST | — | External interrupt input |
| | SDI | PAS0 IFS0 | ST | — | SPI serial data input |
| | SDA | PAS0 IFS0 | ST | NMOS | I ² C data line |
| | RX | PAS0 IFS0 | ST | — | UART RX serial data input |
| | AN4 | PAS0 | AN | — | A/D Converter external input 4 |
| PA2/SCK/SCL/SDO/ TX/ICPCK/OCDSCK | PA2 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | SCK | PAS0 IFS0 | ST | CMOS | SPI serial clock |
| | SCL | PAS0 IFS0 | ST | NMOS | I ² C clock line |
| | SDO | PAS0 | — | CMOS | SPI serial data output |
| | TX | PAS0 | — | CMOS | UART TX serial data output |
| | ICPCK | — | ST | — | ICP clock |
| | OCDSCK | — | ST | — | OCDS clock pin, for EV chip only |
| PA3/PTCK/PTP/ SCS/SDO/TX | PA3 | PAPU PAWU PAS0 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTCK | PAS0 IFS0 | ST | — | PTM clock/capture input |
| | PTP | PAS0 | — | CMOS | PTM output |
| | $\overline{\text{SCS}}$ | PAS0 IFS1 | ST | CMOS | SPI slave select |
| | SDO | PAS0 | — | CMOS | SPI serial data output |
| | TX | PAS0 | — | CMOS | UART TX serial data output |

| Pin Name | Function | OPT | I/T | O/T | Description |
|---|-------------------------|----------------------|-----|------|---|
| PA4/PTPI/SDO/ TX/SCK/SCL | PA4 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTPI | PAS1 IFS0 | ST | — | PTM capture input |
| | SDO | PAS1 | — | CMOS | SPI serial data output |
| | TX | PAS1 | — | CMOS | UART TX serial data output |
| | SCK | PAS1 IFS0 | ST | CMOS | SPI serial clock |
| | SCL | PAS1 IFS0 | ST | NMOS | I ² C clock line |
| PA5/INT/PTP/ SDI/SDA/RX | PA5 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | INT | PAS1 IFS1 | ST | — | External interrupt input |
| | PTP | PAS1 | — | CMOS | PTM output |
| | SDI | PAS1 IFS0 | ST | — | SPI serial data input |
| | SDA | PAS1 IFS0 | ST | NMOS | I ² C data line |
| | RX | PAS1 IFS0 | ST | — | UART RX serial data input |
| PA6/PTPB/ $\overline{\text{SCS}}$ / SDO/TX | PA6 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | PTPB | PAS1 | — | CMOS | PTM inverted output |
| | $\overline{\text{SCS}}$ | PAS0 IFS1 | ST | CMOS | SPI slave select |
| | SDO | PAS1 | — | CMOS | SPI serial data output |
| | TX | PAS1 | — | CMOS | UART TX serial data output |
| PA7/INT/SDO/TX/ SCK/SCL | PA7 | PAPU PAWU PAS1 | ST | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
| | INT | PAS1 IFS1 | ST | — | External interrupt input |
| | SDO | PAS1 | — | CMOS | SPI serial data output |
| | TX | PAS1 | — | CMOS | UART TX serial data output |
| | SCK | PAS1 IFS0 | ST | CMOS | SPI serial clock |
| | SCL | PAS1 IFS0 | ST | NMOS | I ² C clock line |
| PB0/AN0 | PB0 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | AN0 | PBS0 | AN | — | A/D Converter external input 0 |
| PB1/AN1 | PB1 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | AN1 | PBS0 | AN | — | A/D Converter external input 1 |
| PB2/AN2 | PB2 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | AN2 | PBS0 | AN | — | A/D Converter external input 2 |

| Pin Name | Function | OPT | I/T | O/T | Description |
|-------------------------|----------|--------------|-----|------|--|
| PB3/AN3 | PB3 | PBPU PBS0 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | AN3 | PBS0 | AN | — | A/D Converter external input 3 |
| PB4/PTCK/SDI/SDA/RX | PB4 | PBPU PBS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTCK | PBS1 IFS0 | ST | — | PTM clock/capture input |
| | SDI | PBS1 IFS0 | ST | — | SPI serial data input |
| | SDA | PBS1 IFS0 | ST | NMOS | I ² C data line |
| PB5/PTPI/SDO/TX/SCK/SCL | RX | PBS1 IFS0 | ST | — | UART RX serial data input |
| | PB5 | PBPU PBS1 | ST | CMOS | General purpose I/O. Register enabled pull-up |
| | PTPI | PBS1 IFS0 | ST | — | PTM capture input |
| | SDO | PBS1 | — | CMOS | SPI serial data output |
| | TX | PBS1 | — | CMOS | UART TX serial data output |
| OPINP | OPINP | — | AN | — | OPA positive input |
| OPINN | OPINN | — | AN | — | OPA negative input |
| OPOUT | OPOUT | — | — | AN | OPA output |
| VLDO | VLDO | — | — | PWR | LDO output |
| VDD | VDD | — | PWR | — | Positive power supply, LDO input |
| VSS | VSS | — | PWR | — | Negative power supply, ground |
| BDVDD | BDVDD | — | PWR | — | Piezoelectric horn driver positive power supply |
| BDVSS | BDVSS | — | PWR | — | Piezoelectric horn driver negative power supply, ground |
| VOUT | VOUT | — | — | PWR | Boost converter output |
| LX | LX | — | — | PWR | Power switch output |
| ENB | ENB | — | ST | — | Enable pin with pull-high resistor, used to control boost converter and buzzer driver mode |
| FB | FB | — | ST | — | Feedback pin for self-driving(3-pin) buzzer driver |
| VS | VS | — | — | CMOS | Output for buzzer driver |
| VB | VB | — | — | CMOS | Complementary output for buzzer driver |

Legend: I/T: Input type;

O/T: Output type;

OPT: Optional by register option;

PWR: Power;

ST: Schmitt Trigger input;

CMOS: CMOS output;

NMOS: NMOS output;

AN: Analog signal.

Absolute Maximum Ratings

MCU 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..... | $-50^{\circ}C$ to $125^{\circ}C$ |
| Operating Temperature..... | $-40^{\circ}C$ to $85^{\circ}C$ |
| I_{OH} Total | $-80mA$ |
| I_{OL} Total | $80mA$ |
| Total Power Dissipation | $500mW$ |

Piezoelectric Horn Driver Absolute Maximum Ratings

| | |
|----------------------|-------------------------|
| Output Voltage..... | $V_{SS}-0.3V$ to $+18V$ |
| I_{OL} Total | $200mA$ |
| I_{OH} Total | $-200mA$ |

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

$T_a=-40^{\circ}C\sim 85^{\circ}C$

| Symbol | Parameter | Test Conditions | Min. | Typ. | Max. | Unit |
|----------|--------------------------|--------------------------|------|------|------|------|
| V_{DD} | Operating Voltage – HIRC | $f_{SYS}=f_{HIRC}=2MHz$ | 2.2 | — | 5.5 | V |
| | | $f_{SYS}=f_{HIRC}=4MHz$ | 2.2 | — | 5.5 | |
| | | $f_{SYS}=f_{HIRC}=8MHz$ | 2.2 | — | 5.5 | |
| | Operating Voltage – LIRC | $f_{SYS}=f_{LIRC}=32kHz$ | 2.2 | — | 5.5 | V |

Standby Current Characteristics

$T_a=25^{\circ}C$, unless otherwise specified

| Symbol | Standby Mode | Test Conditions | | Min. | Typ. | Max. | Max. @85°C | Unit |
|-----------|--------------|-----------------|------------|------|------|------|---------------|---------|
| | | V_{DD} | Conditions | | | | | |
| I_{STB} | SLEEP Mode | 2.2V | WDT off | — | 0.2 | 0.6 | 0.7 | μA |
| | | 3V | | — | 0.2 | 0.8 | 1.0 | |
| | | 5V | | — | 0.5 | 1.0 | 1.2 | |
| | | 2.2V | WDT on | — | 1.2 | 2.4 | 2.9 | μA |
| | | 3V | | — | 1.5 | 3.0 | 3.6 | |
| | | 5V | | — | 3 | 5 | 6 | |

| Symbol | Standby Mode | Test Conditions | | Min. | Typ. | Max. | Max. @85°C | Unit |
|------------------|-------------------|---|---|------|------|------|---------------|------|
| | | V _{DD} | Conditions | | | | | |
| I _{STB} | IDLE0 Mode | 2.2V | f _{SUB} on | — | 2.4 | 4.0 | 4.8 | μA |
| | | 3V | | — | 3 | 5 | 6 | |
| | | 5V | | — | 5 | 10 | 12 | |
| | IDLE1 Mode – HIRC | 2.2V | f _{SUB} on, f _{SYS} =2MHz | — | 0.03 | 0.06 | 0.08 | mA |
| | | 3V | | — | 0.07 | 0.14 | 0.16 | |
| | | 5V | | — | 0.13 | 0.26 | 0.30 | |
| | | 2.2V | f _{SUB} on, f _{SYS} =4MHz | — | 0.05 | 0.07 | 0.09 | mA |
| | | 3V | | — | 0.11 | 0.22 | 0.25 | |
| | | 5V | | — | 0.21 | 0.42 | 0.50 | |
| | 2.2V | f _{SUB} on, f _{SYS} =8MHz | — | 0.15 | 0.25 | 0.30 | mA | |
| | 3 | | — | 0.30 | 0.36 | 0.40 | | |
| | 5V | | — | 0.55 | 0.74 | 0.85 | | |

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.
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.

Operating Current Characteristics

T_a=25°C

| Symbol | Operating Mode | Test Conditions | | Min. | Typ. | Max. | Unit |
|-----------------|------------------|------------------------|-------------------------|------|------|------|------|
| | | V _{DD} | Conditions | | | | |
| I _{DD} | SLOW Mode – LIRC | 2.2V | f _{SYS} =32kHz | — | 8 | 6 | μA |
| | | 3V | | — | 10 | 20 | |
| | | 5V | | — | 30 | 50 | |
| | FAST Mode – HIRC | 2.2V | f _{SYS} =2MHz | — | 0.1 | 0.2 | mA |
| | | 3V | | — | 0.2 | 0.3 | |
| | | 5V | | — | 0.4 | 0.6 | |
| | | 2.2V | f _{SYS} =4MHz | — | 0.3 | 0.5 | mA |
| | | 3V | | — | 0.4 | 0.6 | |
| | | 5V | | — | 0.8 | 1.2 | |
| | 2.2V | 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.
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.

A.C. Characteristics

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

High Speed Internal Oscillator – HIRC – Frequency Accuracy

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

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|-------------------|------------------------------------|-----------------|--------------|-------|------|-------|------|
| | | V _{DD} | Temperature | | | | |
| f _{HIRC} | 2MHz Writer Trimmed HIRC Frequency | 3V/5V | 25°C | -1% | 2 | +1% | MHz |
| | | | -40°C ~ 85°C | -4% | 2 | +4% | |
| | | 2.2V~5.5V | 25°C | -6% | 2 | +6% | |
| | | | -40°C ~ 85°C | -8% | 2 | +8% | |
| | 4MHz Writer Trimmed HIRC Frequency | 3V/5V | 25°C | -1% | 4 | +1% | MHz |
| | | | -40°C ~ 85°C | -2% | 4 | +2% | |
| | | 2.2V~5.5V | 25°C | -2.5% | 4 | +2.5% | |
| | | | -40°C ~ 85°C | -3% | 4 | +3% | |
| | 8MHz Writer Trimmed HIRC Frequency | 3V/5V | 25°C | -1% | 8 | +1% | MHz |
| | | | -40°C ~ 85°C | -2% | 8 | +2% | |
| | | 2.7V~5.5V | 25°C | -2.5% | 8 | +2.5% | |
| | | | -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.

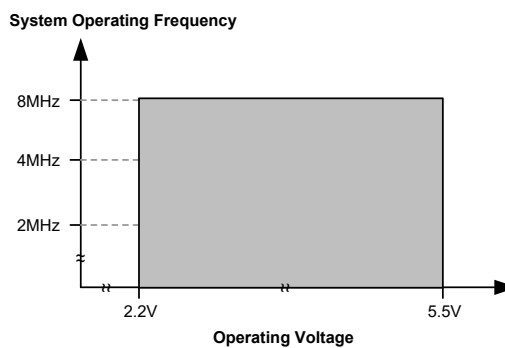
2. The row below the 3V/5V trim voltage row is provided to show the values for the full V_{DD} range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 2.2V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

3. The minimum and maximum tolerance values provided in the table are only for the frequency at which the writer trims the HIRC oscillator. After trimming at this chosen specific frequency any change in HIRC oscillator frequency using the oscillator register control bits by the application program will give a frequency tolerance to within ±20%.

Low Speed Internal Oscillator Characteristics – LIRC

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|--------------------|----------------------|-----------------|--------------|------|------|------|------|
| | | V _{DD} | Temperature | | | | |
| f _{LIRC} | Oscillator Frequency | 2.2V~5.5V | 25°C | -5% | 32 | +5% | kHz |
| | | | -40°C ~ 85°C | -10% | 32 | +10% | |
| t _{START} | LIRC Start Up Time | — | 25°C | — | — | 100 | 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 condition where f _{sys} is off | — | f _{sys} =f _H ~f _H /64, f _H =f _{HIRC} | — | 16 | — | t _{HIRC} |
| | | — | f _{sys} =f _{SUB} =f _{LIRC} | — | 2 | — | t _{LIRC} |
| | System Start-up Time Wake-up from condition where f _{sys} is on | — | f _{sys} =f _H ~f _H /64, f _H =f _{HIRC} | — | 2 | — | t _H |
| | | — | f _{sys} =f _{SUB} =f _{LIRC} | — | 2 | — | t _{SUB} |
| | System Speed Switch Time FAST to SLOW Mode or SLOW to FAST Mode | — | f _{HIRC} switches from off → on | — | 16 | — | t _{HIRC} |
| t _{RSTD} | System Reset Delay Time Reset source from Power-on Reset or LVR Hardware Reset | — | RR _{POR} =5V/ms | 42 | 48 | 54 | ms |
| | System Reset Delay Time WDTC Software Reset | — | — | | | | |
| | System Reset Delay Time Reset source from WDT Overflow | — | — | 14 | 16 | 18 | ms |
| t _{SRESET} | Minimum Software Reset Width to Reset | — | — | 45 | 90 | 120 | μs |

- 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 symbol 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=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit | |
|-----------------|---|-----------------|--|--|------|--------------------|------|---|
| | | V _{DD} | Conditions | | | | | |
| V _{IL} | Input Low Voltage for I/O Ports | 5V | — | 0 | — | 1.5 | V | |
| | | — | — | 0 | — | 0.2V _{DD} | | |
| V _{IH} | Input High Voltage for I/O Ports | 5V | — | 3.5 | — | 5.0 | V | |
| | | — | — | 0.8V _{DD} | — | V _{DD} | | |
| I _{OL} | Sink Current for I/O Ports | 3V | V _{OL} =0.1V _{DD} | 16 | 32 | — | mA | |
| | | 5V | | 32 | 65 | — | | |
| I _{OH} | Source Current for I/O Ports | 3V | V _{OH} =0.9V _{DD} , SLEDC[m+1:m]=00B, (m=0, 2, 4 or 6) | -0.7 | -1.5 | — | mA | |
| | | 5V | | -1.5 | -2.9 | — | | |
| | | 3V | | V _{OH} =0.9V _{DD} , SLEDC[m+1:m]=01B, (m=0, 2, 4 or 6) | -1.3 | -2.5 | | — |
| | | 5V | | | -2.5 | -5.1 | | — |
| | | 3V | | V _{OH} =0.9V _{DD} , SLEDC[m+1:m]=10B, (m=0, 2, 4 or 6) | -1.8 | -3.6 | | — |
| | | 5V | | | -3.6 | -7.3 | | — |
| | | 3V | | V _{OH} =0.9V _{DD} , SLEDC[m+1:m]=11B, (m=0, 2, 4 or 6) | -4 | -8 | | — |
| 5V | -8 | -16 | — | | | | | |
| R _{PH} | Pull-high Resistance for I/O Ports (Note) | 3V | — | 20 | 60 | 100 | kΩ | |
| | | 5V | — | 10 | 30 | 50 | | |

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|--------------------|--|-----------------|--|------------------|------|------|--------------------|
| | | V _{DD} | Conditions | | | | |
| I _{LEAK} | Input Leakage Current for I/O Ports | 5V | V _{IN} =V _{DD} or V _{IN} =V _{SS} | — | — | ±1 | μA |
| t _{TPI} | PTPI Capture Input Minimum Pulse Width | — | — | 0.3 | — | — | μs |
| t _{TCK} | PTCK Clock Input Minimum Pulse Width | — | — | 0.3 | — | — | μs |
| t _{INT} | Interrupt Input Pin Minimum Pulse Width | — | — | 10 | — | — | μs |
| f _{TMCLK} | PTM Maximum Timer Clock Source Frequency | 5V | — | — | — | 1 | f _{sys} |
| t _{CPW} | PTM minimum capture pulse width | — | — | t _{CPW} | — | — | t _{TMCLK} |

Note: 1. 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.

2. If PTCAPTS=0, then t_{CPW}=max(2×t_{TMCLK}, t_{TPI})
 If PTCAPTS=1, then t_{CPW}=max(2×t_{TMCLK}, t_{TCK})
 Ex1: If PTCAPTS=0, f_{TMCLK}=16MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.125μs, 0.3μs)=0.3μs
 Ex2: If PTCAPTS=1, f_{TMCLK}=16MHz, t_{TCK}=0.3μs, then t_{CPW}=max(0.125μs, 0.3μs)=0.3μs
 Ex3: If PTCAPTS=0, f_{TMCLK}=8MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.25μs, 0.3μs)=0.3μs
 Ex4: If PTCAPTS=0, f_{TMCLK}=4MHz, t_{TPI}=0.3μs, then t_{CPW}=max(0.5μs, 0.3μs)=0.5μs
 Where t_{TMCLK}=1/f_{TMCLK}

Memory Characteristics

Ta=-40°C~85°C, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|---|---|-----------------|------------|--------------------|------|--------------------|------|
| | | V _{DD} | Conditions | | | | |
| V _{RW} | V _{DD} for Read/Write | — | — | V _{DDmin} | — | V _{DDmax} | V |
| Flash Program / Data EEPROM Memory | | | | | | | |
| t _{DEW} | Erase/Write Cycle Time – Flash Program Memory | — | — | — | 2 | 3 | ms |
| | Write Cycle Time – Data EEPROM Memory | — | — | — | 4 | 6 | ms |
| I _{DDPGM} | Programming/Erase Current on V _{DD} | — | — | — | — | 5.0 | mA |
| E _P | Cell Endurance – Flash Program Memory | — | — | 10K | — | — | E/W |
| | Cell Endurance – Data EEPROM Memory | — | — | 100K | — | — | E/W |
| t _{RETD} | ROM Data Retention Time | — | Ta=25°C | — | 40 | — | Year |
| RAM Data Memory | | | | | | | |
| V _{DR} | RAM Data Retention Voltage | — | — | 1.0 | — | — | V |

Note: "E/W" means Erase/Write times.

LVR/LVD Electrical Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|------------------|---------------------------|-----------------|-----------------------------|------|------|------|------|
| | | V _{DD} | Conditions | | | | |
| V _{DD} | Operating Voltage | — | — | 2.2 | — | 5.5 | V |
| V _{LVR} | Low Voltage Reset Voltage | — | LVR enable, voltage is 2.1V | -5% | 2.1 | +6% | V |

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|------------------------|--|-----------------|--|------|------|------|------|
| | | V _{DD} | Conditions | | | | |
| V _{LVD} | Low Voltage Detection Voltage | — | LVD enable, voltage select 2.2V | -5% | 2.2 | +6% | V |
| | | — | 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 | | |
| I _{LVR/LVDBG} | Operating Current | 3V | LVD enable, LVR enable, VBGEN=0 | — | 15 | 20 | μA |
| | | 5V | VBGEN=0 | — | 20 | 25 | |
| | | 3V | LVD enable, LVR enable, VBGEN=1 | — | — | 150 | μA |
| | | 5V | VBGEN=1 | — | 180 | 200 | |
| I _{LVR} | Additional Current for LVR Enable | — | LVD disable, VBGEN=0 | — | 20 | 25 | μA |
| I _{LVD} | Additional Current for LVD Enable | — | LVR disable, VBGEN=0 | — | 20 | 25 | μA |
| t _{LVDS} | LVDO Stable Time | — | For LVR enable, VBGEN=0, LVD off → on | — | — | 15 | μs |
| | | — | For LVR disable, VBGEN=0, LVD off → on | — | — | 150 | μs |
| t _{LVR} | Minimum Low Voltage Width to Reset | — | — | 120 | 240 | 480 | μs |
| t _{LVD} | Minimum Low Voltage Width to Interrupt | — | — | 60 | 120 | 240 | μs |

A/D Converter Electrical Characteristics

T_a=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|--------------------|---|-----------------|--|------|------|------------------|-------------------|
| | | V _{DD} | Conditions | | | | |
| V _{DD} | A/D Converter Operating Voltage | — | — | 2.2 | — | 5.5 | V |
| V _{ADI} | A/D Converter Input Voltage | — | — | 0 | — | V _{REF} | V |
| V _{REF} | A/D Converter Reference Voltage | — | — | 2 | — | V _{DD} | V |
| DNL | A/D Differential Non-linearity | 3V | V _{REF} =V _{DD} , t _{ADCK} =0.5μs | -3 | — | +3 | LSB |
| | | 5V | | | | | |
| | | 3V | V _{REF} =V _{DD} , t _{ADCK} =8μs | | | | |
| | | 5V | | | | | |
| INL | A/D Integral Non-linearity | 3V | V _{REF} =V _{DD} , t _{ADCK} =0.5μs | -4 | — | +4 | LSB |
| | | 5V | | | | | |
| | | 3V | V _{REF} =V _{DD} , t _{ADCK} =8μs | | | | |
| | | 5V | | | | | |
| I _{ADC} | Additional Current Consumption for A/D Converter Enable | 3V | No load, t _{ADCK} =0.5μs | — | 1 | 2 | mA |
| | | 5V | | — | 1.5 | 3.0 | |
| t _{ADCK} | A/D Converter Clock Period | — | — | 0.5 | — | 10.0 | μs |
| t _{ADS} | A/D Converter Sampling Time | — | — | — | 4 | — | t _{ADCK} |
| t _{ADC} | Conversion Time (Including A/D Sample and Hold Time) | — | — | — | 16 | — | t _{ADCK} |
| t _{ON2ST} | A/D Converter On-to-Start Time | — | — | 4 | — | — | μs |

Reference Voltage Electrical Characteristics

Ta=25°C

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|------------------|---------------------------------------|-----------------|-------------------------|------|------|------|------|
| | | V _{DD} | Conditions | | | | |
| V _{BG} | Bandgap Reference Voltage | — | — | -5% | 1.04 | +5% | V |
| I _{BG} | Additional Current for Bandgap Enable | — | LVR disable, LVD diable | — | — | 180 | μA |
| t _{BGS} | V _{BG} Start Stable Time | — | No load | — | — | 150 | μs |

Note: The V_{BG} voltage is used as A/D converter internal input.

OPA Electrical Characteristics

V_{DD}=5V, Ta=25°C, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|------------------|-----------------------------------|-----------------|--|---------------------|------|----------------------|------|
| | | V _{DD} | Conditions | | | | |
| V _{DD} | OPA Operating Voltage | — | — | 2.2 | — | 5.5 | V |
| I _{OPA} | Additional Current for OPA Enable | — | No load, OPBW[1:0]=00B | — | 3.0 | 5.0 | μA |
| | | | No load, OPBW[1:0]=01B | — | 10 | 16 | |
| | | | No load, OPBW[1:0]=10B | — | 80 | 128 | |
| | | | No load, OPBW[1:0]=11B | — | 200 | 320 | |
| V _{OS} | Input Offset Voltage | — | Without calibration, OPOF[5:0]=100000B | -15 | — | +15 | mV |
| | | | With calibration | -4 | — | +4 | |
| I _{OS} | Input Offset Current | — | V _{IN} =(1/2)V _{CM} | — | 1 | 10 | nA |
| V _{CM} | Common Mode Voltage Range | — | OPBW[1:0]=00B/01B/10B/11B | V _{SS} | — | V _{DD} -1.4 | V |
| V _{OR} | Maximum Output Voltage Range | — | OPBW[1:0]=00B/01B, no load | V _{SS} +80 | — | V _{DD} -120 | mV |
| | | | OPBW[1:0]=10B/11B, no load | V _{SS} +40 | — | V _{DD} -60 | |
| I _{SC} | Output Short Circuit Current | — | R _{LOAD} =5.1Ω, OPBW[1:0]=00B/01B | ±6 | ±12 | — | mA |
| | | | R _{LOAD} =5.1Ω, OPBW[1:0]=10B/11B | ±10 | ±20 | — | |
| PSRR | Power Supply Rejection Ratio | — | OPBW[1:0]=00B/01B/10B/11B | 50 | 70 | — | dB |
| CMRR | Common Mode Rejection Ratio | — | OPBW[1:0]=00B/01B/10B/11B | 50 | 80 | — | dB |
| A _{OL} | Open Loop Gain | — | OPBW[1:0]=00B/01B/10B/11B | 60 | 80 | — | dB |
| SR | Slew Rate | — | R _{LOAD} =1MΩ, C _{LOAD} =60pF, OPBW[1:0]=00B | 0.5 | 1.5 | — | V/ms |
| | | | R _{LOAD} =1MΩ, C _{LOAD} =60pF, OPBW[1:0]=01B | 5 | 15 | — | |
| | | | R _{LOAD} =1MΩ, C _{LOAD} =60pF, OPBW[1:0]=10B | 180 | 500 | — | |
| | | | R _{LOAD} =1MΩ, C _{LOAD} =60pF, OPBW[1:0]=11B | 600 | 1800 | — | |
| GBW | Gain Bandwidth | — | R _{LOAD} =1MΩ, C _{LOAD} =60pF, OPBW[1:0]=00B | 2.5 | 5.0 | — | kHz |
| | | | R _{LOAD} =1MΩ, C _{LOAD} =60pF, OPBW[1:0]=01B | 20 | 40 | — | |
| | | | R _{LOAD} =1MΩ, C _{LOAD} =60pF, OPBW[1:0]=10B | 400 | 600 | — | |
| | | | R _{LOAD} =1MΩ, C _{LOAD} =60pF, OPBW[1:0]=11B | 1300 | 2000 | — | |

Note: These parameters are characterized but not tested.

$V_{DD}=2.2V\sim 5.5V$, $T_a=25^\circ C$, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|-----------|-----------------------------------|-----------------|---|-------------|------------|--------------|---------|
| | | V_{DD} | Conditions | | | | |
| I_{OPA} | Additional Current for OPA Enable | — | No load, OPBW[1:0]=00B | — | 2.5 | 4.0 | μA |
| | | | No load, OPBW[1:0]=01B | — | 10 | 16 | |
| | | | No load, OPBW[1:0]=10B | — | 80 | 128 | |
| | | | No load, OPBW[1:0]=11B | — | 200 | 320 | |
| V_{OS} | Input Offset Voltage | — | Without calibration, OPOF[5:0]=100000B | -15 | — | +15 | mV |
| | | | With calibration | -4 | — | +4 | |
| I_{OS} | Input Offset Current | — | $V_{IN}=(1/2)V_{CM}$ | — | 1 | 10 | nA |
| V_{CM} | Common Mode Voltage Range | — | OPBW[1:0]=00B/01B/10B/11B | V_{SS} | — | $V_{DD}-1.4$ | V |
| V_{OR} | Maximum Output Voltage Range | — | OPBW[1:0]=00B/01B, no load | $V_{SS}+80$ | — | $V_{DD}-120$ | mV |
| | | | OPBW[1:0]=10B/11B, no load | $V_{SS}+40$ | — | $V_{DD}-60$ | |
| I_{SC} | Output Short Circuit Current | — | $R_{LOAD}=5.1\Omega$, OPBW[1:0]=00B/01B | ± 1.2 | ± 12.0 | — | mA |
| | | | $R_{LOAD}=5.1\Omega$, OPBW[1:0]=10B/11B | ± 2 | ± 20 | — | |
| PSRR | Power Supply Rejection Ratio | — | OPBW[1:0]=00B/01B/10B/11B | 50 | 70 | — | dB |
| CMRR | Common Mode Rejection Ratio | — | OPBW[1:0]=00B/01B/10B/11B | 50 | 80 | — | dB |
| A_{OL} | Open Loop Gain | — | OPBW[1:0]=00B/01B/10B/11B | 60 | 80 | — | dB |
| SR | Slew Rate | — | $R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, OPBW[1:0]=00B | 0.5 | 1.5 | — | V/ms |
| | | | $R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, OPBW[1:0]=01B | 5 | 15 | — | |
| | | | $R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, OPBW[1:0]=10B | 180 | 500 | — | |
| | | | $R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, OPBW[1:0]=11B | 600 | 1800 | — | |
| GBW | Gain Bandwidth | — | $R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, OPBW[1:0]=00B | 2 | 5 | — | kHz |
| | | | $R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, OPBW[1:0]=01B | 15 | 40 | — | |
| | | | $R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, OPBW[1:0]=10B | 250 | 600 | — | |
| | | | $R_{LOAD}=1M\Omega$, $C_{LOAD}=60pF$, OPBW[1:0]=11B | 800 | 2000 | — | |

Note: These parameters are characterized but not tested.

LDO Electrical Characteristics

$V_{IN}=V_{OUT}+0.3V$, $C_{LOAD}=4.7\mu F$, $T_a=25^\circ C$, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|-----------|-------------------------|-----------------|--|------|-----------|-----------|---------------|
| | | V_{DD} | Conditions | | | | |
| V_{IN} | Input Voltage | — | — | 2.5 | — | 5.5 | V |
| V_{OUT} | Output Voltage | — | $I_{LOAD}=1mA$, $V_{OUT}=2.2V$ | -3% | 2.2 | +3% | V |
| | | | $I_{LOAD}=1mA$, $V_{OUT}=3.0V$ | -2% | 3.0 | +2% | |
| I_{OUT} | Output Current | — | $V_{IN}=2.5V$, $\Delta V_{OUT}=0.1V$, $V_{OUT}=2.2V$ | 10 | — | — | mA |
| | | | $V_{IN}=3.4V$, $\Delta V_{OUT}=0.1V$, $V_{OUT}=3.0V$ | 30 | — | — | |
| I_Q | Quiescent Current | 5V | No load | — | — | 5 | μA |
| TC | Temperature Coefficient | — | $T_a=-40^\circ C\sim 85^\circ C$, $I_{LOAD}=10mA$ | — | ± 1.5 | ± 2.0 | $mV/^\circ C$ |

Piezoelectric Horn Driver Electrical Characteristics

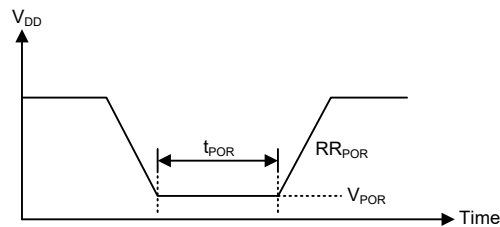
BDV_{DD}=V_{DD}=3V, V_{OUT}=12V, Ta=25°C, unless otherwise specified

| Symbol | Parameter | Test Conditions | | Min. | Typ. | Max. | Unit |
|----------------------------------|-----------------------------------|-------------------|---|----------------------|-----------------|----------------------|------|
| | | BDV _{DD} | Conditions | | | | |
| Supply Voltage | | | | | | | |
| BDV _{DD} | Input Voltage Range | — | — | — | V _{DD} | — | V |
| I _{IN} | Supply Current | — | No load, boost converter enable, external-driving buzzer mode, f=4kHz | — | 2 | 4 | mA |
| I _{SHDN} | Shutdown Current | — | V _{ENB} =BDV _{DD} | — | 0.5 | 1.0 | μA |
| Boost Converter | | | | | | | |
| V _{OUT} | Output Voltage Range | — | — | 10.8 | 12.0 | 13.2 | V |
| I _{OCF} | Over Current Protection Current | — | — | — | 0.95 | — | A |
| f _{SW} | Switching Frequency | — | V _{OUT} =12V | 0.8 | 1.0 | 1.2 | MHz |
| Piezoelectric Horn Driver | | | | | | | |
| V _{IH} | High-Level Input Voltage | — | FB Pin | 0.7V _{OUT} | — | V _{OUT} | V |
| | | | ENB Pin | 0.7BDV _{DD} | — | BDV _{DD} | V |
| V _{IL} | Low-Level Input Voltage | — | FB Pin | 0 | — | 0.3V _{OUT} | V |
| | | | ENB Pin | 0 | — | 0.3BDV _{DD} | V |
| I _{OH} | Source Current for VB and VS Pins | — | V _{OUT} =12V, V _{OH} =0.9V _{OUT} | -120 | -150 | — | mA |
| I _{OL} | Sink Current for VB and VS Pins | — | V _{OUT} =12V, V _{OL} =0.1V _{OUT} | 120 | 150 | — | mA |
| I _{LEAK} | Input Leakage Current | — | V _{FB} =V _{OUT} or =V _{SS} | — | — | ±0.1 | μA |
| | | | V _{ENB} =BDV _{DD} | — | — | -0.1 | μA |
| R _{PH} | Pull-high Resistance | — | ENB Pin | 0.7 | 1.0 | 1.3 | MΩ |

Power-on Reset Characteristics

Ta=25°C

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



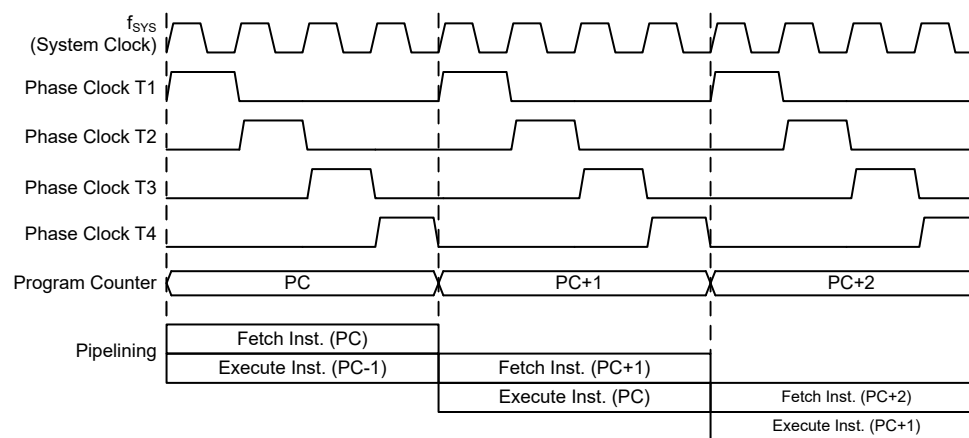
System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The range of the device take advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one or two cycles for most of the standard or extended instructions respectively, with the exception of 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 low-cost, high-volume production for controller applications.

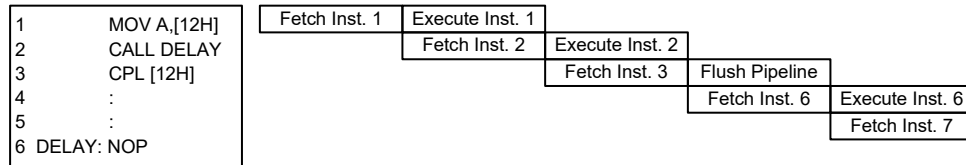
Clocking and Pipelining

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

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



System Clocking and Pipelining



Instruction Fetching

Program Counter

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

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

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

Program Counter

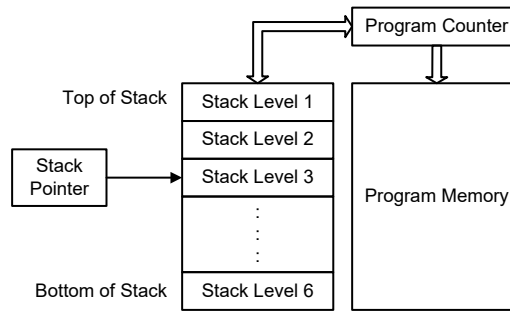
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available for manipulation, the jumps are limited to the present page of memory that is 256 locations. When such program jumps are executed it should also be noted that a dummy cycle will be inserted. Manipulating the PCL register may cause program branching, so an extra cycle is needed to pre-fetch.

Stack

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

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

If the stack is overflow, the first Program Counter save in the stack will be lost.



Arithmetic and Logic Unit – ALU

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

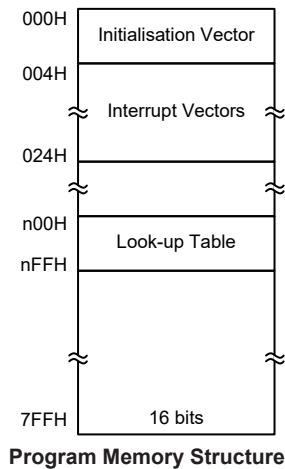
- Arithmetic operations:
ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,
LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:
AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,
LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation:
RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,
LRR, LRRCA, LRRCA, LRRCA, 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,
LSNZ, LSZ, LSZA, LSIZ, LSIZA, LSDZ, LSDZA

Flash Program Memory

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

Structure

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



Special Vectors

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

Look-up Table

Any location within the Program Memory can be defined as a look-up table where programmers can store fixed data. To use the look-up table, the table pointer must first be setup by placing the address of the look up data to be retrieved in the table pointer register, TBLP and TBHP. These registers define the total address of the look-up table.

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

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

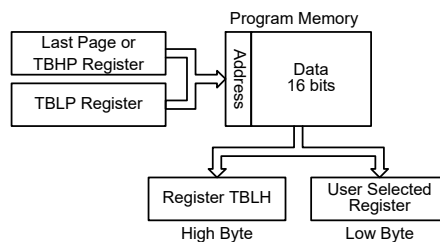


Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is “0700H” which refers to the start address of the last page within the 2K words Program Memory of the device. The table pointer low byte register is setup here to have an initial value of “06H”. This will ensure that the first data read from the data table will be at the Program Memory address “0706H” 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 table pointer - note that this address is referenced
mov tblp,a       ; to the last page or the page that tbhp pointed
mov a,07h         ; initialise high table pointer
mov tbhp,a       ; it is not necessary to set tbhp if executing tabrdl or ltabrdl
:
:
tabrd tempreg1    ; transfers value in table referenced by table pointer
                  ; data at program memory address "0706H" 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 "0705H" transferred to tempreg2 and
                  ; TBLH
                  ; in this example the data "1AH" is transferred to tempreg1 and
                  ; data "0FH" to tempreg2
                  ; the value "00H" will be transferred to the high byte register TBLH
:
org 0700h        ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
  
```

In Circuit Programming – ICP

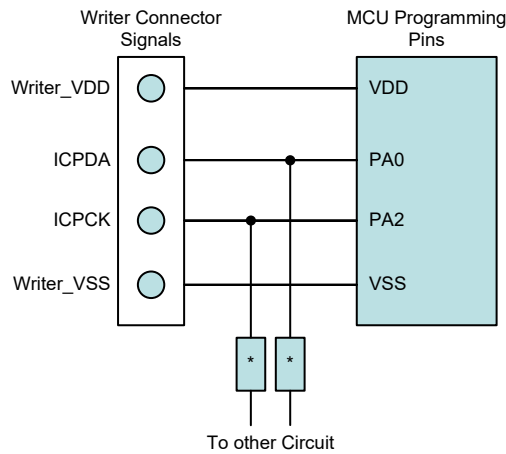
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device.

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

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

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

During the programming process, the user must take care of the ICPDA and ICPCK pins for data and clock programming purposes to ensure that no other outputs are connected to these two pins.



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

On-Chip Debug Support – OCDS

There is an EV chip named BA45V6830 which is used to emulate the BA45F6830 device. The EV chip device also provides an “On-Chip Debug” function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for “On-Chip Debug” function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCSDA 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 OCSDA and OCDSCK pins in the device will have no effect in the EV chip.

However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named “Holtek e-Link for 8-bit MCU OCDS User’s Guide”.

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

Data Memory

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

Categorized into two types, the first of these is an area of RAM, known as the Special Function Data Memory. 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 known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

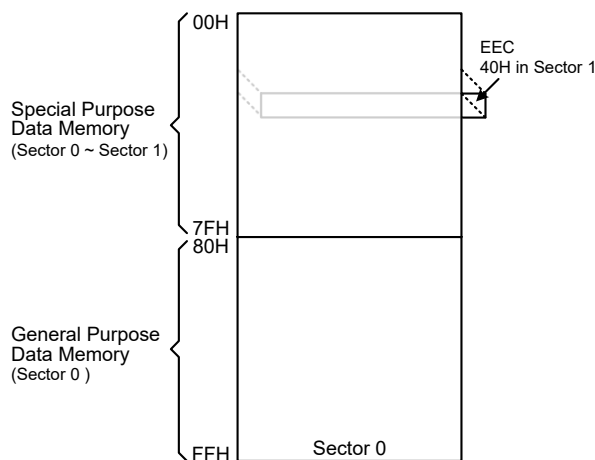
Switching between the different Data Memory sectors can be achieved by properly setting the Memory Pointers to correct value if using the indirect addressing method.

Structure

The Data Memory is subdivided into two sectors, all of which are implemented in 8-bit wide RAM. The Special Purpose Data Memory registers are accessible in all sectors, with the exception of the EEC register at address 40H, which is only accessible in Sector 1. 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.

| Special Purpose Data Memory | General Purpose Data Memory | |
|-----------------------------|-----------------------------|-----------------|
| Located Sectors | Capacity | Sector: Address |
| 0, 1 | 128×8 | 0: 80H~FFH |

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. 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 9 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 programing for both reading and writing operations. By using the bit operation instructions individual bits can be set or reset under program control giving the user a large range of flexibility for bit manipulation in the Data Memory.

Special Purpose Data Memory

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

| Sector 0 | | Sector 1 | Sector 0 | | Sector 1 |
|----------|--------|----------|----------|------------------|----------|
| 00H | IAR0 | | 40H | | EEC |
| 01H | MP0 | | 41H | PAS0 | |
| 02H | IAR1 | | 42H | PAS1 | |
| 03H | MP1L | | 43H | PBS0 | |
| 04H | MP1H | | 44H | PBS1 | |
| 05H | ACC | | 45H | IFS0 | |
| 06H | PCL | | 46H | IFS1 | |
| 07H | TBLP | | 47H | SIMC0 | |
| 08H | TBLH | | 48H | SIMC1/UUCR1 | |
| 09H | TBHP | | 49H | SIMC2/SIMA/UUCR2 | |
| 0AH | STATUS | | 4AH | SIMD/UTXR_RXR | |
| 0BH | | | 4BH | SIMTOC/UBRG | |
| 0CH | IAR2 | | 4CH | UUSR | |
| 0DH | MP2L | | 4DH | | |
| 0EH | MP2H | | 4EH | | |
| 0FH | RSTFC | | 4FH | | |
| 10H | TB0C | | 50H | | |
| 11H | TB1C | | 51H | | |
| 12H | SCC | | 52H | | |
| 13H | HIRCC | | 53H | | |
| 14H | PA | | 54H | | |
| 15H | PAC | | 55H | | |
| 16H | PAPU | | 56H | | |
| 17H | PAWU | | 57H | | |
| 18H | PB | | 58H | | |
| 19H | PBC | | 59H | | |
| 1AH | PBPU | | 5AH | | |
| 1BH | SLEDC | | 5BH | | |
| 1CH | | | 5CH | | |
| 1DH | PSCR | | 5DH | | |
| 1EH | LVDC | | 5EH | | |
| 1FH | PTMC0 | | 5FH | | |
| 20H | PTMC1 | | 60H | | |
| 21H | PTMDL | | 61H | | |
| 22H | PTMDH | | 62H | | |
| 23H | PTMAL | | 63H | | |
| 24H | PTMAH | | 64H | | |
| 25H | PTMRPL | | 65H | | |
| 26H | PTMRPH | | 66H | | |
| 27H | | | 67H | | |
| 28H | | | 68H | | |
| 29H | | | 69H | | |
| 2AH | | | 6AH | | |
| 2BH | | | 6BH | | |
| 2CH | REGC | | 6CH | | |
| 2DH | SADOL | | 6DH | | |
| 2EH | SADOH | | 6EH | | |
| 2FH | SADC0 | | 6FH | | |
| 30H | SADC1 | | 70H | | |
| 31H | OPSW | | 71H | | |
| 32H | OPC | | 72H | | |
| 33H | OPVOS | | 73H | | |
| 34H | | | 74H | | |
| 35H | | | 75H | | |
| 36H | | | 76H | | |
| 37H | | | 77H | | |
| 38H | | | 78H | | |
| 39H | INTEG | | 79H | | |
| 3AH | INTC0 | | 7AH | | |
| 3BH | INTC1 | | 7BH | | |
| 3CH | INTC2 | | 7CH | | |
| 3DH | WDTC | | 7DH | | |
| 3EH | EEA | | 7EH | | |
| 3FH | EED | | 7FH | | |

□ : Unused, read as 00H

Special Purpose Data Memory

Special Function Register Description

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

Indirect Addressing Registers – IAR0, IAR1, 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 the MP1L/MP1H register pair and IAR2 register together with the 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 will return a result of “00H” and writing to the registers will result in no operation.

Memory Pointers – MP0, MP1L, MP1H, MP2L, MP2H

Five Memory Pointers, known as MP0, MP1L, MP1H, MP2L, 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 sectors according to the corresponding MP1H or MP2H register. Direct Addressing can be used in all 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 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:
```

Indirect Addressing Program 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 mplh, 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 examples shown above, no reference is made to specific Data Memory addresses.

Direct Addressing Program Example using extended instructions

```
data .section 'data'
temp db ?
code .section at 0 'code'
org 00h
start:
    lmov a, [m]               ; move [m] data to acc
    lsub a, [m+1]             ; compare [m] and [m+1] data
    snz c                     ; [m]>[m+1]?
    jmp continue              ; no
    lmov a, [m]               ; yes, exchange [m] and [m+1] data
    mov temp, a
    lmov a, [m+1]
    lmov [m], a
    mov a, temp
    lmov [m+1], a
continue:
```

Note: Here “m” is a data memory address located in any data memory sectors. For example, m=1F0H, it indicates address 0F0H in Sector 1.

Accumulator – ACC

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

Program Counter Low 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. TBLP and TBHP are the table pointers and indicate the location where the table data is located. Their value must be setup before any table read commands are executed. Their value can be changed, for example using the “INC” or “DEC” instructions, allowing for easy table data pointing and reading. TBLH is the location where the high order byte of the table data is stored after a table read data instruction has been executed. Note that the lower order table data byte is transferred to a user defined location.

Status Register – STATUS

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

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

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

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

• **STATUS Register**

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

EEPROM Data Memory

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

EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 32×8 bits for the device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address and a data register in Sector 0 and a single control register in Sector 1.

EEPROM Registers

Three registers control the overall operation of the internal EEPROM Data Memory. These are the address register, EEA, the data register, EED and a single control register, EEC. As both the EEA and EED registers are located in Sector 0, they can be directly accessed in the same way as any other Special Function Register. The EEC register however, being located in Sector 1, can only be read from or written to indirectly using the MP1L/MP1H or MP2L/MP2H Memory Pointer pairs and Indirect Addressing Register, IAR1/IAR2. Because the EEC control register is located at address 40H in Sector 1, the MP1L or MP2L Memory Pointer must first be set to the value 40H and the MP1H or MP2H Memory Pointer high byte set to the value, 01H, before any operations on the EEC register are executed.

| Register Name | Bit | | | | | | | |
|---------------|-----|----|----|------|------|------|------|------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| EEA | — | — | — | EEA4 | EEA3 | EEA2 | EEA1 | EEA0 |
| EED | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| EEC | — | — | — | — | WREN | WR | RDEN | RD |

EEPROM Register List

• EEA Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---|------|------|------|------|------|
| Name | — | — | — | EEA4 | EEA3 | EEA2 | EEA1 | EEA0 |
| R/W | — | — | — | R/W | R/W | R/W | R/W | R/W |
| POR | — | — | — | 0 | 0 | 0 | 0 | 0 |

Bit 7~5 Unimplemented, read as “0”

Bit 4~0 **EEA4~EEA0**: Data EEPROM address bit 4 ~ bit 0

• **EED Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | 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**: Data EEPROM data bit 7 ~ bit 0

• **EEC Register**

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

Bit 7~4 Unimplemented, read as “0”

Bit 3 **WREN**: Data EEPROM Write Enable

0: Disable

1: Enable

This is the Data EEPROM Write Enable Bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations.

Bit 2 **WR**: EEPROM Write Control

0: Write cycle has finished

1: Activate a write cycle

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

Bit 1 **RDEN**: Data EEPROM Read Enable

0: Disable

1: Enable

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

Bit 0 **RD**: EEPROM Read Control

0: Read cycle has finished

1: Activate a read cycle

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

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

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

3. Ensure that the write operation is totally complete before changing the contents of the EEPROM related registers.

Reading Data from the EEPROM

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

Writing Data to the EEPROM

To write data to the EEPROM, the EEPROM address of the data to be written must first be placed in the EEA register and the data placed in the EED register. To initiate a write cycle, the write enable bit, WREN, in the EEC register must first be set high to enable the write function. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle. These two instructions must be executed consecutively. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set again after the write cycle has started. Note that setting the WR bit high will not initiate a write cycle if the WREN bit has not been set. As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended.

Write Protection

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

EEPROM Interrupt

The EEPROM write interrupt is generated when an EEPROM write cycle has ended. The EEPROM interrupt must first be enabled by setting the DEE bit in the relevant interrupt register. When an EEPROM write cycle ends, the DEF request flag will be set high. If the global and EEPROM interrupts are enabled and the stack is not full, a jump to the EEPROM Interrupt vector will take place. When the interrupt is serviced the EEPROM interrupt flag will be automatically reset. More details can be obtained in the Interrupt section.

Programming Considerations

Care must be taken that data is not inadvertently written to the EEPROM. Protection can be enhanced by ensuring that the Write Enable bit is normally cleared to zero when not writing. Also the Memory Pointer high byte register, MP1H or MP2H, could be normally cleared to zero as this would inhibit access to Sector 1 where the EEPROM control register exists. Although certainly not necessary, consideration might be given in the application program to the checking of the validity of new write data by a simple read back process.

When writing data the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. The global interrupt bit EMI should also be cleared before a write cycle is executed and then set high again after a valid write activation procedure has completed. Note that the device should not enter the IDLE or SLEEP mode until the EEPROM read or write operation is totally complete. Otherwise, the EEPROM read or write operation will fail.

Programming Examples

Reading data from the EEPROM – polling method

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, 40H                ; setup memory pointer MP1L
MOV MP1L, A               ; MP1L points to EEC register
MOV A, 01H                ; setup memory pointer MP1H
MOV MP1H, A
SET IAR1.1                ; set RDEN bit, enable read operations
SET IAR1.0                 ; start Read Cycle - set RD bit
BACK:
SZ IAR1.0                 ; check for read cycle end
JMP BACK
CLR IAR1                  ; disable EEPROM read if no more read operations are required
CLR MP1H
MOV A, EED                 ; move read data to register
MOV READ_DATA, A
```

Note: For each read operation, the address register should be re-specified followed by setting the RD bit high to activate a read cycle even if the target address is consecutive.

Writing Data to the EEPROM – polling method

```
MOV A, EEPROM_ADRES      ; user defined address
MOV EEA, A
MOV A, EEPROM_DATA        ; user defined data
MOV EED, A
MOV A, 40H                ; setup memory pointer MP1L
MOV MP1L, A               ; MP1L points to EEC register
MOV A, 01H                ; setup memory pointer MP1H
MOV MP1H, A
CLR EMI
SET IAR1.3                ; set WREN bit, enable write operations
SET IAR1.2                 ; start Write Cycle - set WR bit - executed immediately
                           ; after setting WREN bit

SET EMI
BACK:
SZ IAR1.2                 ; check for write cycle end
JMP BACK
CLR MP1H
```

Oscillators

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

Oscillator Overview

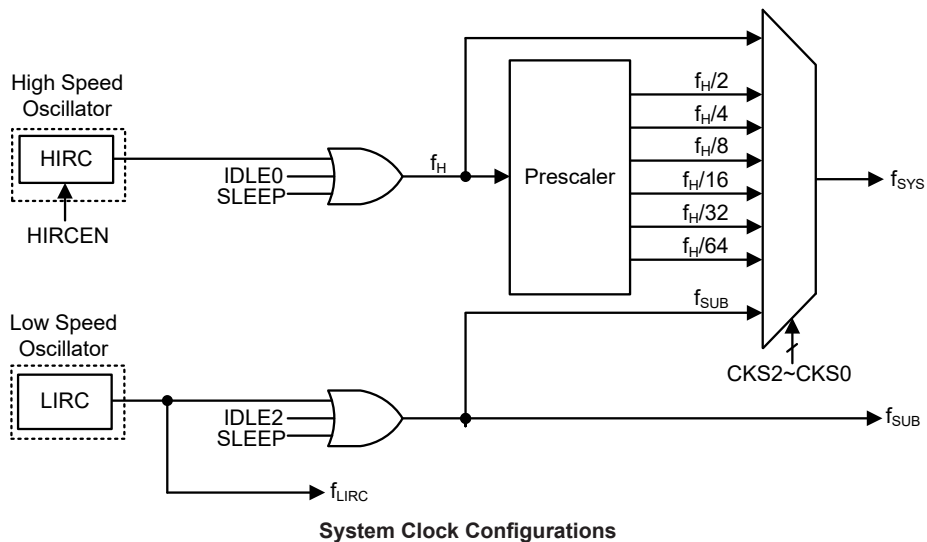
In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. Fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillator. The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

| Type | Name | Frequency |
|------------------------|------|-----------|
| Internal High Speed RC | HIRC | 2/4/8MHz |
| Internal Low Speed RC | LIRC | 32kHz |

Oscillator Types

System Clock Configurations

There are two oscillator sources, one high speed oscillator and one low speed oscillator. The high speed system clock is sourced from the internal 2/4/8MHz 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.



Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 2MHz, 4MHz and 8MHz, which are selected by the HIRC1~HIRC0 bits in the HIRCC register. These bits must also be setup to match the selected configuration option frequency to ensure that the HIRC frequency accuracy specified in the A.C. Characteristics is achieved. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. Note that this internal system clock option requires no external pins for its operation.

Internal 32kHz Oscillator – LIRC

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

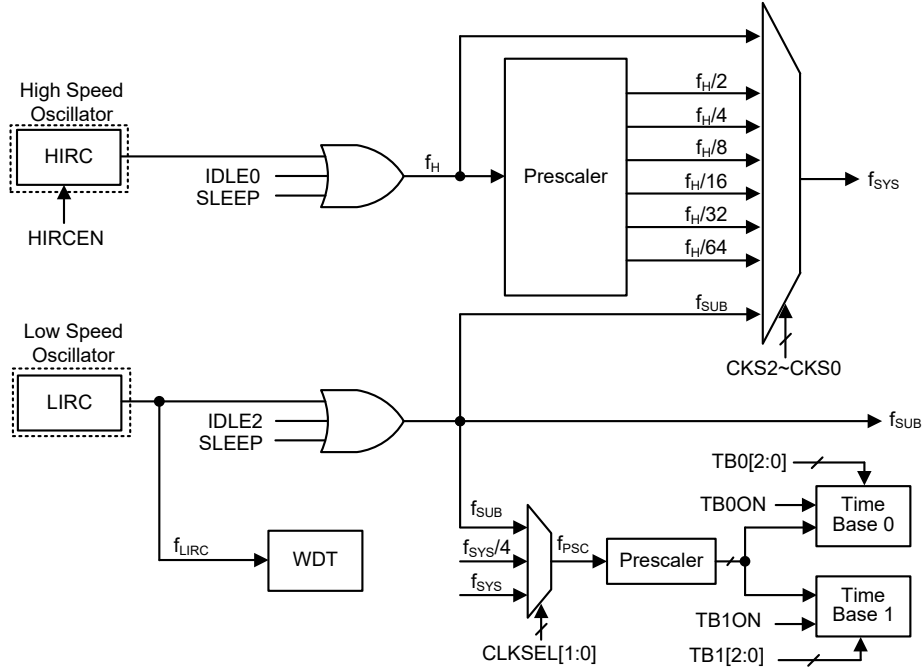
Operating Modes and System Clocks

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

System Clocks

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

The main system clock, can come from a high frequency, f_H , or low frequency, f_{SUB} , source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock 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 $f_H/2 \sim f_H/64$.



Device Clock Configurations

Note: When the system clock source f_{SYS} is switched to f_{SUB} from f_H , the high speed oscillator will stop to conserve the power or continue to oscillate to provide the clock source, $f_H \sim f_H/64$, for peripheral circuit to use, which is determined by configuring the corresponding high speed oscillator enable control bit.

System Operation Modes

There are six different modes of operation for the microcontroller, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the FAST Mode and SLOW Mode. The remaining four modes, the SLEEP, IDLE0, IDLE1 and IDLE2 Modes 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

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

SLOW Mode

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

SLEEP Mode

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

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 turned on to drive some peripheral functions.

IDLE1 Mode

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

IDLE2 Mode

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

Control Registers

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

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

System Operating Mode Control 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 or FSS bit. 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 system clock period.

• **HIRCC Register**

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

Bit 7~4 Unimplemented, read as “0”

Bit 3~2 **HIRC1~HIRC0**: HIRC frequency selection

- 00: 2MHz
- 01: 4MHz
- 10: 8MHz
- 11: 2MHz

When the HIRC oscillator is enabled or the HIRC frequency selection is changed by application program, the clock frequency will automatically be changed after the HIRCF flag is set to 1. It is recommended that the HIRC frequency selected by these bits is the same as the frequency determined by the configuration option to keep the HIRC frequency accuracy specified in the A.C. characteristics.

Bit 1 **HIRCF**: HIRC oscillator stable flag
 0: HIRC unstable
 1: HIRC stable

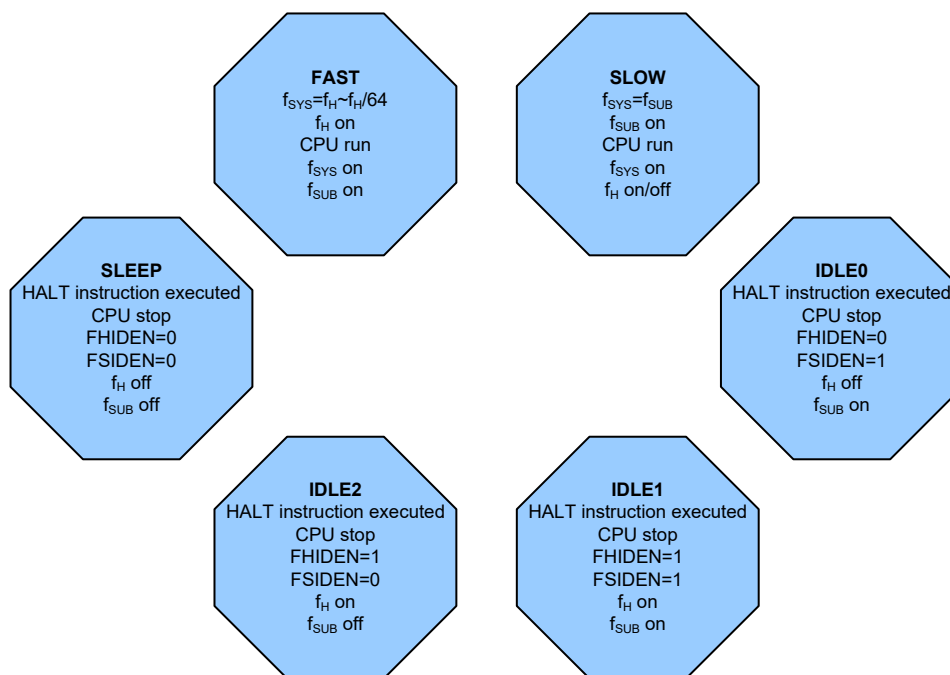
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

Bit 0 **HIRCEN**: HIRC oscillator enable control
 0: Disable
 1: Enable

Operating Mode Switching

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

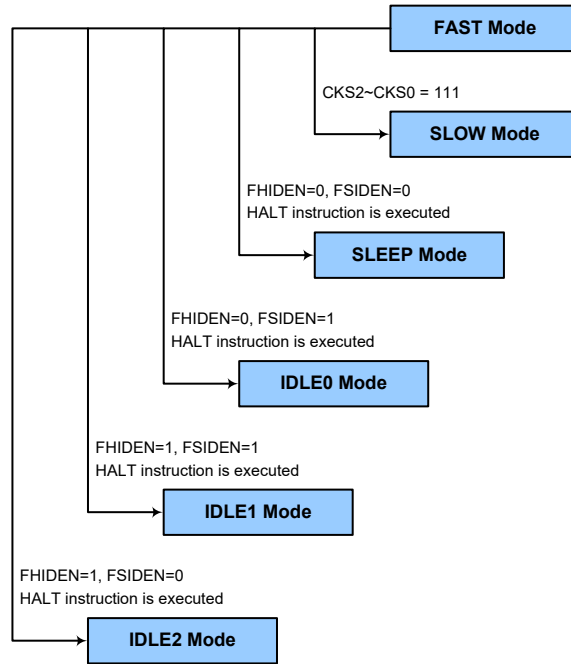
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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



FAST 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.

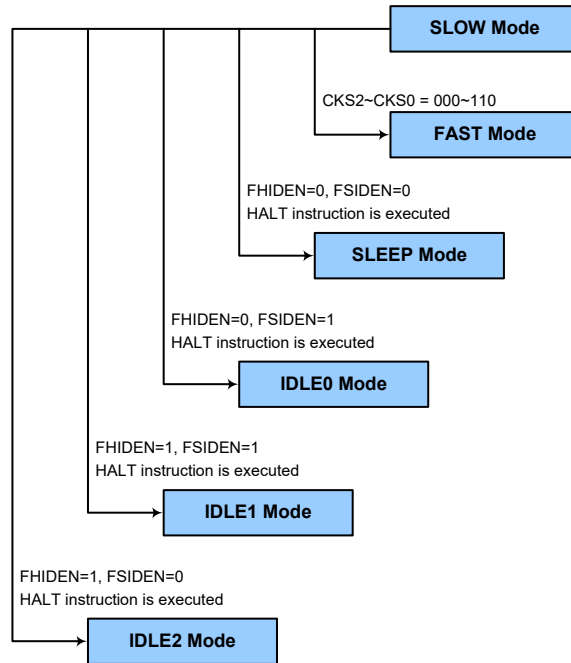
The SLOW Mode is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



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 $f_H \sim f_H/64$.

However, if f_H is not used in SLOW mode and thus switched off, it will take some time to re-oscillate and stabilise when switching to the FAST mode from the SLOW Mode. This is monitored using the HIRCF bit in the HIRCC register. The time duration required for the high speed system oscillator stabilization is specified in the System Start Up Time Characteristics.



Entering the SLEEP Mode

There is only one way for the device to enter the SLEEP Mode and that is to execute the “HALT” instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to “0”. In this mode all the clocks and functions will be switched off except the WDT function. When this instruction is executed under the conditions described above, the following will occur:

- The system clock will be stopped and the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE0 Mode

There is only one way for the device to enter the IDLE0 Mode and that is to execute the “HALT” instruction in the application program with the FHIDEN bit in the SCC register equal to “0” and the FSIDEN bit in the SCC register equal to “1”. When this instruction is executed under the conditions described above, the following will occur:

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

Entering the IDLE1 Mode

There is only one way for the device to enter the IDLE1 Mode and that is to execute the “HALT” instruction in the application program with both the FHIDEN and FSIDEN bits in the SCC register equal to “1”. When this instruction is executed under the conditions described above, the following will occur:

- The f_H and f_{SUB} clocks will be on but the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag, PDF, will be set and the Watchdog time-out flag, TO, will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

Entering the IDLE2 Mode

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

- The f_H clock will be on but the f_{SUB} clock will be off and the application program will stop at the “HALT” instruction.
- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.

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

Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Mode, there are other considerations which must also be taken into account by the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be unbonded pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

Care must also be taken with the loads, which are connected to I/O pins, which are setup as outputs. These should be placed in a condition in which minimum current is drawn or connected only to external circuits that do not draw current, such as other CMOS inputs. Also note that additional standby current will also be required if the LIRC oscillator has enabled.

In the IDLE1 and IDLE2 Mode the high speed oscillator is on, if the peripheral function clock source is derived from the high speed oscillator, the additional standby current will also be perhaps in the order of several hundred micro-amps.

Wake-up

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

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

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

When the device executes the “HALT” instruction, the PDF flag will be set to 1. The PDF flag will be cleared to 0 if the device experiences a system power-up or executes the clear Watchdog Timer instruction. If the system is woken up by a WDT overflow, a Watchdog Timer reset will be initiated and the TO flag will be set to 1. The TO flag is set if a WDT time-out occurs and causes a wake-up that only resets the Program Counter and Stack Pointer, other flags remain in their original status.

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

Watchdog Timer

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

Watchdog Timer Clock Source

The Watchdog Timer clock source is provided by the internal clock, f_{LIRC} , which is sourced from the LIRC oscillator. The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with V_{DD} , temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of 2^8 to 2^{18} to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

Watchdog Timer Control Register

A single register, WDTC, controls the required time-out period as well as the enable/disable and reset MCU operation.

• WDTC Register

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

Bit 7~3 **WE4~WE0**: WDT function software control

10101: Disable

01010: Enable

Other values: Reset MCU

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

Bit 2~0 **WS2~WS0**: WDT time-out period selection

000: $2^8/f_{LIRC}$

001: $2^{10}/f_{LIRC}$

010: $2^{12}/f_{LIRC}$

011: $2^{14}/f_{LIRC}$

100: $2^{15}/f_{LIRC}$

101: $2^{16}/f_{LIRC}$

110: $2^{17}/f_{LIRC}$

111: $2^{18}/f_{LIRC}$

These three bits determine the division ratio of the Watchdog Timer source clock, which in turn determines the timeout period.

• RSTFC Register

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

“x”: unknown

Bit 7~3 Unimplemented, read as “0”

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

Bit 1 Unimplemented, read as “0”

Bit 0 **WRF:** WDT control register software reset flag
 0: Not occurred
 1: Occurred

This bit is set to 1 by the WDT Control register software reset and cleared by the application program. Note that this bit can only be cleared to 0 by the application program.

Watchdog Timer Operation

The Watchdog Timer operates by providing a device reset when its timer overflows. This means that in the application program and during normal operation the user has to strategically clear the Watchdog Timer before it overflows to prevent the Watchdog Timer from executing a reset. This is done using the clear watchdog 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 enable/disable control of the Watchdog Timer and the MCU reset. The WDT function will be disabled when the WE4~WE0 bits are set to a value of 10101B while the WDT function will be enabled if the WE4~WE0 bits are equal to 01010B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the device after a delay time, t_{SRESET} . After power-on these bits will have a value of 01010B.

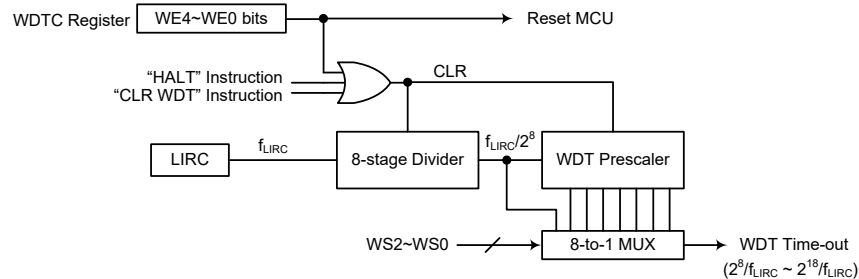
| WE4~WE0 Bits | WDT Function |
|-----------------|--------------|
| 10101B | Disable |
| 01010B | Enable |
| Any other value | Reset MCU |

Watchdog Timer Enable/Disable Control

Under normal program operation, a Watchdog Timer time-out will initialise a device reset and set the status bit TO. However, if the system is in the SLEEP or IDLE Mode, when a Watchdog Timer time-out occurs, the TO and PDF bits 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 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, the third is via a HALT instruction.

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

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



Watchdog Timer

Reset and Initialisation

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

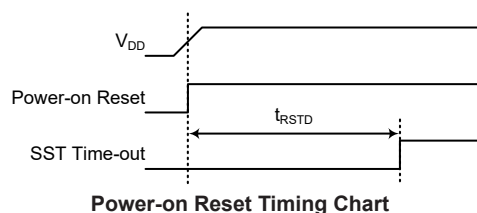
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result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

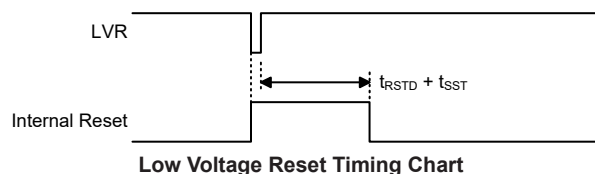
The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. 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.



Low Voltage Reset – LVR

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

The LVR function is always enabled with a specific LVR voltage V_{LVR} , which is fixed at 2.1V. 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 in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between $0.9V \sim V_{LVR}$ must exist for a time greater than that specified by t_{LVR} in the LVR/LVD characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. Note that the LVR function will be automatically disabled when the device enters the IDLE or SLEEP mode.



• **RSTFC Register**

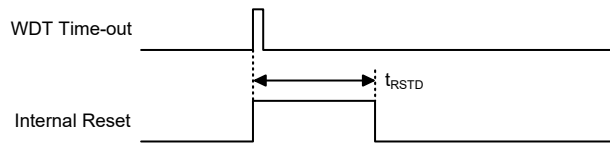
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---|---|---|------|---|-----|
| Name | — | — | — | — | — | LVRF | — | WRF |
| R/W | — | — | — | — | — | R/W | — | R/W |
| POR | — | — | — | — | — | x | — | 0 |

“x”: unknown

- Bit 7~3 Unimplemented, read as “0”
- Bit 2 **LVRF**: LVR function reset flag
 0: Not occurred
 1: Occurred
 This bit is set to 1 when a specific Low Voltage Reset situation condition occurs. This bit can only be cleared to 0 by the application program.
- Bit 1 Unimplemented, read as “0”
- Bit 0 **WRF**: WDT control register software reset flag
 Refer to the Watchdog Timer Control Register section.

Watchdog Time-out Reset during Normal Operation

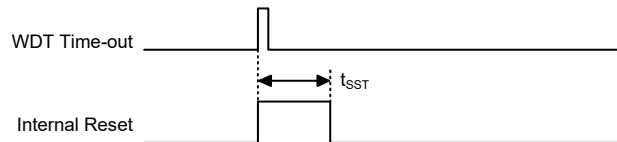
When the Watchdog time-out Reset occurs during normal operations in the FAST or SLOW mode, the Watchdog time-out flag TO will be set to “1”.



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 |

Note: “u” stands for unchanged

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

| Item | Condition After RESET |
|--------------------|--|
| Program Counter | Reset to zero |
| Interrupts | All interrupts will be disabled |
| WDT, Time Base | 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 each of the microcontroller internal registers. Note that as more than one package type exist, the table reflects the situation for the larger package type.

| Register | Power-on Reset | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|----------|----------------|---------------------------------|---------------------------|
| IAR0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| IAR1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP1L | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP1H | 0000 0000 | 0000 0000 | uuuu uuuu |
| ACC | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| PCL | 0000 0000 | 0000 0000 | 0000 0000 |
| TBLP | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TBLH | xxxx xxxx | uuuu uuuu | uuuu uuuu |
| TBHP | ---- -xxx | ---- -uuu | ---- -uuu |
| STATUS | xx00 xxxx | uu1u uuuu | uu11 uuuu |
| IAR2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP2L | 0000 0000 | 0000 0000 | uuuu uuuu |
| MP2H | 0000 0000 | 0000 0000 | uuuu uuuu |
| RSTFC | ---- -x-0 | ---- -u-u | ---- -u-u |
| TB0C | 0--- -000 | 0--- -000 | u--- -uuu |
| TB1C | 0--- -000 | 0--- -000 | u--- -uuu |
| SCC | 000- --00 | 000- --00 | uuu- --uu |
| HIRCC | ---- 0001 | ---- 0001 | ---- uuuu |
| PA | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAC | 1111 1111 | 1111 1111 | uuuu uuuu |
| PAPU | 0000 0000 | 0000 0000 | uuuu uuuu |
| PAWU | 0000 0000 | 0000 0000 | uuuu uuuu |
| PB | --11 1111 | --11 1111 | --uu uuuu |
| PBC | --11 1111 | --11 1111 | --uu uuuu |
| PBPU | --00 0000 | --00 0000 | --uu uuuu |
| SLEDC | 0000 0000 | 0000 0000 | uuuu uuuu |
| PSCR | ---- --00 | ---- --00 | ---- --uu |
| LVDC | --00 0000 | --00 0000 | --uu uuuu |
| PTMC0 | 0000 0--- | 0000 0--- | uuuu u--- |
| PTMC1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PTMDL | 0000 0000 | 0000 0000 | uuuu uuuu |

| Register | Power-on Reset | WDT Time-out (Normal Operation) | WDT Time-out (IDLE/SLEEP) |
|------------------|----------------|---------------------------------|---------------------------|
| PTMDH | ---- --00 | ---- --00 | ---- --uu |
| PTMAL | 0000 0000 | 0000 0000 | uuuu uuuu |
| PTMAH | ---- --00 | ---- --00 | ---- --uu |
| PTMRPL | 0000 0000 | 0000 0000 | uuuu uuuu |
| PTMRPH | ---- --00 | ---- --00 | ---- --uu |
| REGC | 0--- --00 | 0--- --00 | u--- --uu |
| SADOL | xxxx ---- | xxxx ---- | uuuu ---- (ADRF=0) |
| | | | uuuu uuuu (ADRF=1) |
| SADOH | xxxx xxxx | xxxx xxxx | uuuu uuuu (ADRF=0) |
| | | | ---- uuuu (ADRF=1) |
| SADC0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SADC1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| OPSW | 0000 0000 | 0000 0000 | uuuu uuuu |
| OPC | 000- --00 | 000- --00 | uuu- --uu |
| OPVOS | 0010 0000 | 0010 0000 | uuuu uuuu |
| INTEG | ---- --00 | ---- --00 | ---- --uu |
| INTC0 | -000 0000 | -000 0000 | -uuu uuuu |
| INTC1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| INTC2 | -000 -000 | -000 -000 | -uuu -uuu |
| WDTC | 0101 0011 | 0101 0011 | uuuu uuuu |
| EEA | ---0 0000 | ---0 0000 | ---u uuuu |
| EED | 0000 0000 | 0000 0000 | uuuu uuuu |
| EEC | ---- 0000 | ---- 0000 | ---- uuuu |
| PAS0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PAS1 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PBS0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| PBS1 | ---- 0000 | ---- 0000 | ---- uuuu |
| IFS0 | 0000 0000 | 0000 0000 | uuuu uuuu |
| IFS1 | ---- 0000 | ---- 0000 | ---- uuuu |
| SIMC0 | 1110 0000 | 1110 0000 | uuuu uuuu |
| SIMC1 (UMD=0) | 1000 0001 | 1000 0001 | uuuu uuuu |
| UUCR1* (UMD=1) | 0000 00x0 | 0000 00x0 | uuuu uuuu |
| SIMC2/SIMA/UUCR2 | 0000 0000 | 0000 0000 | uuuu uuuu |
| SIMD/UTXR_RXR | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| SIMTOC (UMD=0) | 0000 0000 | 0000 0000 | uuuu uuuu |
| UBRG* (UMD=1) | xxxx xxxx | xxxx xxxx | uuuu uuuu |
| UUSR | 0000 1011 | 0000 1011 | uuuu uuuu |

Note: "u" stands for unchanged

"x" stands for unknown

"-" stands for unimplemented

"*" The UUCR1 and SIMC1 registers share the same memory address while the UBRG and SIMTOC registers share the same memory address. The default value of the UUCR1 or UBRG register can be obtained when the UMD bit is set high by application program after a reset.

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 | — | — | PB5 | PB4 | PB3 | PB2 | PB1 | PB0 |
| PBC | — | — | PBC5 | PBC4 | PBC3 | PBC2 | PBC1 | PBC0 |
| PBPU | — | — | PBPU5 | PBPU4 | PBPU3 | PBPU2 | PBPU1 | PBPU0 |

“—”: Unimplemented, read as “0”

I/O Logic Function Register 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 a digital input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using registers, namely PAPU~PBPU, 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” can be A and 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 Power down mode.

• PAWU Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PAWU7 | PAWU6 | PAWU5 | PAWU4 | PAWU3 | PAWU2 | PAWU1 | PAWU0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7~0 **PAWU7~PAWU0**: PA7~PA0 wake-up function control
 0: Disable
 1: Enable

I/O Port Control Registers

Each I/O port has its own control register known as PAC~PBC, to control the input/output configuration. With this control register, each CMOS output or input can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding bit of the control register must be written as a “1”. This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a “0”, the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

• PxC Register

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

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

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

I/O Port Source Current Selection

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

• **SLEDC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|--------|--------|--------|--------|--------|--------|--------|--------|
| Name | SLEDC7 | SLEDC6 | SLEDC5 | SLEDC4 | SLEDC3 | SLEDC2 | SLEDC1 | SLEDC0 |
| 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 **SLEDC7~SLEDC6:** PB5~PB4 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)
- Bit 5~4 **SLEDC5~SLEDC4:** PB3~PB0 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)
- Bit 3~2 **SLEDC3~SLEDC2:** PA7~PA4 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)
- Bit 1~0 **SLEDC1~SLEDC0:** PA3~PA0 source current selection
00: Source current=Level 0 (Min.)
01: Source current=Level 1
10: Source current=Level 2
11: Source current=Level 3 (Max.)

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 IFSi, 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 INT, PTCK, PTPI, 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 an 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.

Note that as more than one package type exist, the table refelects the situation for the larger package type.

| Register Name | Bit | | | | | | | |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PAS0 | PAS07 | PAS06 | PAS05 | PAS04 | PAS03 | PAS02 | PAS01 | PAS00 |
| PAS1 | PAS17 | PAS16 | PAS15 | PAS14 | PAS13 | PAS12 | PAS11 | PAS10 |
| PBS0 | PBS07 | PBS06 | PBS05 | PBS04 | PBS03 | PBS02 | PBS01 | PBS00 |
| PBS1 | — | — | — | — | PBS13 | PBS12 | PBS11 | PBS10 |
| IFS0 | IFS07 | IFS06 | IFS05 | IFS04 | IFS03 | IFS02 | IFS01 | IFS00 |
| IFS1 | — | — | — | — | IFS13 | IFS12 | IFS11 | IFS10 |

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/PTCK
 01: PTP
 10: SCS
 11: SDO/TX

Bit 5~4 **PAS05~PAS04:** PA2 Pin-Shared function selection
 00: PA2
 01: SCK/SCL
 10: SDO/TX
 11: PA2

Bit 3~2 **PAS03~PAS02:** PA1 Pin-Shared function selection
 00: PA1/INT
 01: SDI/SDA/RX
 10: AN4
 11: PA1/INT

Bit 1~0 **PAS01~PAS00:** PA0 Pin-Shared function selection
 00: PA0
 01: SCS
 10: SDI/SDA/RX
 11: PA0

• **PAS1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PAS17 | PAS16 | PAS15 | PAS14 | PAS13 | PAS12 | PAS11 | PAS10 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7~6 **PAS17~PAS16:** PA7 Pin-Shared function selection

00: PA7/INT
 01: SDO/TX
 10: SCK/SCL
 11: PA7/INT

Bit 5~4 **PAS15~PAS14:** PA6 Pin-Shared function selection

00: PA6
 01: PTPB
 10: \overline{SCS}
 11: SDO/TX

Bit 3~2 **PAS13~PAS12:** PA5 Pin-Shared function selection

00: PA5/INT
 01: PTP
 10: SDI/SDA/RX
 11: PA5/INT

Bit 1~0 **PAS11~PAS10:** PA4 Pin-Shared function selection

00: PA4/PTPI
 01: SDO/TX
 10: SCK/SCL
 11: PA4/PTPI

• **PBS0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PBS07 | PBS06 | PBS05 | PBS04 | PBS03 | PBS02 | PBS01 | PBS00 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit 7~6 **PBS07~PBS06:** PB3 Pin-Shared function selection

00: PB3
 01: AN3
 10: PB3
 11: PB3

Bit 5~4 **PBS05~PBS04:** PB2 Pin-Shared function selection

00: PB2
 01: AN2
 10: PB2
 11: PB2

Bit 3~2 **PBS03~PBS02:** PB1 Pin-Shared function selection

00: PB1
 01: AN1
 10: PB1
 11: PB1

Bit 1~0 **PBS01~PBS00:** PB0 Pin-Shared function selection

00: PB0
 01: AN0
 10: PB0
 11: PB0

• **PBS1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---|---|-------|-------|-------|-------|
| Name | — | — | — | — | PBS13 | PBS12 | PBS11 | PBS10 |
| 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 **PBS13~PBS12:** PB5 Pin-Shared function selection
 00: PB5/PTPI
 01: SDO/TX
 10: SCK/SCL
 11: PB5/PTPI
- Bit 1~0 **PBS11~PBS10:** PB4 Pin-Shared function selection
 00: PB4/PTCK
 01: SDI/SDA/RX
 10: PB4/PTCK
 11: PB4/PTCK

• **IFS0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | IFS07 | IFS06 | IFS05 | IFS04 | IFS03 | IFS02 | IFS01 | IFS00 |
| 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 **IFS07~IFS06:** SCK/SCL input source pin selection
 00: PA2
 01: PA7
 10: PB5
 11: PA4
- Bit 5~4 **IFS05~IFS04:** SDI/SDA/RX input source pin selection
 00: PA0
 01: PA1
 10: PA5
 11: PB4
- Bit 3~2 **IFS03~IFS02:** PTPI input source pin selection
 00: PA4
 01: PA4
 10: PB5
 11: PB5
- Bit 1~0 **IFS01~IFS00:** PTCK input source pin selection
 00: PA3
 01: PA3
 10: PB4
 11: PB4

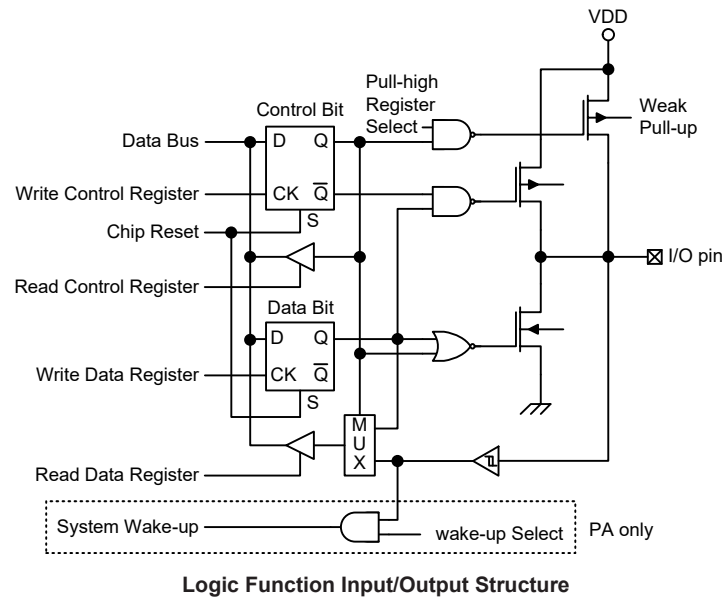
• **IFS1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---|---|-------|-------|-------|-------|
| Name | — | — | — | — | IFS13 | IFS12 | IFS11 | IFS10 |
| 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 **IFS13~IFS12**: INT input source pin selection
 - 00: PA1
 - 01: PA1
 - 10: PA5
 - 11: PA7
- Bit 1~0 **IFS11~IFS10**: $\overline{\text{SCS}}$ input source pin selection
 - 00: PA0
 - 01: PA3
 - 10: PA6
 - 11: PA0

I/O Pin Structures

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



Programming Considerations

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

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

Timer Module – 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 one Timer Module, abbreviated to the name TM. The TM is a multi-purpose timing unit and serves to provide operations such as Timer/Counter, Input Capture, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. The TM has two individual interrupts. The addition of input and output pins for the TM ensures that users are provided with timing unit with a wide and flexible range of features.

Introduction

The device contains one Periodic Type TM. The brief features to the Periodic TM will be described in this section. The detailed operation will be described in the following chapter.

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

TM Function Summary

TM Operation

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

TM Clock Source

The clock source which drives the main counter in the TM can originate from various sources. The selection of the required clock source is implemented using the PTCK2~PTCK0 bits in the PTM 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 PTCK pin. The PTCK pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

TM Interrupts

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

TM External Pins

The TM has two TM input pins, with the label PTCK and PTPI respectively. The PTM input pin, PTCK, is essentially a clock source for the PTM and is selected using the PTCK2~PTCK0 bits in the PTMC0 register. This external TM input pin allows an external clock source to drive the internal TM. The PTCK input pin can be chosen to have either a rising or falling active edge. The PTCK pin is also used as the external trigger input pin in single pulse output mode.

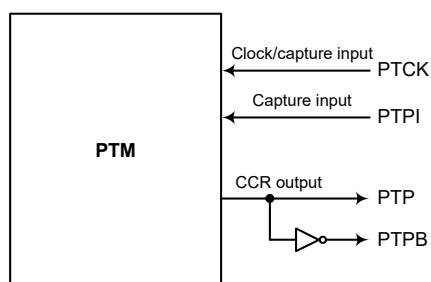
The other PTM input pin, PTPI, is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the PTIO1~PTIO0 bits in the PTMC1 register. There is another capture input, PTCK, for PTM capture input mode, which can be used as the external trigger input source except the PTPI pin.

The PTM has two output pins with the label PTP and PTPB. 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 PTP and PTPB output pins are also the pins where the TM generates the PWM output waveform.

As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be setup using the relevant pin-shared function selection bits described in the Pin-shared Function section. The details of the pin-shared function selection are described in the pin-shared function section.

| PTM | |
|------------|-----------|
| Input | Output |
| PTCK, PTPI | PTP, PTPB |

TM External Pins

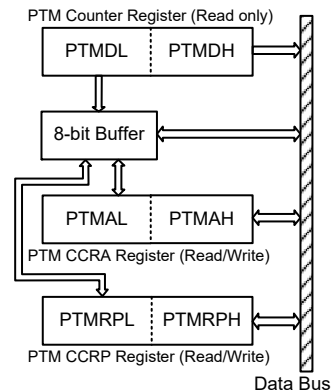


PTM Function Pin Block Diagram

Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA and CCRP 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 and CCRP registers are implemented in the way shown in the following diagram and accessing these register pairs is carried out in a specific way as described above, it is recommended to use the “MOV” instruction to access the CCRA and CCRP low byte registers, named PTMAL and PTMRPL, using the following access procedures. Accessing the CCRA or CCRP low byte registers without following these access procedures will result in unpredictable values.

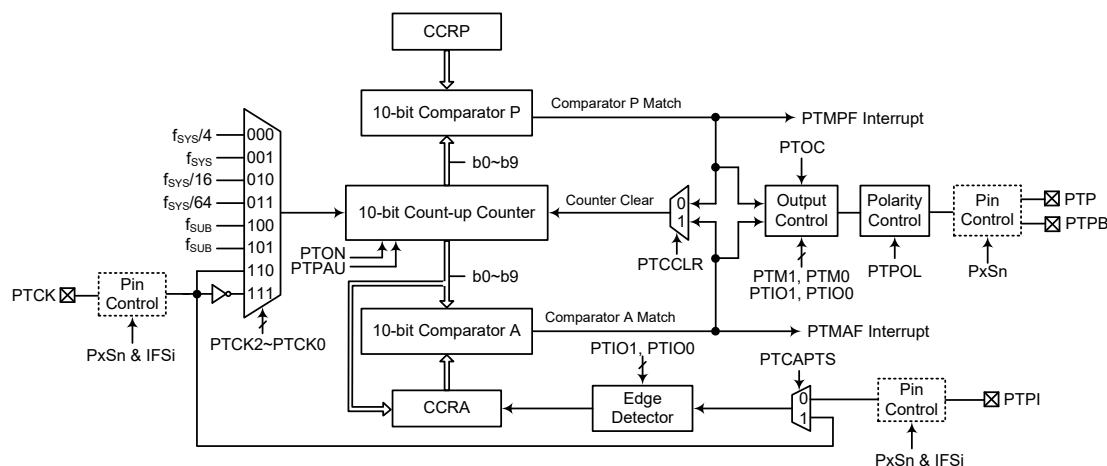


The following steps show the read and write procedures:

- Writing Data to CCRA
 - ♦ Step 1. Write data to Low Byte PTMAL or PTMRPL
 - Note that here data is only written to the 8-bit buffer.
 - ♦ Step 2. Write data to High Byte PTMAH or PTMRPH
 - Here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the Low Byte registers.
- Reading Data from the Counter Registers and or CCRA
 - ♦ Step 1. Read data from the High Byte PTMDH, PTMAH or PTMRPH
 - 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 PTMDL, PTMAL or PTMRPL
 - This step reads data from the 8-bit buffer.

Periodic Type TM – PTM

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



Note: 1. The PTPB is the inverted output of the PTP.

2. The PTM external pins are pin-shared with other functions, therefore before using the PTM function, ensure that the pin-shared function registers have been set properly to enable the PTM pin function. The PTCK and PTPI pins, if used, must also be set as an input by setting the corresponding bits in the port control register.

10-bit Periodic Type TM Block Diagram

Periodic TM Operation

The Periodic Type TM 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 and CCRA comparators are 10-bit wide whose value is respectively compared 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 PTON 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 PTM interrupt signal will also usually be generated. The Periodic Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control the output pins. All operating setup conditions are selected using relevant internal registers.

Periodic Type TM Register Description

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

| Register Name | Bit | | | | | | | |
|---------------|-------|-------|-------|-------|-------|-------|---------|--------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PTMC0 | PTPAU | PTCK2 | PTCK1 | PTCK0 | PTON | — | — | — |
| PTMC1 | PTM1 | PTM0 | PTIO1 | PTIO0 | PTOC | PTPOL | PTCAPTS | PTCCLR |
| PTMDL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PTMDH | — | — | — | — | — | — | D9 | D8 |
| PTMAL | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| PTMAH | — | — | — | — | — | — | D9 | D8 |
| PTMRPL | PTRP7 | PTRP6 | PTRP5 | PTRP4 | PTRP3 | PTRP2 | PTRP1 | PTRP0 |
| PTMRPH | — | — | — | — | — | — | PTRP9 | PTRP8 |

10-bit Periodic TM Register List

• **PTMC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|------|---|---|---|
| Name | PTPAU | PTCK2 | PTCK1 | PTCK0 | PTON | — | — | — |
| R/W | R/W | R/W | R/W | R/W | R/W | — | — | — |
| POR | 0 | 0 | 0 | 0 | 0 | — | — | — |

Bit 7 PTPAU: PTM 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 PTM 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 PTCK2~PTCK0: Select PTM Counter clock
 000: $f_{SYS}/4$
 001: f_{SYS}
 010: $f_H/16$
 011: $f_H/64$
 100: f_{SUB}
 101: f_{SUB}
 110: PTCK rising edge clock
 111: PTCK falling edge clock

These three bits are used to select the clock source for the PTM. The external pin clock source can be chosen to be active on the rising or falling edge. The clock source f_{SYS} is the system clock, while f_H and f_{SUB} are other internal clocks, the details of which can be found in the oscillator section.

Bit 3 PTON: PTM Counter On/Off Control
 0: Off
 1: On

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

Bit 2~0 Unimplemented, read as “0”

• **PTMC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|------|------|-------|-------|------|-------|---------|--------|
| Name | PTM1 | PTM0 | PTIO1 | PTIO0 | PTOC | PTPOL | PTCAPTS | PTCCLR |
| 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 **PTM1~PTM0**: Select PTM Operating Mode
 00: Compare Match Output Mode
 01: Capture Input Mode
 10: PWM Output Mode or Single Pulse Output Mode
 11: Timer/Counter Mode
 These bits setup the required operating mode for the PTM. To ensure reliable operation the PTM should be switched off before any changes are made to the PTM1 and PTM0 bits. In the Timer/Counter Mode, the PTM output pin state is undefined.
- Bit 5~4 **PTIO1~PTIO0**: Select PTM external pin function
 Compare Match Output Mode
 00: No change
 01: Output low
 10: Output high
 11: Toggle output
 PWM Output Mode/Single Pulse Output Mode
 00: PWM Output inactive state
 01: PWM Output active state
 10: PWM output
 11: Single pulse output
 Capture Input Mode
 00: Input capture at rising edge of PTPI or PTCK
 01: Input capture at falling edge of PTPI or PTCK
 10: Input capture at falling/rising edge of PTPI or PTCK
 11: Input capture disabled
 Timer/Counter Mode
 Unused
 These two bits are used to determine how the PTM external pin changes state when a certain condition is reached. The function that these bits select depends upon in which mode the PTM is running.
 In the Compare Match Output Mode, the PTIO1 and PTIO0 bits determine how the PTM output pin changes state when a compare match occurs from the Comparator A. The PTM 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 PTM output pin should be setup using the PTOC bit in the PTMC1 register. Note that the output level requested by the PTIO1 and PTIO0 bits must be different from the initial value setup using the PTOC bit otherwise no change will occur on the PTM output pin when a compare match occurs. After the PTM output pin changes state, it can be reset to its initial level by changing the level of the PTON bit from low to high.
 In the PWM Output Mode, the PTIO1 and PTIO0 bits determine how the PTM 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 PTIO1 and PTIO0 bits only after the TM has been switched off. Unpredictable PWM outputs will occur if the PTIO1 and PTIO0 bits are changed when the PTM is running.
- Bit 3 **PTOC**: PTM PTP Output control bit
 Compare Match Output Mode
 0: Initial low
 1: Initial high

PWM Output Mode/Single Pulse Output Mode

- 0: Active low
- 1: Active high

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

Bit 2 **PTPOL**: PTM PTP Output polarity Control
 0: Non-invert
 1: Invert

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

Bit 1 **PTCAPTS**: PTM Capture Trigger Source Selection
 0: From PTPI pin
 1: From PTCK pin

Bit 0 **PTCCLR**: Select PTM Counter clear condition
 0: PTM Comparator P match
 1: PTM Comparator A match

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

• **PTMDL 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**: PTM Counter Low Byte Register bit 7 ~ bit 0
 PTM 10-bit Counter bit 7 ~ bit 0

• **PTMDH 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**: PTM Counter High Byte Register bit 1 ~ bit 0
 PTM 10-bit Counter bit 9 ~ bit 8

• **PTMAL 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**: PTM CCRA Low Byte Register bit 7 ~ bit 0
 PTM 10-bit CCRA bit 7 ~ bit 0

• **PTMAH 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**: PTM CCRA High Byte Register bit 1 ~ bit 0
 PTM 10-bit CCRA bit 9 ~ bit 8

• **PTMRPL Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PTRP7 | PTRP6 | PTRP5 | PTRP4 | PTRP3 | PTRP2 | PTRP1 | PTRP0 |
| 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 **PTRP7~PTRP0**: PTM CCRP Low Byte Register bit 7 ~ bit 0
 PTM 10-bit CCRP bit 7 ~ bit 0

• **PTMRPH Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---|---|---|---|-------|-------|
| Name | — | — | — | — | — | — | PTRP9 | PTRP8 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

Bit 7~2 Unimplemented, read as “0”
 Bit 1~0 **PTRP9~PTRP8**: PTM CCRP High Byte Register bit 1 ~ bit 0
 PTM 10-bit CCRP bit 9 ~ bit 8

Periodic Type TM Operating Modes

The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the PTM1 and PTM0 bits in the PTMC1 register.

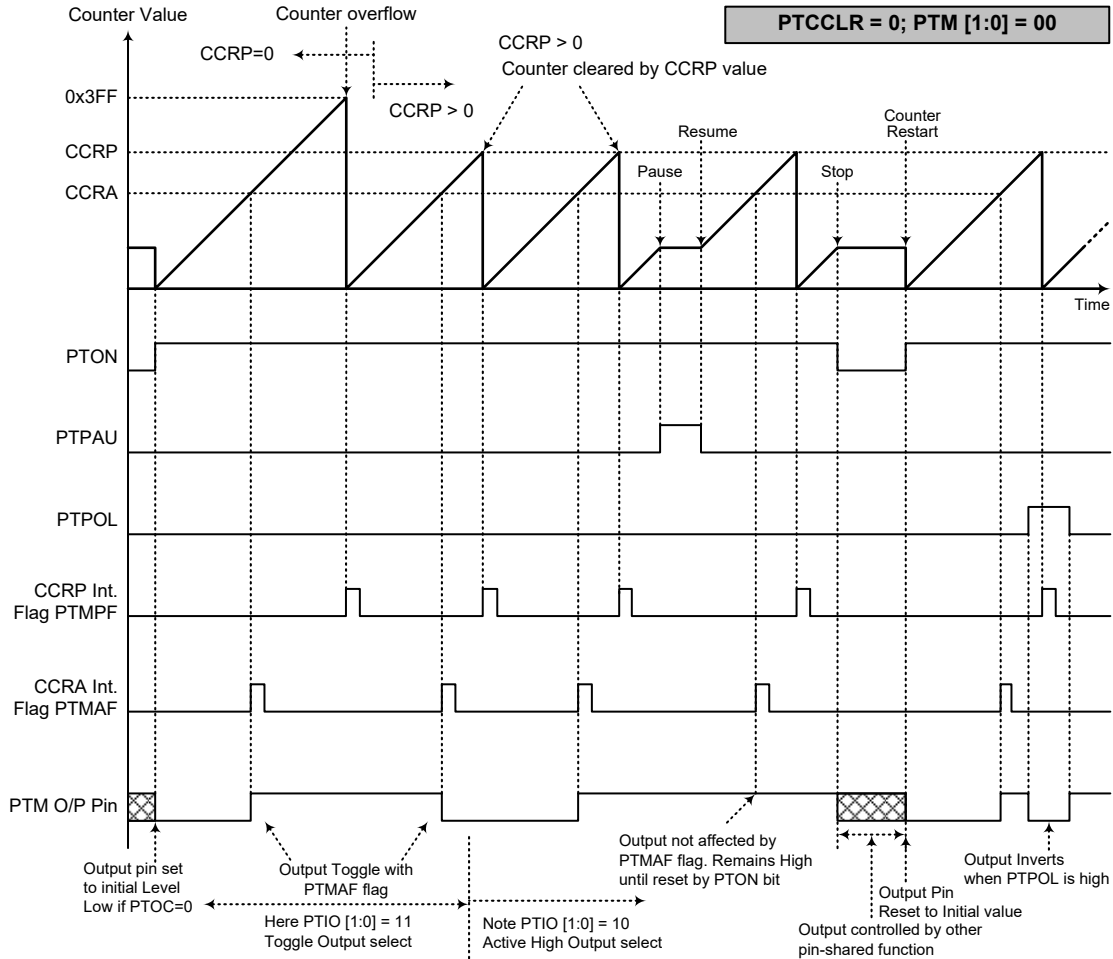
Compare Match Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 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 PTCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both PTMAF and PTMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the PTCCLR bit in the PTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTCCLR is high no PTMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be cleared to zero.

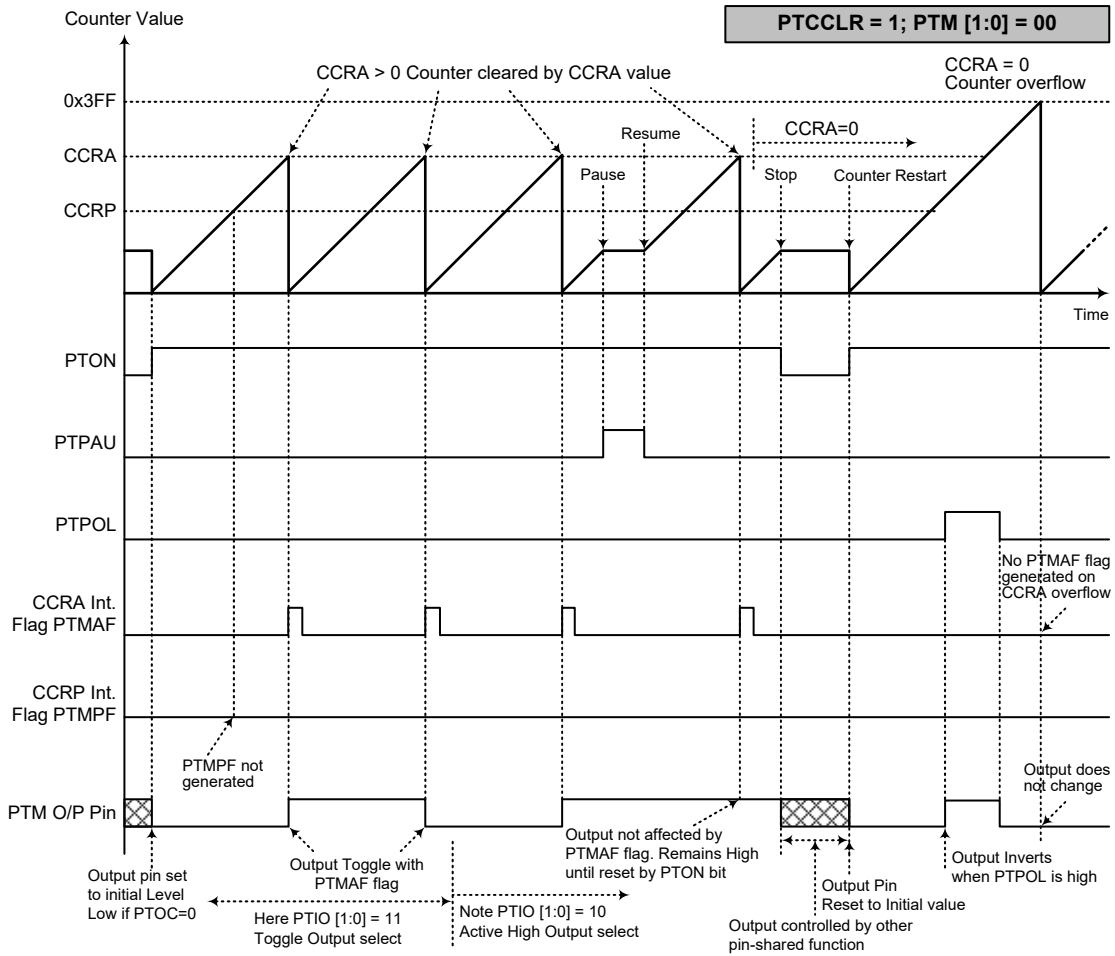
If the CCRA bits are all zero, the counter will overflow when its reaches its maximum 10-bit, 3FF Hex, value, however here the PTMAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the PTM output pin, will change state. The PTM output pin condition however only changes state when a PTMAF interrupt request flag is generated after a compare match occurs from Comparator A. The PTMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the PTM output pin. The way in which the PTM output pin changes state are determined by the condition of the PTIO1 and PTIO0 bits in the PTMC1 register. The PTM output pin can be selected using the PTIO1 and PTIO0 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 PTM output pin, which is setup after the PTON bit changes from low to high, is setup using the PTOC bit. Note that if the PTIO1 and PTIO0 bits are zero then no pin change will take place.



Compare Match Output Mode – PTCCCLR=0

- Note: 1. With PTCCCLR=0 a Comparator P match will clear the counter
2. The PTM output pin is controlled only by the PTMAF flag
3. The output pin is reset to its initial state by a PTON bit rising edge



Compare Match Output Mode – PTCCLR=1

- Note: 1. With PTCCLR=1 a Comparator A match will clear the counter
2. The PTM output pin is controlled only by the PTMAF flag
3. The output pin is reset to its initial state by a PTON bit rising edge
4. A PTMPF flag is not generated when PTCCLR=1

Timer/Counter Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 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 PTM 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 PTM output pins are not used in this mode, the pins can be used as normal I/O pins or other pin-shared functions.

PWM Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively. The PWM function within the PTM 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 PTM 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 PTCCLR 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. 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 PTOC bit in the PTMC1 register is used to select the required polarity of the PWM waveform while the two PTIO1 and PTIO0 bits are used to enable the PWM output or to force the PTM output pin to a fixed high or low level. The PTPOL bit is used to reverse the polarity of the PWM output waveform.

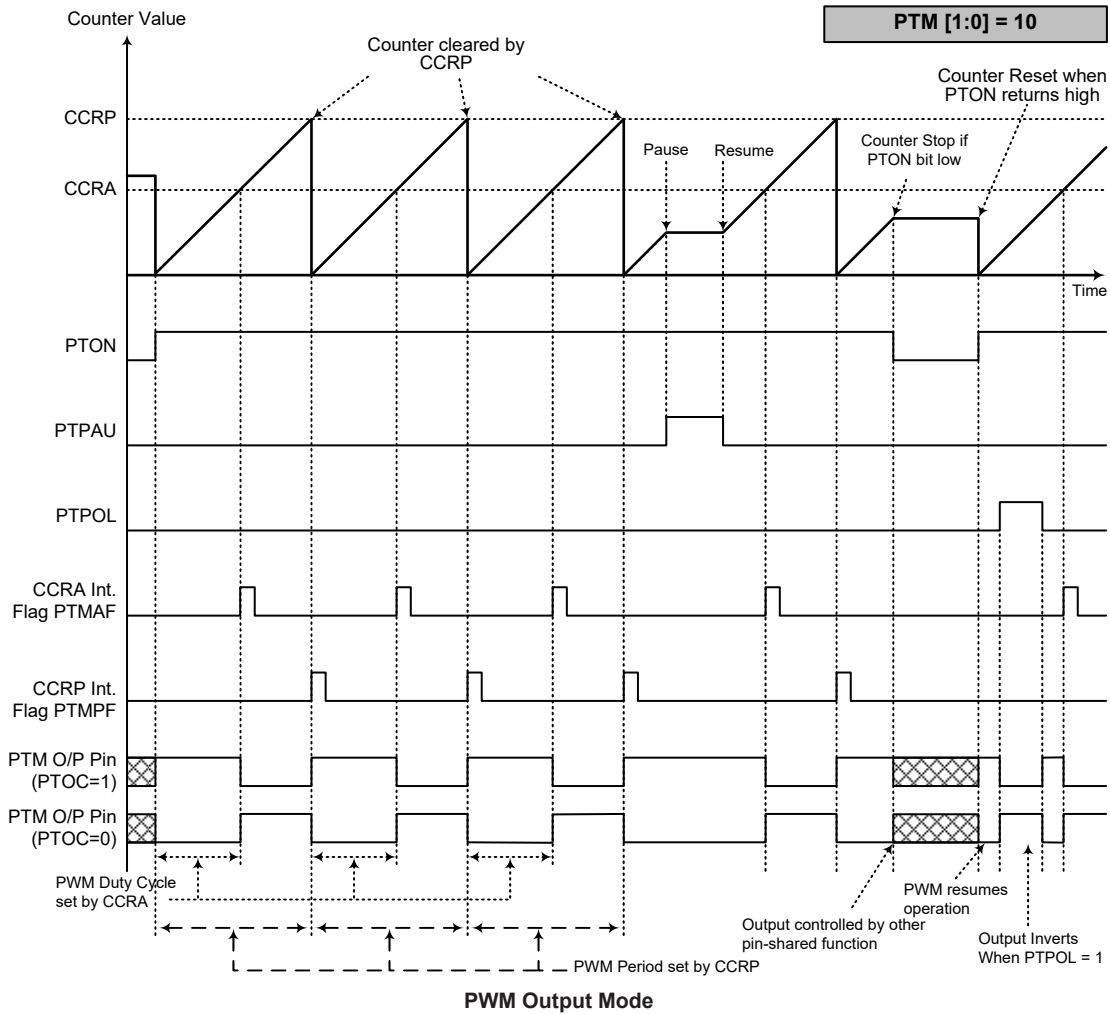
10-bit PTM, PWM Output Mode, Edge-aligned Mode

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

If $f_{SYS}=4\text{MHz}$, PTM clock source select $f_{SYS}/4$, $CCRP=512$ and $CCRA=128$,

The PTM PWM output frequency= $(f_{SYS}/4)/512=f_{SYS}/2048=1.9531\text{kHz}$, duty= $128/512=25\%$.

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



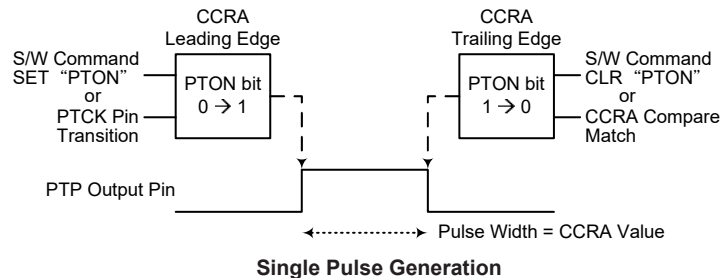
- Note:
1. Counter cleared by CCRP
 2. A counter clear sets the PWM Period
 3. The internal PWM function continues running even when PTIO[1:0]=00 or 01
 4. The PTCLLR bit has no influence on PWM operation

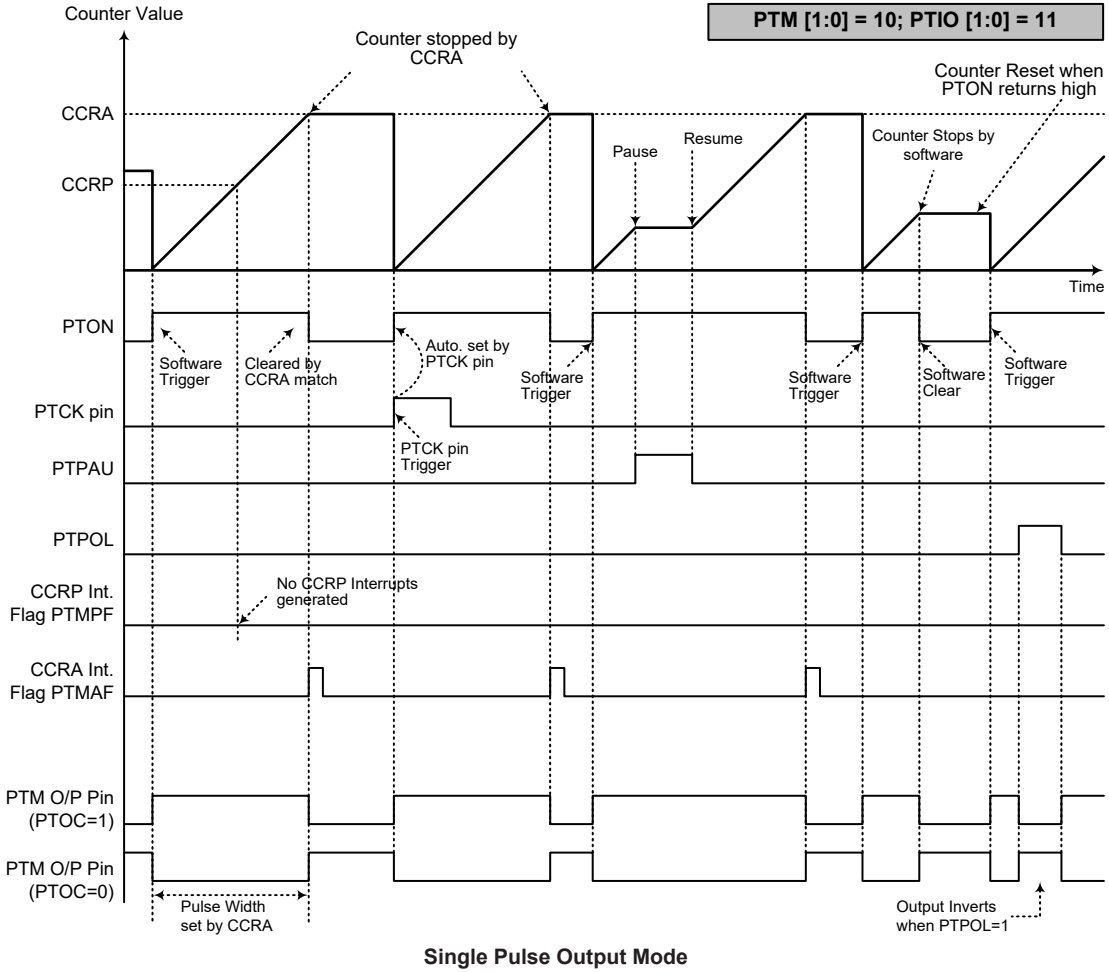
Single Pulse Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTM output pin.

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

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





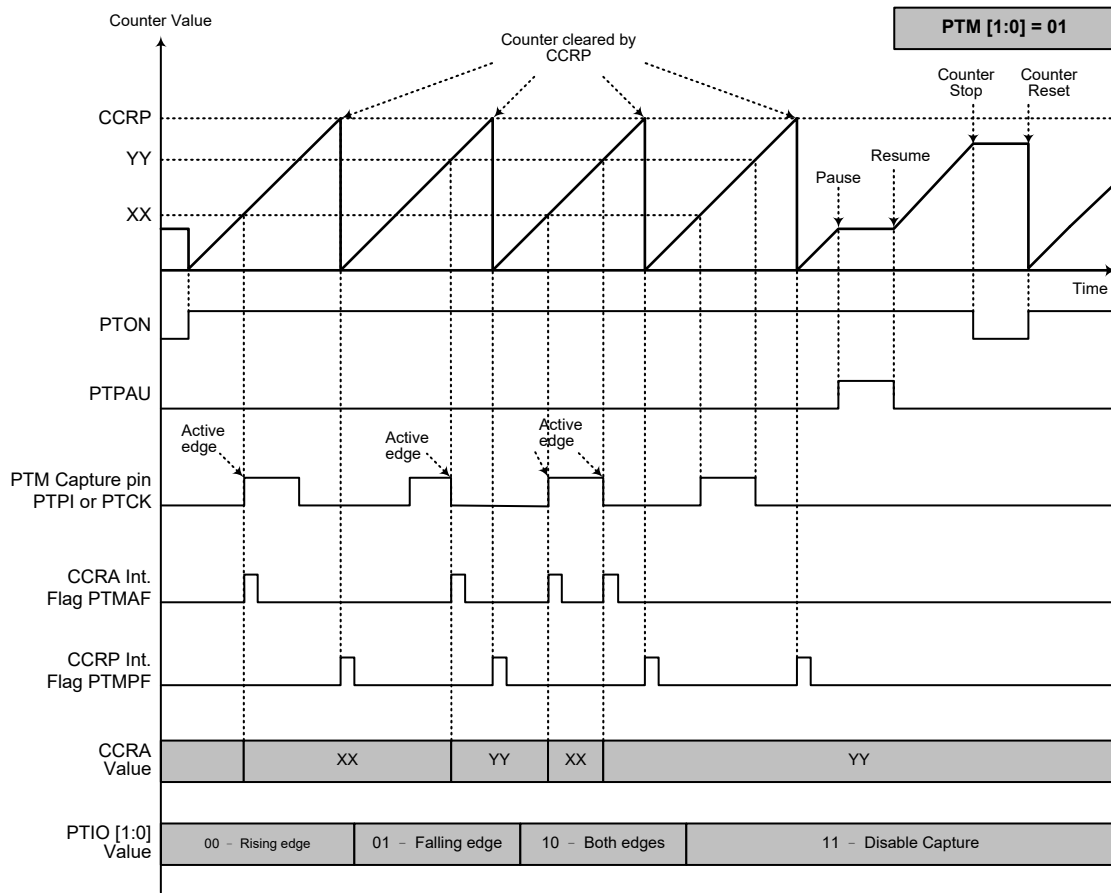
- Note: 1. Counter stopped by CCRA
 2. CCRP is not used
 3. The pulse is triggered by the PTCK pin or by setting the PTON bit high
 4. A PTCK pin active edge will automatically set the PTON bit high
 5. In the Single Pulse Output Mode, PTIO[1:0] must be set to "11" and cannot be changed.

Capture Input Mode

To select this mode bits PTM1 and PTM0 in the PTMC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the PTPI or PTCK pin which is selected using the PTCAPTS bit in the PTMC1 register. The input pin active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the PTIO1 and PTIO0 bits in the PTMC1 register. The counter is started when the PTON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the PTPI or PTCK pin the present value in the counter will be latched into the CCRA registers and a PTM interrupt generated. Irrespective of what events occur on the PTPI or PTCK pin, the counter will continue to free run until the PTON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTIO1 and PTIO0 bits can select the active trigger edge on the PTPI or PTCK pin to be a rising edge, falling edge or both edge types. If the PTIO1 and PTIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the PTPI or PTCK pin, however it must be noted that the counter will continue to run. The PTCCLR, PTOC and PTPOL bits are not used in this mode.

There are some considerations that should be noted. If PTCK is used as the capture input source, then it cannot be selected as the PTM clock source. If the captured pulse width is less than 2 timer clock periods, it may be ignored by hardware. After the counter value is latched to the CCRA registers by an active capture edge, the PTMAF flag will be set high after 0.5 timer clock period. The delay time from the active capture edge received to the action of latching counter value to CCRA registers is less than 1.5 timer clock periods.

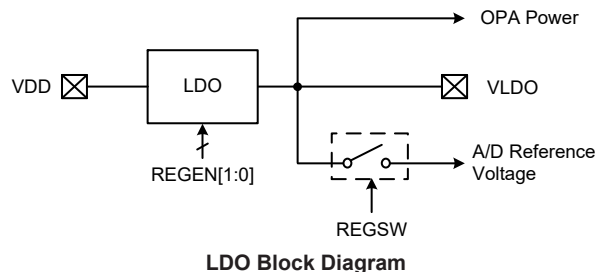


Capture Input Mode

- Note: 1. PTM[1:0]=01 and active edge set by the PTIO[1:0] bits
 2. A PTM Capture input pin active edge transfers the counter value to CCRA
 3. PTCLLR bit not used
 4. No output function – PTOC and PTPOL bits are not used
 5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero.

Voltage Regulator – LDO

The device includes a voltage regulator, LDO. The REGC register controls the regulator module to work in four modes. In the Hi-impedance mode, the LDO will be turned off and the VLDO pin will be floating. In the bypass mode, the LDO is turned off and the V_{DD} will bypass the LDO circuit and be connected to the VLDO pin directly. In the third mode the regulator is turned on, when the input voltage is larger than 2.5V, the LDO will output a fixed voltage of 2.2V on the VLDO pin. In the fourth mode the regulator is turned on, when the input voltage is larger than 3.3V, the LDO will output a fixed voltage of 3.0V. The LDO output can be used as the OPA power supply and A/D converter internal signal.



• REGC Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|---|---|---|---|---|--------|--------|
| Name | REGSW | — | — | — | — | — | REGEN1 | REGEN0 |
| R/W | R/W | — | — | — | — | — | R/W | R/W |
| POR | 0 | — | — | — | — | — | 0 | 0 |

Bit 7 **REGSW**: Switch on/off control
 0: Off
 1: On

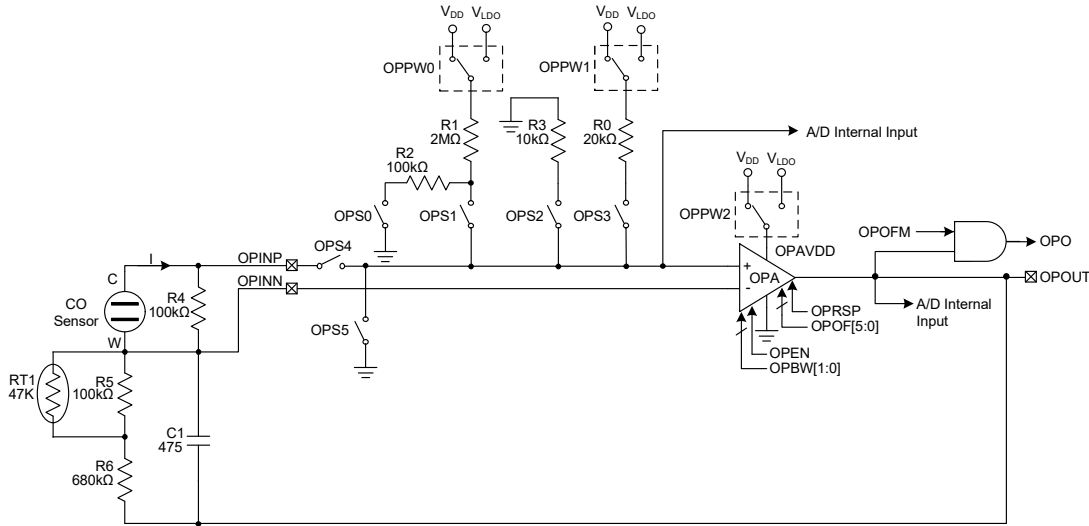
When the A/D converter reference voltage is selected to come from V_{LDO} , this bit should be set high. However when the A/D converter reference voltage is selected to come from V_{DD} , this bit should be cleared to zero, otherwise V_{LDO} will be connected together with V_{DD} , which will result in unpredictable situations such as an irreversible damage.

Bit 6~2 Unimplemented, read as “0”

Bit 1~0 **REGEN1~REGEN0**: Regulator on/off control
 00: Regulator off, Hi-impedance mode, the VLDO pin is floating
 01: Regulator off, Bypass mode, $V_{LDO}=V_{DD}$
 10: Regulator on, the VLDO pin outputs 2.2V
 11: Regulator on, the VLDO pin outputs 3.0V

CO/Gas Detector AFE

The device includes a CO/Gas Detector module which is mainly composed of an Operational Amplifier, OPA, and two groups of voltage dividers. This module together with proper peripheral circuits can be used for CO or other gas detector applications. In applications, a 100kΩ resistor is connected to the sensor in parallel by connecting its two ends to the OPINN and OPINP pins respectively. When the sensor detects the CO or other gases, the current will flow from the C end, at the same time a same amount of current from the OPOUT end will flow through the R6, R5 and RT1 resistors and finally to the W end, in which condition, the output voltage on the OPOUT pin is equal to $(V_{OPINP} + I \times (R5 // RT1 + R6))$.



CO/Gas Detector Block Diagram

CO/Gas Detector Registers

The overall CO/Gas Detector circuits are controlled by a series of registers. The OPSW register is used to control the OPA output voltage by controlling a series of switches. The OPC register is used for OPA power selection, OPA enable/disable control, OPA output status monitoring and OPA bandwidth selection. The OPVOS register is used for OPA input offset calibration voltage selection and control.

| Register Name | Bit | | | | | | | |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OPSW | OPPW1 | OPPW0 | OPS5 | OPS4 | OPS3 | OPS2 | OPS1 | OPS0 |
| OPC | OPPW2 | OPEN | OPO | — | — | — | OPBW1 | OPBW0 |
| OPVOS | OPOFM | OPRSP | OPOF5 | OPOF4 | OPOF3 | OPOF2 | OPOF1 | OPOF0 |

CO/Gas Detector Register List

• OPSW Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|------|------|------|------|------|------|
| Name | OPPW1 | OPPW0 | OPS5 | OPS4 | OPS3 | OPS2 | OPS1 | OPS0 |
| 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 **OPPW1**: Power switch 1 selection bit
 0: V_{DD}
 1: V_{LDO}

- Bit 6 **OPPW0**: Power switch 0 selection bit
 0: V_{DD}
 1: V_{LDO}
- Bit 5 **OPS5**: Switch 5 control bit
 0: Off
 1: On
- Bit 4 **OPS4**: Switch 4 control bit
 0: Off
 1: On
- Bit 3 **OPS3**: Switch 3 control bit
 0: Off
 1: On
- Bit 2 **OPS2**: Switch 2 control bit
 0: Off
 1: On
- Bit 1 **OPS1**: Switch 1 control bit
 0: Off
 1: On
- Bit 0 **OPS0**: Switch 0 control bit
 0: Off
 1: On

• **OPC Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|------|-----|---|---|---|-------|-------|
| Name | OPPW2 | OPEN | OPO | — | — | — | OPBW1 | OPBW0 |
| R/W | R/W | R/W | R | — | — | — | R/W | R/W |
| POR | 0 | 0 | 0 | — | — | — | 0 | 0 |

- Bit 7 **OPPW2**: Power switch 2 selection bit
 0: V_{DD}
 1: V_{LDO}
- Bit 6 **OPEN**: OPA enable/disable control bit
 0: Disable
 1: Enable
- Bit 5 **OPO**: OPA output status (positive logic)
 This bit is read only.
 When OPOFM=1, the OPO bit defines the OPA output status, refer to the Operational Amplifier Input Calibration section. When OPOFM= 0, this bit is fixed at 0.
- Bit 4~2 Unimplemented, read as “0”
- Bit 1~0 **OPBW1~OPBW0**: OPA bandwidth selection
 00: 5kHz
 01: 40kHz
 10: 600kHz
 11: 2MHz

| OPBW[1:0] | A/D Converter Clock Frequency (kHz) | | | | | | | |
|-----------|-------------------------------------|-------|------|-----|-----|-----|------|------|
| | 15.625 | 31.25 | 62.5 | 125 | 250 | 500 | 1000 | 2000 |
| 00 | √ | × | × | × | × | × | × | × |
| 01 | √ | √ | √ | √ | × | × | × | × |
| 10 | √ | √ | √ | √ | √ | √ | √ | √ |
| 11 | √ | √ | √ | √ | √ | √ | √ | √ |

The OPA bandwidth selection and A/D converter clock frequency configuration should follow this table to ensure higher measurement precision. Refer to the Operational Amplifier Electrical Characteristics for more details.

• **OPVOS Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | OPOFM | OPRSP | OPOF5 | OPOF4 | OPOF3 | OPOF2 | OPOF1 | OPOF0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |

- Bit 7 **OPOFM**: OPA normal operation or input offset calibration mode selection
0: Normal operation mode
1: Input offset calibration mode
- Bit 6 **OPRSP**: OPA input offset calibration reference input selection
0: OPINN is selected as reference input
1: OPINP is selected as reference input
- Bit 5~0 **OPOF5~OPOF0**: OPA input offset calibration value
This bit field is used to perform the OPA input offset calibration operation and the value after the input offset calibration can be restored into this bit field. Refer to the “Input Offset Calibration” section for more detailed information.

Input Offset Calibration

To operate in the input offset calibration mode, the OPA input pins to be used should first be enabled by setting the OPEN bit high.

Step1: Set OPOFM=1, the OPA is now under offset calibration mode. To make sure V_{OS} as minimise as possible after calibration, the input reference voltage in calibration mode should be the same as input DC operating voltage in normal mode operation.

Step2: Set OPOF[5:0]=000000, then read the OPO bit.

Step3: Let OPOF[5:0]=OPOF[5:0] + 1 then read the OPO bit.

If the OPO bit state is not changed, repeat Step3 until the OPO bit state is changed.

If the OPO bit state is changed, record the current OPOF[5:0] data as V_{OS1} , then go to Step4.

Step4: Set OPOF[5:0] = 111111, then read OPO bit.

Step5: Let OPOF[5:0] = OPOF[5:0] – 1 then read the OPO bit.

If the OPO bit state is not changed, repeat Step5 until the OPO bit state is changed.

If the OPO bit state is changed, record the current OPOF[5:0] data as V_{OS2} , then go to Step6.

Step6: Restore $V_{OS} = (V_{OS1} + V_{OS2})/2$ to OPAOF[5:0] bits, the calibration is finished.

If $(V_{OS1} + V_{OS2})/2$ is not integral, discard the decimal.

Analog to Digital Converter

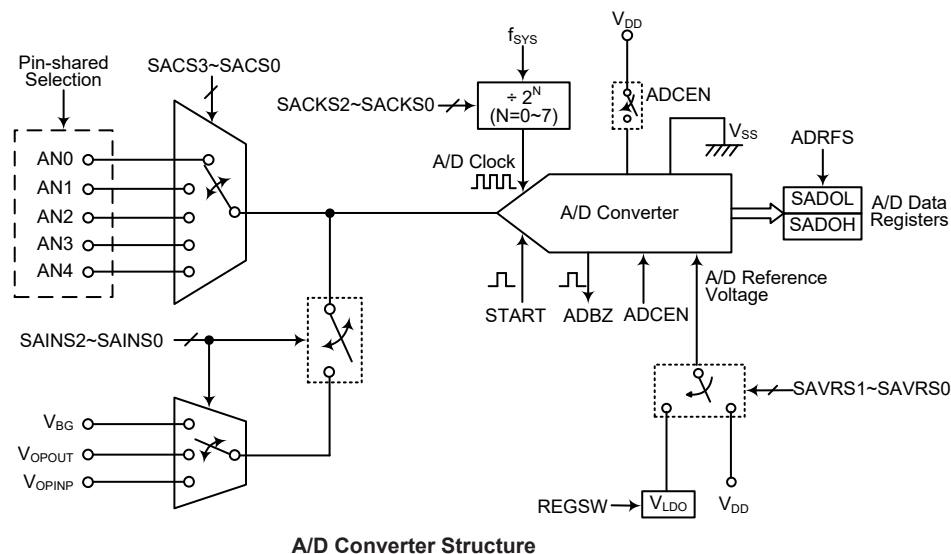
The need to interface to real world analog signals is a common requirement for many electronic systems. However, to properly process these signals by a microcontroller, they must first be converted into digital signals by A/D converters. By integrating the A/D conversion electronic circuitry into the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

A/D Converter Overview

This device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS2~SAINS0 bits together with the SACS3~SACS0 bits. When the external analog signal is to be converted, the corresponding pin-shared control bits should first be properly configured and then desired external channel input should be selected using the SAINS2~SAINS0 and SACS3~SACS0 bits. Note that when the internal analog signal is to be converted, the SAINS and SACS fields should be properly configured to avoid external channel input. More detailed information about the A/D input signal is described in the “A/D Converter Control Registers” and “A/D Converter Input Signals” sections respectively.

| External Input Channels | Internal Signal | A/D Channel Select Bits |
|-------------------------|--|-------------------------------|
| 5: AN0~AN4 | 3: V _{BG} , V _{OPOUT} , V _{OPINP} | SAINS2~SAINS0, SACS3~SACS0 |

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



A/D Converter Register Description

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

| Register Name | Bit | | | | | | | |
|-----------------|--------|--------|--------|--------|--------|--------|--------|--------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SADOL (ADRF5=0) | D3 | D2 | D1 | D0 | — | — | — | — |
| SADOL (ADRF5=1) | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SADOH (ADRF5=0) | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 |
| SADOH (ADRF5=1) | — | — | — | — | D11 | D10 | D9 | D8 |
| SADC0 | START | ADBZ | ADCEN | ADRF5 | SACS3 | SACS2 | SACS1 | SACS0 |
| SADC1 | SAINS2 | SAINS1 | SAINS0 | SAVRS1 | SAVRS0 | SACKS2 | SACKS1 | SACKS0 |

A/D Converter Register List

A/D Converter Data Registers – SADOL, SADOH

As the internal A/D converter provides a 12-bit digital conversion value, it requires two data registers to store the converted value. These are a high byte register, known as SADOH, and a low byte register, known as SADOL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. As only 12 bits of the 16-bit register space is utilised, the format in which the data is stored is controlled by the ADRF5 bit in the SADC0 register, as shown in the accompanying table. Any unused bits will be read as zero. Note that A/D data registers contents will be unchanged if the A/D converter is disabled.

| ADRF5 | SADOH | | | | | | | | SADOL | | | | | | | |
|-------|-------|-----|----|----|-----|-----|----|----|-------|----|----|----|----|----|----|----|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

A/D Converter Data Registers

A/D Converter Control Registers – SADC0, SADC1

To control the function and operation of the A/D converter, two control registers known as SADC0 and SADC1 are provided. These 8-bit registers define functions such as the selection of which analog channel is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. As the device contains only one actual analog to digital converter hardware circuit, each of the external or internal analog signal inputs must be routed to the converter. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. The SAINS2~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the internal analog signal or external analog channel input.

The relevant pin-shared function selection bits determine which pins on I/O Ports are used as analog inputs for the A/D converter input and which pins are not to be used as the A/D converter input. When the pin is selected to be an A/D input, its original function whether it is an I/O or other pin-shared function will be removed. In addition, any internal pull-high resistor connected to the pin will be automatically removed if the pin is selected to be an A/D converter input.

• **SADC0 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|------|-------|-------|-------|-------|-------|-------|
| Name | START | ADBZ | ADCEN | ADRFS | SACS3 | SACS2 | SACS1 | SACS0 |
| R/W | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- Bit 7** **START:** Start the A/D conversion
0→1→0: Start
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process.
- Bit 6** **ADBZ:** A/D converter busy flag
0: No A/D conversion is in progress
1: A/D conversion is in progress
This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set to 1 to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is complete.
- Bit 5** **ADCEN:** A/D converter function enable control
0: Disable
1: Enable
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is cleared to zero, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair known as SADOH and SADOL will be unchanged.
- Bit 4** **ADRFS:** A/D converter data format select
0: A/D converter data format → SADOH=D[11:4]; SADOL=D[3:0]
1: A/D converter data format → SADOH=D[11:8]; SADOL=D[7:0]
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D data register section.
- Bit 3~0** **SACS3~SACS0:** A/D converter external analog channel input select
0000: AN0
0001: AN1
0010: AN2
0011: AN3
0100: AN4
0101~1111: undefined, floating
These bits are used to select external analog input channel. When the external analog channel is selected to be converted, the SAINS bit field should be set to “000” or “101”~”111”. Refer to “A/D Converter Input Signal Selection” table for more details.

• **SADC1 Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|--------|--------|--------|--------|--------|--------|--------|--------|
| Name | SAINS2 | SAINS1 | SAINS0 | SAVRS1 | SAVRS0 | SACKS2 | SACKS1 | SACKS0 |
| 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 7~5** **SAINS2~SAINS0:** A/D converter input signal select
000: External input – External analog channel input
001: Internal input – Internal Bandgap reference voltage, V_{BG}
010: Internal input – Internal OPA output, V_{OPOUT}
011: Internal input – Internal OPA positive input, V_{OPINP}
100: Internal input – unused, connected to ground
101~111: External input – External analog channel input

Care must be taken if the SAINS2~SAINS0 bits are set to “001”~“011” to select the internal analog signal to be converted. When the internal analog signal is selected to be converted, the external input pin must never be selected as the A/D input signal by properly setting the SACS3~SACS0 bits with a value from 0101 to 1111. Otherwise, the external channel input will be connected together with the internal analog signal. This will result in unpredictable situations such as an irreversible damage.

- Bit 4~3 **SAVRS1~SAVRS0**: A/D converter reference voltage select
- 00: LDO output voltage, V_{LDO}
 - 01: Internal A/D converter power supply, V_{DD}
 - 1x: LDO output voltage, V_{LDO}

These bits are used to select the A/D converter reference voltage. Care must be taken if the SAVRS1~SAVRS0 bits are set to “01” to select the internal A/D converter power supply as the reference voltage source. When the internal A/D converter power supply is selected as the reference voltage, the LDO output voltage can not be configured as the reference voltage input by properly configuring the REGSW bit in the REGC register. Otherwise, the LDO output voltage will be connected to the A/D converter along with the A/D converter power supply.

- Bit 2~0 **SACKS2~SACKS0**: A/D conversion clock source select
- 000: f_{SYS}
 - 001: $f_{SYS}/2$
 - 010: $f_{SYS}/4$
 - 011: $f_{SYS}/8$
 - 100: $f_{SYS}/16$
 - 101: $f_{SYS}/32$
 - 110: $f_{SYS}/64$
 - 111: $f_{SYS}/128$

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

A/D Converter Reference Voltage

The reference voltage supply to the A/D converter can be supplied from the LDO output, V_{LDO} , or from the A/D converter power supply, V_{DD} . The desired selection is made using the SAVRS1 and SAVRS0 bits. When the SAVRS bit field is set to “00” or “1x”, the A/D converter reference voltage will come from the LDO output voltage. When the SAVRS bit field is set to “01”, the A/D converter reference voltage will come from the A/D converter power supply. However, if the A/D converter power supply is selected as the reference voltage, the LDO output voltage can not be configured as the reference voltage input function by properly configuring the REGSW bit in the REGC register to avoid the internal connection between the V_{LDO} and V_{DD} . The analog input values must not be allowed to exceed the selected reference voltage.

| SAVRS[1:0] | Reference | Description |
|------------|-----------|-------------------------------------|
| 00, 1x | V_{LDO} | LDO output voltage |
| 01 | V_{DD} | Internal A/D converter power supply |

A/D Converter Reference Voltage Selection

A/D Converter Input Signals

All the external A/D analog channel input pins are pin-shared with the I/O pins as well as other functions. The corresponding control bits for each A/D external input pin in the PxSn register determine whether the input pins are setup as A/D converter analog inputs or whether they have other functions. If the pin is setup to be as an A/D analog channel input, the original pin functions will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D inputs. Note that it

is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D input as when the pin-shared function control bits enable an A/D input, the status of the port control register will be overridden.

If the SAINS2~SAINS0 bits are set to “000” or “101~111”, the external analog channel input is selected to be converted and the SACS3~SACS0 bits can determine which actual external channel is selected to be converted. If the SAINS2~SAINS0 bits are set to “001~011”, the internal Bandgap reference voltage, OPA output voltage or the OPA positive input voltage is selected to be converted. Note that if the internal analog signal is selected to be converted, the external input channel determined by the SACS3~SACS0 bits must be switched to a non-existent A/D input channel by properly setting the SACS bit field with a value from 0101 to 1111. Otherwise, the external channel input will be connected together with the internal analog signal, which will result in unpredictable errors.

| SAINS[2:0] | SACS[3:0] | Input Signals | Description |
|-----------------|-----------|--------------------|---|
| 000, 101~111 | 0000~0100 | AN0~AN4 | External pin analog input |
| | 0101~1111 | — | Non-existent channel, input is floating |
| 001 | 0101~1111 | V _{BG} | Internal Bandgap reference voltage |
| 010 | 0101~1111 | V _{OPOUT} | Internal OPA output voltage |
| 011 | 0101~1111 | V _{OPINP} | Internal OPA positive input voltage |
| 100 | 0101~1111 | GND | Unused, connected to ground |

A/D Converter Input Signal Selection

A/D Converter Operation

The START bit in the SADC0 register is used to start the AD conversion. When the microcontroller sets this bit from low to high and then low again, an analog to digital conversion cycle will be initiated.

The ADBZ bit in the SADC0 register is used to indicate whether the analog to digital conversion process is in progress or not. This bit will be automatically set to 1 by the microcontroller after an A/D conversion is successfully initiated. When the A/D conversion is complete, the ADBZ will be cleared to 0. In addition, the corresponding A/D interrupt request flag will be set in the interrupt control register, and if the interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D internal interrupt signal will direct the program flow to the associated A/D internal interrupt address for processing. If the A/D internal interrupt is disabled, the microcontroller can poll the ADBZ bit in the SADC0 register to check whether it has been cleared as an alternative method of detecting the end of an A/D conversion cycle.

The clock source for the A/D converter, which originates from the system clock f_{SYS} , can be chosen to be either f_{SYS} or a subdivided version of f_{SYS} . The division ratio value is determined by the SACKS2~SACKS0 bits in the SADC1 register. Although the A/D clock source is determined by the system clock f_{SYS} and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended range of permissible A/D clock period, t_{ADCK} , is from 0.5 μ s to 10 μ s, care must be taken for system clock frequencies. For example, as the system clock operates at a frequency of 8MHz, the SACKS2~SACKS0 bits should not be set to 000, 001 or 111. Doing so will give A/D clock periods that are less than the minimum A/D clock period or larger than the maximum A/D clock period, which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk * show where special care must be taken.

| f _{sys} | A/D Clock Period (t _{ADCK}) | | | | | | | |
|------------------|--|--|--|--|---|---|---|--|
| | SACKS[2:0] = 000 (f _{sys}) | SACKS[2:0] = 001 (f _{sys} /2) | SACKS[2:0] = 010 (f _{sys} /4) | SACKS[2:0] = 011 (f _{sys} /8) | SACKS[2:0] = 100 (f _{sys} /16) | SACKS[2:0] = 101 (f _{sys} /32) | SACKS[2:0] = 110 (f _{sys} /64) | SACKS[2:0] = 111 (f _{sys} /128) |
| | 1MHz | 1μs | 2μs | 4μs | 8μs | 16μs * | 32μs * | 64μs * |
| 2MHz | 500ns | 1μs | 2μs | 4μs | 8μs | 16μs * | 32μs * | 64μs * |
| 4MHz | 250ns * | 500ns | 1μs | 2μs | 4μs | 8μs | 16μs * | 32μs * |
| 8MHz | 125ns * | 250ns * | 500ns | 1μs | 2μs | 4μs | 8μs | 16μs * |

A/D Clock Period Examples

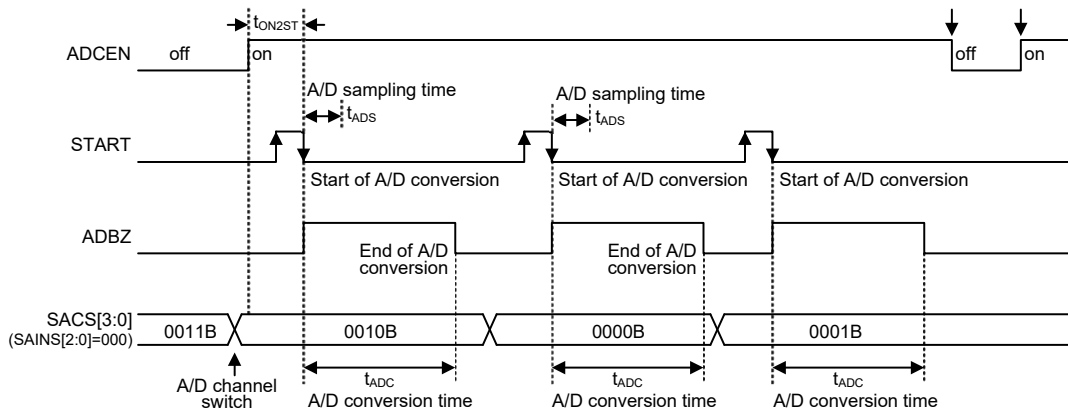
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry a certain delay, as indicated in the timing diagram, must be allowed before an A/D conversion is initiated. Even if no pins are selected for use as A/D inputs, if the ADCEN bit is high, then some power will still be consumed. In power conscious applications it is therefore recommended that the ADCEN is set low to reduce power consumption when the A/D converter function is not being used.

Conversion Rate and Timing Diagram

A complete A/D conversion contains two parts, data sampling and data conversion. The data sampling which is defined as t_{ADS} takes 4 A/D clock periods and the data conversion takes 12 A/D clock periods. Therefore a total of 16 A/D clock periods for an external input A/D conversion which is defined as t_{ADC} are necessary.

$$\text{Maximum single A/D conversion rate} = 1/(\text{A/D clock period} \times 16)$$

The accompanying diagram shows graphically the various stages involved in an analog to digital conversion process and its associated timing. After an A/D conversion process has been initiated by the application program, the microcontroller internal hardware will begin to carry out the conversion, during which time the program can continue with other functions. The time taken for the A/D conversion is 16 t_{ADCK} clock periods where t_{ADCK} is equal to the A/D clock period.



A/D Conversion Timing – External Channel Input

Summary of A/D Conversion Steps

The following summarises the individual steps that should be executed in order to implement an A/D conversion process.

- Step 1
 Select the required A/D conversion clock by correctly programming bits SACKS2~SACKS0 in the SADC1 register.

- Step 2
Enable the A/D by setting the ADCEN bit in the SADC0 register to one.
- Step 3
Select which signal is to be connected to the internal A/D converter by correctly configuring the SAINS2~SAINS0 bits.
Select the external channel input to be converted, go to Step 4.
Select the internal analog signal to be converted, go to Step 5.
- Step 4
If the A/D input signal comes from the external channel input selected by configuring the SAINS bit field, the corresponding pins should be configured as A/D input function by configuring the relevant pin-shared function control bits. The desired analog channel then should be selected by configuring the SACS bit field. After this step, go to Step 6.
- Step 5
Before the A/D input signal is selected to come from the internal analog signal by configuring the SAINS bit field, the corresponding external input pin must be switched to a non-existed channel input by setting the SACS3~SACS0 bits with a value from 0101 to 1111. The desired internal analog signal then can be selected by configuring the SAINS bit field. After this step, go to Step 6.
- Step 6
Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC1 register. If the power supply is selected as the reference voltage, the default reference source should be disconnected by properly configuring the related bit.
- Step 7
Select A/D converter output data format by setting the ADRFS bit in the SADC0 register.
- Step 8
If the A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.
- Step 9
The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.
- Step 10
If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.

Note: When checking for the end of the conversion process, if the method of polling the ADBZ bit in the SADC0 register is used, the interrupt enable step above can be omitted.

Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D internal circuitry can be switched off to reduce power consumption, by clearing bit ADCEN to 0 in the SADC0 register. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines. If the A/D converter input lines are used as normal I/Os, then care must be taken as if the input voltage is not at a valid logic level, then this may lead to some increase in power consumption.

A/D Conversion Function

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

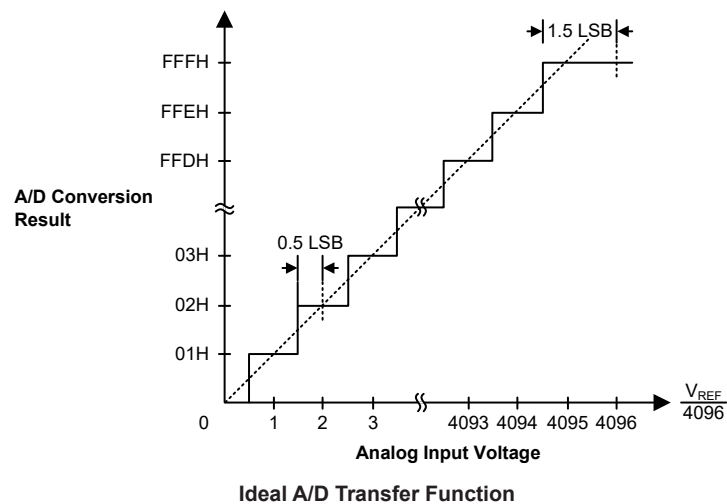
$$1 \text{ LSB} = V_{REF} \div 4096$$

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

$$\text{A/D input voltage} = \text{A/D output digital value} \times V_{REF} \div 4096$$

The diagram shows the ideal transfer function between the analog input value and the digitised output value for the A/D converter. Except for the digitised zero value, the subsequent digitised values will change at a point 0.5 LSB below where they would change without the offset, and the last full scale digitised value will change at a point 1.5 LSB below the V_{REF} level.

Note that here the V_{REF} voltage is the actual A/D converter reference voltage determined by the SAVRS field.



A/D Conversion Programming Examples

The following two programming examples illustrate how to setup and implement an A/D conversion. In the first example, the method of polling the ADBZ bit in the SADC0 register is used to detect when the conversion cycle is complete, whereas in the second example, the A/D interrupt is used to determine when the conversion is complete.

Example: using an ADBZ polling method to detect the end of conversion

```

clr ADE           ; disable ADC interrupt
mov a,03H
mov SADC1,a       ; select input signal from external channel, reference voltage from
                  ; VLDO, fsys/8 as A/D clock
mov a,01H
mov PBS0,a        ; setup PBS0 to configure pin AN0
mov a,20H
mov SADC0,a       ; enable A/D and connect AN0 channel to A/D converter
:
:
start_conversion:
clr START         ; high pulse on start bit to initiate conversion
set START         ; reset A/D

```



```
clr START          ; start A/D
polling_EOC:
sz  ADBZ           ; poll the SADC0 register ADBZ bit to detect end of A/D conversion
jmp polling_EOC   ; continue polling
mov a,SADOL        ; read low byte conversion result value
mov SADOL_buffer,a ; save result to user defined register
mov a,SADOH        ; read high byte conversion result value
mov SADOH_buffer,a ; save result to user defined register
:
:
jmp start_conversion ; start next A/D conversion
```

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

```
clr ADE           ; disable ADC interrupt
mov a,03H
mov SADC1,a       ; select input signal from external channel, reference voltage from
                  ; VLDO, fSYS/8 as A/D clock
mov a,01H         ; setup PBS0 to configure pin AN0
mov PBS0,a
mov a,20H
mov SADC0,a       ; enable A/D and connect AN0 channel to A/D converter
:
:
start_conversion:
clr START         ; high pulse on START bit to initiate conversion
set  START        ; reset A/D
clr  START        ; start A/D
clr  ADF          ; clear ADC interrupt request flag
set  ADE          ; enable ADC interrupt
set  EMI          ; enable global interrupt
:
:
                  ; ADC interrupt service routine
ADC_ISR:
mov  acc_stack,a  ; save ACC to user defined memory
mov  a,STATUS
mov  status_stack,a ; save STATUS to user defined memory
:
:
mov  a,SADOL      ; read low byte conversion result value
mov  SADOL_buffer,a ; save result to user defined register
mov  a,SADOH      ; read high byte conversion result value
mov  SADOH_buffer,a ; save result to user defined register
:
:
EXIT_INT_ISR:
mov  a,status_stack
mov  STATUS,a     ; restore STATUS from user defined memory
mov  a,acc_stack  ; restore ACC from user defined memory
reti
```

Universal Serial Interface Module – USIM

The device contains a Universal Serial Interface Module, which includes the four-line SPI interface, the two-line I²C interface and the two-line UART 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, I²C or UART based hardware such as sensors, Flash or EEPROM memory, etc. The USIM interface pins are pin-shared with other I/O pins therefore the USIM interface functional pins must first be selected using the corresponding pin-shared function selection bits. As all the interface types share the same pins and registers, the choice of whether the UART, SPI or I²C type is used is made using the UART mode selection bit, named UMD, and the SPI/I²C operating mode control bits, named SIM2~SIM0, in the SIMC0 register. These pull-high resistors of the USIM pin-shared I/O are selected using pull-high control registers when the USIM function is enabled and the corresponding pins are used as USIM input pins.

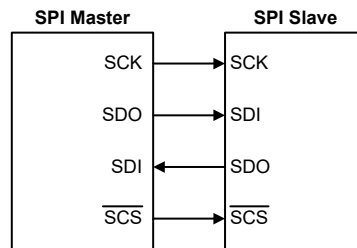
SPI Interface

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

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

SPI Interface Operation

The SPI interface is a full duplex synchronous serial data link. It is a four-line interface with pin names SDI, SDO, SCK and \overline{SCS} . Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, the SCK pin is the Serial Clock line and \overline{SCS} is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I²C/UART function pins, the SPI interface pins must first be selected by configuring the pin-shared function selection bits and setting the correct bits in the SIMC0 and SIMC2 registers. Communication between devices connected to the SPI interface is carried out in a slave/master mode with all data transfer initiations being implemented by the master. The Master also controls the clock signal. As the device only contains a single \overline{SCS} pin only one slave device can be utilized. The \overline{SCS} pin is controlled by software, set CSEN bit to 1 to enable \overline{SCS} pin function, set CSEN bit to 0 the \overline{SCS} pin will be floating state.

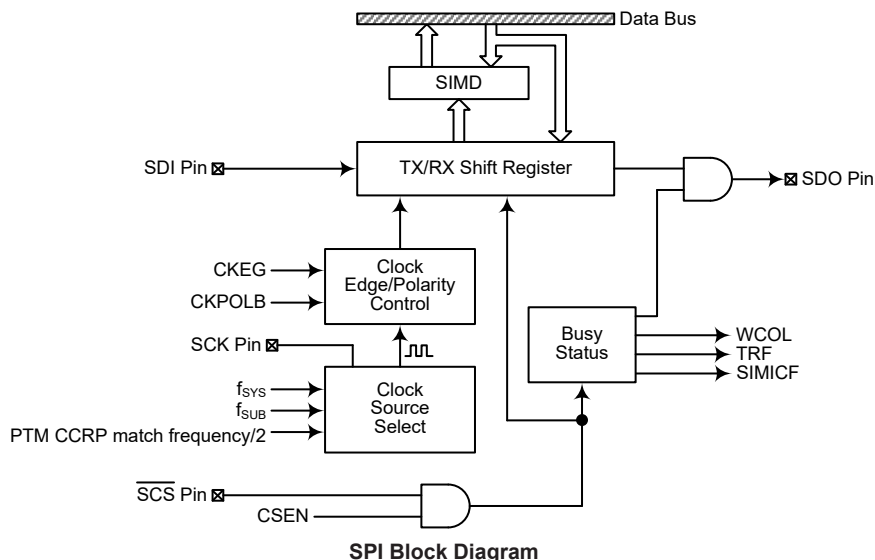


SPI Master/Slave Connection

The SPI function in the device offers the following features:

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

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



SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two control registers, SIMC0 and SIMC2. Note that the SIMC2 and SIMD registers and their POR values are only available when the SPI mode is selected by properly configuring the UMD and SIM2~SIM0 bits in the SIMC0 register.

| Register Name | Bit | | | | | | | |
|---------------|------|------|--------|------|---------|---------|-------|--------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SIMC0 | SIM2 | SIM1 | SIM0 | UMD | 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**: USIM SPI/I²C data register bit 7~bit 0

SPI Control Registers

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

• **SIMC0 Register**

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C 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 PTM CCRP match frequency/2
 101: SPI slave mode
 110: I²C slave mode
 111: Undefined

When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM 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 PTM and f_{SUB} . If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **UMD**: UART mode selection bit
 0: SPI or I²C mode
 1: UART mode

This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be set low for SPI or I²C mode.

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

Bit 1 **SIMEN**: USIM SPI/I²C Enable Control
 0: Disable
 1: Enable

The bit is the overall on/off control for the USIM SPI/I²C interface. When the SIMEN bit is cleared to zero to disable the USIM SPI/I²C interface, the SDI, SDO, SCK and SCS, or SDA and SCL lines will lose their SPI or I²C function and the USIM operating current will be reduced to a minimum value. When the bit is high the USIM SPI/I²C interface is enabled. If the USIM is configured to operate as an SPI interface via the UMD and 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 USIM is configured to operate as

an I²C interface via the UMD and 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**: USIM SPI Incomplete Flag
 0: USIM SPI incomplete condition is not occurred
 1: USIM SPI incomplete condition is occurred

This bit is only available when the USIM 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 high but the SCS line is pulled high by the external master device before the SPI data transfer is completely finished, the SIMICF bit will be set high 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 high if the SIMICF bit is set high 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 SCS pin control
 0: Disable
 1: Enable

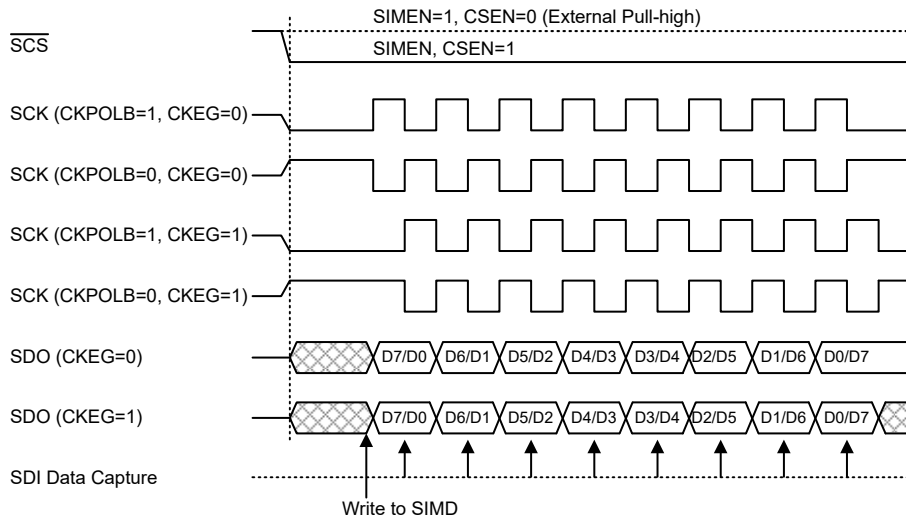
The CSEN bit is used as an enable/disable for the SCS pin. If this bit is low, then the SCS pin will be disabled and placed into a floating condition. If the bit is high the SCS pin will be enabled and used as a select pin.

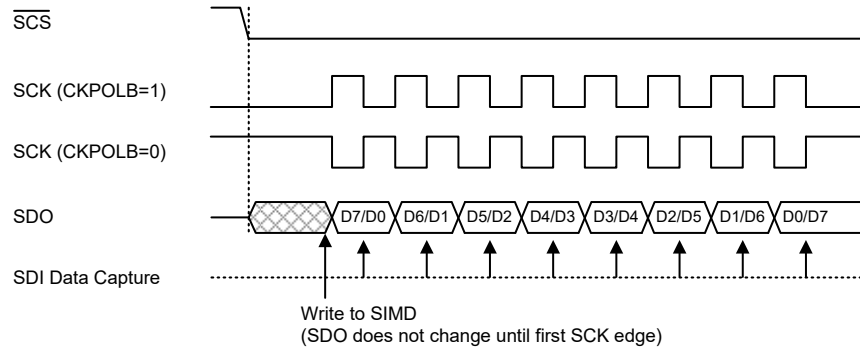
- Bit 1 **WCOL:** SPI write collision flag
 0: No collision
 1: Collision
- The WCOL flag is used to detect if a data collision has occurred. If this bit is high it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. The bit can be cleared to zero by the application program.
- Bit 0 **TRF:** SPI Transmit/Receive complete flag
 0: SPI data is being transferred
 1: SPI data transmission is completed
- The TRF bit is the Transmit/Receive Complete flag and is set “1” automatically when an SPI data transmission is completed, but must set to “0” by the application program. It can be used to generate an interrupt.

SPI Communication

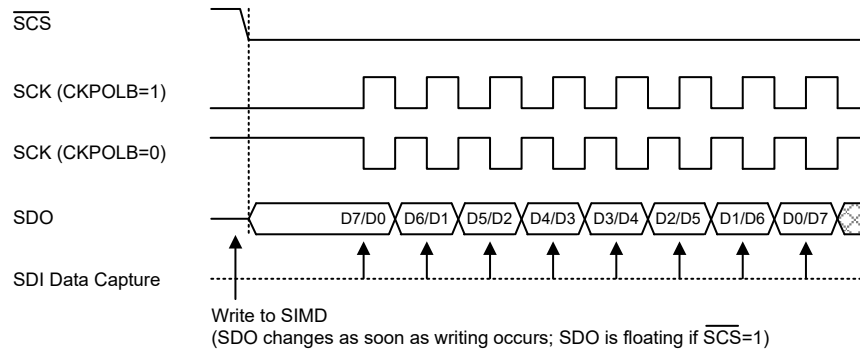
After the SPI interface is enabled by setting the SIMEN bit high, then in the Master Mode, when data is written to the SIMD register, transmission/reception will begin simultaneously. When the data transfer is completed, the TRF flag will be set high automatically, but must be cleared using the application program. In the Slave Mode, when the clock signal from the master has been received, any data in the SIMD register will be transmitted and any data on the SDI pin will be shifted into the SIMD register. The master should output an \overline{SCS} signal to enable the slave devices before a clock signal is provided. The slave data to be transferred should be well prepared at the appropriate moment relative to the 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 will continue to function in certain IDLE Modes if the clock source used by the SPI interface is still active.



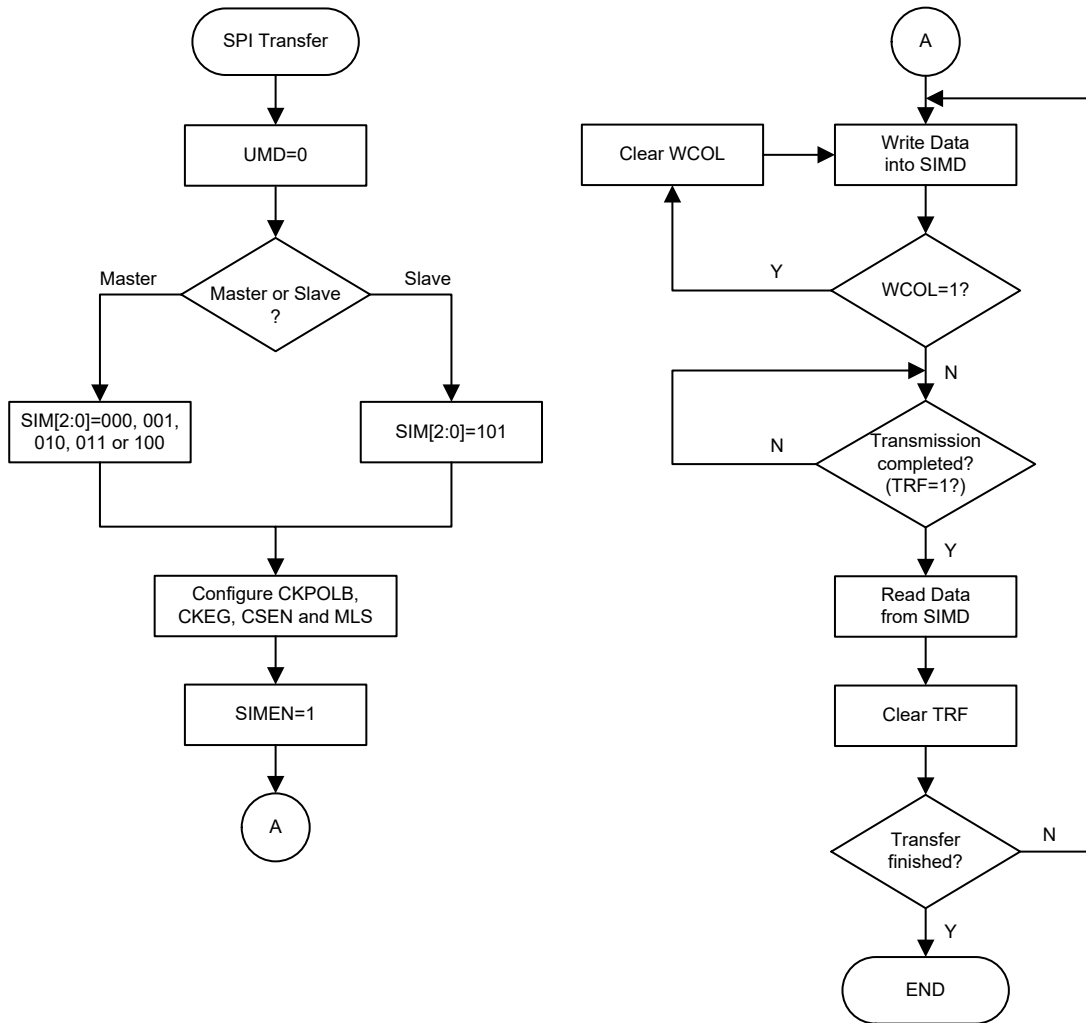


SPI Slave Mode Timing – CKEG=0



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

SPI Slave Mode Timing – CKEG=1



SPI Transfer Control Flowchart

SPI Bus Enable/Disable

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

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

SPI Operation Steps

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

The CSEN bit in the SIMC2 register controls the overall function of the SPI interface. Setting this bit high will enable the SPI interface by allowing the \overline{SCS} line to be active, which can then be used to control the SPI interface. If the CSEN bit is low, the SPI interface will be disabled and the \overline{SCS} line will be in a floating condition and can therefore not be used for control of the SPI interface. If the CSEN bit and the SIMEN bit in the SIMC0 are set high, this will place the SDI

line in a floating condition and the SDO line high. If in Master Mode the SCK line will be either high or low depending upon the clock polarity selection bit CKPOLB in the SIMC2 register. If in Slave Mode the SCK line will be in a floating condition. If the SIMEN bit is low, then the bus will be disabled and \overline{SCS} , SDI, SDO and SCK will all become I/O pins or the other functions using the corresponding pin-shared control bits. In the Master Mode the Master will always generate the clock signal. The clock and data transmission will be initiated after data has been written into the SIMD register. In the Slave Mode, the clock signal will be received from an external master device for both data transmission and reception. The following sequences show the order to be followed for data transfer in both Master and Slave Mode.

Master Mode

- Step 1
Select the SPI Master mode and clock source using the UMD and 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 USIM 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 UMD and 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.

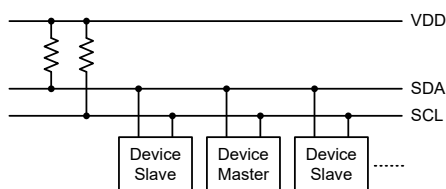
- Step 5
Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6
Check the TRF bit or wait for an USIM SPI serial bus interrupt.
- Step 7
Read data from the SIMD register.
- Step 8
Clear TRF.
- Step 9
Go to step 4.

Error Detection

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

I²C Interface

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

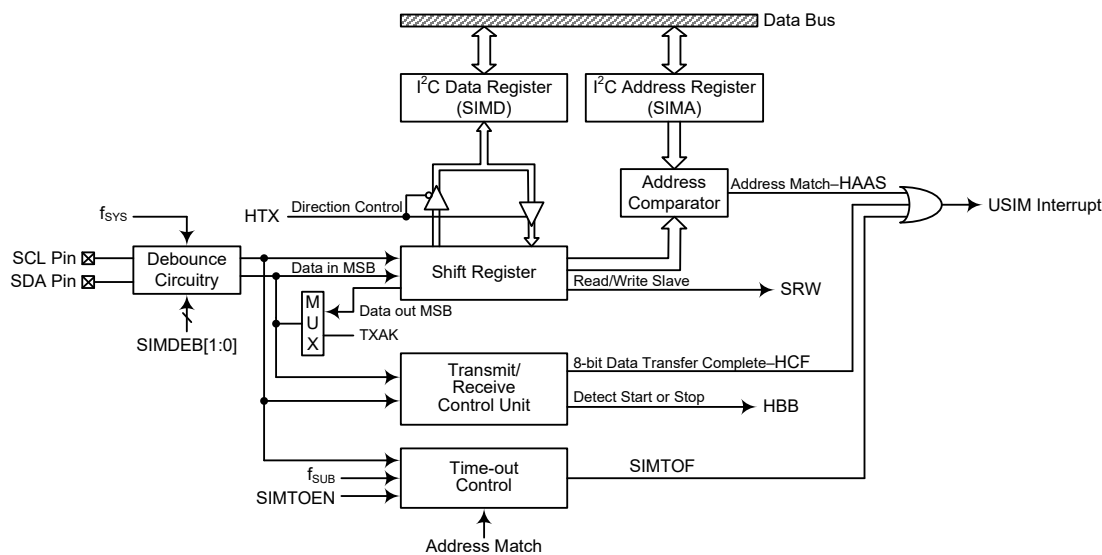


I²C Master Slave Bus Connection

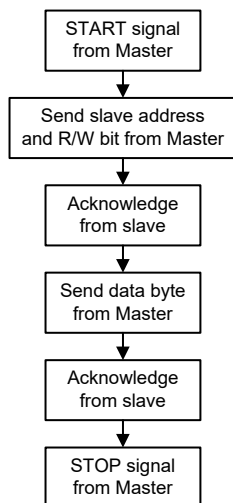
I²C Interface Operation

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

When two devices communicate with each other on the bidirectional I²C bus, one is known as the master device and one as the slave device. Both master and slave can transmit and receive data, however, it is the master device that has overall control of the bus. For the device, which only operates in slave mode, there are two methods of transferring data on the I²C bus, the slave transmit mode and the slave receive mode. The pull-high control function pin-shared with SCL/SDA pin is still applicable even if I²C device is activated and the related internal pull-high register could be controlled by its corresponding pull-high control register.



I²C Block Diagram



I²C Interface Operation

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

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

I²C Minimum f_{SYS} Frequency Requirements

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. Note that the SIMC1, SIMD, SIMA and SIMTOC registers and their POR values are only available when the I²C mode is selected by properly configuring the UMD and SIM2~SIM0 bits in the SIMC0 register.

| Register Name | Bit | | | | | | | |
|---------------|---------|--------|---------|---------|---------|---------|---------|---------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SIMC0 | SIM2 | SIM1 | SIM0 | UMD | SIMDEB1 | SIMDEB0 | SIMEN | SIMICF |
| SIMC1 | HCF | HAAS | HBB | HTX | TXAK | SRW | IAMWU | RXAK |
| SIMD | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
| SIMA | SIMA6 | SIMA5 | SIMA4 | SIMA3 | SIMA2 | SIMA1 | SIMA0 | 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**: USIM SPI/I²C data register bit 7~bit 0

I²C Address Register

The SIMA register is also used by the SPI interface but has the name of SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. Bit 7~1 of the SIMA register define the device slave address. Bit 0 is not defined. When a master device, which is connected to the I²C bus, sends out an address, which matches the slave address in the SIMA register, the slave device will be selected.

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

• SIMC0 Register

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C 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 PTM CCRP match frequency/2
- 101: SPI slave mode
- 110: I²C slave mode
- 111: Undefined

When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM 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 PTM and f_{SUB} . If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 **UMD**: UART mode selection bit

- 0: SPI or I²C mode
- 1: UART mode

This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be set low for SPI or I²C mode.

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 USIM is configured as the I²C interface function by setting the UMD bit to "0" and SIM2~SIM0 bits to "110".

Bit 1 **SIMEN**: USIM SPI/I²C Enable Control

- 0: Disable
- 1: Enable

The bit is the overall on/off control for the USIM SPI/I²C interface. When the SIMEN bit is cleared to zero to disable the USIM SPI/I²C interface, the SDI, SDO, SCK and SCS, or SDA and SCL lines will lose their SPI or I²C function and the USIM operating current will be reduced to a minimum value. When the bit is high the USIM SPI/I²C interface is enabled. If the USIM is configured to operate as an SPI interface via the UMD and 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 USIM is configured to operate as an I²C interface via the UMD and 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:** USIM SPI Incomplete Flag
 This bit is only available when the USIM is configured to operate in an SPI slave mode. Refer to the SPI register section.

• **SIMC1 Register**

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

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

Bit 6 **HAAS:** I²C Bus address match flag
 0: Not address match
 1: Address match
 The HAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.

Bit 5 **HBB:** I²C Bus busy flag
 0: I²C Bus is not busy
 1: I²C Bus is busy
 The HBB flag is the I²C busy flag. This flag will be “1” when the I²C bus is busy which will occur when a START signal is detected. The flag will be set to “0” when the bus is free which will occur when a STOP signal is detected.

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

Bit 3 **TXAK:** I²C Bus transmit acknowledge flag
 0: Slave send acknowledge flag
 1: Slave do not send acknowledge flag
 The TXAK bit is the transmit acknowledge flag. After the slave device receipt of 8 bits of data, this bit will be transmitted to the bus on the 9th clock from the slave device. The slave device must always set TXAK bit to “0” before further data is received.

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

Bit 1 **IAMWU:** I²C Address Match Wake-up control
 0: Disable
 1: Enable
 This bit should be set high 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 to zero by the application program after wake-up to ensure correction device operation.

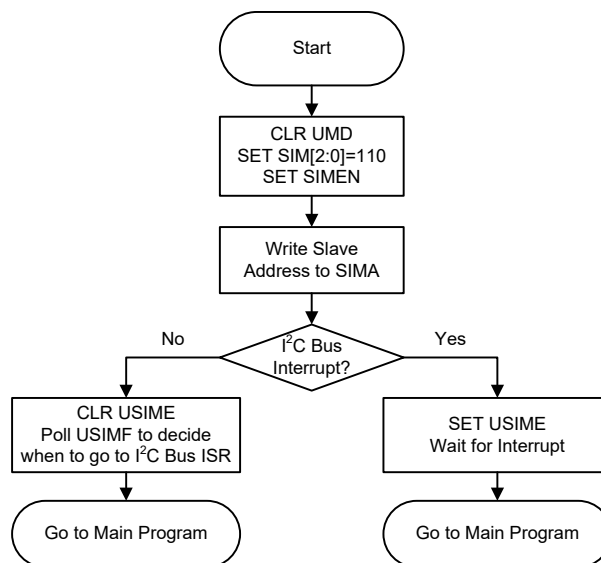
- Bit 0 **RXAK:** I²C Bus Receive acknowledge flag
 0: Slave receive acknowledge flag
 1: Slave does not receive acknowledge flag

The RXAK flag is the receiver acknowledge flag. When the RXAK flag is “0”, it means that a acknowledge signal has been received at the 9th clock, after 8 bits of data have been transmitted. When the slave device in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is “1”. When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the I²C Bus.

I²C Bus Communication

Communication on the I²C bus requires four separate steps, a START signal, a slave device address transmission, a data transmission and finally a STOP signal. When a START signal is placed on the I²C bus, all devices on the bus will receive this signal and be notified of the imminent arrival of data on the bus. The first seven bits of the data will be the slave address with the first bit being the MSB. If the address of the slave device matches that of the transmitted address, the HAAS bit in the SIMC1 register will be set and an USIM interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and SIMTOF bits to determine whether the interrupt source originates from an address match or from the completion of an 8-bit data transfer completion or from the I²C bus time-out occurrence. During a data transfer, note that after the 7-bit slave address has been transmitted, the following bit, which is the 8th bit, is the read/write bit whose value will be placed in the SRW bit. This bit will be checked by the slave device to determine whether to go into transmit or receive mode. Before any transfer of data to or from the I²C bus, the microcontroller must initialise the bus, the following are steps to achieve this:

- Step 1
Set the UMD, SIM2~SIM0 and SIMEN bits in the SIMC0 register to “0”, “110” and “1” respectively to enable the I²C bus.
- Step 2
Write the slave address of the device to the I²C bus address register SIMA.
- Step 3
Set the USIME interrupt enable bit of the interrupt control register to enable the USIM interrupt.



I²C Bus Initialisation Flow Chart

I²C Bus Start Signal

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

I²C Slave Address

The transmission of a START signal by the master will be detected by all devices on the I²C bus. To determine which slave device the master wishes to communicate with, the address of the slave device will be sent out immediately following the START signal. All slave devices, after receiving this 7-bit address data, will compare it with their own 7-bit slave address. If the address sent out by the master matches the internal address of the microcontroller slave device, then an internal USIM 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 USIM I²C bus interrupt signal can come from three sources, when the program enters the interrupt subroutine, the HAAS and SIMTOF bits should be examined to see whether the interrupt source has come from a matching slave address or from the completion of a data byte transfer or from the I²C bus time-out occurrence. When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.

I²C Bus Read/Write Signal

The SRW bit in the SIMC1 register defines whether the master device wishes to read data from the I²C bus or write data to the I²C bus. The slave device should examine this bit to determine if it is to be a transmitter or a receiver. If the SRW flag is “1” then this indicates that the master device wishes to read data from the I²C bus, therefore the slave device must be setup to send data to the I²C bus as a transmitter. If the SRW flag is “0” then this indicates that the master wishes to send data to the I²C bus, therefore the slave device must be setup to read data from the I²C bus as a receiver.

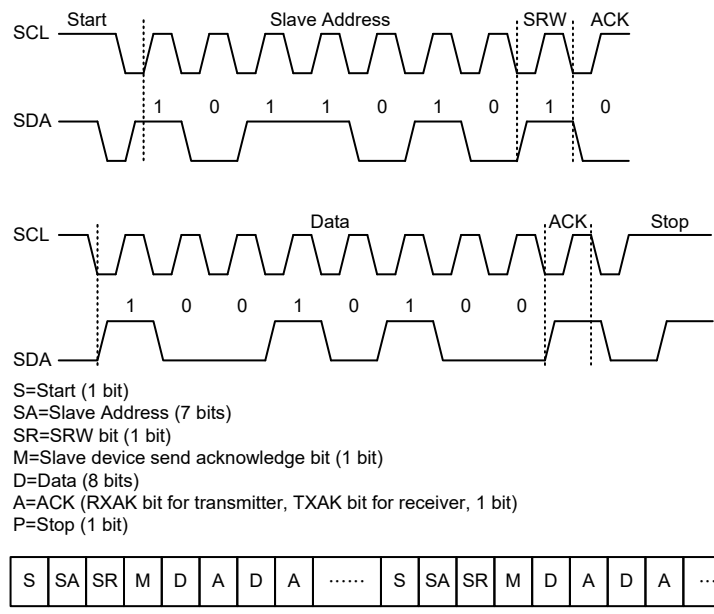
I²C Bus Slave Address Acknowledge Signal

After the master has transmitted a calling address, any slave device on the I²C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge signal is received by the master then a STOP signal must be transmitted by the master to end the communication. When the HAAS flag is high, the addresses have matched and the slave device must check the SRW flag to determine if it is to be a transmitter or a receiver. If the SRW flag is high, the slave device should be setup to be a transmitter so the HTX bit in the SIMC1 register should be set to “1”. If the SRW flag is low, then the microcontroller slave device should be setup as a receiver and the HTX bit in the SIMC1 register should be set to “0”.

I²C Bus Data and Acknowledge Signal

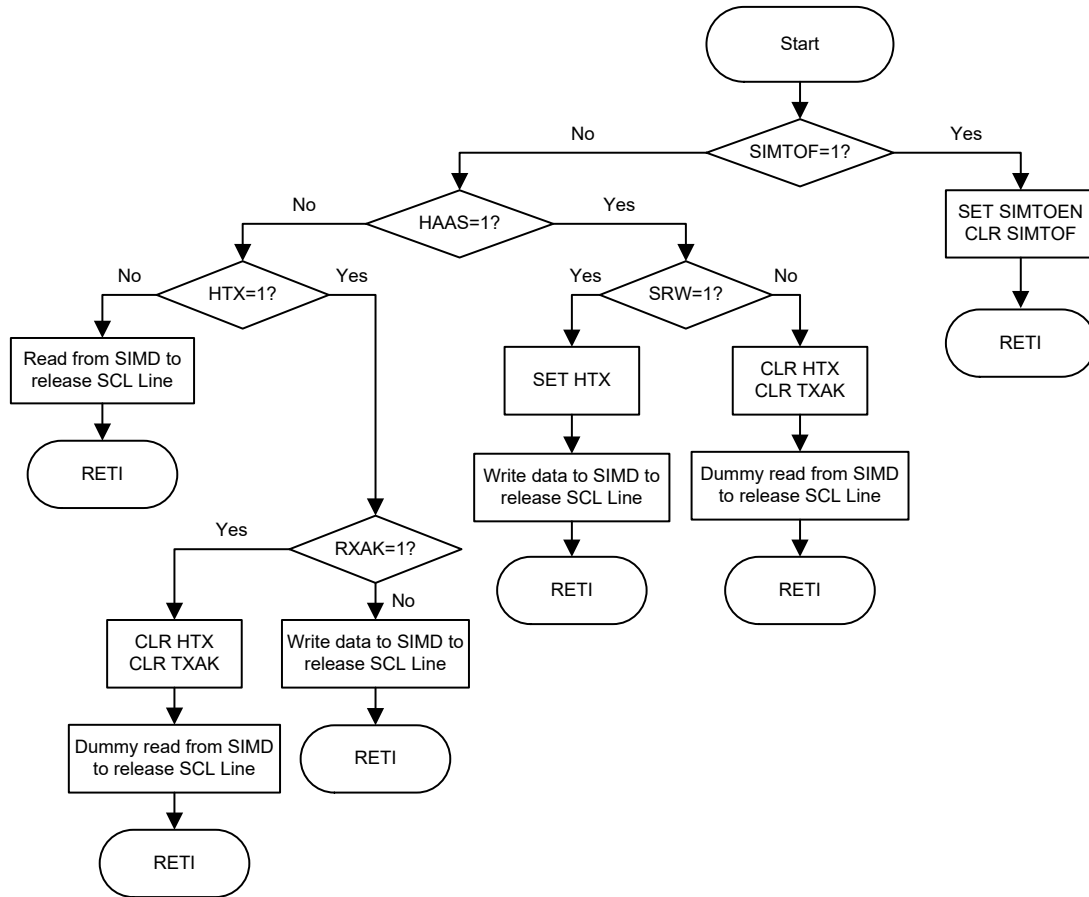
The transmitted data is 8-bit wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8 bits of data, the receiver must transmit an acknowledge signal, level “0”, before it can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

When the slave receiver receives the data byte, it must generate an acknowledge bit, known as TXAK, on the 9th clock. The slave device, which is setup as a transmitter will check the RXAK bit in the SIMC1 register to determine if it is to send another data byte, if not then it will release the SDA line and await the receipt of a STOP signal from the master.



I²C Communication Timing Diagram

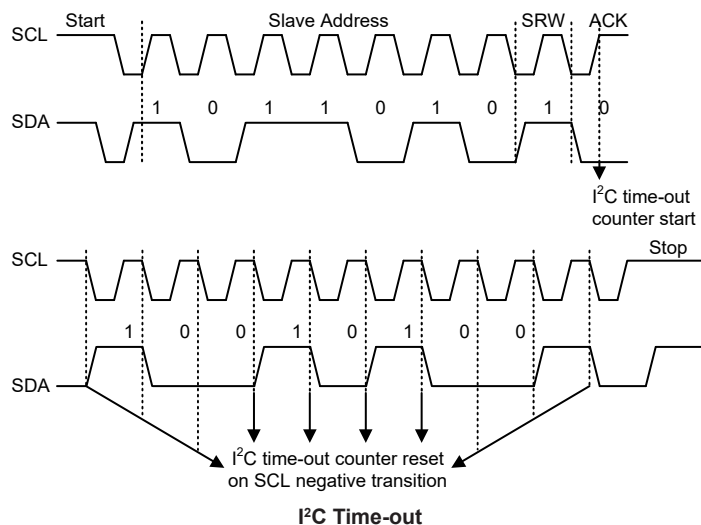
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.



I²C Bus ISR Flow Chart

I²C Time-out Control

In order to reduce the problem of I²C lockup due to reception of erroneous clock sources, a time-out function is provided. If the clock source to the I²C is not received for a while, then the I²C circuitry and registers will be reset after a certain time-out period. The time-out counter starts counting on an I²C bus “START” & “address match” condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out setup by the SIMTOC register, then a time-out condition will occur. The time-out function will stop when an I²C “STOP” condition occurs.



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 USIM 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 I ² C Time-out |
|-------------------|---------------------------------|
| SIMD, SIMA, SIMC0 | No change |
| SIMC1 | Reset to POR condition |

I²C Registers after Time-out

The SIMTOF flag can be cleared by the application program. There are 64 time-out periods which can be selected using SIMTOS bit field in the SIMTOC register. The time-out time is given by the formula: $((1\sim64)\times 32)/f_{SUB}$. 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**: USIM I²C Time-out control
 0: Disable
 1: Enable

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

This bit is set high when time-out occurs and can only be cleared to zero by application program.

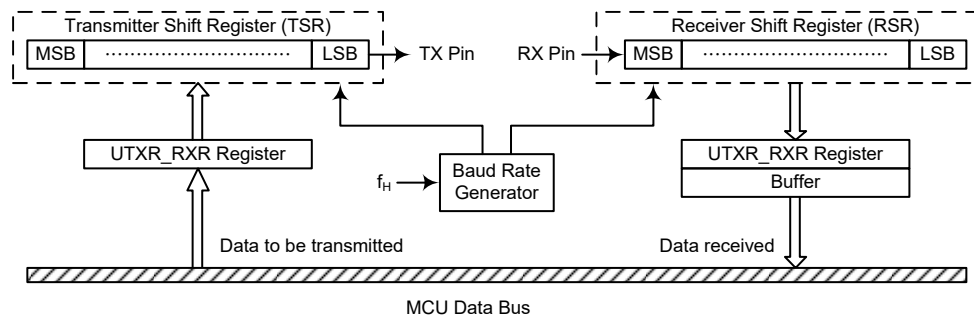
Bit 5~0 **SIMTOS5~SIMTOS0**: USIM I²C Time-out period selection
 I²C time-out clock source is $f_{SUB}/32$.
 I²C time-out time is equal to $(SIMTOS[5:0]+1) \times (32/f_{SUB})$.

UART Interface

The device contains an integrated full-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 shares the same internal interrupt vector with the SPI and I²C interfaces 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 asynchronous communication
- 8 or 9 bits character length
- Even, odd or no parity options
- One or two stop bits
- Baud rate generator with 8-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- 2-byte Deep FIFO Receive Data Buffer
- RX pin wake-up function
- Transmit and receive interrupts
- Interrupts can be triggered by the following conditions:
 - ♦ Transmitter Empty
 - ♦ Transmitter Idle
 - ♦ Receiver Full
 - ♦ Receiver Overrun
 - ♦ Address Mode Detect



UART Data Transfer Block Diagram

UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as TX and RX. The TX and RX pins, which are pin-shared with I/O or other pin functions. The TX and RX pin function should first be selected by the corresponding pin-shared function selection register before the UART function is used. Along with the UMD bit, the UREN and URXEN bits, if set, will setup 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 pin function is disabled by clearing the UMD, UREN, UTXEN or URXEN bit, the TX or RX pin will be set to a floating state. At this time whether the internal pull-high resistor is connected to the TX or RX pin or not is determined by the corresponding I/O pull-high function control bit.

UART Data Transfer Scheme

The above block diagram shows the overall data transfer structure arrangement for the UART. The actual data to be transmitted from the MCU is first transferred to the UTXR_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 UTXR_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 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 UTXR_RXR register, where it is buffered and can be manipulated by the application program. Only the UTXR_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 in the Data Memory. This shared register known as the UTXR_RXR register is used for both data transmission and data reception.

UART Status and Control Registers

There are six control registers associated with the UART function. The UMD bit in the SIMC0 register can be used to select the UART mode. The UUSR, UUCR1 and UUCR2 registers control the overall function of the UART, while the UBRG register controls the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the UTXR_RXR data register. Note that UART related registers and their POR values are only available when the UART mode is selected by setting the UMD bit in the SIMC0 register to “1”.

| Register Name | Bit | | | | | | | |
|---------------|-------|-------|-------|--------|---------|---------|--------|--------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SIMC0 | SIM2 | SIM1 | SIM0 | UMD | SIMDEB1 | SIMDEB0 | SIMEN | SIMICF |
| UUSR | UPERR | UNF | UFERR | UOERR | URIDLE | URXIF | UTIDLE | UTXIF |
| UUCR1 | UREN | UBNO | UPREN | UPRT | USTOPS | UTXBRK | URX8 | UTX8 |
| UUCR2 | UTXEN | URXEN | UBRGH | UADDEN | UWAKE | URIE | UTIIE | UTEIE |
| UTXR_RXR | UTXR7 | UTXR6 | UTXR5 | UTXR4 | UTXR3 | UTXR2 | UTXR1 | UTXR0 |
| UBRG | UBRG7 | UBRG6 | UBRG5 | UBRG4 | UBRG3 | UBRG2 | UBRG1 | UBRG0 |

UART Register List

• SIMC0 Register

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

Bit 7~5 **SIM2~SIM0**: USIM SPI/I²C Operating Mode Control

When the UMD bit is cleared to zero, these bits setup the SPI or I²C operating mode of the USIM function. Refer to the SPI or I²C register section for more details.

- Bit 4 **UMD**: UART mode selection bit
 0: SPI or I²C mode
 1: UART mode
 This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I²C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be set low for SPI or I²C mode.
- Bit 3~2 **SIMDEB1~SIMDEB0**: I²C Debounce Time Selection
 Refer to the I²C register section.
- Bit 1 **SIMEN**: USIM SPI/I²C Enable Control
 This bit is only available when the USIM is configured to operate in an SPI or I²C mode with the UMD bit low. Refer to the SPI or I²C register section for more details.
- Bit 0 **SIMICF**: USIM SPI Incomplete Flag
 Refer to the SPI register section.

• **UUSR Register**

The UUSR 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 UUSR register are read only. Further explanation on each of the flags is given below:

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-----|-------|-------|--------|-------|--------|-------|
| Name | UPERR | UNF | UFERR | UOERR | URIDLE | URXIF | UTIDLE | UTXIF |
| R/W | R | R | R | R | R | R | R | R |
| POR | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |

- Bit 7 **UPERR**: Parity error flag
 0: No parity error is detected
 1: Parity error is detected
 The UPERR 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 Parity mode (odd or even) is selected. The flag can also be cleared to zero by a software sequence which involves a read to the status register UUSR followed by an access to the UTXR_RXR data register.
- Bit 6 **UNF**: Noise flag
 0: No noise is detected
 1: Noise is detected
 The UNF 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 UNF flag is set during the same cycle as the URXIF flag but will not be set in the case of as overrun. The UNF flag can be cleared to zero by a software sequence which will involve a read to the status register UUSR followed by an access to the UTXR_RXR data register.
- Bit 5 **UFERR**: Framing error flag
 0: No framing error is detected
 1: Framing error is detected
 The UFERR 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 UUSR followed by an access to the UTXR_RXR data register.

- Bit 4 **UOERR:** Overrun error flag
 0: No overrun error is detected
 1: Overrun error is detected
The UOERR 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 UTXR_RXR receive data register. The flag is cleared to zero by a software sequence, which is a read to the status register UUSR followed by an access to the UTXR_RXR data register.
- Bit 3 **URIDLE:** Receiver status
 0: Data reception is in progress (Data being received)
 1: No data reception is in progress (Receiver is idle)
The URIDLE 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 URIDLE bit is "1" indicating that the UART receiver is idle and the RX pin stays in logic high condition.
- Bit 2 **URXIF:** Receive UTXR_RXR data register status
 0: UTXR_RXR data register is empty
 1: UTXR_RXR data register has available data
The URXIF flag is the receive data register status flag. When this read only flag is "0", it indicates that the UTXR_RXR read data register is empty. When the flag is "1", it indicates that the UTXR_RXR read data register contains new data. When the contents of the shift register are transferred to the UTXR_RXR register, an interrupt is generated if URIE=1 in the UUCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags UNF, UFERR, and/or UPERR are set within the same clock cycle. The URXIF flag will eventually be cleared to zero when the UUSR register is read with URXIF set, followed by a read from the UTXR_RXR register, and if the UTXR_RXR register has no more new data available.
- Bit 1 **UTIDLE:** Transmission idle
 0: Data transmission is in progress (Data being transmitted)
 1: No data transmission is in progress (Transmitter is idle)
The UTIDLE 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 UTXIF flag is "1" and when there is no transmit data or break character being transmitted. When UTIDLE is equal to "1", the TX pin becomes idle with the pin state in logic high condition. The UTIDLE flag is cleared to zero by reading the UUSR register with UTIDLE set and then writing to the UTXR_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.
- Bit 0 **UTXIF:** Transmit UTXR_RXR data register status
 0: Character is not transferred to the transmit shift register
 1: Character has transferred to the transmit shift register (UTXR_RXR data register is empty)
The UTXIF 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 UTXR_RXR data register. The UTXIF flag is cleared to zero by reading the UART status register (UUSR) with UTXIF set and then writing to the UTXR_RXR data register. Note that when the UTXEN bit is set, the UTXIF flag bit will also be set since the transmit data register is not yet full.

• **UUCR1 Register**

The UUCR1 register together with the UUCR2 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 | UREN | UBNO | UPREN | UPRT | USTOPS | UTXBRK | URX8 | UTX8 |
| 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 **UREN**: UART function enable control

- 0: Disable UART. TX and RX pins are in a floating state
- 1: Enable UART. TX and RX pins function as UART pins

The UREN bit is the UART enable bit. When this bit is equal to "0", the UART will be disabled and the RX pin as well as the TX pin will be set in a floating state. When the bit is equal to "1", the UART will be enabled if the UMD bit is set and the TX and RX pins will function as defined by the UTXEN and URXEN 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 UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF bits will be cleared to zero, while the UTIDLE, UTXIF and URIDLE bits will be set high. Other control bits in UUCR1, UUCR2 and UBRG registers will remain unaffected. If the UART is active and the UREN 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 **UBNO**: 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 URX8 and UTX8 will be used to store the 9th bit of the received and transmitted data respectively.

Bit 5 **UPREN**: 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 **UPRT**: Parity type selection bit

- 0: Even parity for parity generator
- 1: Odd parity for parity generator

This bit is the parity type selection bit. When this bit is equal to "1", odd parity type will be selected. If the bit is equal to "0", then even parity type will be selected.

Bit 3 **USTOPS**: Number of Stop bits selection for transmitter

- 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 transmitter. 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.

Bit 2 **UTXBRK**: Transmit break character

- 0: No break character is transmitted
- 1: Break characters transmit

The UTXBRK 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 UTXBRK bit is reset.

- Bit 1 **URX8**: 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 URX8. The UBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.
- Bit 0 **UTX8**: 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 UTX8. The UBNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• **UUCR2 Register**

The UUCR2 register is the second of the two 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 USIM UART mode interrupt sources. The register also serves to control the baud rate speed, 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 | UTXEN | URXEN | UBRGH | UADDEN | UWAKE | URIE | UTIIE | UTEIE |
| 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 **UTXEN**: UART Transmitter enabled control
0: UART transmitter is disabled
1: UART transmitter is enabled
The bit named UTXEN 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 set in a floating state.
If the UTXEN bit is equal to "1" and the UMD and UREN bit are also equal to "1", the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the UTXEN 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 set in a floating state.
- Bit 6 **URXEN**: UART Receiver enabled control
0: UART receiver is disabled
1: UART receiver is enabled
The bit named URXEN 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 pin will be set in a floating state. If the URXEN bit is equal to "1" and the UMD and UREN bit are also equal to "1", the receiver will be enabled and the RX pin will be controlled by the UART. Clearing the URXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX pin will be set in a floating state.
- Bit 5 **UBRGH**: Baud Rate speed selection
0: Low speed baud rate
1: High speed baud rate
The bit named UBRGH selects the high or low speed mode of the Baud Rate Generator. This bit, together with the value placed in the baud rate register UBRG, controls the Baud Rate of the UART. If this bit is equal to "1", the high speed mode is selected. If the bit is equal to "0", the low speed mode is selected.

- Bit 4 **UADDEN:** Address detect function enable control
 0: Address detect function is disabled
 1: Address detect function is enabled
 The bit named UADDEN 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 UTXRX7 if UBNO=0 or the 9th bit, which corresponds to URX8 if UBNO=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 UBNO. 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 **UWAKE:** RX pin wake-up UART function enable control
 0: RX pin wake-up UART function is disabled
 1: RX 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 pin occurs. Note that this bit is only available when the UART clock (f_{H}) is switched off. There will be no RX pin wake-up UART function if the UART clock (f_{H}) exists. If the UWAKE bit is set high as the UART clock (f_{H}) is switched off, a UART wake-up request will be initiated when a falling edge on the RX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX 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 pin when the UWAKE bit is cleared to zero.
- Bit 2 **URIE:** 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 UOERR or receive data available flag URXIF is set, the USIM interrupt request flag USIMF will be set. If this bit is equal to "0", the USIM interrupt request flag USIMF will not be influenced by the condition of the UOERR or URXIF flags.
- Bit 1 **UTIE:** 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 UTIDLE is set, due to a transmitter idle condition, the USIM interrupt request flag USIMF will be set. If this bit is equal to "0", the USIM interrupt request flag USIMF will not be influenced by the condition of the UTIDLE flag.
- Bit 0 **UTEIE:** 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 UTXIF is set, due to a transmitter empty condition, the USIM interrupt request flag USIMF will be set. If this bit is equal to "0", the USIM interrupt request flag USIMF will not be influenced by the condition of the UTXIF flag.

• **UTXR_RXR Register**

The UTXR_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 pin.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|--------|--------|--------|--------|--------|--------|--------|--------|
| Name | UTXRX7 | UTXRX6 | UTXRX5 | UTXRX4 | UTXRX3 | UTXRX2 | UTXRX1 | UTXRX0 |
| 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 **UTXRX7~UTXRX0**: UART Transmit/Receive Data bit 7 ~ bit 0

• **UBRG Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | UBRG7 | UBRG6 | UBRG5 | UBRG4 | UBRG3 | UBRG2 | UBRG1 | UBRG0 |
| 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 **UBRG7~UBRG0**: Baud Rate values

By programming the UBRGH bit in UUCR2 register which allows selection of the related formula described above and programming the required value in the UBRG register, the required baud rate can be setup.

Note: Baud rate= $f_{H}/[64 \times (N+1)]$ if UBRGH=0.

Baud rate= $f_{H}/[16 \times (N+1)]$ if UBRGH=1.

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 8-bit timer, the period of which is determined by two factors. The first of these is the value placed in the baud rate register UBRG and the second is the value of the UBRGH bit in the control register UUCR2. The UBRGH bit decides if the baud rate generator is to be used in a high speed mode or low speed mode, which in turn determines the formula that is used to calculate the baud rate. The value N in the UBRG register which is used in the following baud rate calculation formula determines the division factor. Note that N is the decimal value placed in the UBRG register and has a range of between 0 and 255.

| UUCR2 UBRGH Bit | 0 | 1 |
|-----------------|---------------------------|---------------------------|
| Baud Rate (BR) | $f_{H}/[64 \times (N+1)]$ | $f_{H}/[16 \times (N+1)]$ |

By programming the UBRGH bit which allows selection of the related formula and programming the required value in the UBRG register, the required baud rate can be setup. Note that because the actual baud rate is determined using a discrete value, N, placed in the UBRG register, there will be an error associated between the actual and requested value. The following example shows how the UBRG register value N and the error value can be calculated.

Calculating the Baud Rate and Error Values

For a clock frequency of 4MHz, and with UBRGH cleared to zero determine the UBRG register value N, the actual baud rate and the error value for a desired baud rate of 4800.

From the above table the desired baud rate $BR=f_{H}/[64 \times (N+1)]$

Re-arranging this equation gives $N=[f_{H}/(BR \times 64)]-1$

Giving a value for $N=[4000000/(4800 \times 64)]-1=12.0208$

To obtain the closest value, a decimal value of 12 should be placed into the UBRG register. This gives an actual or calculated baud rate value of $BR=4000000/[64 \times (12+1)]=4808$

Therefore the error is equal to $(4808-4800)/4800=0.16\%$

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 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 and stop bits, along with the parity, are setup by programming the corresponding UBNO, UPRT, UPREN, and USTOPS bits in the UUCR1 register. The baud rate used to transmit and receive data is setup using the internal 8-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 UREN bit in the UUCR1 register. When the UART mode is selected by setting the UMD bit in the SIMC0 register to “1”, if the UREN, UTXEN and URXEN bits are set, then these two UART pins will act as normal TX output pin and RX input pin respectively. If no data is being transmitted on the TX pin, then it will default to a logic high value.

Clearing the UREN bit will disable the TX and RX pins and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. 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 UTXEN, URXEN, UTXBRK, URXIF, UOERR, UFERR, UPERR and UNF being cleared while bits UTIDLE, UTXIF and URIDLE will be set. The remaining control bits in the UUCR1, UUCR2 and UBRG registers will remain unaffected. If the UREN bit in the UUCR1 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

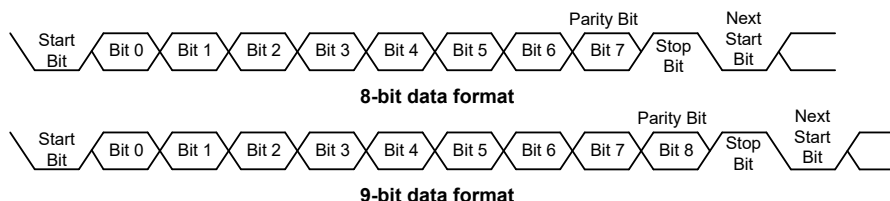
The format of the data to be transferred is composed of various factors such as data bit length, parity on/off, parity type, address bits and the number of stop bits. These factors are determined by the setup of various bits within the UUCR1 register. The UBNO bit controls the number of data bits which can be set to either 8 or 9, the UPRT bit controls the choice of odd or even parity, the UPREN bit controls the parity on/off function and the USTOPS bit decides whether one or two stop bits are to be used. 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 used for the transmitter. There is only one stop bit for the receiver.

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

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

Transmitter Receiver Data Format

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



UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the UBNO bit in the UUCR1 register. When UBNO 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 UTX8 bit in the UUCR1 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 UTXR_RXR register. The data to be transmitted is loaded into this UTXR_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 UTXR_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 UTXEN bit is set, but the data will not be transmitted until the UTXR_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 UTXR_RXR register, after which the UTXEN bit can be set. When a transmission of data begins, the TSR is normally empty, in which case a transfer to the UTXR_RXR register will result in an immediate transfer to the TSR. If during a transmission the UTXEN 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 UTXR_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 UTX8 bit in the UUCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the UBNO, UPRT, UPREN and USTOPS bits to define the required word length, parity type and number of stop bits.
- Setup the UBRG register to select the desired baud rate.
- Set the UTXEN bit to ensure that the TX pin is used as a UART transmitter pin.
- Access the UUSR register and write the data that is to be transmitted into the UTXR_RXR register. Note that this step will clear the UTXIF bit.

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

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

1. A UUSR register access
2. A UTXR_RXR register write execution

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

During a data transmission, a write instruction to the UTXR_RXR register will place the data into the UTXR_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 UTXR_RXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the UTXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the UTIDLE bit will be set. To clear the UTIDLE bit the following software sequence is used:

1. A UUSR register access
2. A UTXR_RXR register write execution

Note that both the UTXIF and UTIDLE bits are cleared by the same software sequence.

Transmitting Break

If the UTXBRK bit is set high and the state keeps for a time of greater than $[(UBRG+1) \times t_{th}]$ while UTIDLE=1, then break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by $13 \times N$ '0' bits and stop bits, where $N=1, 2, \text{etc.}$ If a break character is to be transmitted then the UTXBRK 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 UTXBRK 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 UTXBRK bit, the transmitter will finish transmitting the last break character and subsequently send out one or 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 UBNO bit is set, the word length will be set to 9 bits with the MSB being stored in the URX8 bit of the UUCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSR. The data which is received on the RX external input 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 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 input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RX 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 input pin, LSB first. In the read mode, the UTXR_RXR register forms a buffer between the internal bus and the receiver shift register. The UTXR_RXR register is a two-byte deep FIFO data buffer, where two bytes can be held in the FIFO while a third byte can continue to be received. Note that the

application program must ensure that the data is read from UTXR_RXR before the third byte has been completely shifted in, otherwise this third byte will be discarded and an overrun error UOERR will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of UBNO, UPRT and UPREN bits to define the word length, parity type
- Setup the UBRG register to select the desired baud rate
- Set the URXEN bit to ensure that the RX 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 URXIF bit in the UUSR register will be set when the UTXR_RXR register has data available. There will be at most one more character available before an overrun error occurs.
- When the contents of the shift register have been transferred to the UTXR_RXR register, then if the URIE 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 URXIF bit can be cleared using the following software sequence:

1. A UUSR register access
2. A UTXR_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 UBNO bit plus one stop bit. 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 UBNO plus one stop bit. The URXIF bit is set, UFERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the URIDLE bit is set. A break is regarded as a character that contains only zeros with the UFERR 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 UFERR 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 stop bits are received. It should be noted that the URIDLE 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, UFERR, will be set.
- The receive data register, UTXR_RXR, will be cleared.
- The UOERR, UNF, UPERR, URIDLE or URXIF 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 UUSR register, otherwise known as the URIDLE flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the URIDLE flag will have a high value, which indicates the receiver is in an idle condition.

Receiver Interrupt

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

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 – UOERR

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

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

- The UOERR flag in the UUSR register will be set.
- The UTXR_RXR contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the URIE bit is set.

The UOERR flag can be cleared by an access to the UUSR register followed by a read to the UTXR_RXR register.

Noise Error – UNF

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, UNF, in the UUSR register will be set on the rising edge of the URXIF bit.
- Data will be transferred from the Shift register to the UTXR_RXR register.
- No interrupt will be generated. However this bit rises at the same time as the URXIF bit which itself generates an interrupt.

Note that the UNF flag is reset by a UUSR register read operation followed by a UTXR_RXR register read operation.

Framing Error – UFERR

The read only framing error flag, UFERR, in the UUSR 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 UFERR flag will be set. The UFERR flag and the received data will be recorded in the UUSR and UTXR_RXR registers respectively, and the flag is cleared in any reset.

Parity Error – UPERR

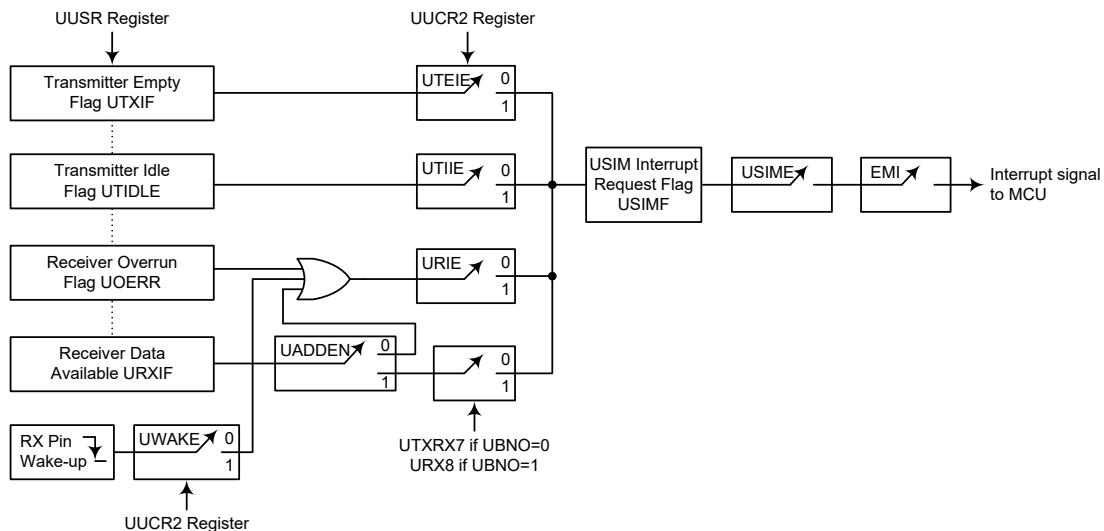
The read only parity error flag, UPERR, in the UUSR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, UPREN=1, and if the parity type, odd or even is selected. The read only UPERR flag and the received data will be recorded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

UART Interrupt Structure

Several individual UART conditions can trigger an USIM 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 data available, receiver overrun, address detect and an RX pin wake-up. When any of these conditions are created, if the global interrupt enable bit and the USIM 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 UUSR register flags which will generate an USIM interrupt if its associated interrupt enable control bit in the UUCR2 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 USIM UART mode interrupt sources.

The address detect condition, which is also an USIM UART mode interrupt source, does not have an associated flag, but will generate an USIM interrupt when an address detect condition occurs if its function is enabled by setting the UADDEN bit in the UUCR2 register. An RX pin wake-up, which is also an USIM UART mode interrupt source, does not have an associated flag, but will generate an USIM interrupt if the UART clock (f_{IH}) source is switched off and the UWAKE and URIE bits in the UUCR2 register are set when a falling edge on the RX pin occurs. Note that in the event of an RX wake-up interrupt occurring, there will be a certain period of delay, commonly known as the System Start-up Time, for the oscillator to restart and stabilize before the system resumes normal operation.

Note that the UUSR 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 USIM interrupt enable control bit in the interrupt control register 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, UADDEN, in the UUCR2 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 URXIF flag. If the UADDEN 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 USIME and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if UBNO=1 or the 8th bit if UBNO=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 UADDEN bit is not enabled, then a Receiver Data Available interrupt will be generated each time the URXIF 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 UPREN to zero.

| UADDEN | 9th Bit if UBNO=1 8th Bit if UBNO=0 | USIM Interrupt Generated |
|--------|--|-----------------------------|
| 0 | 0 | √ |
| | 1 | √ |
| 1 | 0 | × |
| | 1 | √ |

UADDEN 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 UUSR, UUCR1, UUCR2, UTXR_RXR as well as the UBRG register 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.

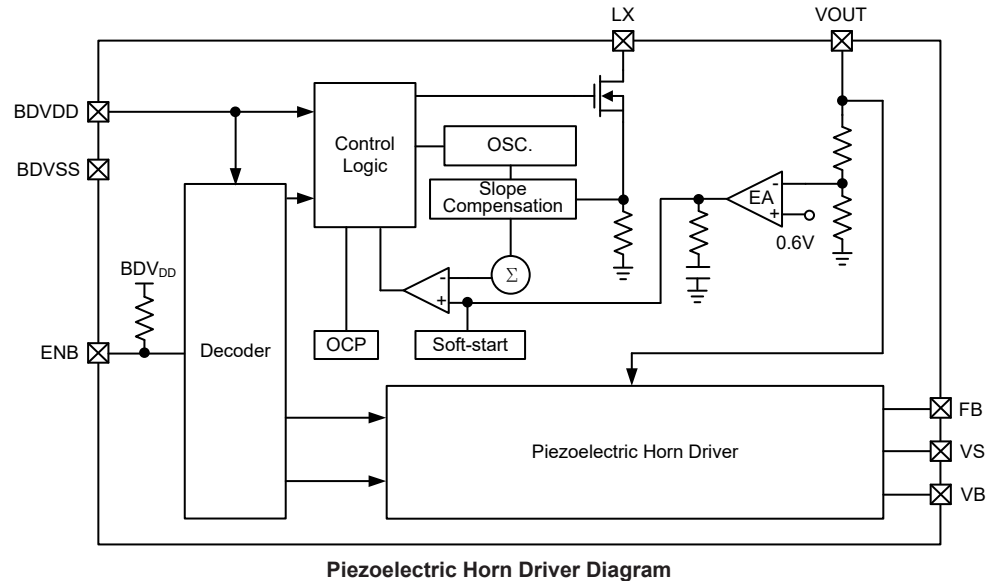
The UART function contains a receiver RX pin wake-up function, which is enabled or disabled by the UWAKE bit in the UUCR2 register. If this bit, along with the UART mode selection bit, UMD, the UART enable bit, UREN, the receiver enable bit, URXEN and the receiver interrupt bit, URIE, are all set when the UART clock (f_{H}) is off, then a falling edge on the RX pin will trigger an RX 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 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 USIM interrupt enable bit, USIME, must be set. If the EMI and USIME 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 USIM interrupt will not be generated until after this time has elapsed.

Piezoelectric Horn Driver

The device contains a piezoelectric horn driver with a 12V boost converter. The driver is used to drive an external-driving(2-pin) buzzer or a self-driving(3-pin) buzzer.

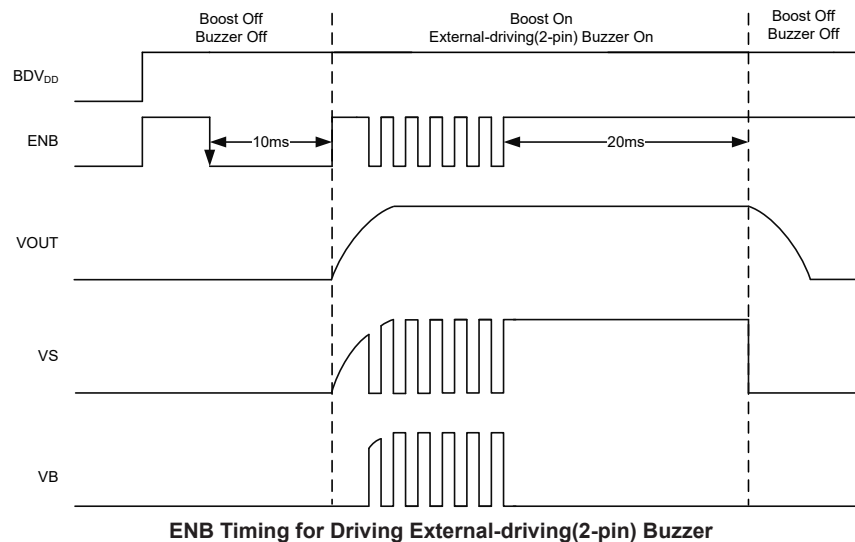
The ENB pin is used to select buzzer mode and boost converter on/off. When the ENB pin is set to high, the driver is in the shutdown mode, the buzzer driver and boost converter are both off.



External-driving(2-pin) Buzzer Driver Mode Settings

The driver is in the shutdown mode by default. Clear the ENB pin to 0 for 10ms, the driver will enter external-driving buzzer driver mode. The boost converter will be on. The control signal of external-driving buzzer is generated by the ENB pin. The VS pin outputs the non-inverted signal of the ENB pin, and the VB pin outputs the inverted signal of the ENB pin. The FB pin should be connect to VSS.

If the ENB pin is always high for 20ms, the driver will enter the shutdown mode.



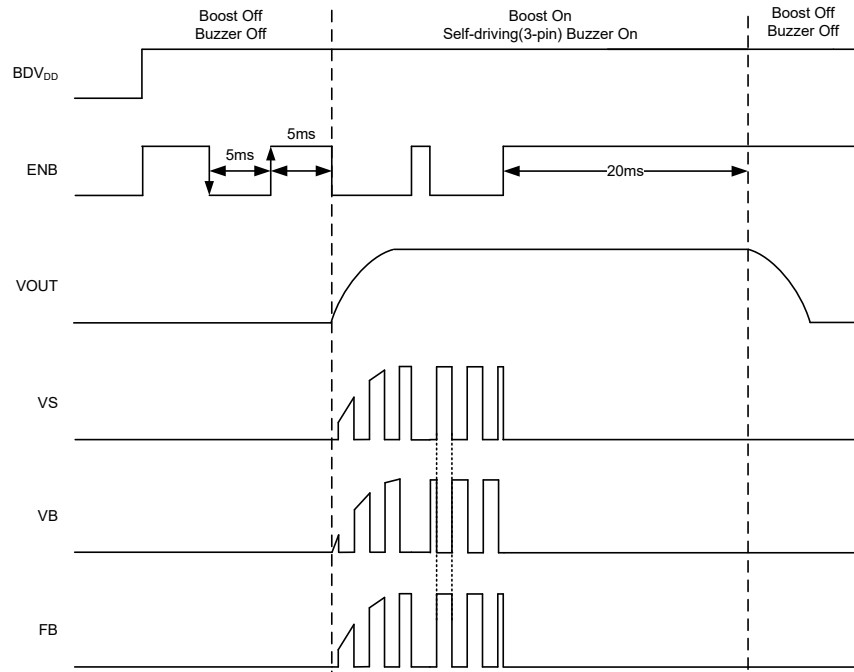
Self-driving(3-pin) Buzzer Driver Mode Settings

The driver is in the shutdown mode by default. Clear the ENB pin to 0 for 5ms, after which set the ENB pin high for 5ms, the driver will enter self-driving buzzer driver mode. The boost converter will be on.

Clear the ENB pin to 0, the self-driving buzzer driver is enabled. Set the ENB pin high, the self-driving buzzer driver is disabled.

If the ENB pin is always high for 20ms, the driver will enter the shutdown mode.

Note: The buzzer mode and shutdown mode settings are achieved by time counting. Attention should be paid to the system noise to avoid IC misjudgment which may reset the internal counter and restart the counting.



ENB Timing for Driving Self-driving(3-pin) Buzzer

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

• LVDC Register

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

Bit 7~6 Unimplemented, read as “0”

Bit 5 **LVDO**: LVD Output flag
 0: No Low Voltage Detected
 1: Low Voltage Detected

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

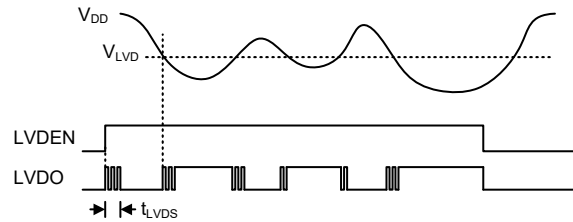
Bit 3 **VBGEN**: Bandgap Voltage Output Enable control
 0: Disable
 1: Enable

Note that the Bandgap circuit is enabled when the LVD or LVR function is enabled or when the VBGEN bit is set to 1.

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

LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage, V_{DD} , with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.2V and 4.0V. When the power supply voltage, V_{DD} , falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. 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 which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of t_{LVD} after the LVDO bit has been set high by a low voltage condition. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if V_{DD} falls below the preset LVD voltage. This will cause the device to wake-up from the 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 interrupt functions. The external interrupts are generated by the action of the external INT pin, while the internal interrupts are generated by various internal functions such as the TM, Time Base, USIM, LVD, EEPROM and the A/D converter, etc.

Interrupt Registers

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

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

| Function | Enable Bit | Request Flag | Notes |
|---------------|------------|--------------|-------|
| Global | EMI | — | — |
| INT Pin | INTE | INTF | — |
| A/D Converter | ADE | ADF | — |
| Time Base | TBnE | TBnF | n=0~1 |
| USIM | USIME | USIMF | — |
| EEPROM | DEE | DEF | — |
| LVD | LVE | LVF | — |
| PTM | PTMPE | PTMPF | — |
| | PTMAE | PTMAF | |

Interrupt Register Bit Naming Conventions

| Register Name | Bit | | | | | | | |
|---------------|-------|-------|-------|------|-------|-------|-------|-------|
| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| INTEG | — | — | — | — | — | — | INTS1 | INTS0 |
| INTC0 | — | LVF | USIMF | INTF | LVE | USIME | INTE | EMI |
| INTC1 | PTMAF | PTMPF | DEF | ADF | PTMAE | PTMPE | DEE | ADE |
| INTC2 | — | — | TB1F | TB0F | — | — | TB1E | TB0E |

Interrupt Register List

• **INTEG Register**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|------|---|---|---|---|---|---|-------|-------|
| Name | — | — | — | — | — | — | INTS1 | INTS0 |
| R/W | — | — | — | — | — | — | R/W | R/W |
| POR | — | — | — | — | — | — | 0 | 0 |

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **INTS1~INTS0**: Interrupt edge control for INT 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 | — | LVF | USIMF | INTF | LVE | USIME | INTE | 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 **LVF**: LVD interrupt request flag

- 0: No request
- 1: Interrupt request

Bit 5 **USIMF**: USIM interrupt request flag

- 0: No request
- 1: Interrupt request

Bit 4 **INTF**: INT interrupt request flag

- 0: No request
- 1: Interrupt request

- Bit 3 **LVE**: LVD interrupt control
 0: Disable
 1: Enable
- Bit 2 **USIME**: USIM interrupt control
 0: Disable
 1: Enable
- Bit 1 **INTE**: INT 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 | PTMAF | PTMPF | DEF | ADF | PTMAE | PTMPE | DEE | ADE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

- Bit 7 **PTMAF**: PTM Comparator A match interrupt request flag
 0: No request
 1: Interrupt request
- Bit 6 **PTMPF**: PTM Comparator P match interrupt request flag
 0: No request
 1: Interrupt request
- Bit 5 **DEF**: Data EEPROM interrupt request flag
 0: No request
 1: Interrupt request
- Bit 4 **ADF**: A/D Converter interrupt request flag
 0: No request
 1: Interrupt request
- Bit 3 **PTMAE**: PTM Comparator A match interrupt control
 0: Disable
 1: Enable
- Bit 2 **PTMPE**: PTM Comparator P match interrupt control
 0: Disable
 1: Enable
- Bit 1 **DEE**: Data EEPROM interrupt control
 0: Disable
 1: Enable
- Bit 0 **ADE**: A/D Converter interrupt control
 0: Disable
 1: Enable

• **INTC2 Register**

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

- Bit 7~6 Unimplemented, read as “0”
- Bit 5 **TB1F**: Time Base 1 request flag
 0: No request
 1: Interrupt request

| | |
|---------|---|
| Bit 4 | TB0F : Time Base 0 request flag 0: No request 1: Interrupt request |
| Bit 3~2 | Unimplemented, read as “0” |
| Bit 1 | TB1E : Time Base 1 interrupt control 0: Disable 1: Enable |
| Bit 0 | TB0E : Time Base 0 interrupt control 0: Disable 1: Enable |

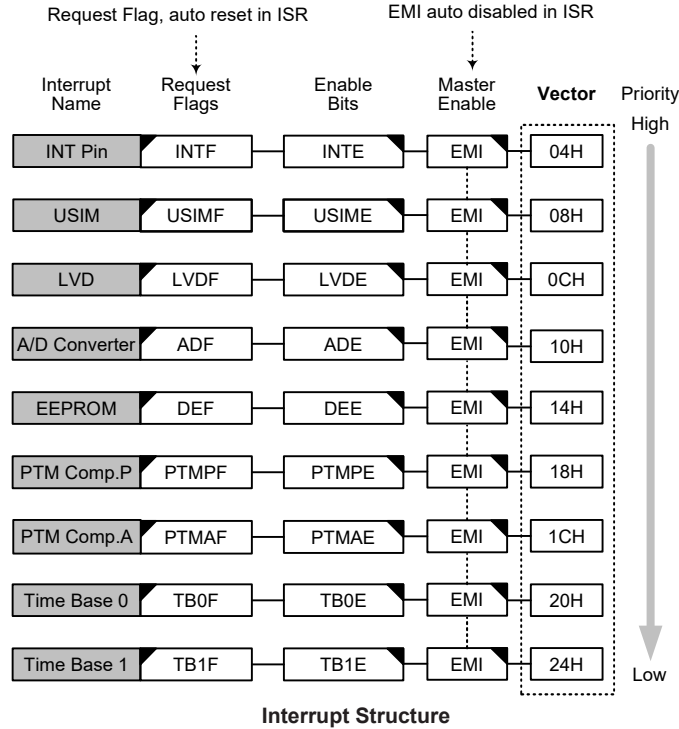
Interrupt Operation

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

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

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

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



External Interrupt

The external interrupt is controlled by signal transitions on the INT pin. An external interrupt request will take place when the external interrupt request flag, INTF, is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pin. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the external interrupt enable bit, INTE, 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 pin is pin-shared with I/O pin, it can only be configured as external interrupt pin if its external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register. When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that the pull-high resistor selection on the external interrupt pin 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 A/D Converter Interrupt is serviced, the A/D Interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

EEPROM Interrupt

An EEPROM Interrupt request will take place when the EEPROM Interrupt request flag, DEF, is set, which occurs when an EEPROM Write cycle ends. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and EEPROM Interrupt enable bit, DEE, must first be set. When the interrupt is enabled, the stack is not full and an EEPROM Write cycle ends, a subroutine call to the EEPROM Interrupt vector will take place. When the EEPROM Interrupt is serviced, the DEF flag will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

TM Interrupts

The Periodic Type TM has two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. There are two interrupt request flags, PTMPF and PTMAF, and two enable control bits, PTMPE and PTMAE. A PTM interrupt request will take place when any of the PTM request flags are set, a situation which occurs when a PTM comparator P or A match situation happens.

To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and the respective PTM Interrupt enable bit must first be set. When the interrupt is enabled, the stack is not full and a PTM comparator match situation occurs, a subroutine call to the relevant PTM Interrupt vector location, will take place. When the PTM interrupt is serviced, the PTM interrupt flag will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Universal Serial Interface Module Interrupt

The Universal Serial Interface Module Interrupt, also known as the USIM interrupt, will take place when the USIM Interrupt request flag, USIMF, is set. As the USIM interface can operate in three modes which are SPI mode, I²C mode and UART mode, the USIMF flag can be set by different conditions depending on the selected interface mode.

If the SPI or I²C mode is selected, the USIM interrupt can be triggered when a byte of data has been received or transmitted by the SPI/I²C interface, or an I²C slave address match occurs, or an I²C bus time-out occurs. If the UART mode is selected, several individual UART conditions including a transmitter data register empty, transmitter idle, receiver data available, receiver overrun, address detect and an RX pin wake-up, can generate an USIM interrupt with the USIMF flag bit set high.

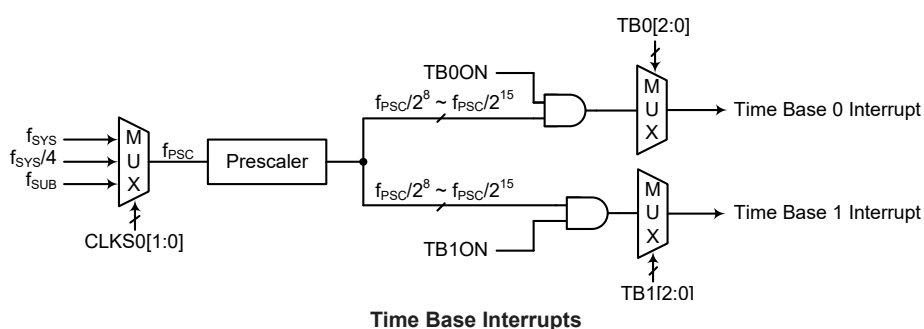
To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and the Serial Interface Interrupt enable bit, USIME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the USIM Interrupt vector, will take place. When the interrupt is serviced, the Universal Serial Interface Interrupt flag, USIMF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Note that if the USIM interrupt is triggered by the UART interface, after the interrupt has been serviced, the UUSR register flags will only be cleared when certain actions are taken by the UART, the details of which are given in the UART section.

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 reset and the EMI bit will be cleared to disable other interrupts.

The clock source, f_{PSC} , originates from the internal clock source f_{SYS} , $f_{SYS}/4$ or f_{SUB} and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL1~CLKSEL0 bits in the PSCR register.



• PSCR Register

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

Bit 7~2 Unimplemented, read as “0”

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

• TB0C Register

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

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

Bit 6~3 Unimplemented, read as “0”

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

• **TB1C Register**

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

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

Bit 6~3 Unimplemented, read as “0”

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

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, and the 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 subroutine call to the LVD Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the LVF flag will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Interrupt Wake-up Function

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

Programming Considerations

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

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.

Configuration Options

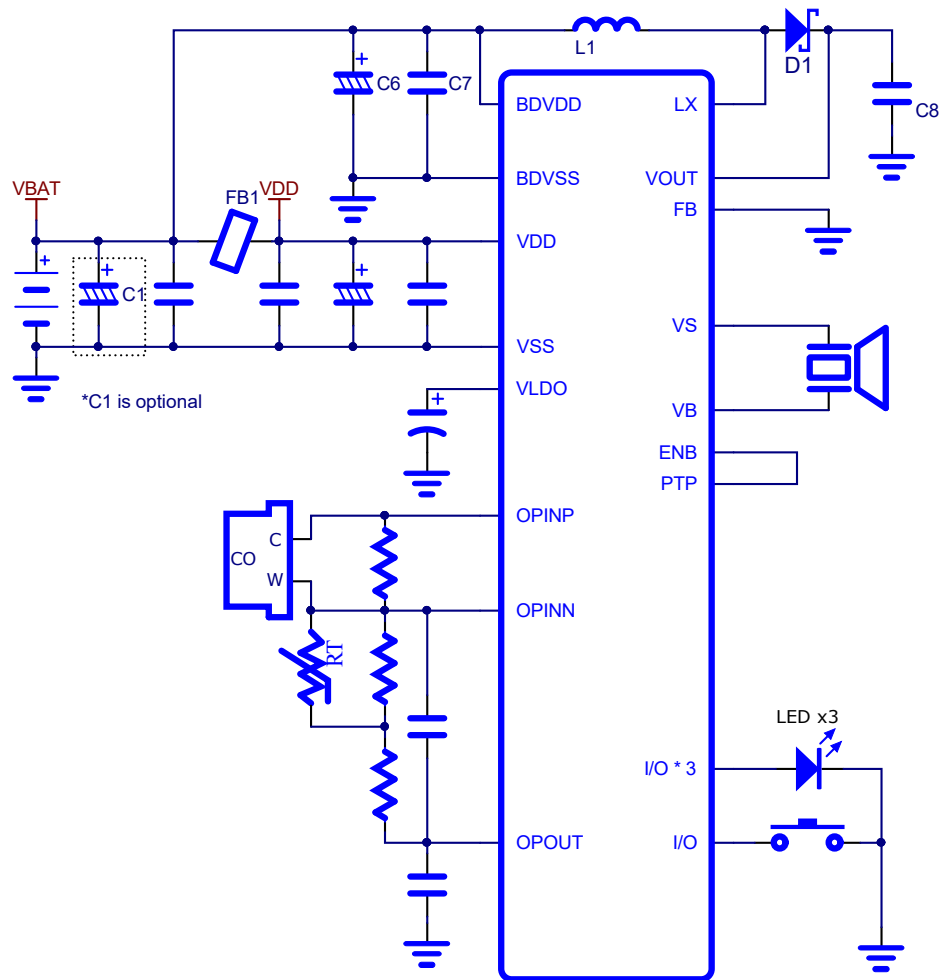
Configuration options refer to certain options within the MCU that are programmed into the device during the programming process. During the development process, these options are selected using the HT-IDE software development tools. All options must be defined for proper system function, the details of which are shown in the table.

| No. | Options |
|--------------------------|---|
| Oscillator Option | |
| 1 | HIRC frequency selection - f_{HIRC} : 2MHz, 4MHz or 8MHz |

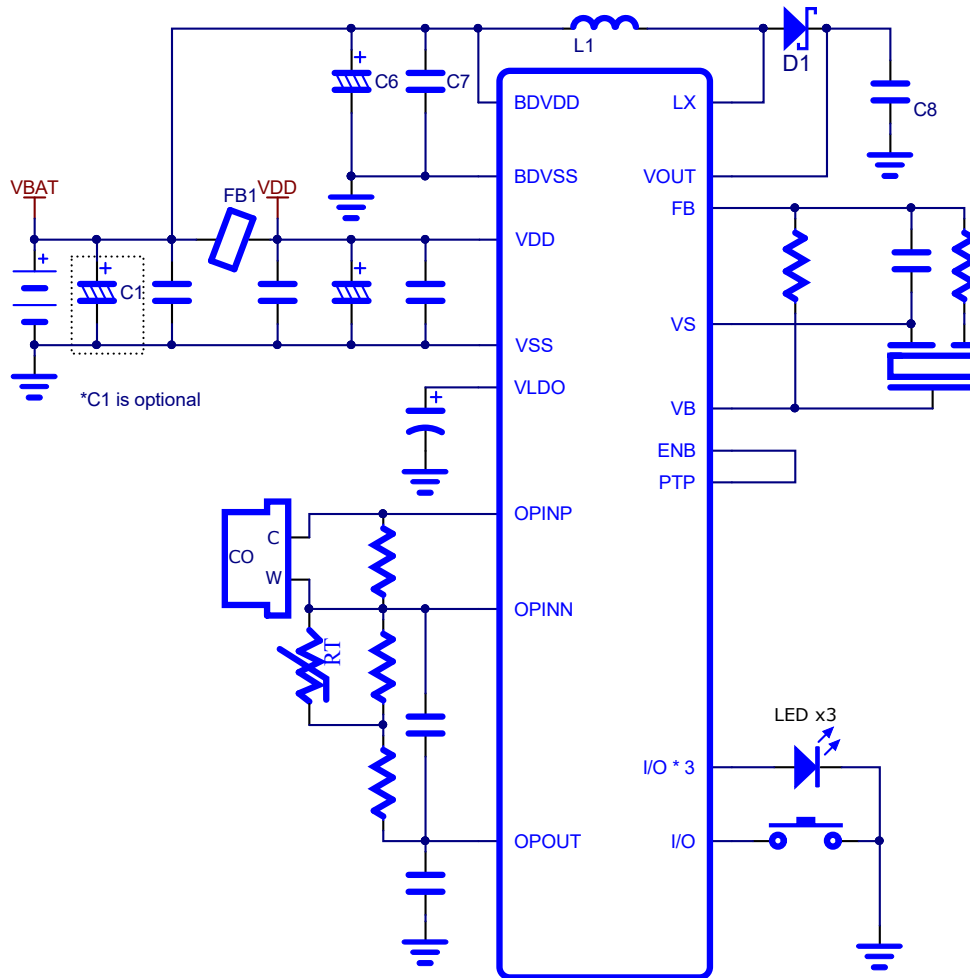
Note: When the HIRC has been configured at a frequency shown in this table, the HIRC1 and HIRC0 bits should also be setup to select the same frequency to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.

Application Circuits

External-driving (2-pin) Buzzer



Self-driving (3-pin) Buzzer



Note: L1=10μH; C8=2.2μF; C6=47μF; C7=0.1μF.

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 0.5 μ s and branch or call instructions would be implemented within 1 μ s. 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 | | | |
| LRRRA [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 |
| LDZ [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 |
| LDZA [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 | $ACC \leftarrow ACC \text{ "AND" } [m]$ |
| 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 | $ACC \leftarrow ACC \text{ "AND" } x$ |
| 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 | $[m] \leftarrow ACC \text{ "AND" } [m]$ |
| Affected flag(s) | Z |

| | |
|------------------|--|
| CALL addr | Subroutine call |
| Description | Unconditionally calls a subroutine at the specified address. The Program Counter then increments by 1 to obtain the address of the next instruction which is then pushed onto the stack. The specified address is then loaded and the program continues execution from this new address. As this instruction requires an additional operation, it is a two cycle instruction. |
| Operation | Stack ← Program Counter + 1 Program Counter ← addr |
| Affected flag(s) | None |
| | |
| CLR [m] | Clear Data Memory |
| Description | Each bit of the specified Data Memory is cleared to 0. |
| Operation | [m] ← 00H |
| Affected flag(s) | None |
| | |
| CLR [m].i | Clear bit of Data Memory |
| Description | Bit i of the specified Data Memory is cleared to 0. |
| Operation | [m].i ← 0 |
| Affected flag(s) | None |
| | |
| CLR WDT | Clear Watchdog Timer |
| Description | The TO, PDF flags and the WDT are all cleared. |
| Operation | WDT cleared TO ← 0 PDF ← 0 |
| Affected flag(s) | TO, PDF |
| | |
| CPL [m] | Complement Data Memory |
| Description | Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa. |
| Operation | [m] ← $\overline{[m]}$ |
| Affected flag(s) | Z |
| | |
| CPLA [m] | Complement Data Memory with result in ACC |
| Description | Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged. |
| Operation | ACC ← $\overline{[m]}$ |
| Affected flag(s) | Z |
| | |
| DAA [m] | Decimal-Adjust ACC for addition with result in Data Memory |
| Description | Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition. |
| Operation | [m] ← ACC + 00H or [m] ← ACC + 06H or [m] ← ACC + 60H or [m] ← ACC + 66H |
| Affected flag(s) | C |

| | |
|------------------|--|
| DEC [m] | Decrement Data Memory |
| Description | Data in the specified Data Memory is decremented by 1. |
| Operation | $[m] \leftarrow [m] - 1$ |
| Affected flag(s) | Z |
| DECA [m] | Decrement Data Memory with result in ACC |
| Description | Data in the specified Data Memory is decremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged. |
| Operation | $ACC \leftarrow [m] - 1$ |
| Affected flag(s) | Z |
| HALT | Enter power down mode |
| Description | This instruction stops the program execution and turns off the system clock. The contents of the Data Memory and registers are retained. The WDT and prescaler are cleared. The power down flag PDF is set and the WDT time-out flag TO is cleared. |
| Operation | $TO \leftarrow 0$ $PDF \leftarrow 1$ |
| Affected flag(s) | TO, PDF |
| INC [m] | Increment Data Memory |
| Description | Data in the specified Data Memory is incremented by 1. |
| Operation | $[m] \leftarrow [m] + 1$ |
| Affected flag(s) | Z |
| INCA [m] | Increment Data Memory with result in ACC |
| Description | Data in the specified Data Memory is incremented by 1. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged. |
| Operation | $ACC \leftarrow [m] + 1$ |
| Affected flag(s) | Z |
| JMP addr | Jump unconditionally |
| Description | The contents of the Program Counter are replaced with the specified address. Program execution then continues from this new address. As this requires the insertion of a dummy instruction while the new address is loaded, it is a two cycle instruction. |
| Operation | Program Counter \leftarrow 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 | $ACC \leftarrow [m]$ |
| Affected flag(s) | None |
| MOV A,x | Move immediate data to ACC |
| Description | The immediate data specified is loaded into the Accumulator. |
| Operation | $ACC \leftarrow x$ |
| 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 | $[m] \leftarrow ACC$ |
| Affected flag(s) | None |

| | |
|------------------|--|
| NOP | No operation |
| Description | No operation is performed. Execution continues with the next instruction. |
| Operation | No operation |
| Affected flag(s) | None |
| | |
| OR A,[m] | Logical OR Data Memory to ACC |
| Description | Data in the Accumulator and the specified Data Memory perform a bitwise logical OR operation. The result is stored in the Accumulator. |
| Operation | ACC ← ACC "OR" [m] |
| Affected flag(s) | Z |
| | |
| OR A,x | Logical OR immediate data to ACC |
| Description | Data in the Accumulator and the specified immediate data perform a bitwise logical OR operation. The result is stored in the Accumulator. |
| Operation | ACC ← ACC "OR" x |
| Affected flag(s) | Z |
| | |
| ORM A,[m] | Logical OR ACC to Data Memory |
| Description | Data in the specified Data Memory and the Accumulator perform a bitwise logical OR operation. The result is stored in the Data Memory. |
| Operation | [m] ← ACC "OR" [m] |
| Affected flag(s) | Z |
| | |
| RET | Return from subroutine |
| Description | The Program Counter is restored from the stack. Program execution continues at the restored address. |
| Operation | Program Counter ← Stack |
| Affected flag(s) | None |
| | |
| RET A,x | Return from subroutine and load immediate data to ACC |
| Description | The Program Counter is restored from the stack and the Accumulator loaded with the specified immediate data. Program execution continues at the restored address. |
| Operation | Program Counter ← Stack ACC ← x |
| Affected flag(s) | None |
| | |
| RETI | Return from interrupt |
| Description | The Program Counter is restored from the stack and the interrupts are re-enabled by setting the EMI bit. EMI is the master interrupt global enable bit. If an interrupt was pending when the RETI instruction is executed, the pending Interrupt routine will be processed before returning to the main program. |
| Operation | Program Counter ← Stack EMI ← 1 |
| Affected flag(s) | None |
| | |
| RL [m] | Rotate Data Memory left |
| Description | The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. |
| Operation | [m].(i+1) ← [m].i; (i=0~6) [m].0 ← [m].7 |
| Affected flag(s) | None |

| | |
|------------------|---|
| RLA [m] | Rotate Data Memory left with result in ACC |
| Description | The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged. |
| Operation | ACC.(i+1) ← [m].i; (i=0~6) ACC.0 ← [m].7 |
| Affected flag(s) | None |
| RLC [m] | Rotate Data Memory left through Carry |
| Description | The contents of the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into bit 0. |
| Operation | [m].(i+1) ← [m].i; (i=0~6) [m].0 ← C C ← [m].7 |
| Affected flag(s) | C |
| RLCA [m] | Rotate Data Memory left through Carry with result in ACC |
| Description | Data in the specified Data Memory and the carry flag are rotated left by 1 bit. Bit 7 replaces the Carry bit and the original carry flag is rotated into the bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged. |
| Operation | ACC.(i+1) ← [m].i; (i=0~6) ACC.0 ← C C ← [m].7 |
| Affected flag(s) | C |
| RR [m] | Rotate Data Memory right |
| Description | The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7. |
| Operation | [m].i ← [m].(i+1); (i=0~6) [m].7 ← [m].0 |
| Affected flag(s) | None |
| RRA [m] | Rotate Data Memory right with result in ACC |
| Description | Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged. |
| Operation | ACC.i ← [m].(i+1); (i=0~6) ACC.7 ← [m].0 |
| Affected flag(s) | None |
| RRC [m] | Rotate Data Memory right through Carry |
| Description | The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. |
| Operation | [m].i ← [m].(i+1); (i=0~6) [m].7 ← C C ← [m].0 |
| Affected flag(s) | C |

| | |
|-------------------|---|
| RRCA [m] | Rotate Data Memory right through Carry with result in ACC |
| Description | Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged. |
| Operation | ACC.i ← [m].(i+1); (i=0~6) ACC.7 ← C C ← [m].0 |
| Affected flag(s) | C |
| | |
| SBC A,[m] | Subtract Data Memory from ACC with Carry |
| Description | The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1. |
| Operation | ACC ← ACC – [m] – \bar{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] – \bar{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] – \bar{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 Skip if [m]=0 |
| Affected flag(s) | None |
| | |
| SDZA [m] | Skip if decrement Data Memory is zero with result in ACC |
| Description | The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction. |
| Operation | ACC ← [m] – 1 Skip if ACC=0 |
| Affected flag(s) | None |

| | |
|------------------|--|
| SET [m] | Set Data Memory |
| Description | Each bit of the specified Data Memory is set to 1. |
| Operation | $[m] \leftarrow FFH$ |
| Affected flag(s) | None |
| | |
| SET [m].i | Set bit of Data Memory |
| Description | Bit i of the specified Data Memory is set to 1. |
| Operation | $[m].i \leftarrow 1$ |
| Affected flag(s) | None |
| | |
| SIZ [m] | Skip if increment Data Memory is 0 |
| Description | The contents of the specified Data Memory are first incremented by 1. If the result is 0, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction. |
| Operation | $[m] \leftarrow [m] + 1$ 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 | $ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$ |
| 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 | $ACC \leftarrow [m]$ 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 | $ACC \leftarrow ACC + [m] + C$ |
| 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 | $[m] \leftarrow ACC + [m] + C$ |
| 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 | $ACC \leftarrow ACC + [m]$ |
| 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 | $[m] \leftarrow ACC + [m]$ |
| 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 | $ACC \leftarrow ACC \text{ "AND" } [m]$ |
| 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 | $[m] \leftarrow ACC \text{ "AND" } [m]$ |
| Affected flag(s) | Z |
| LCLR [m] | Clear Data Memory |
| Description | Each bit of the specified Data Memory is cleared to 0. |
| Operation | $[m] \leftarrow 00H$ |
| 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 | $[m].i \leftarrow 0$ |
| 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 | $[m] \leftarrow \overline{[m]}$ |
| 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 | $ACC \leftarrow \overline{[m]}$ |
| 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 | $[m] \leftarrow ACC + 00H$ or $[m] \leftarrow ACC + 06H$ or $[m] \leftarrow ACC + 60H$ or $[m] \leftarrow ACC + 66H$ |
| Affected flag(s) | C |
| | |
| LDEC [m] | Decrement Data Memory |
| Description | Data in the specified Data Memory is decremented by 1. |
| Operation | $[m] \leftarrow [m] - 1$ |
| 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 | $ACC \leftarrow [m] - 1$ |
| Affected flag(s) | Z |
| | |
| LINC [m] | Increment Data Memory |
| Description | Data in the specified Data Memory is incremented by 1. |
| Operation | $[m] \leftarrow [m] + 1$ |
| 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 | $ACC \leftarrow [m] + 1$ |
| 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 | $ACC \leftarrow [m]$ |
| 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 | $[m] \leftarrow ACC$ |
| 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 | $ACC \leftarrow ACC \text{ "OR" } [m]$ |
| 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 | $[m] \leftarrow ACC \text{ "OR" } [m]$ |
| 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 | $[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow [m].7$ |
| 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 | $ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow [m].7$ |
| 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 | $[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$ |
| 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 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 |

| | |
|--------------------|---|
| 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 | $[m].i \leftarrow [m].(i+1); (i=0\sim 6)$ $[m].7 \leftarrow [m].0$ |
| Affected flag(s) | None |
| LRRRA [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 | $ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$ $ACC.7 \leftarrow [m].0$ |
| Affected flag(s) | None |
| LRRRC [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 | $[m].i \leftarrow [m].(i+1); (i=0\sim 6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$ |
| 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 | $ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$ |
| 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 | $ACC \leftarrow ACC - [m] - \bar{C}$ |
| 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 | $[m] \leftarrow ACC - [m] - \bar{C}$ |
| 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 | $ACC \leftarrow [m] + 1$ 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 $[m].i \neq 0$ |
| 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 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] 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) 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) 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) 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) 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 |

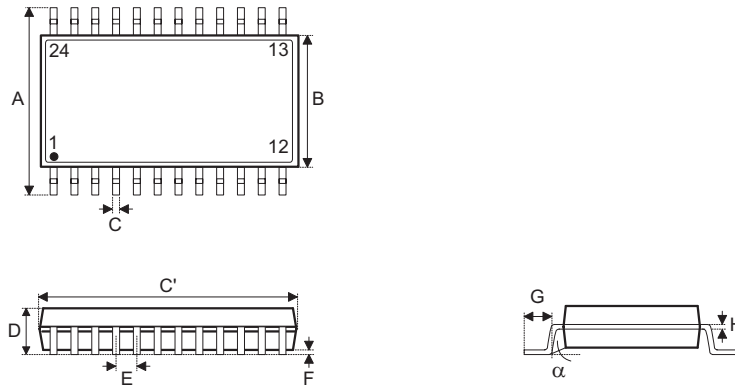
Package Information

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](#)

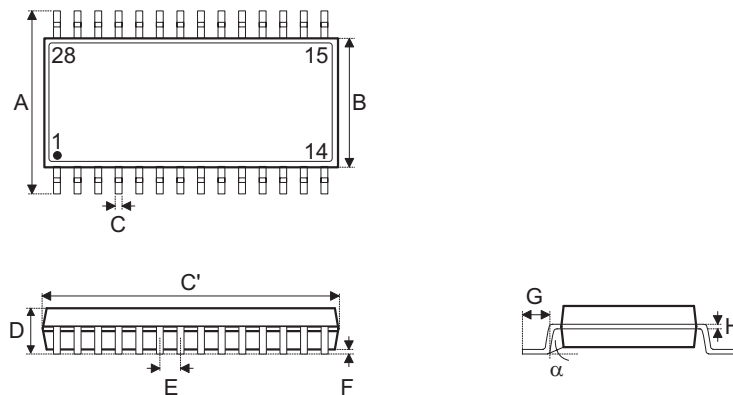
24-pin SSOP (150mil) Outline Dimensions



| Symbol | Dimensions in inch | | |
|--------|--------------------|-----------|-------|
| | Min. | Nom. | Max. |
| A | — | 0.236 BSC | — |
| B | — | 0.154 BSC | — |
| C | 0.008 | — | 0.012 |
| C' | — | 0.341 BSC | — |
| D | — | — | 0.069 |
| E | — | 0.025 BSC | — |
| F | 0.004 | — | 0.010 |
| G | 0.016 | — | 0.050 |
| H | 0.004 | — | 0.010 |
| α | 0° | — | 8° |

| Symbol | Dimensions in mm | | |
|--------|------------------|-----------|------|
| | Min. | Nom. | Max. |
| A | — | 6.000 BSC | — |
| B | — | 3.900 BSC | — |
| C | 0.20 | — | 0.30 |
| C' | — | 8.660 BSC | — |
| D | — | — | 1.75 |
| E | — | 0.635 BSC | — |
| F | 0.10 | — | 0.25 |
| G | 0.41 | — | 1.27 |
| H | 0.10 | — | 0.25 |
| α | 0° | — | 8° |

28-pin SSOP (150mil) Outline Dimensions



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

| Symbol | Dimensions in mm | | |
|----------|------------------|-----------|------|
| | Min. | Nom. | Max. |
| A | — | 6.0 BSC | — |
| B | — | 3.9 BSC | — |
| C | 0.20 | — | 0.30 |
| C' | — | 9.9 BSC | — |
| D | — | — | 1.75 |
| E | — | 0.635 BSC | — |
| F | 0.10 | — | 0.25 |
| G | 0.41 | — | 1.27 |
| H | 0.10 | — | 0.25 |
| α | 0° | — | 8° |

Copyright© 2021 by HOLTEK SEMICONDUCTOR INC.

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