



**Power Bank Flash MCU**

**BP45F4NB/BP45FH4NB**

Revision: V1.20    Date: February 01, 2023

[www.holtek.com](http://www.holtek.com)

## Table of Contents

|                                                                      |           |
|----------------------------------------------------------------------|-----------|
| <b>Features .....</b>                                                | <b>6</b>  |
| CPU Features .....                                                   | 6         |
| Peripheral Features.....                                             | 6         |
| <b>General Description.....</b>                                      | <b>7</b>  |
| <b>Selection Table.....</b>                                          | <b>7</b>  |
| <b>Block Diagram.....</b>                                            | <b>8</b>  |
| <b>Pin Assignment.....</b>                                           | <b>8</b>  |
| <b>Pin Description .....</b>                                         | <b>11</b> |
| Level Shift Input/Output Relationship and Reset Condition .....      | 14        |
| <b>Absolute Maximum Ratings.....</b>                                 | <b>14</b> |
| <b>D.C. Characteristics.....</b>                                     | <b>15</b> |
| Operating Voltage Characteristics.....                               | 15        |
| Operating Current Characteristics.....                               | 15        |
| Standby Current Characteristics .....                                | 15        |
| <b>A.C. Characteristics.....</b>                                     | <b>16</b> |
| High Speed Internal Oscillator – HIRC – Frequency Accuracy .....     | 16        |
| Low Speed Internal Oscillator Characteristics – LIRC .....           | 16        |
| Operating Frequency Characteristic Curves .....                      | 16        |
| System Start Up Time Characteristics .....                           | 17        |
| <b>Input/Output Characteristics .....</b>                            | <b>17</b> |
| Input/Output (without Multi-power) D.C Characteristics .....         | 17        |
| Input/Output (with Multi-power) D.C Characteristics .....            | 19        |
| <b>Memory Characteristics .....</b>                                  | <b>20</b> |
| <b>LVR/LVD Electrical Characteristics .....</b>                      | <b>20</b> |
| <b>A/D Converter Electrical Characteristics.....</b>                 | <b>20</b> |
| <b>Over/Under Voltage Protection Electrical Characteristics.....</b> | <b>21</b> |
| <b>Over Current Protection Electrical Characteristics.....</b>       | <b>22</b> |
| <b>USB Auto Detection Electrical Characteristics.....</b>            | <b>23</b> |
| <b>LDO Regulator Electrical Characteristics .....</b>                | <b>24</b> |
| <b>Level Converter Electrical Characteristics.....</b>               | <b>24</b> |
| <b>Power-on Reset Characteristics.....</b>                           | <b>25</b> |
| <b>System Architecture .....</b>                                     | <b>25</b> |
| Clocking and Pipelining.....                                         | 25        |
| Program Counter.....                                                 | 26        |
| Stack .....                                                          | 27        |
| Arithmetic and Logic Unit – ALU .....                                | 27        |
| <b>Flash Program Memory .....</b>                                    | <b>28</b> |
| Structure.....                                                       | 28        |

|                                                        |           |
|--------------------------------------------------------|-----------|
| Special Vectors .....                                  | 28        |
| Look-up Table.....                                     | 28        |
| Table Program Example.....                             | 29        |
| In Circuit Programming – ICP .....                     | 30        |
| On-Chip Debug Support – OCDS .....                     | 30        |
| In Application Programming – IAP .....                 | 31        |
| <b>Data Memory .....</b>                               | <b>47</b> |
| Structure.....                                         | 47        |
| Data Memory Addressing.....                            | 48        |
| General Purpose Data Memory .....                      | 48        |
| Special Purpose Data Memory .....                      | 48        |
| <b>Special Function Register Description.....</b>      | <b>50</b> |
| Indirect Addressing Registers – IAR0, IAR1, IAR2 ..... | 50        |
| Memory Pointers – MP0, MP1L/MP1H, MP2L/MP2H.....       | 50        |
| Accumulator – ACC.....                                 | 51        |
| Program Counter Low Byte Register – PCL.....           | 52        |
| Look-up Table Registers – TBLP, TBHP, TBLH.....        | 52        |
| Status Register – STATUS.....                          | 52        |
| <b>Oscillators .....</b>                               | <b>53</b> |
| Oscillator Overview .....                              | 53        |
| System Clock Configurations.....                       | 54        |
| Internal High Speed RC Oscillator – HIRC .....         | 54        |
| Internal 32kHz Oscillator – LIRC.....                  | 54        |
| <b>Operating Modes and System Clocks .....</b>         | <b>55</b> |
| System Clocks .....                                    | 55        |
| System Operation Modes.....                            | 55        |
| Control Registers .....                                | 57        |
| Operating Mode Switching.....                          | 58        |
| Standby Current Considerations.....                    | 61        |
| Wake-up .....                                          | 62        |
| <b>Watchdog Timer.....</b>                             | <b>62</b> |
| Watchdog Timer Clock Source.....                       | 62        |
| Watchdog Timer Control Register .....                  | 62        |
| Watchdog Timer Operation .....                         | 64        |
| <b>Reset and Initialisation.....</b>                   | <b>65</b> |
| Reset Functions .....                                  | 65        |
| Reset Initial Conditions .....                         | 67        |
| <b>Input/Output Ports .....</b>                        | <b>71</b> |
| Pull-high Resistors .....                              | 71        |
| Port A Wake-up .....                                   | 72        |
| I/O Port Control Registers.....                        | 72        |
| I/O Port Power Source Control.....                     | 73        |
| I/O Port Source Current Control.....                   | 74        |

|                                                       |            |
|-------------------------------------------------------|------------|
| Pin-shared Functions .....                            | 75         |
| I/O Pin Structure.....                                | 80         |
| Programming Considerations.....                       | 80         |
| <b>Timer Modules – TM .....</b>                       | <b>81</b>  |
| Introduction .....                                    | 81         |
| TM Operation .....                                    | 81         |
| TM Clock Source.....                                  | 81         |
| TM Interrupts.....                                    | 82         |
| TM External Pins.....                                 | 82         |
| Programming Considerations.....                       | 83         |
| <b>Compact Type TM – CTM .....</b>                    | <b>84</b>  |
| Compact Type TM Operation .....                       | 84         |
| Compact Type TM Register Description.....             | 84         |
| Compact Type TM Operating Modes .....                 | 88         |
| <b>Periodic Type TM – PTM.....</b>                    | <b>94</b>  |
| Periodic Type TM Operation.....                       | 94         |
| Periodic Type TM Register Description .....           | 94         |
| Periodic Type TM Operation Modes.....                 | 99         |
| <b>Complementary PWM Output with Dead Time.....</b>   | <b>106</b> |
| Dead Time Insertion .....                             | 106        |
| Complementary PWM Registers.....                      | 107        |
| <b>Analog to Digital Converter .....</b>              | <b>108</b> |
| A/D Converter Overview .....                          | 108        |
| A/D Converter Register Description .....              | 109        |
| A/D Converter Reference Voltage.....                  | 112        |
| A/D Converter Input Signals.....                      | 112        |
| A/D Converter Operation.....                          | 113        |
| Conversion Rate and Timing Diagram .....              | 114        |
| Summary of A/D Conversion Steps.....                  | 115        |
| Programming Considerations.....                       | 115        |
| A/D Conversion Function .....                         | 115        |
| A/D Conversion Programming Examples.....              | 116        |
| <b>Universal Serial Interface Module – USIM .....</b> | <b>118</b> |
| SPI Interface .....                                   | 118        |
| I <sup>2</sup> C Interface .....                      | 126        |
| UART Interface.....                                   | 136        |
| <b>Over Current Protection.....</b>                   | <b>152</b> |
| Over Current Protection Operation .....               | 152        |
| Over Current Protection Registers .....               | 152        |
| Input Voltage Range.....                              | 155        |
| OCP OPA and Comparator Offset Calibration.....        | 155        |
| <b>Over/Under Voltage Protection.....</b>             | <b>157</b> |
| OUVP Circuit Operation .....                          | 157        |

|                                                               |            |
|---------------------------------------------------------------|------------|
| OUPV Register Description.....                                | 158        |
| OVP and UVP Comparator Offset calibration .....               | 160        |
| <b>USB Auto Detection.....</b>                                | <b>162</b> |
| D1+/D1- and D2+/D2- for Auto Detection.....                   | 162        |
| USB Auto Detection Registers .....                            | 163        |
| <b>Interrupts .....</b>                                       | <b>165</b> |
| Interrupt Registers.....                                      | 165        |
| Interrupt Operation .....                                     | 169        |
| External Interrupts.....                                      | 170        |
| A/D Converter Interrupt.....                                  | 171        |
| Over Current Protection Interrupt.....                        | 171        |
| Over Voltage Protection Interrupt.....                        | 171        |
| Under Voltage Protection Interrupts .....                     | 171        |
| Multi-function Interrupts.....                                | 171        |
| Timer Module Interrupts .....                                 | 172        |
| USIM Interrupt.....                                           | 172        |
| Time Base Interrupts .....                                    | 173        |
| LVD Interrupt.....                                            | 174        |
| Interrupt Wake-up Function.....                               | 175        |
| Programming Considerations.....                               | 175        |
| <b>Low Voltage Detector – LVD .....</b>                       | <b>176</b> |
| LVD Register .....                                            | 176        |
| LVD Operation.....                                            | 177        |
| <b>Application Circuits.....</b>                              | <b>178</b> |
| <b>Instruction Set.....</b>                                   | <b>179</b> |
| Introduction .....                                            | 179        |
| Instruction Timing .....                                      | 179        |
| Moving and Transferring Data.....                             | 179        |
| Arithmetic Operations.....                                    | 179        |
| Logical and Rotate Operation .....                            | 180        |
| Branches and Control Transfer .....                           | 180        |
| Bit Operations .....                                          | 180        |
| Table Read Operations .....                                   | 180        |
| Other Operations.....                                         | 180        |
| <b>Instruction Set Summary .....</b>                          | <b>181</b> |
| Table Conventions.....                                        | 181        |
| Extended Instruction Set.....                                 | 183        |
| <b>Instruction Definition.....</b>                            | <b>185</b> |
| Extended Instruction Definition .....                         | 194        |
| <b>Package Information .....</b>                              | <b>201</b> |
| 24-pin SSOP (150mil) Outline Dimensions .....                 | 202        |
| 28-pin SSOP (150mil) Outline Dimensions .....                 | 203        |
| SAW Type 28-pin QFN (4mm×4mm×0.75mm) Outline Dimensions ..... | 204        |

## Features

### CPU Features

- Operating voltage
  - ♦  $f_{SYS}=7.5\text{MHz}$ : 2.6V~5.5V
  - ♦  $f_{SYS}=15\text{MHz}$ : 3.3V~5.5V
- Up to 0.27 $\mu\text{s}$  instruction cycle with 15MHz system clock at  $V_{DD}=5\text{V}$
- Power down and wake-up functions to reduce power consumption
- Oscillator types:
  - ♦ Internal High Speed 30MHz 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
- 8-level subroutine nesting
- Bit manipulation instruction

### Peripheral Features

- Flash Program Memory: 4K $\times$ 16
- Data Memory: 256 $\times$ 8
- Watchdog Timer function
- In Application Programming – IAP
- Up to 26 bidirectional I/O lines
- Two pin-shared external interrupts
- Multiple Timer Modules for time measurement, compare match output or PWM output or single pulse output function
- One complementary PWM output with dead time control
- Two over current protections (OCPn) with interrupt
- Over/Under voltage protection (OUVP) with interrupt
- USB charge/discharge auto detection function
- Dual Time-Base functions for generation of fixed time interrupt signals
- 11 external channel 12-bit resolution A/D converter with an internal reference voltage  $V_{VR}$
- Universal Serial Interface Module – USIM for SPI, I<sup>2</sup>C or UART communication
- Low voltage reset function
- Low voltage detect function
- One integrated LDO: 5V output – BP45FH4NB only
- 2 level shift output pins – BP45FH4NB only
- Package types: 24/28-pin SSOP, 28-pin QFN

## General Description

The devices are Flash Memory 8-bit high performance RISC architecture microcontrollers, specifically designed for Power Bank applications.

For memory features, the Flash Memory offers users the convenience of multi-programming features. Other memory includes an area of RAM Data Memory. By using the In Application Programming technology, users have a convenient means to directly store their measured data in the Flash Program Memory as well as having the ability to easily update their application programs.

Analog features include a multi-channel 12-bit A/D converter, an USB charge/discharge auto detection function and a LDO regulator. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated SPI, I<sup>2</sup>C and UART interface functions, these popular interfaces which provide designers with a means of easy communication with external peripheral hardware. 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.

The devices also include fully integrated high and low speed 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 inclusion of flexible I/O programming features, Time-Base functions along with many other features ensure that the devices will find excellent use in different power bank applications.

Circuitry specific to Power Bank applications is also fully integrated within the devices. These include functions such as over/under voltage protection, over current protection and USB auto detection. These features combine to ensure that a minimum of external components is required to implement Power Bank applications, providing the benefits of reduced component counts and reduced circuit board areas.

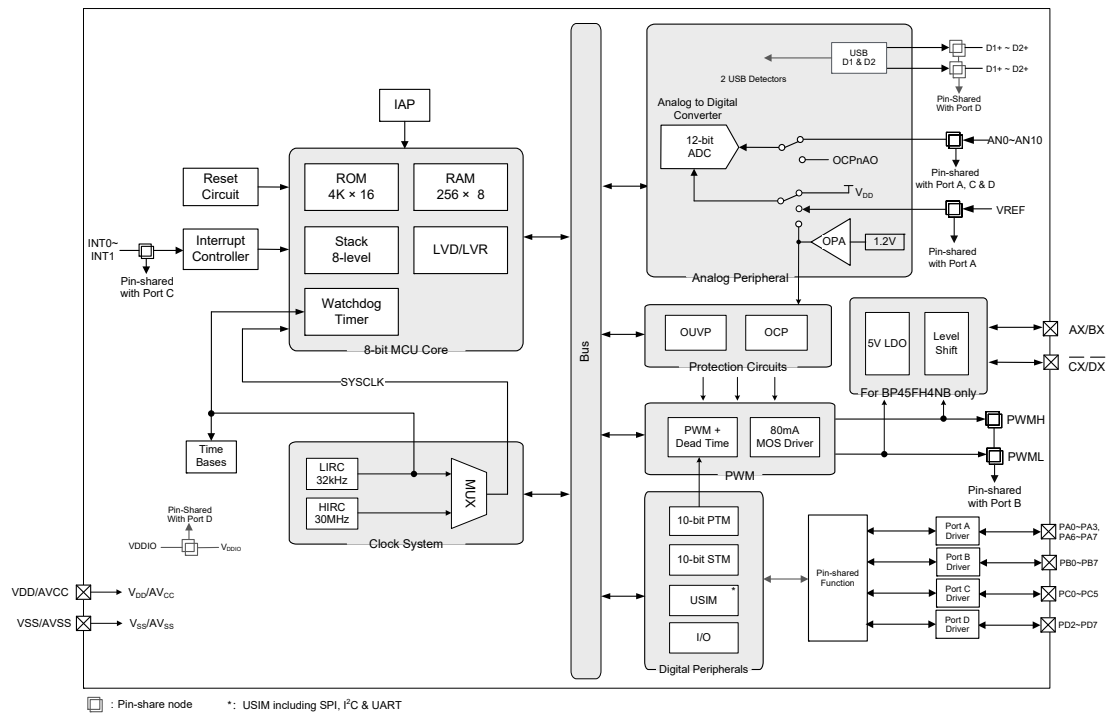
## Selection Table

Most features are common to these devices, the main features distinguishing them are I/O count, Level Shift output pins and package types. The following table summarises the main features of each devices. As the devices exist in more than one package format, the tables reflect the situation for the package with the most pins.

| Part No.  | V <sub>DD</sub> | Program Memory | Data Memory | I/O | Ext. Interrupt | A/D       | USIM | Time Base |
|-----------|-----------------|----------------|-------------|-----|----------------|-----------|------|-----------|
| BP45F4NB  | 2.6V~5.5V       | 4K×16          | 256×8       | 26  | 2              | 12-bit×11 | √    | 2         |
| BP45FH4NB |                 |                |             | 21  |                |           |      |           |

| Part No.  | Timer Module                 | OCP | OUVP | Stacks | USB | LDO | Level Shift Output Pins | Package             |
|-----------|------------------------------|-----|------|--------|-----|-----|-------------------------|---------------------|
| BP45F4NB  | 10-bit CTM×2<br>10-bit PTM×1 | 2   | 1    | 8      | 2   | —   | —                       | 24SSOP/28SSOP/28QFN |
| BP45FH4NB |                              |     |      |        |     | √   | AX/BX<br>CX/DX          | 28SSOP              |

## Block Diagram



## Pin Assignment

|                             |    |    |                   |
|-----------------------------|----|----|-------------------|
| PA0/CTCK1/ICPDA/OCDSDA      | 1  | 24 | PB0/PWMH          |
| PA2/PTCK/ICPCK/OCDSCK       | 2  | 23 | PB1/PWML          |
| PC5/INT1/CTCK0              | 3  | 22 | VDD/AVCC          |
| PB7/SCS/UVPO                | 4  | 21 | VSS/AVSS          |
| PB4/SCK/SCL                 | 5  | 20 | PC2/OCPI          |
| PB5/SDI/SDA/URX/UTX         | 6  | 19 | PC0/OCPI          |
| PB6/SDO/UTX                 | 7  | 18 | PA1/AN9/BATV      |
| PD4/AN0/D1+/SCK/SCL         | 8  | 17 | PA3/AN7           |
| PD5/AN1/D1-/SDI/SDA/URX/UTX | 9  | 16 | PD3/CTP1/OCPOCOUT |
| PD6/AN2/D2+/SDO/UTX         | 10 | 15 | PA7/AN4/VREF      |
| PD7/AN3/D2-/SCS             | 11 | 14 | PA6/AN10/OUVPI    |
| PD2/VDDIO/OVPO              | 12 | 13 | PC4/INT0/AN5/CTP0 |

**BP45F4NB/BP45V4NB**  
**24 SSOP-A**

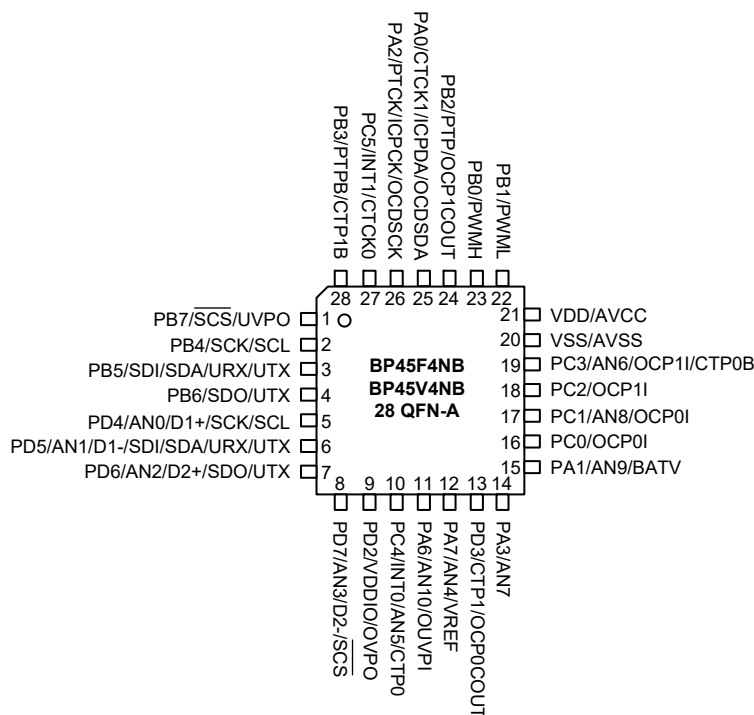


|                               |    |    |                    |
|-------------------------------|----|----|--------------------|
| PA0/CTCK1/ICPDA/OCSDA         | 1  | 28 | PB2/PTP/OC1COUT    |
| PA2/PTCK/ICPCK/OCDSCK         | 2  | 27 | PB0/PWMH           |
| PC5/INT1/CTCK0                | 3  | 26 | PB1/PWML           |
| PB3/PTPB/CTP1B                | 4  | 25 | VDD/AVCC           |
| PB7/ $\overline{SCS}$ /UVPO   | 5  | 24 | VSS/AVSS           |
| PB4/SCK/SCL                   | 6  | 23 | PC3/AN6/OC1I/CTP0B |
| PB5/SDI/SDA/URX/UTX           | 7  | 22 | PC2/OC1I           |
| PB6/SDO/UTX                   | 8  | 21 | PC1/AN8/OC0I       |
| PD4/AN0/D1+/SCK/SCL           | 9  | 20 | PC0/OC0I           |
| PD5/AN1/D1-/SDI/SDA/URX/UTX   | 10 | 19 | PA1/AN9/BATV       |
| PD6/AN2/D2+/SDO/UTX           | 11 | 18 | PA3/AN7            |
| PD7/AN3/D2-/ $\overline{SCS}$ | 12 | 17 | PD3/CTP1/OC0COUT   |
| PD2/VDDIO/OVPO                | 13 | 16 | PA7/AN4/VREF       |
| PC4/INT0/AN5/CTP0             | 14 | 15 | PA6/AN10/OUVPI     |

**BP45F4NB/BP45V4NB**  
**28 SSOP-A**

|                               |    |    |                    |
|-------------------------------|----|----|--------------------|
| $\overline{DX}/\overline{CX}$ | 1  | 28 | VCC                |
| AX/BX                         | 2  | 27 | VSS                |
| PA0/CTCK1/ICPDA/OCSDA         | 3  | 26 | V5/VDD             |
| PA2/PTCK/ICPCK/OCDSCK         | 4  | 25 | VDD/AVCC           |
| PB7/ $\overline{SCS}$ /UVPO   | 5  | 24 | VSS/AVSS           |
| PB4/SCK/SCL                   | 6  | 23 | PC3/AN6/OC1I/CTP0B |
| PB5/SDI/SDA/URX/UTX           | 7  | 22 | PC2/OC1I           |
| PB6/SDO/UTX                   | 8  | 21 | PC1/AN8/OC0I       |
| PD4/AN0/D1+/SCK/SCL           | 9  | 20 | PC0/OC0I           |
| PD5/AN1/D1-/SDI/SDA/URX/UTX   | 10 | 19 | PA1/AN9/BATV       |
| PD6/AN2/D2+/SDO/UTX           | 11 | 18 | PA3/AN7            |
| PD7/AN3/D2-/ $\overline{SCS}$ | 12 | 17 | PD3/CTP1/OC0COUT   |
| PD2/VDDIO/OVPO                | 13 | 16 | PA7/AN4/VREF       |
| PC4/INT0/AN5/CTP0             | 14 | 15 | PA6/AN10/OUVPI     |

**BP45FH4NB/BP45VH4NB**  
**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 BP45V4NB/BP45VH4NB device which is the OCDS EV chip for the BP45F4NB/BP45FH4NB 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.
4. For the BP45FH4NB device:
- The I/O lines, PB0/PWMH and PB1/PWML, are internally connected to the level shift inputs, A and C respectively.
  - The I/O lines, PB2/PTP/OCPI0/OCPI1, PB3/PTPB/CTP1B and PC5/INT1/CTCK0 are not connected to the external pins.
  - The pin 25 and pin 26 must be connected together in the application PCB.

## 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 each Pin Description table shows the situation for the package with the most pins, not all pins in the tables will be available on smaller package sizes.

| Pin Name               | Function | OPT                  | I/T | O/T  | Description                                               |
|------------------------|----------|----------------------|-----|------|-----------------------------------------------------------|
| PA0/CTCK1/ICPDA/OCDSDA | PA0      | PAWU<br>PAPU<br>PAS0 | ST  | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
|                        | CTCK1    | PAS0                 | ST  | —    | CTM1 clock input                                          |
|                        | ICPDA    | —                    | ST  | CMOS | ICP data/address pin                                      |
|                        | OCDSDA   | —                    | ST  | CMOS | OCDs data/address pin, for EV chip only                   |
| PA1/AN9/BATV           | PA1      | PAWU<br>PAPU<br>PAS0 | ST  | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
|                        | AN9      | PAS0                 | AN  | —    | A/D converter external input channel 9                    |
|                        | BATV     | PAS0                 | AN  | —    | A/D converter external input channel                      |
| PA2/PTCK/ICPCK/OCDSCCK | PA2      | PAWU<br>PAPU<br>PAS0 | ST  | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
|                        | PTCK     | PAS0                 | ST  | —    | PTM clock input                                           |
|                        | ICPCK    | —                    | ST  | —    | ICP clock pin                                             |
|                        | OCDSCCK  | —                    | ST  | —    | OCDs clock pin, for EV chip only                          |
| PA3/AN7                | PA3      | PAWU<br>PAPU<br>PAS0 | ST  | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
|                        | AN7      | PAS0                 | AN  | —    | A/D converter external input channel 7                    |
| PA6/AN10/OUVPI         | PA6      | PAWU<br>PAPU<br>PAS1 | ST  | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
|                        | AN10     | PAS1                 | AN  | —    | A/D converter external input channel 10                   |
|                        | OUVPI    | PAS1                 | AN  | —    | OUVP input                                                |
| PA7/AN4/VREF           | PA7      | PAWU<br>PAPU<br>PAS1 | ST  | CMOS | General purpose I/O. Register enabled pull-up and wake-up |
|                        | AN4      | PAS1                 | AN  | —    | A/D converter external input channel 4                    |
|                        | VREF     | PAS1                 | AN  | —    | External reference voltage input                          |
| PB0/PWMH               | PB0      | PBPU<br>PBS0         | ST  | CMOS | General purpose I/O. Register enabled pull-up             |
|                        | PWMH     | PBS0                 | —   | CMOS | PWM output                                                |
| PB1/PWML               | PB1      | PBPU<br>PBS0         | ST  | CMOS | General purpose I/O. Register enabled pull-up             |
|                        | PWML     | PBS0                 | —   | CMOS | Complementary PWM output                                  |
| PB2/PTP/OCP1COUT       | PB2      | PBPU<br>PBS0         | ST  | CMOS | General purpose I/O. Register enabled pull-up             |
|                        | PTP      | PBS0                 | —   | CMOS | PTM output                                                |
|                        | OCP1COUT | PBS0                 | —   | CMOS | OCP1 comparator output                                    |
| PB3/PTPB/CTP1B         | PB3      | PBPU<br>PBS0         | ST  | CMOS | General purpose I/O. Register enabled pull-up             |
|                        | PTPB     | PBS0                 | —   | CMOS | PTM inverting output                                      |
|                        | CTP1B    | PBS0                 | —   | CMOS | CTM1 inverting output                                     |

| Pin Name                           | Function                | OPT                    | I/T | O/T  | Description                                                                                                                  |
|------------------------------------|-------------------------|------------------------|-----|------|------------------------------------------------------------------------------------------------------------------------------|
| PB4/SCK/SCL                        | PB4                     | PBPU<br>PBS1           | ST  | CMOS | General purpose I/O. Register enabled pull-up                                                                                |
|                                    | SCK                     | PBS1<br>IFS            | ST  | CMOS | SPI serial clock                                                                                                             |
|                                    | SCL                     | PBS1<br>IFS            | ST  | NMOS | I <sup>2</sup> C clock line                                                                                                  |
| PB5/SDI/SDA/URX/<br>UTX            | PB5                     | PBPU<br>PBS1           | ST  | CMOS | General purpose I/O. Register enabled pull-up                                                                                |
|                                    | SDI                     | PBS1<br>IFS            | ST  | —    | SPI serial data input                                                                                                        |
|                                    | SDA                     | PBS1<br>IFS            | ST  | NMOS | I <sup>2</sup> C data line                                                                                                   |
|                                    | URX/UTX                 | PBS1<br>IFS            | ST  | CMOS | UART serial data input in full-duplex communication<br>or UART serial data input/output in Single Wire Mode<br>communication |
| PB6/SDO/UTX                        | PB6                     | PBPU<br>PBS1           | ST  | CMOS | General purpose I/O. Register enabled pull-up                                                                                |
|                                    | SDO                     | PBS1                   | —   | CMOS | SPI serial data output                                                                                                       |
|                                    | UTX                     | PBS1                   | —   | CMOS | UART serial data output                                                                                                      |
| PB7/ $\overline{\text{SCS}}$ /UVPO | PB7                     | PBPU<br>PBS1           | ST  | CMOS | General purpose I/O. Register enabled pull-up                                                                                |
|                                    | $\overline{\text{SCS}}$ | PBS1<br>IFS            | ST  | CMOS | SPI slave select                                                                                                             |
|                                    | UVPO                    | PBS1                   | —   | CMOS | UVP comparator output                                                                                                        |
| PC0/OCPOI                          | PC0                     | PCPU<br>PCS0           | ST  | CMOS | General purpose I/O. Register enabled pull-up                                                                                |
|                                    | OCPOI                   | PCS0                   | AN  | —    | OCPOI input signal                                                                                                           |
| PC1/AN8/OCPOI                      | PC1                     | PCPU<br>PCS0           | ST  | CMOS | General purpose I/O. Register enabled pull-up and<br>wake-up                                                                 |
|                                    | AN8                     | PCS0                   | AN  | —    | A/D converter external input channel 8                                                                                       |
|                                    | OCPOI                   | PCS0                   | AN  | —    | OCPOI input signal                                                                                                           |
| PC2/OCPI1                          | PC2                     | PCPU<br>PCS0           | ST  | CMOS | General purpose I/O. Register enabled pull-up and<br>wake-up                                                                 |
|                                    | OCPI1                   | PCS0                   | AN  | —    | OCPI1 input signal                                                                                                           |
| PC3/AN6/OCPI1/<br>CTP0B            | PC3                     | PCPU<br>PCS0           | ST  | CMOS | General purpose I/O. Register enabled pull-up and<br>wake-up                                                                 |
|                                    | AN6                     | PCS0                   | AN  | —    | A/D converter external input channel 6                                                                                       |
|                                    | OCPI1                   | PCS0                   | AN  | —    | OCPI1 input signal                                                                                                           |
|                                    | CTP0B                   | PCS0                   | —   | CMOS | CTM0 inverting output                                                                                                        |
| PC4/INT0/AN5/CTP0                  | PC4                     | PCPU<br>PCS1           | ST  | CMOS | General purpose I/O. Register enabled pull-up and<br>wake-up                                                                 |
|                                    | INT0                    | INTEG<br>INTC1<br>PCS1 | ST  | —    | External Interrupt 0                                                                                                         |
|                                    | AN5                     | PCS1                   | AN  | —    | A/D converter external input channel5                                                                                        |
|                                    | CTP0                    | PCS1                   | —   | CMOS | CTM0 output                                                                                                                  |
| PC5/INT1/CTCK0                     | PC5                     | PCPU<br>PCS1           | ST  | CMOS | General purpose I/O. Register enabled pull-up and<br>wake-up                                                                 |
|                                    | INT1                    | INTEG<br>INTC1<br>PCS1 | ST  | —    | External Interrupt 1                                                                                                         |
|                                    | CTCK0                   | PCS1                   | ST  | —    | CTM0 clock input                                                                                                             |

| Pin Name                    | Function | OPT      | I/T | O/T  | Description                                                                                                            |
|-----------------------------|----------|----------|-----|------|------------------------------------------------------------------------------------------------------------------------|
| PD2/VDDIO/OVPO              | PD2      | PDPUPDS0 | ST  | CMOS | General purpose I/O. Register enabled pull-up and wake-up                                                              |
|                             | VDDIO    | PDS0PMPS | PWR | —    | PD2 pin power                                                                                                          |
|                             | OVPO     | PDS0     | —   | CMOS | OVP comparator output                                                                                                  |
| PD3/CTP1/OCP0COUT           | PD3      | PDPUPDS0 | ST  | CMOS | General purpose I/O. Register enabled pull-up and wake-up                                                              |
|                             | CTP1     | PDS0     | —   | CMOS | CTM1 output                                                                                                            |
|                             | OCP0COUT | PDS0     | —   | CMOS | OCP0 comparator output                                                                                                 |
| PD4/AN0/D1+/SCK/SCL         | PD4      | PDPUPDS1 | ST  | CMOS | General purpose I/O. Register enabled pull-up and wake-up                                                              |
|                             | AN0      | PDS1     | AN  | —    | A/D converter external input channel 0                                                                                 |
|                             | D1+      | PDS1     | —   | AN   | USB DAC0 output pin                                                                                                    |
|                             | SCK      | PDS1IFS  | ST  | CMOS | SPI serial clock                                                                                                       |
|                             | SCL      | PDS1IFS  | ST  | NMOS | I <sup>2</sup> C clock line                                                                                            |
| PD5/AN1/D1-/SDI/SDA/URX/UTX | PD5      | PDPUPDS1 | ST  | CMOS | General purpose I/O. Register enabled pull-up and wake-up                                                              |
|                             | AN1      | PDS1     | AN  | —    | A/D converter external input channel 1                                                                                 |
|                             | D1-      | PDS1     | —   | AN   | USB DAC1 output pin                                                                                                    |
|                             | SDI      | PDS1IFS  | ST  | —    | SPI serial data input                                                                                                  |
|                             | SDA      | PDS1IFS  | ST  | NMOS | I <sup>2</sup> C data line                                                                                             |
|                             | URX/UTX  | PDS1IFS  | ST  | CMOS | UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication |
| PD6/AN2/D2+/SDO/UTX         | PD6      | PDPUPDS1 | ST  | CMOS | General purpose I/O. Register enabled pull-up and wake-up                                                              |
|                             | AN2      | PDS1     | AN  | —    | A/D converter external input channel 2                                                                                 |
|                             | D2+      | PDS1     | —   | AN   | USB DAC0 output pin                                                                                                    |
|                             | SDO      | PDS1     | —   | CMOS | SPI serial data output                                                                                                 |
|                             | UTX      | PDS1     | —   | CMOS | UART serial data output                                                                                                |
| PD7/AN3/D2-/SCS             | PD7      | PDPUPDS1 | ST  | CMOS | General purpose I/O. Register enabled pull-up and wake-up                                                              |
|                             | AN3      | PDS1     | AN  | —    | A/D converter external input channel 3                                                                                 |
|                             | D2-      | PDS1     | —   | AN   | USB DAC1 output pin                                                                                                    |
|                             | SCS      | PDS1IFS  | ST  | CMOS | SPI slave select                                                                                                       |
| VDD/AVCC                    | VDD      | —        | PWR | —    | Digital positive power supply                                                                                          |
|                             | AVCC     | —        | PWR | —    | Analog positive power supply                                                                                           |
| VSS/AVSS                    | VSS      | —        | PWR | —    | Digital negative power supply                                                                                          |
|                             | AVSS     | —        | PWR | —    | Analog negative power supply                                                                                           |

| Pin Name                          | Function                          | OPT | I/T | O/T | Description                                                                     |
|-----------------------------------|-----------------------------------|-----|-----|-----|---------------------------------------------------------------------------------|
| <b>BP45FH4NB Only</b>             |                                   |     |     |     |                                                                                 |
| VCC                               | VCC                               | —   | PWR | —   | LDO power supply and Level shifter output driving supply                        |
| V5                                | V5                                | —   | —   | PWR | 5V LDO output                                                                   |
| AX, BX                            | AX, BX                            | —   | —   | —   | Level shift outputs of input A<br>Internally connected to PB0/PWMH respectively |
| $\overline{CX}$ , $\overline{DX}$ | $\overline{CX}$ , $\overline{DX}$ | —   | —   | —   | Level shift outputs of input C<br>Internally connected to PB1/PWML respectively |

Legend: I/T: Input type;

O/T: Output type;

OPT: Optional by register option;

PWR: Power;

ST: Schmitt Trigger input;

CMOS: CMOS output;

AN: Analog signal;

NMOS: NMOS output.

### Level Shift Input/Output Relationship and Reset Condition

| Level Shift Output | Level Shift Input |              | Reset |
|--------------------|-------------------|--------------|-------|
|                    | A input Low       | A input High |       |
| AX, BX             | Low               | High         | High  |

| Level Shift Output                | Level Shift Input |              | Reset |
|-----------------------------------|-------------------|--------------|-------|
|                                   | C input Low       | C input High |       |
| $\overline{CX}$ , $\overline{DX}$ | High              | Low          | Low   |

### Absolute Maximum Ratings

|                               |                                  |
|-------------------------------|----------------------------------|
| Supply Voltage .....          | $V_{SS}-0.3V$ to $6.0V$          |
| Input Voltage .....           | $V_{SS}-0.3V$ to $V_{DD}+0.3V$   |
| Storage Temperature.....      | $-60^{\circ}C$ to $150^{\circ}C$ |
| Operating Temperature.....    | $-40^{\circ}C$ to $85^{\circ}C$  |
| $I_{OH}$ Total .....          | $-80mA$                          |
| $I_{OL}$ Total .....          | $80mA$                           |
| Total Power Dissipation ..... | $500mW$                          |

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

## D.C. Characteristics

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

### Operating Voltage Characteristics

Ta=-40°C~85°C

| Symbol          | Parameter                | Test Conditions                               | Min. | Typ. | Max. | Unit |
|-----------------|--------------------------|-----------------------------------------------|------|------|------|------|
| V <sub>DD</sub> | Operating Voltage – HIRC | f <sub>SYS</sub> =f <sub>HIRC</sub> /4=7.5MHz | 2.6  | —    | 5.5  | V    |
|                 |                          | f <sub>SYS</sub> =f <sub>HIRC</sub> /2=15MHz  | 3.3  | —    | 5.5  |      |
|                 | Operating Voltage – LIRC | f <sub>SYS</sub> =32kHz                       | 2.6  | —    | 5.5  |      |

### Operating Current Characteristics

Ta=-40°C~85°C

| Symbol          | Operating Mode   | Test Conditions |                                               | Min. | Typ. | Max. | Unit |
|-----------------|------------------|-----------------|-----------------------------------------------|------|------|------|------|
|                 |                  | V <sub>DD</sub> | Conditions                                    |      |      |      |      |
| I <sub>DD</sub> | SLOW Mode – LIRC | 2.6V            | f <sub>SYS</sub> =32kHz                       | —    | 8    | 16   | μA   |
|                 |                  | 3V              |                                               | —    | 10   | 20   |      |
|                 |                  | 5V              |                                               | —    | 30   | 50   |      |
|                 | FAST Mode – HIRC | 2.6V            | f <sub>SYS</sub> =f <sub>HIRC</sub> /4=7.5MHz | —    | 1.7  | 2.1  | mA   |
|                 |                  | 3V              |                                               | —    | 2.1  | 2.5  |      |
|                 |                  | 5V              |                                               | —    | 4.4  | 5.3  |      |
|                 |                  | 3.3V            | f <sub>SYS</sub> =f <sub>HIRC</sub> /2=15MHz  | —    | 3.1  | 3.7  |      |
|                 |                  | 5V              |                                               | —    | 4.8  | 5.8  |      |

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.

### Standby Current Characteristics

Ta=25°C, unless otherwise specified

| Symbol           | Standby Mode      | Test Conditions |                                                                    | Min. | Typ. | Max. | Max.<br>@85°C | Unit |
|------------------|-------------------|-----------------|--------------------------------------------------------------------|------|------|------|---------------|------|
|                  |                   | V <sub>DD</sub> | Conditions                                                         |      |      |      |               |      |
| I <sub>STB</sub> | SLEEP Mode        | 2.6V            | WDT on                                                             | —    | 2.4  | 4.0  | 4.6           | μA   |
|                  |                   | 3V              |                                                                    | —    | 3.0  | 5.0  | 5.7           |      |
|                  |                   | 5V              |                                                                    | —    | 5    | 10   | 11            |      |
|                  | IDLE0 Mode – LIRC | 2.6V            | f <sub>SUB</sub> on                                                | —    | 2.4  | 4.0  | 4.6           | μA   |
|                  |                   | 3V              |                                                                    | —    | 3.0  | 5.0  | 5.7           |      |
|                  |                   | 5V              |                                                                    | —    | 5    | 10   | 11            |      |
|                  | IDLE1 Mode – HIRC | 2.6V            | f <sub>SUB</sub> on, f <sub>SYS</sub> =f <sub>HIRC</sub> /4=7.5MHz | —    | 1.5  | 2.3  | 2.3           | mA   |
|                  |                   | 3V              |                                                                    | —    | 1.9  | 2.8  | 2.8           |      |
|                  |                   | 5V              |                                                                    | —    | 3.8  | 5.6  | 5.6           |      |
|                  |                   | 3.3V            | f <sub>SYS</sub> =f <sub>HIRC</sub> /2=15MHz                       | —    | 2.3  | 2.7  | 2.7           |      |
|                  |                   | 5V              |                                                                    | —    | 4.2  | 5.0  | 5.0           |      |

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.

## 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 5V.

| Symbol            | Parameter                           | Test Conditions |                                                                | Min. | Typ. | Max. | Unit |
|-------------------|-------------------------------------|-----------------|----------------------------------------------------------------|------|------|------|------|
|                   |                                     | V <sub>DD</sub> | Condition                                                      |      |      |      |      |
| f <sub>HIRC</sub> | 30MHz Writer Trimmed HIRC Frequency | 5V              | Ta=25°C<br>f <sub>SYS</sub> =f <sub>HIRC</sub> /4=7.5MHz       | -2%  | 30   | +2%  | MHz  |
|                   |                                     | 5V              | Ta=-40°C~85°C<br>f <sub>SYS</sub> =f <sub>HIRC</sub> /4=7.5MHz | -7%  | 30   | +7%  |      |
|                   |                                     | 2.6V~5.5V       | Ta=-40°C~85°C<br>f <sub>SYS</sub> =f <sub>HIRC</sub> /4=7.5MHz | -18% | 30   | +18% |      |

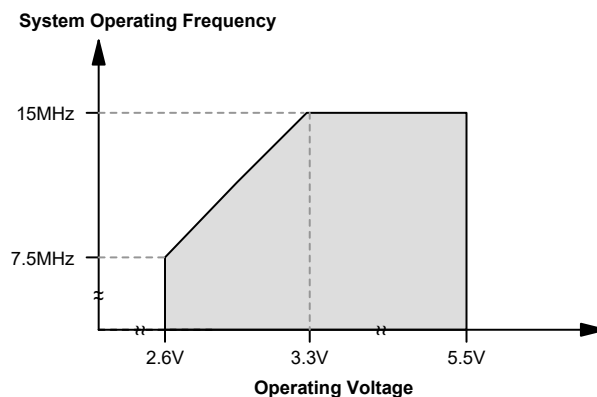
Note: 1. The 5V values for V<sub>DD</sub> are provided as this is the fixed voltage at which the HIRC frequency is trimmed by the writer.

2. The row below the 5V trim voltage row is provided to show the values for the full V<sub>DD</sub> range operating voltage. It is recommended that the trim voltage is fixed at 5V for application voltage ranges from 2.6V to 5.5V.

### Low Speed Internal Oscillator Characteristics – LIRC

| Symbol             | Parameter          | Test Conditions |            | Min. | Typ. | Max. | Unit |
|--------------------|--------------------|-----------------|------------|------|------|------|------|
|                    |                    | V <sub>DD</sub> | Temp.      |      |      |      |      |
| f <sub>LIRC</sub>  | LIRC Frequency     | 5V              | 25°C       | 25.6 | 32.0 | 38.4 | kHz  |
|                    |                    | 2.6V~5.5V       | 25°C       | 12.8 | 32.0 | 41.6 |      |
|                    |                    | —               | -40°C~85°C | 8    | 32   | 60   |      |
| t <sub>START</sub> | LIRC Start Up Time | —               | —          | —    | —    | 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 <sub>DD</sub> | Conditions                                                                              |      |      |      |                   |
| t <sub>SST</sub>    | System Start-up Time                                                                 | —               | f <sub>SYS</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HIRC</sub> | —    | 16   | —    | t <sub>HIRC</sub> |
|                     | Wake-up from Condition where f <sub>SYS</sub> is off                                 | —               | f <sub>SYS</sub> =f <sub>SUB</sub> =f <sub>LIRC</sub>                                   | —    | 2    | —    | t <sub>LIRC</sub> |
|                     | System Start-up Time                                                                 | —               | f <sub>SYS</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HIRC</sub> | —    | 2    | —    | t <sub>H</sub>    |
|                     | Wake-up from condition where f <sub>SYS</sub> is on                                  | —               | f <sub>SYS</sub> =f <sub>SUB</sub> =f <sub>LIRC</sub>                                   | —    | 2    | —    | t <sub>SUB</sub>  |
|                     | System Speed Switch Time<br>FAST to SLOW Mode or<br>SLOW to FAST Mode                | —               | f <sub>HIRC</sub> switches from off → on                                                | —    | 16   | —    | t <sub>HIRC</sub> |
| t <sub>RSTD</sub>   | System Reset Delay Time<br>Reset Source from Power-on reset or<br>LVR Hardware Reset | —               | RR <sub>POR</sub> =5V/ms                                                                | 10   | 16   | 38   | ms                |
|                     | System Reset Delay Time<br>LVRC/WDTC Software Reset                                  | —               | —                                                                                       |      |      |      | ms                |
|                     | System Reset Delay Time<br>Reset Source from WDT Overflow                            | —               | —                                                                                       |      |      |      |                   |
| t <sub>SRESET</sub> | Minimum Software Reset Width to Reset                                                | —               | —                                                                                       | 45   | 90   | 375  | μs                |

- Note: 1. For the System Start-up time values, whether f<sub>SYS</sub> is on or off depends upon the mode type and the chosen f<sub>SYS</sub> system oscillator. Details are provided in the System Operating Modes section.
2. The time units, shown by the symbols t<sub>HIRC</sub> etc. are the inverse of the corresponding frequency values as provided in the frequency tables. For example t<sub>HIRC</sub>=1/f<sub>HIRC</sub>, t<sub>SYS</sub>=1/f<sub>SYS</sub> 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<sub>START</sub>, as provided in the LIRC frequency table, must be added to the t<sub>SST</sub> 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

### Input/Output (without Multi-power) D.C Characteristics

Ta=-40°C~85°C

| Symbol          | Parameter                                                             | Test Conditions |                                     | Min.               | Typ. | Max.               | Unit |
|-----------------|-----------------------------------------------------------------------|-----------------|-------------------------------------|--------------------|------|--------------------|------|
|                 |                                                                       | V <sub>DD</sub> | Conditions                          |                    |      |                    |      |
| V <sub>IL</sub> | Input Low Voltage for I/O Ports except<br>PB4~PB7, PD4~PD7 Pins       | 5V              | —                                   | 0                  | —    | 1.5                | V    |
|                 |                                                                       | —               | —                                   | 0                  | —    | 0.2V <sub>DD</sub> |      |
| V <sub>IH</sub> | Input High Voltage for I/O Ports except<br>PB4~PB7, PD4~PD7 Pins      | 5V              | —                                   | 3.5                | —    | 5.0                | V    |
|                 |                                                                       | —               | —                                   | 0.8V <sub>DD</sub> | —    | V <sub>DD</sub>    |      |
| I <sub>OL</sub> | Sink Current for I/O Pins except PB0~PB1,<br>PB4~PB7 and PD4~PD7 Pins | 3V              | V <sub>OL</sub> =0.1V <sub>DD</sub> | 16                 | 32   | —                  | mA   |
|                 |                                                                       | 5V              |                                     | 32                 | 65   | —                  |      |
|                 | Sink Current for PB0 and PB1 Pins                                     | 3V              | V <sub>OL</sub> =0.1V <sub>DD</sub> | 16                 | 32   | —                  | mA   |
|                 |                                                                       | 5V              |                                     | 40                 | 80   | —                  |      |

| Symbol            | Parameter                                                                                                      | Test Conditions |                                                                            | Min. | Typ. | Max. | Unit |
|-------------------|----------------------------------------------------------------------------------------------------------------|-----------------|----------------------------------------------------------------------------|------|------|------|------|
|                   |                                                                                                                | V <sub>DD</sub> | Conditions                                                                 |      |      |      |      |
| I <sub>OH</sub>   | Source Current for PD3 Pins                                                                                    | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub>                                        | -4   | -8   | —    | mA   |
|                   |                                                                                                                | 5V              |                                                                            | -8   | -16  | —    |      |
|                   | Source Current for PB0 and PB1 Pins                                                                            | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub>                                        | -16  | -32  | —    | mA   |
|                   |                                                                                                                | 5V              |                                                                            | -40  | -80  | —    |      |
|                   | Source Current for I/O Ports for PA0~PA3, PA6~PA7, PB2~PB3, PC0~PC5 and PD2 Pins                               | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDCn[m+1:m]=00B (n=0~1; m=0, 2, 4) | -0.7 | -1.5 | —    | mA   |
|                   |                                                                                                                | 5V              | SLEDCn[m+1:m]=00B (n=0~1; m=0, 2, 4)                                       | -1.5 | -2.9 | —    |      |
|                   |                                                                                                                | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDCn[m+1:m]=01B (n=0~1; m=0, 2, 4) | -1.3 | -2.5 | —    |      |
|                   |                                                                                                                | 5V              | SLEDCn[m+1:m]=01B (n=0~1; m=0, 2, 4)                                       | -2.5 | -5.1 | —    |      |
|                   |                                                                                                                | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDCn[m+1:m]=10B (n=0~1; m=0, 2, 4) | -1.8 | -3.6 | —    |      |
|                   |                                                                                                                | 5V              | SLEDCn[m+1:m]=10B (n=0~1; m=0, 2, 4)                                       | -3.6 | -7.3 | —    |      |
|                   |                                                                                                                | 3V              | V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDCn[m+1:m]=11B (n=0~1; m=0, 2, 4) | -4   | -8   | —    |      |
|                   |                                                                                                                | 5V              | SLEDCn[m+1:m]=11B (n=0~1; m=0, 2, 4)                                       | -8   | -16  | —    |      |
| I <sub>LEAK</sub> | Input Leakage Current for I/O Ports except PB4~PB7, PD4~PD7 Pins                                               | 5V              | V <sub>IN</sub> =V <sub>DD</sub> or V <sub>IN</sub> =V <sub>SS</sub>       | —    | —    | ±1   | μA   |
| R <sub>PL</sub>   | Internal Pull-low Resistance for PB1 <sup>(1)</sup>                                                            | 3V              | —                                                                          | 20   | 30   | 40   | kΩ   |
|                   |                                                                                                                | 5V              | —                                                                          | 20   | 30   | 40   |      |
| R <sub>PH1</sub>  | Register Controlled Pull-high Resistance for I/O Ports except PB0~PB1, PB4~PB7 and PD3~PD7 Pins <sup>(2)</sup> | 3V              | —                                                                          | 20   | 60   | 100  | kΩ   |
|                   |                                                                                                                | 5V              | —                                                                          | 10   | 30   | 50   |      |
| R <sub>PH2</sub>  | Internal Pull-high Resistance for PB0 and PD3 <sup>(3)</sup>                                                   | 3V              | —                                                                          | 20   | 30   | 40   | kΩ   |
|                   |                                                                                                                | 5V              | —                                                                          | 20   | 30   | 40   |      |
| t <sub>INT</sub>  | External Interrupt Minimum Pulse Width                                                                         | —               | —                                                                          | 10   | —    | —    | μs   |
| t <sub>TCK</sub>  | xTM Clock Input Pin Minimum Pulse Width                                                                        | —               | —                                                                          | 0.3  | —    | —    | μs   |

- Note: 1. The R<sub>PL</sub> internal pull low resistance value is calculated by connecting to V<sub>DD</sub> and enabling input pin without 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<sub>PL</sub> value.
2. The R<sub>PH1</sub> internal pull high resistance value is calculated by connecting to ground and enabling the input pin with a register controlled 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<sub>PH1</sub> value.
3. The R<sub>PH2</sub> internal pull high resistance value is calculated by connecting to ground and enabling input pin without R<sub>PH1</sub> 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<sub>PH2</sub> value.

**Input/Output (with Multi-power) D.C Characteristics**

Ta=-40°C~85°C

| Symbol            | Parameter                                                | Test Conditions |                                                                                                                          | Min.                                             | Typ.  | Max.                                             | Unit |
|-------------------|----------------------------------------------------------|-----------------|--------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------|-------|--------------------------------------------------|------|
|                   |                                                          | V <sub>DD</sub> | Conditions                                                                                                               |                                                  |       |                                                  |      |
| V <sub>DD</sub>   | V <sub>DD</sub> Power Supply for PB4~PB7, PD4~PD7 Pins   | —               | —                                                                                                                        | 2.6                                              | 5.0   | 5.5                                              | V    |
| V <sub>DDIO</sub> | V <sub>DDIO</sub> Power Supply for PB4~PB7, PD4~PD7 Pins | —               | —                                                                                                                        | 2.6                                              | —     | V <sub>DD</sub>                                  | V    |
| V <sub>IL</sub>   | Input Low Voltage for PB4~PB7, PD4~PD7 Pins              | 5V              | Pin power=V <sub>DD</sub> or V <sub>DDIO</sub><br>V <sub>DDIO</sub> =V <sub>DD</sub>                                     | 0                                                | —     | 1.5                                              | V    |
|                   |                                                          | —               | Pin power=V <sub>DD</sub> or V <sub>DDIO</sub>                                                                           | 0                                                | —     | 0.2<br>(V <sub>DD</sub> /<br>V <sub>DDIO</sub> ) |      |
| V <sub>IH</sub>   | Input High Voltage for PB4~PB7, PD4~PD7 Pins             | 5V              | Pin power=V <sub>DD</sub> or V <sub>DDIO</sub><br>V <sub>DDIO</sub> =V <sub>DD</sub>                                     | 3.5                                              | —     | 5.0                                              | V    |
|                   |                                                          | —               | Pin power=V <sub>DD</sub> or V <sub>DDIO</sub>                                                                           | 0.8<br>(V <sub>DD</sub> /<br>V <sub>DDIO</sub> ) | —     | V <sub>DD</sub> /<br>V <sub>DDIO</sub>           |      |
| I <sub>OL</sub>   | Sink Current for PB4~PB7, PD4~PD7 Pins                   | 3V              | V <sub>OL</sub> =0.1(V <sub>DD</sub> /V <sub>DDIO</sub> ), V <sub>DDIO</sub> =V <sub>DD</sub>                            | 16                                               | 32    | —                                                | mA   |
|                   |                                                          | 5V              | V <sub>OL</sub> =0.1(V <sub>DD</sub> /V <sub>DDIO</sub> ), V <sub>DDIO</sub> =V <sub>DD</sub>                            | 32                                               | 65    | —                                                |      |
|                   |                                                          | 5V              | V <sub>OL</sub> =0.1V <sub>DDIO</sub> , V <sub>DDIO</sub> =3V                                                            | 20                                               | 40    | —                                                | mA   |
| I <sub>OH</sub>   | Source Current for PB4~PB7, PD4~PD7 Pins                 | 3V              | V <sub>OH</sub> =0.9(V <sub>DD</sub> /V <sub>DDIO</sub> ), V <sub>DDIO</sub> =V <sub>DD</sub><br>SLEDCn[7:6]=00B (n=0~1) | -0.7                                             | -1.5  | —                                                | mA   |
|                   |                                                          | 5V              | V <sub>OH</sub> =0.9(V <sub>DD</sub> /V <sub>DDIO</sub> ), V <sub>DDIO</sub> =V <sub>DD</sub><br>SLEDCn[7:6]=00B (n=0~1) | -1.5                                             | -2.9  | —                                                |      |
|                   |                                                          | 5V              | V <sub>OH</sub> =0.9V <sub>DDIO</sub> , V <sub>DDIO</sub> =3V<br>SLEDCn[7:6]=00B (n=0~1)                                 | -0.40                                            | -0.85 | —                                                | mA   |
|                   |                                                          | 3V              | V <sub>OH</sub> =0.9(V <sub>DD</sub> /V <sub>DDIO</sub> ), V <sub>DDIO</sub> =V <sub>DD</sub><br>SLEDCn[7:6]=01B (n=0~1) | -1.3                                             | -2.5  | —                                                | mA   |
|                   |                                                          | 5V              | V <sub>OH</sub> =0.9(V <sub>DD</sub> /V <sub>DDIO</sub> ), V <sub>DDIO</sub> =V <sub>DD</sub><br>SLEDCn[7:6]=01B (n=0~1) | -2.5                                             | -5.1  | —                                                |      |
|                   |                                                          | 5V              | V <sub>OH</sub> =0.9V <sub>DDIO</sub> , V <sub>DDIO</sub> =3V<br>SLEDCn[7:6]=01B (n=0~1)                                 | -0.70                                            | -1.35 | —                                                | mA   |
|                   |                                                          | 3V              | V <sub>OH</sub> =0.9(V <sub>DD</sub> /V <sub>DDIO</sub> ), V <sub>DDIO</sub> =V <sub>DD</sub><br>SLEDCn[7:6]=10B (n=0~1) | -1.8                                             | -3.6  | —                                                | mA   |
|                   |                                                          | 5V              | V <sub>OH</sub> =0.9(V <sub>DD</sub> /V <sub>DDIO</sub> ), V <sub>DDIO</sub> =V <sub>DD</sub><br>SLEDCn[7:6]=10B (n=0~1) | -3.6                                             | -7.3  | —                                                |      |
|                   |                                                          | 5V              | V <sub>OH</sub> =0.9V <sub>DDIO</sub> , V <sub>DDIO</sub> =3V<br>SLEDCn[7:6]=10B (n=0~1)                                 | -0.95                                            | -1.90 | —                                                | mA   |
|                   |                                                          | 3V              | V <sub>OH</sub> =0.9(V <sub>DD</sub> /V <sub>DDIO</sub> ), V <sub>DDIO</sub> =V <sub>DD</sub><br>SLEDCn[7:6]=11B (n=0~1) | -4                                               | -8    | —                                                |      |
|                   |                                                          | 5V              | V <sub>OH</sub> =0.9(V <sub>DD</sub> /V <sub>DDIO</sub> ), V <sub>DDIO</sub> =V <sub>DD</sub><br>SLEDCn[7:6]=11B (n=0~1) | -8                                               | -16   | —                                                | mA   |
|                   |                                                          | 5V              | V <sub>OH</sub> =0.9V <sub>DDIO</sub> , V <sub>DDIO</sub> =3V<br>SLEDCn[7:6]=11B (n=0~1)                                 | -2.5                                             | -5.0  | —                                                |      |
| R <sub>PH</sub>   | Pull-high Resistance for PB4~PB7, PD4~PD7 Pins (Note)    | 3V              | V <sub>DDIO</sub> =V <sub>DD</sub> , PxPU=FFH(x=B, D)                                                                    | 20                                               | 60    | 100                                              | kΩ   |
|                   |                                                          | 5V              | V <sub>DDIO</sub> =V <sub>DD</sub> , PxPU=FFH(x=B, D)                                                                    | 10                                               | 30    | 50                                               |      |
|                   |                                                          | 5V              | V <sub>DDIO</sub> =3V, PxPU=FFH(x=B, D)                                                                                  | 36                                               | 110   | 180                                              |      |
| I <sub>LEAK</sub> | Input Leakage Current for PB4~PB7, PD4~PD7 Pins          | 5V              | V <sub>IN</sub> =V <sub>SS</sub> or V <sub>IN</sub> =V <sub>DD</sub> or V <sub>DDIO</sub>                                | —                                                | —     | ±1                                               | μA   |

Note: The R<sub>PH</sub> internal pull high resistance value is calculated by connecting to ground and enabling the input pin with 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<sub>PH</sub> value.

## Memory Characteristics

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

| Symbol                      | Parameter                               | Test Conditions |            | Min.               | Typ. | Max.               | Unit |
|-----------------------------|-----------------------------------------|-----------------|------------|--------------------|------|--------------------|------|
|                             |                                         | V <sub>DD</sub> | Conditions |                    |      |                    |      |
| V <sub>DD</sub>             | Operating Voltage for Read/Write        | —               | —          | V <sub>DDmin</sub> | —    | V <sub>DDmax</sub> | V    |
| <b>Flash Program Memory</b> |                                         |                 |            |                    |      |                    |      |
| t <sub>DEW</sub>            | Erase/Write Time – Flash Program Memory | 5.0V            | —          | —                  | 2    | 3                  | ms   |
| E <sub>P</sub>              | Cell Endurance                          | —               | —          | 10K                | —    | —                  | E/W  |
| t <sub>RETD</sub>           | ROM Data Retention time                 | —               | Ta=25°C    | —                  | 40   | —                  | Year |
| <b>RAM Data Memory</b>      |                                         |                 |            |                    |      |                    |      |
| V <sub>DR</sub>             | RAM Data Retention voltage              | —               | —          | 1.0                | —    | —                  | V    |

Note: “E/W” means Erase/Write times.

## LVR/LVD Electrical Characteristics

Ta=-40°C~85°C

| Symbol              | Parameter                              | Test Conditions |                                                                         | Min. | Typ. | Max. | Unit |
|---------------------|----------------------------------------|-----------------|-------------------------------------------------------------------------|------|------|------|------|
|                     |                                        | V <sub>DD</sub> | Conditions                                                              |      |      |      |      |
| V <sub>LVR</sub>    | Low Voltage Reset Voltage              | —               | LVR enable, voltage select 2.55V                                        | - 5% | 2.55 | +5%  | V    |
| V <sub>LVD</sub>    | Low Voltage Detection Voltage          | —               | LVD enable, voltage select 2.7V                                         | -5%  | 2.7  | +5%  | V    |
|                     |                                        | —               | LVD enable, voltage select 3.0V                                         |      | 3.0  |      |      |
|                     |                                        | —               | LVD enable, voltage select 3.3V                                         |      | 3.3  |      |      |
|                     |                                        | —               | LVD enable, voltage select 3.6V                                         |      | 3.6  |      |      |
|                     |                                        | —               | LVD enable, voltage select 4.0V                                         |      | 4.0  |      |      |
| I <sub>LVRLVD</sub> | Operating current                      | 3V              | LVD enable, LVR enable, V <sub>LVR</sub> =2.55V, V <sub>LVD</sub> =2.7V | —    | —    | 10   | μA   |
|                     |                                        | 5V              | LVD enable, LVR enable, V <sub>LVR</sub> =2.55V, V <sub>LVD</sub> =2.7V | —    | 8    | 15   | μA   |
| t <sub>LVDS</sub>   | LVDO Stable Time                       | —               | For LVR enable, LVD off → on                                            | —    | —    | 18   | μs   |
| t <sub>LVR</sub>    | Minimum Low Voltage Width to Reset     | —               | —                                                                       | 140  | 600  | 1000 | μs   |
| t <sub>LVD</sub>    | Minimum Low voltage Width to Interrupt | —               | —                                                                       | 40   | 150  | 320  | μs   |
| I <sub>LVR</sub>    | Additional Current for LVR Enable      | —               | LVD disable                                                             | —    | —    | 8    | μA   |

## A/D Converter Electrical Characteristics

Ta=-40°C~85°C

| Symbol             | Parameter                                   | Test Conditions |                                                              | Min. | Typ. | Max.             | Unit              |
|--------------------|---------------------------------------------|-----------------|--------------------------------------------------------------|------|------|------------------|-------------------|
|                    |                                             | V <sub>DD</sub> | Conditions                                                   |      |      |                  |                   |
| V <sub>ADI</sub>   | A/D Converter Input Voltage                 | —               | —                                                            | 0    | —    | V <sub>REF</sub> | V                 |
| V <sub>REF</sub>   | A/D Converter Reference Voltage             | —               | —                                                            | 2    | —    | V <sub>DD</sub>  | V                 |
| N <sub>R</sub>     | A/D Converter Resolution                    | —               | —                                                            | —    | —    | 12               | Bit               |
| DNL                | Differential Non-linearity                  | —               | V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs | -3   | —    | +3               | LSB               |
| INL                | Integral Non-linearity                      | —               | V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs | -4   | —    | +4               | LSB               |
| I <sub>ADC</sub>   | Additional Current for A/D Converter Enable | 2.6V            | No load (t <sub>ADCK</sub> =0.5μs)                           | —    | 300  | 420              | μA                |
|                    |                                             | 3V              |                                                              | —    | 340  | 500              |                   |
|                    |                                             | 5V              |                                                              | —    | 500  | 700              |                   |
| t <sub>ADCK</sub>  | Clock period                                | —               | —                                                            | 0.5  | —    | 10.0             | μs                |
| t <sub>ON2ST</sub> | A/D Converter On-to-start Time              | —               | —                                                            | 4    | —    | —                | μs                |
| t <sub>ADS</sub>   | Sampling Time                               | —               | —                                                            | —    | 4    | —                | t <sub>ADCK</sub> |

| Symbol             | Parameter                                             | Test Conditions |                                   | Min. | Typ. | Max. | Unit              |
|--------------------|-------------------------------------------------------|-----------------|-----------------------------------|------|------|------|-------------------|
|                    |                                                       | V <sub>DD</sub> | Conditions                        |      |      |      |                   |
| t <sub>ADC</sub>   | Conversion Time<br>(Include A/D Sample and Hold Time) | —               | —                                 | —    | 16   | —    | t <sub>ADCK</sub> |
| GERR               | A/D conversion gain error                             | —               | V <sub>REF</sub> =V <sub>DD</sub> | -4   | —    | +4   | LSB               |
| OSRR               | A/D conversion offset error                           | —               | V <sub>REF</sub> =V <sub>DD</sub> | -4   | —    | +4   | LSB               |
| V <sub>VR</sub>    | OPA Output Voltage                                    | 2.6V~5.5V       | —                                 | -1%  | 2.4  | +1%  | V                 |
| R <sub>BATV</sub>  | The Sum of BATV_R1 and BATV_R2                        | 3V              | —                                 | 2    | 4    | 6    | kΩ                |
|                    |                                                       | 5V              | —                                 | 2    | 4    | 6    |                   |
| R <sub>RBATV</sub> | The Ratio of BATV_R1/BATV_R2                          | 3V              | —                                 | -1%  | 1:1  | +1%  | —                 |
|                    |                                                       | 5V              | —                                 | -1%  | 1:1  | +1%  |                   |
| R <sub>OVP</sub>   | The Sum of OVP_R1, OVP_R2 and OVP_R3                  | 3V              | —                                 | 1.5  | 3.0  | 4.5  | kΩ                |
|                    |                                                       | 5V              | —                                 | 1.5  | 3.0  | 4.5  |                   |
| RR <sub>OVP</sub>  | The Ratio of (OVP_R1+OVP_R2)/OVP_R3                   | 3V              | —                                 | -1%  | 2:1  | +1%  | —                 |
|                    |                                                       | 5V              | —                                 | -1%  | 2:1  | +1%  |                   |

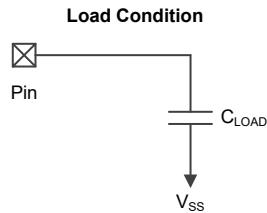
## Over/Under Voltage Protection Electrical Characteristics

Ta=-40°C~85°C

| Symbol            | Parameter                  | Test Conditions |                                               | Min.            | Typ. | Max.                 | Unit |
|-------------------|----------------------------|-----------------|-----------------------------------------------|-----------------|------|----------------------|------|
|                   |                            | V <sub>DD</sub> | Conditions                                    |                 |      |                      |      |
| I <sub>OUVP</sub> | Operating Current          | 3V              | UVPEN=1, OVPEN=1, DAC V <sub>REF</sub> =2.5V  | —               | 520  | 720                  | μA   |
|                   |                            | 5V              |                                               | —               | 550  | 750                  |      |
| V <sub>OS</sub>   | Input Offset Voltage       | 3V              | With calibration                              | -2              | —    | 2                    | mV   |
|                   |                            | 5V              |                                               | -2              | —    | 2                    |      |
|                   |                            | 3V              | Without calibration<br>OVPCOF[4:0]=10000B     | -10             | —    | 10                   | mV   |
|                   |                            | 3V              | Without calibration<br>UVPCOF[4:0]=10000B     | -10             | —    | 10                   |      |
|                   |                            | 5V              | Without calibration<br>OVPCOF[4:0]=10000B     | -10             | —    | 10                   | mV   |
|                   |                            | 5V              | Without calibration<br>UVPCOF[4:0]=10000B     | -10             | —    | 10                   |      |
| V <sub>HYS</sub>  | Hysteresis                 | 3V              | —                                             | 10              | —    | 45                   | mV   |
|                   |                            | 5V              | —                                             | 10              | —    | 45                   |      |
| V <sub>CM</sub>   | Common Mode Voltage Range  | 3V              | —                                             | V <sub>SS</sub> | —    | V <sub>DD</sub> -1.4 | V    |
|                   |                            | 5V              | —                                             | V <sub>SS</sub> | —    | V <sub>DD</sub> -1.4 |      |
| R <sub>O</sub>    | R2R Output Resistance      | 3V              | —                                             | —               | —    | —                    | kΩ   |
|                   |                            | 5V              | —                                             | —               | —    | —                    |      |
| DNL               | Differential Non-linearity | 3V              | DAC V <sub>REF</sub> =V <sub>DD</sub> @ -45°C | -16             | —    | +16                  | LSB  |
|                   |                            | 3V              | DAC V <sub>REF</sub> =V <sub>DD</sub> @ 25°C  | -3              | —    | +3                   |      |
|                   |                            | 3V              | DAC V <sub>REF</sub> =V <sub>DD</sub> @ 85°C  | -3              | —    | +3                   |      |
|                   |                            | 5V              | DAC V <sub>REF</sub> =V <sub>DD</sub> @ -45°C | -16             | —    | +16                  |      |
|                   |                            | 5V              | DAC V <sub>REF</sub> =V <sub>DD</sub> @ 25°C  | -3              | —    | +3                   |      |
|                   |                            | 5V              | DAC V <sub>REF</sub> =V <sub>DD</sub> @ 85°C  | -3              | —    | +3                   |      |

| Symbol | Parameter              | Test Conditions |                                               | Min. | Typ. | Max. | Unit |
|--------|------------------------|-----------------|-----------------------------------------------|------|------|------|------|
|        |                        | V <sub>DD</sub> | Conditions                                    |      |      |      |      |
| INL    | Integral Non-linearity | 3V              | DAC V <sub>REF</sub> =V <sub>DD</sub> @ -45°C | -16  | —    | +16  | LSB  |
|        |                        | 3V              | DAC V <sub>REF</sub> =V <sub>DD</sub> @ 25°C  | -4   | —    | +4   |      |
|        |                        | 3V              | DAC V <sub>REF</sub> =V <sub>DD</sub> @ 85°C  | -4   | —    | +4   |      |
|        |                        | 5V              | DAC V <sub>REF</sub> =V <sub>DD</sub> @ -45°C | -16  | —    | +16  |      |
|        |                        | 5V              | DAC V <sub>REF</sub> =V <sub>DD</sub> @ 25°C  | -4   | —    | +4   |      |
|        |                        | 5V              | DAC V <sub>REF</sub> =V <sub>DD</sub> @ 85°C  | -4   | —    | +4   |      |

Note: Load Condition: C<sub>LOAD</sub>=50pF



## Over Current Protection Electrical Characteristics

T<sub>a</sub>=-40°C~85°C

| Symbol              | Parameter                            | Test Conditions |                                                | Min.                    | Typ. | Max.                    | Unit |
|---------------------|--------------------------------------|-----------------|------------------------------------------------|-------------------------|------|-------------------------|------|
|                     |                                      | V <sub>DD</sub> | Conditions                                     |                         |      |                         |      |
| I <sub>OCP</sub>    | OCP Operating Current                | 3V              | OCPnEN[1:0]=01B<br>OCPnVRS[1:0]=10B            | —                       | 300  | 500                     | μA   |
|                     |                                      | 5V              | OCPnCHY=1<br>Gn[2:0]=000B                      | —                       | 450  | 600                     |      |
| V <sub>REF</sub>    | DAC Reference voltage                | 3V              | OCPnVRS[1:0]=01                                | 1.8                     | —    | V <sub>DD</sub>         | V    |
|                     |                                      | 5V              | OCPnVRS[1:0]=01                                | 1.8                     | —    | V <sub>DD</sub>         | V    |
| V <sub>OS_CMP</sub> | Comparator Input Offset Voltage      | 3V              | Without calibration<br>(OCPnCOF[4:0]=10000B)   | -15                     | —    | 15                      | mV   |
|                     |                                      | 5V              | Without calibration<br>(OCPnCOF[4:0]=10000B)   | -15                     | —    | 15                      |      |
|                     |                                      | 3V              | With calibration                               | -2                      | —    | 2                       |      |
|                     |                                      | 5V              | With calibration                               | -2                      | —    | 2                       |      |
| V <sub>HYS</sub>    | Hysteresis                           | 3V              | —                                              | 10                      | 40   | 60                      | mV   |
|                     |                                      | 5V              | —                                              | 10                      | 40   | 60                      |      |
| V <sub>CM_CMP</sub> | Comparator Common Mode Voltage Range | 3V              | —                                              | V <sub>SS</sub>         | —    | V <sub>DD</sub><br>-1.0 | V    |
|                     |                                      | 5V              | —                                              | V <sub>SS</sub>         | —    | V <sub>DD</sub><br>-1.0 |      |
| V <sub>OS_OPA</sub> | OPA Input Offset Voltage             | 3V              | Without calibration<br>(OCPnOOF [5:0]=100000B) | -15                     | —    | 15                      | mV   |
|                     |                                      | 5V              | Without calibration<br>(OCPnOOF [5:0]=100000B) | -15                     | —    | 15                      |      |
|                     |                                      | 3V              | With calibration                               | -2                      | —    | 2                       |      |
|                     |                                      | 5V              | With calibration                               | -2                      | —    | 2                       |      |
| V <sub>CM_OPA</sub> | OPA Common Mode Voltage Range        | 3V              | —                                              | V <sub>SS</sub>         | —    | V <sub>DD</sub><br>-1.4 | V    |
|                     |                                      | 5V              | —                                              | V <sub>SS</sub>         | —    | V <sub>DD</sub><br>-1.4 |      |
| V <sub>OR</sub>     | OPA Maximum Output Voltage Range     | 3V              | —                                              | V <sub>SS</sub><br>+0.1 | —    | V <sub>DD</sub><br>-0.1 | V    |
|                     |                                      | 5V              | —                                              | V <sub>SS</sub><br>+0.1 | —    | V <sub>DD</sub><br>-0.1 |      |

| Symbol | Parameter                  | Test Conditions |                                       | Min. | Typ. | Max. | Unit |
|--------|----------------------------|-----------------|---------------------------------------|------|------|------|------|
|        |                            | V <sub>DD</sub> | Conditions                            |      |      |      |      |
| Ga     | PGA Gain Accuracy          | 3V              | All gain                              | -5   | —    | 5    | %    |
|        |                            | 5V              |                                       | -5   | —    | 5    |      |
| Ro     | R2R Output Resistance      | 3V              | —                                     | —    | 10   | —    | kΩ   |
|        |                            | 5V              | —                                     | —    | 10   | —    |      |
| DNL    | Differential Non-linearity | 3V              | DAC V <sub>REF</sub> =V <sub>DD</sub> | -1.5 | —    | +1.5 | LSB  |
|        |                            | 5V              |                                       | -1   | —    | +1   |      |
| INL    | Integral Non-linearity     | 3V              | DAC V <sub>REF</sub> =V <sub>DD</sub> | -2   | —    | +2   | LSB  |
|        |                            | 5V              |                                       | -1.5 | —    | +1.5 |      |

## USB Auto Detection Electrical Characteristics

Ta=-40°C~85°C

| Symbol             | Parameter                                               | Test Conditions |                                                            | Min.                  | Typ. | Max.                 | Unit |
|--------------------|---------------------------------------------------------|-----------------|------------------------------------------------------------|-----------------------|------|----------------------|------|
|                    |                                                         | V <sub>DD</sub> | Conditions                                                 |                       |      |                      |      |
| V <sub>DAC</sub>   | DAC Operating Voltage                                   | —               | —                                                          | 2.6                   | —    | 5.5                  | V    |
| I <sub>DAC</sub>   | DAC Operating Current                                   | 3V              | No Load                                                    | —                     | 0.6  | 0.9                  | mA   |
|                    |                                                         | 5V              |                                                            | —                     | 1.0  | 1.5                  |      |
| I <sub>DACS</sub>  | DAC Shutdown Current                                    | —               | No Load                                                    | —                     | —    | 0.1                  | μA   |
| N <sub>R</sub>     | DAC Resolution                                          | —               | —                                                          | —                     | 8    | —                    | bits |
| DNL                | DAC Differential Non-linearity                          | —               | No Load, DAC V <sub>REF</sub> =V <sub>DD</sub>             | -1                    | —    | +1                   | LSB  |
| INL                | DAC Integral Non-linearity                              | —               | No Load, DAC V <sub>REF</sub> =V <sub>DD</sub>             | -2                    | —    | +2                   | LSB  |
| V <sub>DACO</sub>  | Output Voltage Range                                    | —               | Code=0000H                                                 | V <sub>SS</sub>       | —    | V <sub>SS</sub> +0.2 | V    |
|                    |                                                         | —               | Code=0FFFH                                                 | V <sub>REF</sub> -0.2 | —    | V <sub>REF</sub>     |      |
| V <sub>REF</sub>   | Reference Voltage                                       | —               | —                                                          | 2                     | —    | V <sub>DD</sub>      | V    |
| t <sub>ST</sub>    | Settling Time                                           | 3V              | C <sub>LOAD</sub> =50pF                                    | —                     | —    | 5                    | μs   |
|                    |                                                         | 5V              |                                                            | —                     | —    | 5                    |      |
| R <sub>O</sub>     | R2R Output Resistance                                   | 3V              | —                                                          | —                     | 3    | —                    | kΩ   |
|                    |                                                         | 5V              |                                                            | —                     | 5    | —                    |      |
| OSRR               | Offset Error                                            | 3V              | V <sub>REF</sub> =V <sub>DD</sub> =3V, Data word=128       | —                     | —    | 50                   | mV   |
|                    |                                                         | 5V              | V <sub>REF</sub> =V <sub>DD</sub> =5V, Data word=128       | —                     | —    | 80                   |      |
| GERR               | Gain Error                                              | 3V              | V <sub>REF</sub> =V <sub>DD</sub> =3V, Data word=128       | —                     | —    | 50                   | mV   |
|                    |                                                         | 5V              | V <sub>REF</sub> =V <sub>DD</sub> =5V, Data word=128       | —                     | —    | 80                   |      |
| I <sub>DACOL</sub> | Output Sink Current                                     | 3V              | Data word=0000H,<br>V <sub>DACO</sub> =0.1V <sub>REF</sub> | 20                    | —    | —                    | μA   |
|                    |                                                         | 5V              |                                                            | 40                    | —    | —                    |      |
| I <sub>DACOH</sub> | Output Source Current                                   | 3V              | Data word=0FFFH,<br>V <sub>DACO</sub> =0.9V <sub>REF</sub> | 20                    | —    | —                    | μA   |
|                    |                                                         | 5V              |                                                            | 40                    | —    | —                    |      |
| I <sub>SC</sub>    | Output Short-circuit Current                            | 3V              | Data word=0FFFH                                            | 0.25                  | —    | —                    | mA   |
|                    |                                                         | 5V              |                                                            | 0.4                   | —    | —                    |      |
| R <sub>ON</sub>    | Analog Switch on Resistance Between D1+/D2+ and D1-/D2- | 5V              | —                                                          | —                     | 20   | 35                   | Ω    |
| R <sub>PL</sub>    | Pull-low Resistance for D1+, D2+                        | 5V              | —                                                          | 400                   | 700  | 1400                 | kΩ   |
|                    | Pull-low Resistance for D1-, D2-                        | 5V              | —                                                          | 15                    | 20   | 30                   | kΩ   |

| Symbol               | Parameter                                       | Test Conditions |                                                                                                     | Min. | Typ. | Max. | Unit |
|----------------------|-------------------------------------------------|-----------------|-----------------------------------------------------------------------------------------------------|------|------|------|------|
|                      |                                                 | V <sub>DD</sub> | Conditions                                                                                          |      |      |      |      |
| ERR                  | The Error for D1+, D1-, D2+, D2- output voltage | 5V              | DAC reference=V <sub>DD</sub> , ADUDAn[7:0]=10010100B, D1+, D1-, D2+ or D2- connect 150kΩ to ground | 2.57 | 2.70 | 2.84 | V    |
|                      |                                                 | 5V              | DAC reference=V <sub>DD</sub> , ADUDAn[7:0]=01101110B, D1+, D1-, D2+, D2- connect 150kΩ to ground   | 1.9  | 2.0  | 2.1  | V    |
| t <sub>VDP_SRC</sub> | V <sub>DP_SRC</sub> Turn On Stable Time         | —               | V <sub>BG</sub> Off                                                                                 | —    | —    | 200  | μs   |
|                      |                                                 | —               | V <sub>BG</sub> On                                                                                  | —    | 5    | 10   | μs   |

## LDO Regulator Electrical Characteristics

V<sub>5</sub>=5V, T<sub>a</sub>=25°C, unless otherwise specified

| Symbol             | Parameter                      | Test Conditions |                                                                                            | Min. | Typ. | Max. | Unit  |
|--------------------|--------------------------------|-----------------|--------------------------------------------------------------------------------------------|------|------|------|-------|
|                    |                                | V <sub>IN</sub> | Conditions                                                                                 |      |      |      |       |
| V <sub>IN</sub>    | Input voltage                  | —               | —                                                                                          | 6    | —    | 28   | V     |
| V <sub>OUT</sub>   | Output Voltage                 | —               | T <sub>a</sub> =25°C, I <sub>LOAD</sub> =1mA, V <sub>IN</sub> =V <sub>OUT</sub> +1V        | -2%  | 5    | 2%   | V     |
|                    |                                | —               | T <sub>a</sub> =-40°C~85°C, I <sub>LOAD</sub> =1mA, V <sub>IN</sub> =V <sub>OUT</sub> +1V  | -5%  | 5    | 5%   | V     |
| ΔV <sub>LOAD</sub> | Load Regulation <sup>(1)</sup> | —               | 1mA≤I <sub>LOAD</sub> ≤30mA, V <sub>IN</sub> =V <sub>OUT</sub> +1V                         | —    | 0.09 | 0.18 | %mA   |
| V <sub>DROP</sub>  | Dropout Voltage <sup>(2)</sup> | —               | ΔV <sub>OUT</sub> =2%, I <sub>LOAD</sub> =1mA, V <sub>IN</sub> =V <sub>OUT</sub> +2V       | —    | —    | 100  | mV    |
| I <sub>Q</sub>     | Quiescent Current              | 12V             | No load                                                                                    | —    | 2    | 4    | μA    |
| ΔV <sub>LINE</sub> | Line Regulation                | —               | 6V≤V <sub>IN</sub> ≤8V, I <sub>LOAD</sub> =1mA                                             | —    | —    | 0.2  | %/V   |
| TC                 | Temperature Coefficient        | —               | T <sub>a</sub> =-40°C~85°C, I <sub>LOAD</sub> =10mA, V <sub>IN</sub> =V <sub>OUT</sub> +1V | —    | ±0.9 | ±2   | mV/°C |

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

2. Dropout voltage is defined as the input voltage minus the output voltage that produces a 2% change in the output voltage from the value at V<sub>IN</sub>=V<sub>OUT</sub>+2V.

## Level Converter Electrical Characteristics

V<sub>CC</sub>=12V, T<sub>a</sub>=25°C

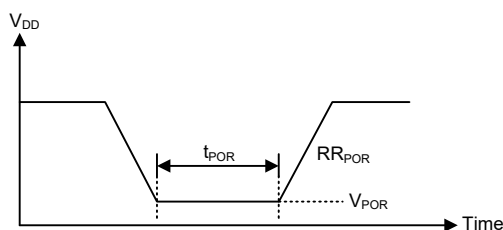
| Symbol              | Parameter                               | Test Conditions |                        | Min. | Typ. | Max. | Unit |
|---------------------|-----------------------------------------|-----------------|------------------------|------|------|------|------|
|                     |                                         | V <sub>IN</sub> | Conditions             |      |      |      |      |
| I <sub>source</sub> | Output Source Current of AX, BX, CX, DX | —               | V <sub>OH</sub> =10.4V | -60  | -90  | —    | mA   |
| I <sub>sink</sub>   | Output Sink Current of AX, BX, CX, DX   | —               | V <sub>OL</sub> =1.6V  | 60   | 90   | —    | mA   |



## Power-on Reset Characteristics

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

| 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   |



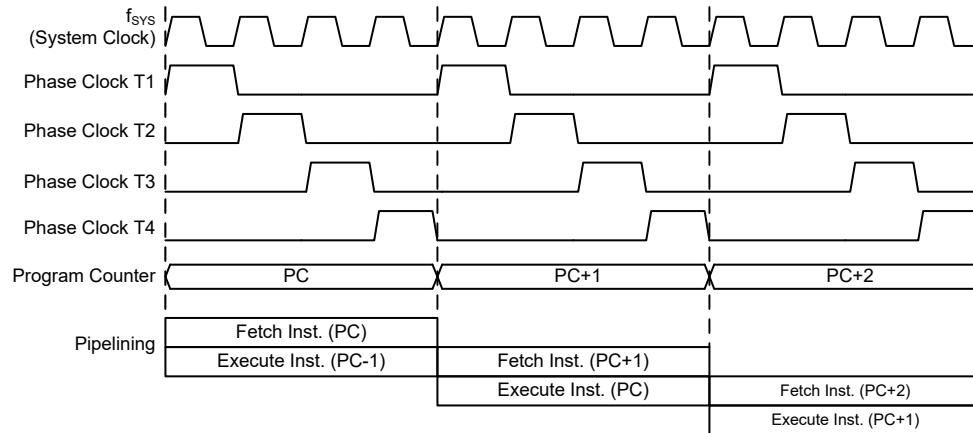
## System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The devices 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 devices suitable for low-cost, high-volume production for controller applications.

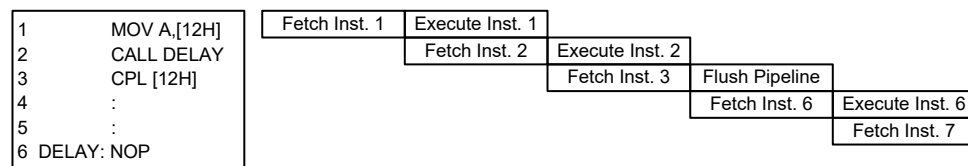
### Clocking and Pipelining

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

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 |
| PC11~PC8                  | PCL7~PCL0    |

**Program Counter**

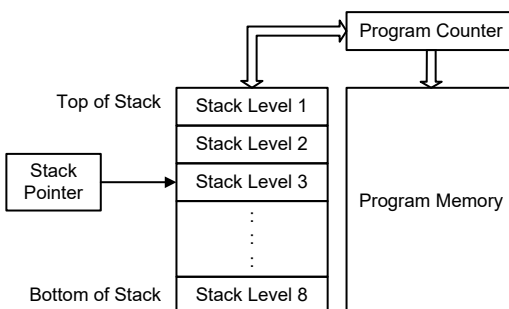
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available 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, organized into 8 levels, is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, and is neither readable nor writeable. At a subroutine call or interrupt acknowledge signal, the contents of the Program Counter are pushed onto the stack. At the end of a subroutine or an interrupt routine, signaled by a return instruction, RET or RETI, the Program Counter is restored to its previous value from the stack. After a device reset, the Stack Pointer will point to the top of the stack.

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

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



## Arithmetic and Logic Unit – ALU

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

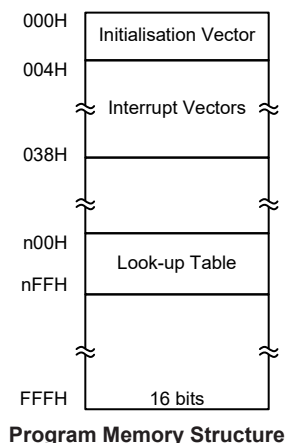
- Arithmetic operations:  
ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,  
LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:  
AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,  
LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation:  
RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,  
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 devices 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 devices offer users the flexibility to conveniently debug and develop their applications while also offering a means of field programming and updating.

### Structure

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



**Program Memory Structure**

### Special Vectors

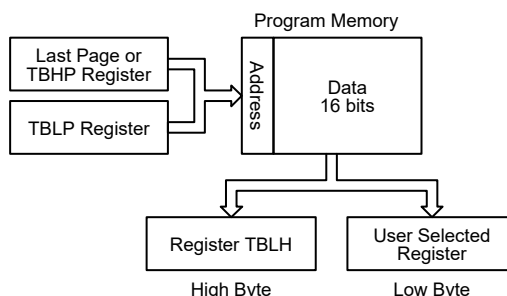
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

### Look-up Table

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the corresponding table read instruction such as “TABRD [m]” when the memory [m] is located in sector 0. If the memory [m] is located in other sectors, the data can be retrieved from the program memory using the corresponding extended table read instruction such as “LTABRD [m]”. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register. 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 microcontrollers. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is “0F00H” which refers to the start address of the last page within the 4K Program Memory of the devices. The table pointer low byte register is set here to have an initial value of “06H”. This will ensure that the first data read from the data table will be at the Program Memory address “0F06H” or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the specific address pointed by the TBLP and TBHP registers if the “TABRD [m]” or “LTABRD [m]” instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the “TABRD [m]” or “LTABRD [m]” instruction is executed.

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

### Table Read Program Example

```

tempreg1 db ?      ; temporary register #1
tempreg2 db ?      ; temporary register #2
:
:
mov a,06h          ; initialise low table pointer - note that this address is referenced
mov tblp,a         ; to the last page or the page that tbhp pointed
mov a,0Fh          ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1      ; transfers value in table referenced by table pointer,
                   ; data at program memory address "0F06H" transferred to tempreg1 and
                   ; TBLH
dec tblp            ; reduce value of table pointer by one
tabrd tempreg2      ; transfers value in table referenced by table pointer,
                   ; data at program memory address "0F05H" transferred to tempreg2 and
                   ; TBLH
                   ; in this example the data "1AH" is transferred to tempreg1 and data "0FH"
                   ; to register tempreg2
                   ; the value "00H" will be transferred to the high byte register TBLH
:

```

```

:
org 0F00h          ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh

```

## In Circuit Programming – ICP

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

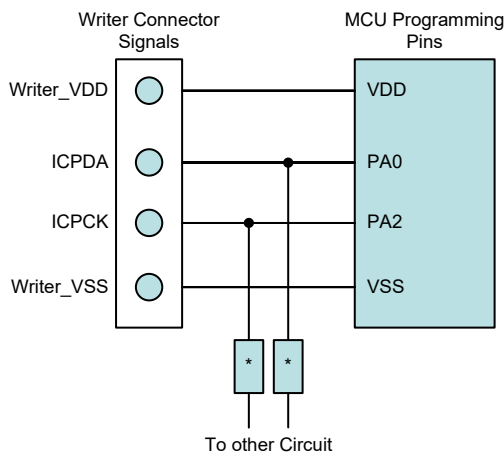
As an additional convenience, Holtek has provided a means of programming the microcontrollers in-circuit using a 4-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or 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 devices.

The Flash MCU to Writer programming pins correspondence table is as follows:

| 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. The technical details regarding the in-circuit programming of the devices 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 BP45V4NB/BP45VH4NB which is used to emulate the BP45F4NB/BP45FH4NB devices. 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 OCDSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCDSDA and OCDSCK pins in the device will have no effect in the EV chip. However, the two OCDS pins which are pin-shared with the ICP programming pins are still used as the Flash Memory programming pins for ICP. For more detailed OCDS information, refer to the corresponding document named “Holtek e-Link for 8-bit MCU OCDS User’s Guide”.

| Holtek e-Link Pins | EV Chip Pins | Pin Description                                 |
|--------------------|--------------|-------------------------------------------------|
| OCDSDA             | OCDSDA       | On-chip debug support data/address input/output |
| OCDSCK             | OCDSCK       | On-chip debug support clock input               |
| VDD                | VDD          | Power supply                                    |
| VSS                | VSS          | Ground                                          |

## In Application Programming – IAP

Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. The provision of the IAP function offers users the convenience of Flash Memory multi-programming features. The convenience of the IAP function is that it can execute the updated program procedure using its internal firmware, without requiring an external Program Writer or PC. In addition, the IAP interface can also be any type of communication protocol, such as UART or USB, using I/O pins. Regarding the internal firmware, the user can select versions provided by Holtek or create their own. The following section illustrates the procedures regarding how to implement the IAP firmware.

### Flash Memory Read/Write Size

The Flash memory Erase and Write operations are carried out in a page format while the Read operation is carried out in a word format. The page size and write buffer size are both assigned with a capacity of 32 words. Note that the Erase operation should be executed before the Write operation is executed.

When the Flash Memory Erase/Write Function is successfully enabled, the CFWEN bit will be set high. When the CFWEN bit is set high, the data can be written into the write buffer. The FWT bit is used to initiate the write process and then indicate the write operation status. This bit is set high by application programs to initiate a write process and will be cleared by hardware if the write process is finished.

The Read operation can be carried out by executing a specific read procedure. The FRDEN bit is used to enable the read function and the FRD bit is used to initiate the read process by application programs and then indicate the read operation status. When the read process is finished, this bit will be cleared by hardware.

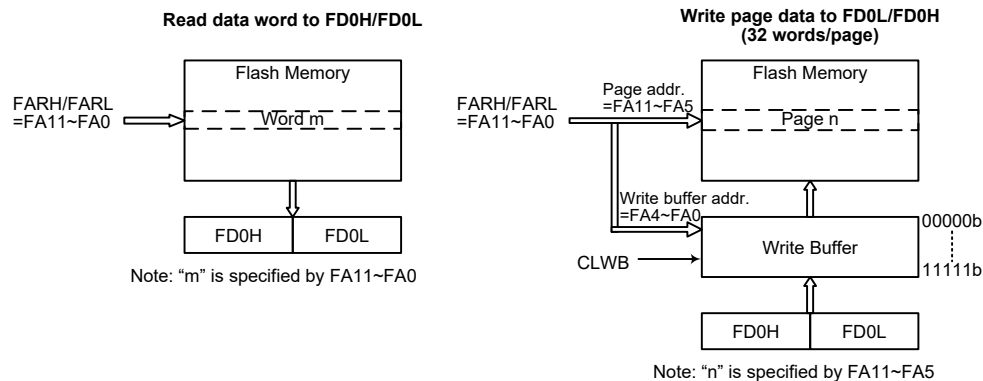
| Operations                                  | Format        |
|---------------------------------------------|---------------|
| Erase                                       | 32 words/page |
| Write                                       | 32 words/time |
| Read                                        | 1 word/time   |
| Note: Page size=Write buffer size=32 words. |               |

**IAP Operation Format**

| Erase Page | FARH      | FARL[7:5] | FARL[4:0] |
|------------|-----------|-----------|-----------|
| 0          | 0000 0000 | 000       | x xxxx    |
| 1          | 0000 0000 | 001       | x xxxx    |
| 2          | 0000 0000 | 010       | x xxxx    |
| 3          | 0000 0000 | 011       | x xxxx    |
| 4          | 0000 0000 | 100       | x xxxx    |
| :          | :         | :         | :         |
| 126        | 0000 1111 | 110       | x xxxx    |
| 127        | 0000 1111 | 111       | x xxxx    |

"x": Don't care

**Erase Page Number and Selection**



**Flash Memory IAP Read/Write Structure**

### Write Buffer

The write buffer is used to store the written data temporarily when executing the write operation. The Write Buffer can be filled with written data after the Flash Memory Erase/Write Function has been successfully enabled by executing the Flash Memory Erase/Write Function Enable procedure. The write buffer can be cleared by configuring the CLWB bit in the FC2 register. The CLWB bit can be set high to enable the Clear Write Buffer procedure. When the procedure is finished this bit will be cleared to low by the hardware. It is recommended that the write buffer should be cleared by setting the CLWB bit high before the write buffer is used for the first time or when the data in the write buffer is updated.

The write buffer size is 32 words corresponding to a page. The write buffer address is mapped to a specific flash memory page specified by the memory address bits, FA11~FA5. The data written into the FD0L and FD0H registers will be loaded into the write buffer. When data is written into the high byte data register, FD0H, it will result in the data stored in the high and low byte data registers both being written into the write buffer. It will also cause the flash memory address to be incremented by one, after which the new address will be loaded into the FARH and FARL address registers. When the flash memory address reaches the page boundary, 11111b of a page with 32 words, the address will now not be incremented but will stop at the last address of the page. At this point a new page address should be specified for any other erase/write operations.

After a write process is finished, the write buffer will automatically be cleared by the hardware. Note that the write buffer should be cleared manually by the application program when the data written into the flash memory is incorrect in the data verification step. The data should again be written into the write buffer after the write buffer has been cleared when the data is found to be incorrect during the data verification step.



## IAP Flash Program Memory Registers

There are two address registers, four 16-bit data registers and three control registers. All the registers are located in Sector 0. Read and Write operations to the Flash memory are carried out by 16-bit data operations using the address and data registers and the control register. Several registers control the overall operation of the internal Flash Program Memory. The address registers are named FARL and FARH, the data registers are named FDnL and FDnH and the control registers are named FC0, FC1 and FC2.

| Register Name | Bit   |       |       |       |       |      |       |      |
|---------------|-------|-------|-------|-------|-------|------|-------|------|
|               | 7     | 6     | 5     | 4     | 3     | 2    | 1     | 0    |
| FC0           | CFWEN | FMOD2 | FMOD1 | FMOD0 | FWPEN | FWT  | FRDEN | FRD  |
| FC1           | D7    | D6    | D5    | D4    | D3    | D2   | D1    | D0   |
| FC2           | —     | —     | —     | —     | —     | —    | —     | CLWB |
| FARL          | FA7   | FA6   | FA5   | FA4   | FA3   | FA2  | FA1   | FA0  |
| FARH          | —     | —     | —     | —     | FA11  | FA10 | FA9   | FA8  |
| FD0L          | D7    | D6    | D5    | D4    | D3    | D2   | D1    | D0   |
| FD0H          | D15   | D14   | D13   | D12   | D11   | D10  | D9    | D8   |
| FD1L          | D7    | D6    | D5    | D4    | D3    | D2   | D1    | D0   |
| FD1H          | D15   | D14   | D13   | D12   | D11   | D10  | D9    | D8   |
| FD2L          | D7    | D6    | D5    | D4    | D3    | D2   | D1    | D0   |
| FD2H          | D15   | D14   | D13   | D12   | D11   | D10  | D9    | D8   |
| FD3L          | D7    | D6    | D5    | D4    | D3    | D2   | D1    | D0   |
| FD3H          | D15   | D14   | D13   | D12   | D11   | D10  | D9    | D8   |

IAP Register List

### • FARL Register

| Bit  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
|------|-----|-----|-----|-----|-----|-----|-----|-----|
| Name | FA7 | FA6 | FA5 | FA4 | FA3 | FA2 | FA1 | FA0 |
| 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 **FA7~FA0**: Flash Memory Address bit 7~bit 0

### • FARH Register

| Bit  | 7 | 6 | 5 | 4 | 3    | 2    | 1   | 0   |
|------|---|---|---|---|------|------|-----|-----|
| Name | — | — | — | — | FA11 | FA10 | FA9 | FA8 |
| R/W  | — | — | — | — | R/W  | R/W  | R/W | R/W |
| POR  | — | — | — | — | 0    | 0    | 0   | 0   |

Bit 7~4 Unimplemented, read as “0”

Bit 3~0 **FA11~FA8**: Flash Memory Address bit 11~bit 8

### • FD0L Register

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

Bit 7~0 **D7~D0**: The first Flash Memory data bit 7~bit 0

Note that data written into the low byte data register FD0L will only be stored in the FD0L register and not loaded into the lower 8-bit write buffer.

**• FD0H Register**

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

Bit 7~0 **D15~D8**: The first Flash Memory data bit 15~bit 8

Note that when 8-bit data is written into the high byte data register FD0H, the whole 16 bits of data stored in the FD0H and FD0L registers will simultaneously be loaded into the 16-bit write buffer after which the contents of the Flash memory address register pair, FARH and FARL, will be incremented by one.

**• FD1L Register**

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

Bit 7~0 **D7~D0**: The second Flash Memory data bit 7~bit 0

**• FD1H Register**

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

Bit 7~0 **D15~D8**: The second Flash Memory data bit 15~bit 8

**• FD2L Register**

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

Bit 7~0 **D7~D0**: The third Flash Memory data bit 7~bit 0

**• FD2H Register**

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

Bit 7~0 **D7~D0**: The third Flash Memory data bit 7~bit 0

**• FD3L Register**

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

Bit 7~0 **D7~D0**: The fourth Flash Memory data bit 7~bit 0

• **FD3H Register**

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

Bit 7~0      **D15~D8**: The fourth Flash Memory data bit 15~bit 8

• **FC0 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2   | 1     | 0   |
|------|-------|-------|-------|-------|-------|-----|-------|-----|
| Name | CFWEN | FMOD2 | FMOD1 | FMOD0 | FWPEN | FWT | FRDEN | FRD |
| 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      **CFWEN**: Flash Memory Erase/Write function enable control

0: Flash Memory erase/write function is disabled

1: Flash Memory erase/write function has been successfully enabled

When this bit is cleared to zero by application program, the Flash Memory erase/write function is disabled. Note that this bit cannot be set high by application programs. Writing “1” into this bit results in no action. This bit is used to indicate that the Flash Memory erase/write function status. When this bit is set high by hardware, it means that the Flash Memory erase/write function is enabled successfully. Otherwise, the Flash Memory erase/write function is disabled as the bit content is zero.

Bit 6~4      **FMOD2~FMOD0**: Flash Memory Mode selection

000: Write Mode

001: Page Erase Mode

010: Reserved

011: Read Mode

100: Reserved

101: Reserved

110: Flash Memory Erase/Write function Enable Mode

111: Reserved

These bits are used to select the Flash Memory operation modes. Note that the “Flash memory Erase/Write function Enable Mode” should first be successfully enabled before the Erase or Write Flash memory operation is executed.

Bit 3      **FWPEN**: Flash Memory Erase/Write function enable procedure trigger

0: Erase/Write function enable procedure is not triggered or procedure timer times out

1: Erase/Write function enable procedure is triggered and procedure timer starts to count

This bit is used to activate the flash memory Erase/Write function enable procedure and an internal timer. It is set by the application programs and then cleared to zero by the hardware when the internal timer times out. The correct patterns must be written into the FD1L/FD1H, FD2L/FD2H and FD3L/FD3H register pairs respectively as soon as possible after the FWPEN bit is set high.

Bit 2      **FWT**: Flash Memory write initiate control

0: Do not initiate Flash Memory write or indicating that a Flash Memory write process has completed

1: Initiate a Flash Memory write process

This bit is set by software and cleared to zero by the hardware when the Flash memory write process has completed. Note that all CPU operations will be stopped cease when this bit is set to 1.

Bit 1 **FRDEN**: Flash Memory read enabled bit  
 0: Flash Memory read disable  
 1: Flash Memory read enable  
 This is the Flash memory Read Enable bit which must be set high before any Flash memory read operations are carried out. Clearing this bit to zero will inhibit Flash memory read operations.

Bit 0 **FRD**: Flash Memory read control bit  
 0: Do not initiate Flash Memory read or indicating that a Flash Memory read process has completed  
 1: Initiate a Flash Memory read process  
 This bit is set by software and cleared to zero by the hardware when the Flash memory read process has completed. Note that all CPU operations will be stopped cease when this bit is set to 1.

- Note: 1. The FWT, FRDEN and FRD bits cannot be set to “1” at the same time with a single instruction.  
 2. Ensure that the  $f_{SUB}$  clock is stable before executing the erase or write operation.  
 3. Note that the CPU will be stopped when a read, erase or write operation is successfully activated.  
 4. Ensure that the read, erase or write operation is totally complete before executing other operations.

#### • FC1 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**: Chip Reset Pattern  
 When a specific value of “55H” is written into this register, a reset signal will be generated to reset the whole chip.

#### • FC2 Register

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

Bit 7~1 Unimplemented, read as “0”

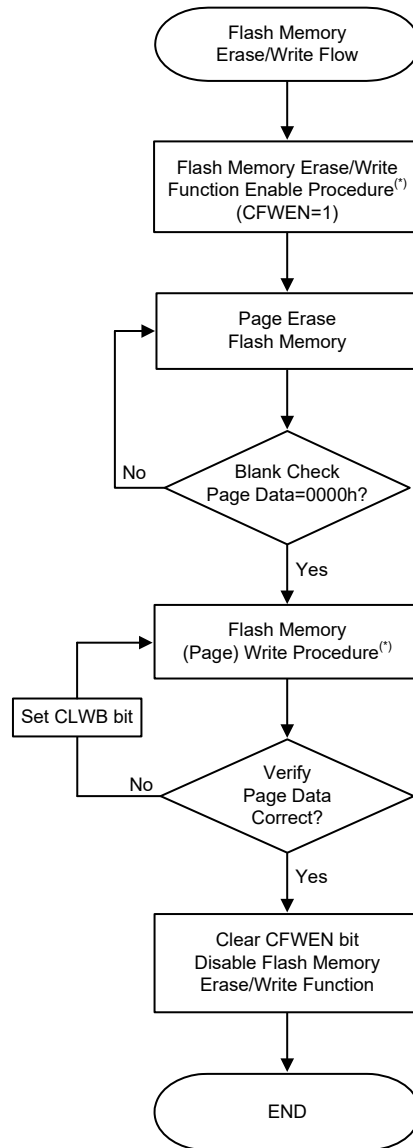
Bit 0 **CLWB**: Flash Memory Write buffer clear control  
 0: Do not initiate a Write Buffer Clear process or indicating that a Write Buffer Clear process has completed  
 1: Initiate a Write Buffer Clear process  
 This bit is set by software and cleared to zero by hardware when the Write Buffer Clear process has completed.

### **Flash Memory Erase/Write Flow**

It is important to understand the Flash memory Erase/Write flow before the Flash memory contents are updated. Users can refer to the corresponding operation procedures when developing their IAP program to ensure that the flash memory contents are correctly updated.

### **Flash Memory Erase/Write Flow Descriptions**

1. Activate the “Flash Memory Erase/Write function enable procedure” first. When the Flash Memory Erase/Write function is successfully enabled, the CFWEN bit in the FC0 register will automatically be set high by hardware. After this, Erase or Write operations can be executed on the Flash memory. Refer to the “Flash Memory Erase/Write Function Enable Procedure” for details.
2. Configure the flash memory address to select the desired erase page and then erase this page.
3. Execute a Blank Check operation to ensure whether the page erase operation is successful or not. The “TABRD” instruction should be executed to read the flash memory contents and to check if the contents is 0000h or not. If the flash memory page erase operation fails, users should go back to Step 2 and execute the page erase operation again.
4. Write data into the specific page. Refer to the “Flash Memory Write Procedure” for details.
5. Execute the “TABRD” instruction to read the flash memory contents and check if the written data is correct or not. If the data read from the flash memory is different from the written data, it means that the page write operation has failed. The CLWB bit should be set high to clear the write buffer and then write the data into the specific page again if the write operation has failed.
6. Clear the CFWEN bit to disable the Flash Memory Erase/Write function enable mode if the current page Erase and Write operations are completed and no more pages need to be erased or written.



**Flash Memory Erase/Write Flow**

Note: The Flash Memory Erase/Write Function Enable procedure and Flash Memory Write procedure will be described in the following sections.

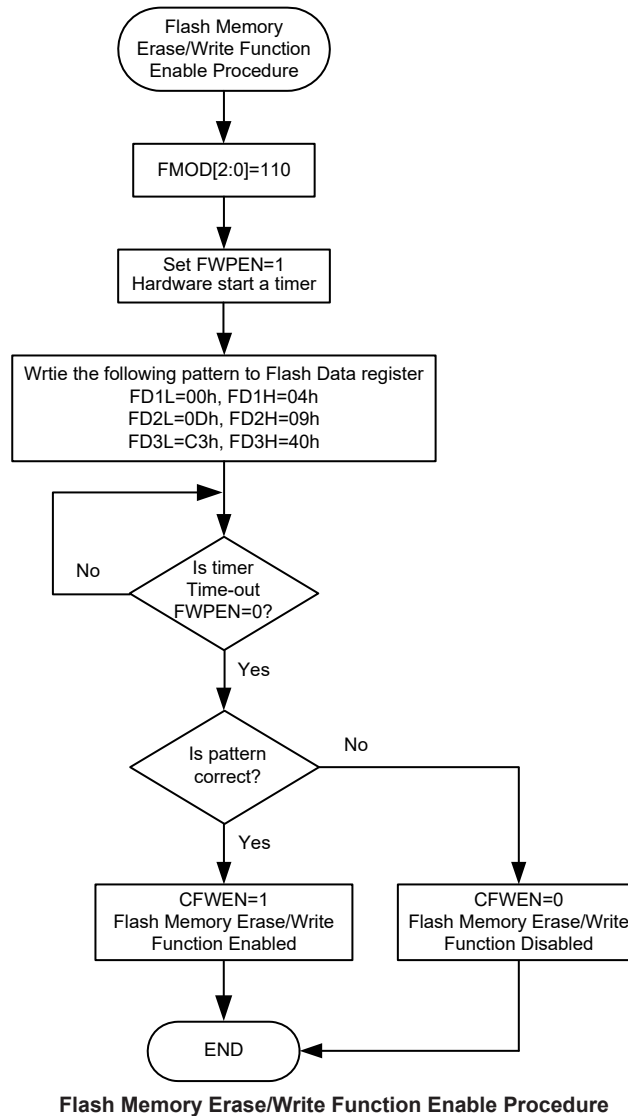
### **Flash Memory Erase/Write Function Enable Procedure**

The Flash Memory Erase/Write Function Enable Mode is specially designed to prevent the flash memory contents from being wrongly modified. In order to allow users to change the Flash memory data using the IAP control registers, users must first enable the Flash memory Erase/Write function.

### **Flash Memory Erase/Write Function Enable Procedure Description**

1. Write data “110” to the FMOD [2:0] bits in the FC0 register to select the Flash Memory Erase/Write Function Enable Mode.
2. Set the FWPEN bit in the FC0 register to “1” to activate the Flash Memory Erase/Write Enable Function. This will also activate an internal timer.
3. Write the correct data pattern into the Flash data registers, FD1L~FD3L and FD1H~FD3H, as soon as possible after the FWPEN bit is set high. The enable Flash memory erase/write function data pattern is 00H, 0DH, C3H, 04H, 09H and 40H corresponding to the FD1L~FD3L and FD1H~FD3H registers respectively.
4. Once the timer has timed out, the FWPEN bit will automatically be cleared to zero by hardware regardless of the input data pattern.
5. If the written data pattern is incorrect, the Flash memory erase/write function will not be enabled successfully and the above steps should be repeated. If the written data pattern is correct, the Flash memory erase/write function will be enabled successfully.
6. Once the Flash memory erase/write function is enabled, the Flash memory contents can be updated by executing the page erase and write operations using the IAP control registers.

To disable the Flash memory erase/write function, the CFWEN bit in the FC0 register can be cleared. There is no need to execute the above procedure.





### **Flash Memory Write Procedure**

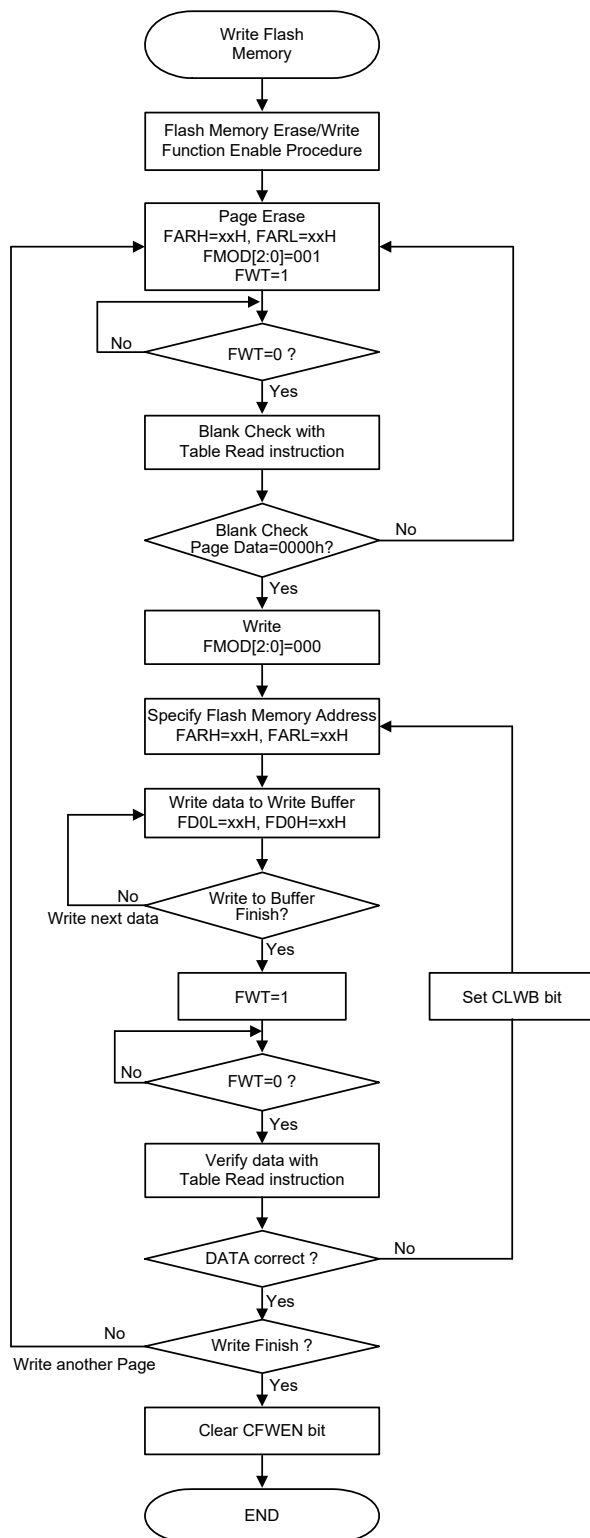
After the Flash memory erase/write function has been successfully enabled as the CFWEN bit is set high, the data to be written into the flash memory can be loaded into the write buffer. The selected flash memory page data should be erased by properly configuring the IAP control registers before the data write procedure is executed.

The write buffer size is 32 words, known as a page, whose address is mapped to a specific flash memory page specified by the memory address bits, FA11~FA5. It is important to ensure that the page where the write buffer data is located is the same one which the memory address bits, FA11~FA5, specify.

### **Flash Memory Consecutive Write Description**

The maximum amount of write data is 32 words for each write operation. The write buffer address will be automatically incremented by one when consecutive write operations are executed. The start address of a specific page should first be written into the FARL and FARH registers. Then the data word should first be written into the FD0L register and then the FD0H register. At the same time the write buffer address will be incremented by one and then the next data word can be written into the FD0L and FD0H registers for the next address without modifying the address register pair, FARH and FARL. When the write buffer address reaches the page boundary the address will not be further incremented but will stop at the last address of the page.

1. Activate the “Flash Memory Erase/Write function enable procedure”. Check the CFWEN bit value and then execute the erase/write operations if the CFWEN bit is set high. Refer to the “Flash Memory Erase/Write function enable procedure” for more details.
2. Set the FMOD field to “001” to select the erase operation. Set the FWT bit high to erase the desired page which is specified by the FARH and FARL registers. Wait until the FWT bit goes low.
3. Execute a Blank Check operation using the table read instruction to ensure that the erase operation has successfully completed.  
Go to step 2 if the erase operation is not successful.  
Go to step 4 if the erase operation is successful.
4. Set the FMOD field to “000” to select the write operation.
5. Setup the desired start address in the FARH and FARL registers. Write the desired data words consecutively into the FD0L and FD0H registers within a page as specified by their consecutive addresses. The maximum written data number is 32 words.
6. Set the FWT bit high to write the data words from the write buffer to the flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.  
If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 5.  
Go to step 8 if the write operation is successful.
8. Clear the CFWEN bit low to disable the Flash memory erase/write function.



**Flash Memory Consecutive Write Procedure**

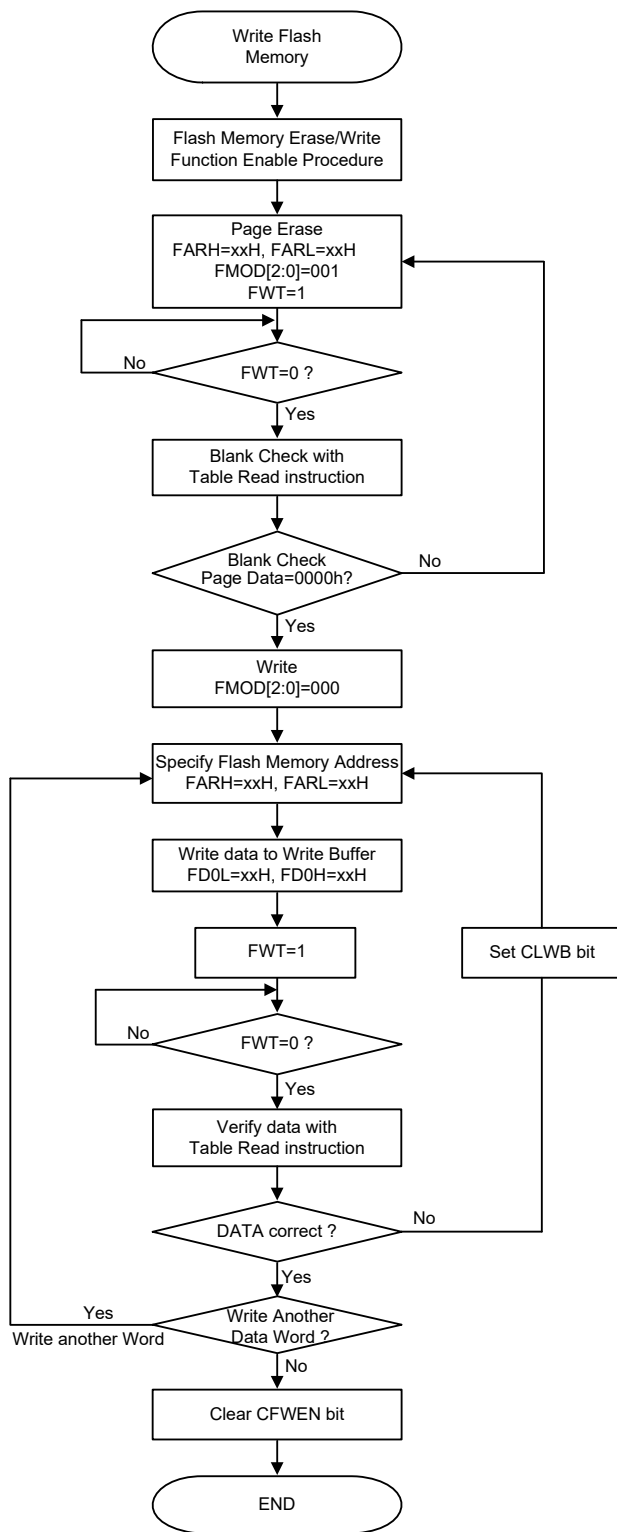
Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.  
2. It will take a typical time of 2.2ms for the FWT bit state changing from high to low.

### **Flash Memory Non-Consecutive Write Description**

The main difference between Flash Memory Consecutive and Non-Consecutive Write operations is whether the data words to be written are located in consecutive addresses or not. If the data to be written is not located in consecutive addresses the desired address should be re-assigned after a data word is successfully written into the Flash Memory.

A two data word non-consecutive write operation is taken as an example here and described as follows:

1. Activate the “Flash Memory Erase/Write function enable procedure”. Check the CFWEN bit value and then execute the erase/write operation if the CFWEN bit is set high. Refer to the “Flash Memory Erase/Write function enable procedure” for more details.
2. Set the FMOD field to “001” to select the erase operation. Set the FWT bit high to erase the desired page which is specified by the FARH and FARL registers. Wait until the FWT bit goes low.
3. Execute a Blank Check operation using the table read instruction to ensure that the erase operation has successfully completed.  
Go to step 2 if the erase operation is not successful.  
Go to step 4 if the erase operation is successful.
4. Set the FMOD field to “000” to select the write operation.
5. Setup the desired address ADDR1 in the FARH and FARL registers. Write the desired data word DATA1 first into the FD0L register and then into the FD0H register.
6. Set the FWT bit high to transfer the data word from the write buffer to the flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.  
If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 5.  
Go to step 8 if the write operation is successful.
8. Setup the desired address ADDR2 in the FARH and FARL registers. Write the desired data word DATA2 first into the FD0L register and then into the FD0H register.
9. Set the FWT bit high to transfer the data word from the write buffer to the flash memory. Wait until the FWT bit goes low.
10. Verify the data using the table read instruction to ensure that the write operation has successfully completed.  
If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 8.  
Go to step 11 if the write operation is successful.
11. Clear the CFWEN bit low to disable the Flash memory erase/write function.



**Flash Memory Non-Consecutive Write Procedure**

Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.

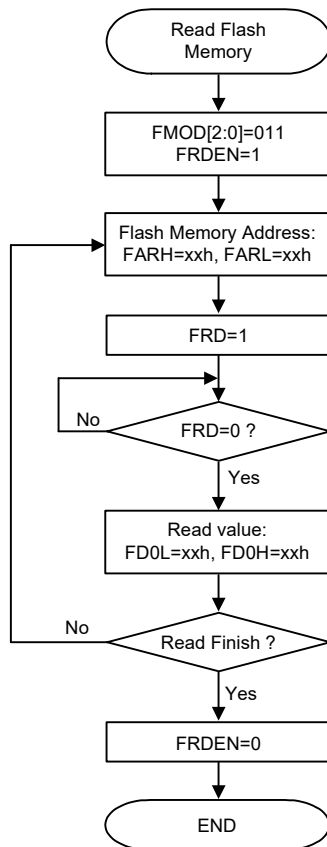
2. It will take a typical time of 2.2ms for the FWT bit state changing from high to low.

#### **Important Points to Note for Flash Memory Write Operations**

1. The “Flash Memory Erase/Write Function Enable Procedure” must be successfully activated before the Flash Memory erase/write operation is executed.
2. The Flash Memory erase operation is executed to erase a whole page.
3. The whole write buffer data will be written into the flash memory in a page format. The corresponding address cannot exceed the page boundary.
4. After the data is written into the flash memory the flash memory contents must be read out using the table read instruction, TABRD, and checked if it is correct or not. If the data written into the flash memory is incorrect, the write buffer should be cleared by setting the CLWB bit high and then write the data again into the write buffer. Then activate a write operation on the same flash memory page without erasing it. The data check, buffer clear and data re-write steps should be repeatedly executed until the data written into the flash memory is correct.
5. The system frequency should be setup to the maximum application frequency when data write and data check operations are executed using the IAP function.

#### **Flash Memory Read Procedure**

To activate the Flash Memory Read procedure, the FMOD field should be set to “011” to select the flash memory read mode and the FRDEN bit should be set high to enable the read function. The desired flash memory address should be written into the FARH and FARL registers and then the FRD bit should be set high. After this the flash memory read operation will be activated. The data stored in the specified address can be read from the data registers, FD0H and FD0L, when the FRD bit goes low. There is no need to first activate the Flash Memory Erase/Write Function Enable Procedure before the flash memory read operation is executed.



**Flash Memory Read Procedure**

- Note: 1. When the read operation is successfully activated, all CPU operations will temporarily cease.  
2. It will take a typical time of three instruction cycles for the FRD bit state changing from high to low.

## 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.

Categorised 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 devices. 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.

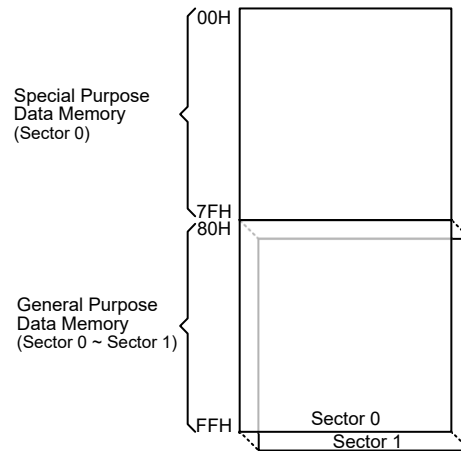
### Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide Memory.

Each of the Data Memory sectors is categorized into two types, the Special Purpose Data Memory and the General Purpose Data Memory. The address range of the Special Purpose Data Memory for the devices are from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH.

Switching between the different Data Memory sectors is achieved by setting the Memory Pointers to the correct value if using the indirect addressing method. The start address of the Data Memory for the devices are the address 00H.

| Special Purpose Data Memory | General Purpose Data Memory |                          |
|-----------------------------|-----------------------------|--------------------------|
| Available Sectors           | Capacity                    | Sector: Address          |
| 0, 1                        | 256×8                       | 0: 80H~FFH<br>1: 80H~FFH |



**Data Memory Structure**

## **Data Memory Addressing**

For the devices that support 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 devices, 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 |        |
|----------|--------|
| 00H      | IAR0   |
| 01H      | MP0    |
| 02H      | IAR1   |
| 03H      | MP1L   |
| 04H      | MP1H   |
| 05H      | ACC    |
| 06H      | PCL    |
| 07H      | TBLP   |
| 08H      | TBLH   |
| 09H      | TBHP   |
| 0AH      | STATUS |
| 0BH      |        |
| 0CH      | IAR2   |
| 0DH      | MP2L   |
| 0EH      | MP2H   |
| 0FH      | RSTFC  |
| 10H      | LVRC   |
| 11H      | LVDC   |
| 12H      | SCC    |
| 13H      | HIRCC  |
| 14H      | PA     |
| 15H      | PAC    |
| 16H      | PAPU   |
| 17H      | PAWU   |
| 18H      | PB     |
| 19H      | PBC    |
| 1AH      | PBPU   |
| 1BH      | PC     |
| 1CH      | PCC    |
| 1DH      | PCPU   |
| 1EH      | PD     |
| 1FH      | PDC    |
| 20H      | PDPU   |
| 21H      | PAS0   |
| 22H      | PAS1   |
| 23H      | PBS0   |
| 24H      | PBS1   |
| 25H      | PCS0   |
| 26H      | PCS1   |
| 27H      | PDS0   |
| 28H      | PDS1   |
| 29H      | WDTC   |
| 2AH      | CPR    |
| 2BH      | OCVPC  |
| 2CH      | CTM0C0 |
| 2DH      | CTM0C1 |
| 2EH      | CTM0DL |
| 2FH      | CTM0DH |
| 30H      | CTM0AL |
| 31H      | CTM0AH |
| 32H      | CTM1C0 |
| 33H      | CTM1C1 |
| 34H      | CTM1DL |
| 35H      | CTM1DH |
| 36H      | CTM1AL |
| 37H      | CTM1AH |
| 38H      | PTMC0  |
| 39H      | PTMC1  |
| 3AH      | PTMDL  |
| 3BH      | PTMDH  |
| 3CH      | PTMAL  |
| 3DH      | PTMAH  |
| 3EH      | PTMRPL |
| 3FH      | PTMRPH |

| Sector 0 |                  |
|----------|------------------|
| 40H      | SWS0             |
| 41H      | SADC0            |
| 42H      | SADC1            |
| 43H      | SADOH            |
| 44H      | SADOL            |
| 45H      | INTC0            |
| 46H      | INTC1            |
| 47H      | INTC2            |
| 48H      | INTC3            |
| 49H      | MFI              |
| 4AH      | INTEG            |
| 4BH      | PSC0R            |
| 4CH      | PSC1R            |
| 4DH      | PMP5             |
| 4EH      | SLEDC0           |
| 4FH      | SLEDC1           |
| 50H      | OCP0C0           |
| 51H      | OCP0C1           |
| 52H      | OCP0DA           |
| 53H      | OCP0OCAL         |
| 54H      | OCP0CCAL         |
| 55H      | OCP1C0           |
| 56H      | OCP1C1           |
| 57H      | OCP1DA           |
| 58H      | OCP1OCAL         |
| 59H      | OCP1CCAL         |
| 5AH      | OUVPC0           |
| 5BH      | OUVPC1           |
| 5CH      | OUVPC2           |
| 5DH      | OUVPC3           |
| 5EH      | OVPDAH           |
| 5FH      | OVPDAL           |
| 60H      | UVPDAH           |
| 61H      | UVPDAL           |
| 62H      | ADUDA0           |
| 63H      | ADUDA1           |
| 64H      | ADUC0            |
| 65H      | ADUC1            |
| 66H      | SIMC0            |
| 67H      | SIMC1/UUCR1      |
| 68H      | SIMD/UTXR_RXR    |
| 69H      | SIMC2/SIMA/UUCR2 |
| 6AH      | UUCR3            |
| 6BH      | SIMTOC/UBRG      |
| 6CH      | UUSR             |
| 6DH      | FC0              |
| 6EH      | FC1              |
| 6FH      | FC2              |
| 70H      | FARL             |
| 71H      | FARH             |
| 72H      | FD0L             |
| 73H      | FD0H             |
| 74H      | FD1L             |
| 75H      | FD1H             |
| 76H      | FD2L             |
| 77H      | FD2H             |
| 78H      | FD3L             |
| 79H      | FD3H             |
| 7AH      | IFS              |
| 7BH      |                  |
| 7CH      | TB0C             |
| 7DH      | TB1C             |
| 7EH      |                  |
| 7FH      |                  |

□ : Unused, read as 00H

▤ : Reserved, cannot be changed otherwise specified

### 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 corresponding instruction which can address all available data memory space.

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

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

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

## 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 operations are selected through the 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 Interrupt. The fully integrated internal oscillators, requiring no external components, are provided to form a wide range of both fast and slow system oscillators. The higher frequency oscillator provides higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillator. With the capability of dynamically switching between fast and slow system clock, the devices have

the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

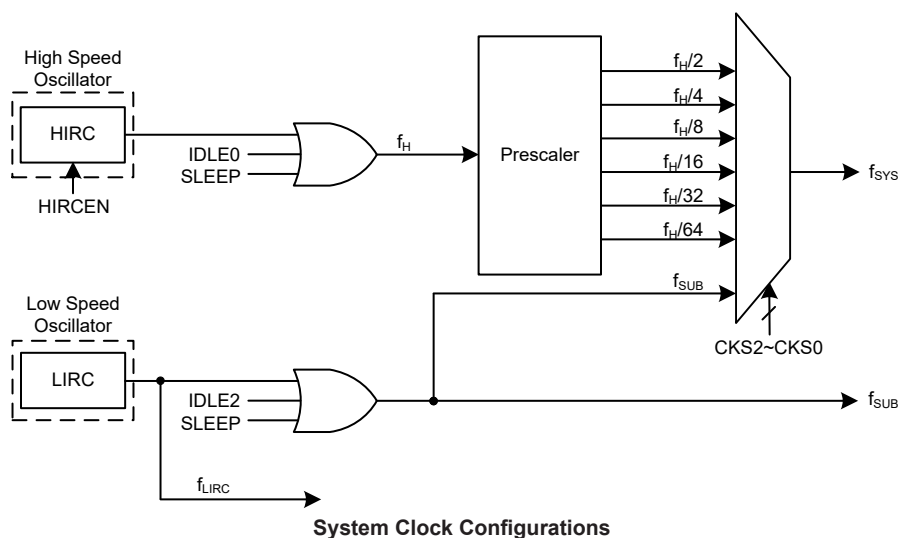
| Type                   | Name | Frequency |
|------------------------|------|-----------|
| Internal High Speed RC | HIRC | 30MHz     |
| Internal Low Speed RC  | LIRC | 32kHz     |

**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 30MHz 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 the system clock can be dynamically selected.

The frequency of the slow speed or high speed system clock is determined using the CKS2~CKS0 bits in the SCC register. Note that two oscillator selections must be made namely one high speed and one low speed system oscillators.



## Internal High Speed RC Oscillator – HIRC

The internal RC oscillator is a fully integrated system oscillator requiring no external components. The internal high speed RC oscillator has a fixed frequency of 30MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

## Internal 32kHz Oscillator – LIRC

The Internal 32kHz System Oscillator is a fully integrated low frequency RC oscillator with a typical frequency of 32kHz, 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.

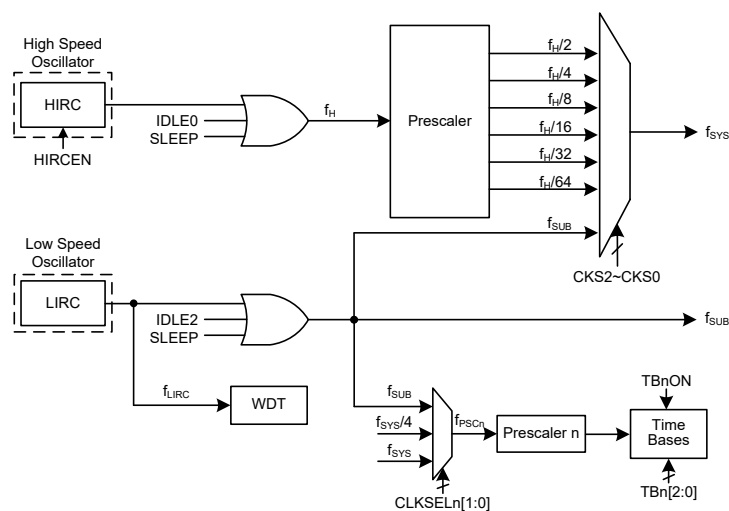
## Operating Modes and System Clocks

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 devices 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 devices have 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 either a divided version of the high speed system oscillator with a range of  $f_H/2 \sim f_H/64$  or a 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.



**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,  $f_H/2 \sim f_H/64$ , 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 microcontrollers, each one with its own special characteristics and which can be chosen according to the specific performance and power requirements of the application. There are two modes allowing normal operation of the microcontroller, the FAST Mode and SLOW Mode. The remaining four modes, the SLEEP, IDLE0, IDLE1 and IDLE2 Mode are used when the microcontroller CPU is switched off to conserve power.

| Operation Mode | CPU | Register Setting |        |           | $f_{SYS}$           | $f_H$                 | $f_{SUB}$ | $f_{LIRC}$        |
|----------------|-----|------------------|--------|-----------|---------------------|-----------------------|-----------|-------------------|
|                |     | FHIDEN           | FSIDEN | CKS2~CKS0 |                     |                       |           |                   |
| FAST           | On  | x                | x      | 000~110   | $f_H/2 \sim f_H/64$ | On                    | On        | On                |
| SLOW           | On  | x                | x      | 111       | $f_{SUB}$           | On/Off <sup>(1)</sup> | 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 <sup>(2)</sup> |

"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 is switched on as the WDT function is always enabled.

### FAST Mode

This is one of the main operating modes where the microcontrollers have all of their functions operational and where the system clock is provided by the high speed oscillator. This mode operates allowing the microcontrollers to operate normally with a clock source which will come from the high speed oscillator, HIRC. The high speed oscillator will however first be divided by a ratio ranging from 2 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 microcontrollers at a divided clock ratio reduces the operating current.

### SLOW Mode

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

### SLEEP Mode

The SLEEP Mode is entered when a HALT instruction is executed and when the FHIDEN and FSIDEN 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 as the WDT function is always 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 SCC and HIRCC registers are used to control the system clock and the corresponding oscillator configurations.

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

**System Operating Mode Control Register List**

### • SCC Register

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

Bit 7~5 **CKS2~CKS0**: System clock selection

000:  $f_H/2$   
 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  $f_{SUB}$ , a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4~2 Unimplemented, read as “0”

Bit 1 **FHIDEN**: High frequency oscillator control when CPU is switched off

0: Disable  
 1: Enable

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

Bit 0 **FSIDEN**: Low frequency oscillator control when CPU is switched off

0: Disable  
 1: Enable

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

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

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

• **HIRCC Register**

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

Bit 7~2 Unimplemented, read as “0”

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

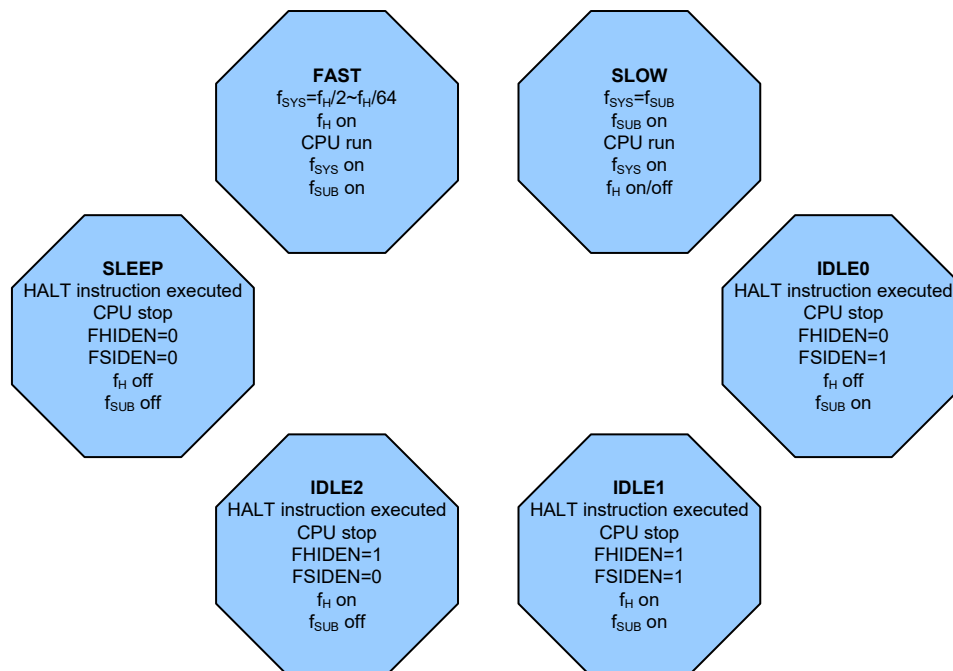
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.

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

## Operating Mode Switching

The devices 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.

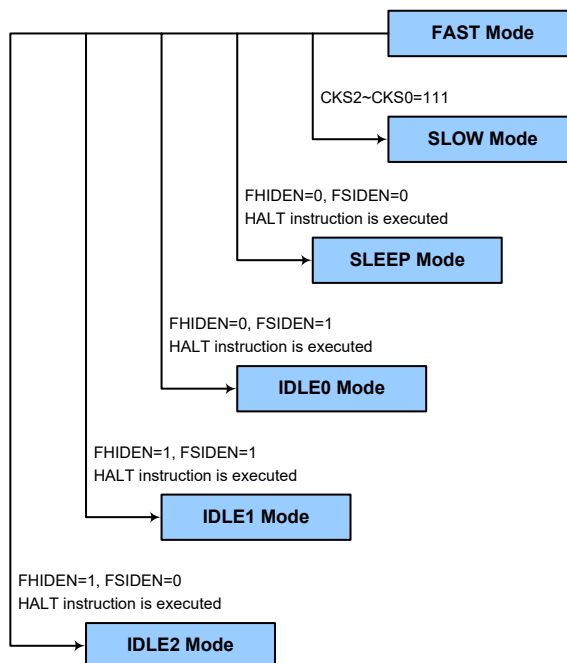
In simple terms, mode switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while mode switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the devices enters the 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.

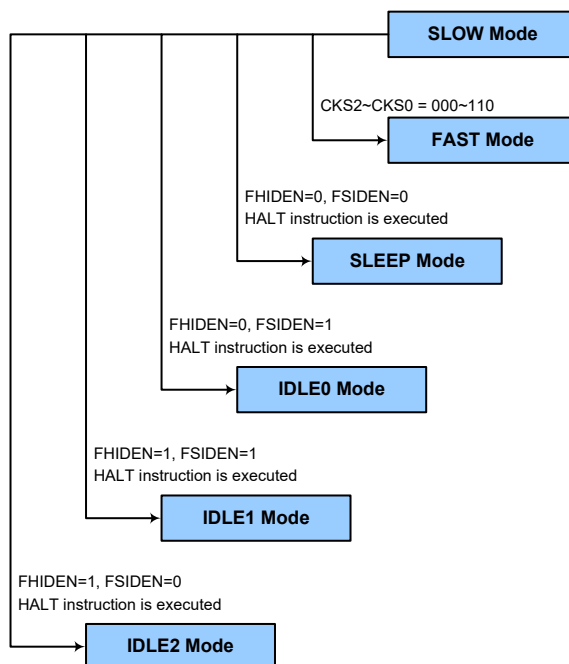
The SLOW Mode system clock is sourced from the LIRC oscillator and therefore requires this oscillator to be stable before full mode switching occurs.



### SLOW Mode to FAST Mode Switching

In the 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  $f_H/2 \sim f_H/64$ .

However, if  $f_H$  is not used in the 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 stabilisation is specified in the System Start Up Time Characteristics.



### Entering the SLEEP Mode

There is only one way for the devices 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 as the WDT function is always enabled.

### Entering the IDLE0 Mode

There is only one way for the devices 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 as the WDT function is always enabled.

### Entering the IDLE1 Mode

There is only one way for the devices 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 as the WDT function is always enabled.

### Entering the IDLE2 Mode

There is only one way for the devices 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 as the WDT function is always enabled.

### Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the devices 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 devices. 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 devices which has different package types, as there may be unbonded pins. These must either be set as outputs or if set 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 set 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 devices can enter the SLEEP or any IDLE Mode, where the CPU will be switched off. However, when the devices are woken up again, they 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 devices execute the “HALT” instruction, it will enter the IDLE or SLEEP mode and the PDF flag will be set to 1. The PDF flag will be cleared to 0 if the devices experience a system power-up or execute 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 set 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 devices 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, the WDT enable operation as well as the MCU reset operation.

• **WDTC Register**

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

Bit 7~3 **WE4~WE0**: WDT function software control  
 01010/10101: Enable  
 Other values: MCU reset  
 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,  $t_{\text{RESET}}$ , and the WRF bit in the RSTFC register will be set high.

Bit 2~0 **WS2~WS0**: WDT time-out period selection  
 000:  $2^8/f_{\text{LIRC}}$   
 001:  $2^{10}/f_{\text{LIRC}}$   
 010:  $2^{12}/f_{\text{LIRC}}$   
 011:  $2^{14}/f_{\text{LIRC}}$   
 100:  $2^{15}/f_{\text{LIRC}}$   
 101:  $2^{16}/f_{\text{LIRC}}$   
 110:  $2^{17}/f_{\text{LIRC}}$   
 111:  $2^{18}/f_{\text{LIRC}}$   
 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 | LRF | WRF |
| R/W  | — | — | — | — | — | R/W  | R/W | R/W |
| POR  | — | — | — | — | — | x    | 0   | 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 **LRF**: LVR control register software reset flag  
 Refer to the Low Voltage Reset section.  
 Bit 0 **WRF**: WDT control register software reset flag  
 0: Not occurred  
 1: Occurred  
 This bit is set to 1 by the WDT control register software reset and cleared by the application program. 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 devices. There are five bits, WE4~WE0, in the WDTC register to offer the enable control and reset control of the Watchdog Timer. The WDT function will be enabled if the WE4~WE0 bits are equal to 01010B or 10101B. If the WE4~WE0 bits are set to any other values, other than 01010B and 10101B, it will reset the devices after a delay time,  $t_{SRESET}$ . After power on these bits will have a value of 01010B.

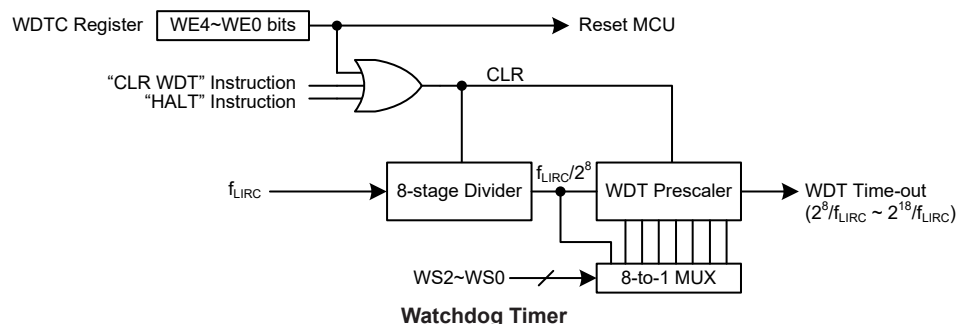
| WE4~WE0 Bits    | WDT Function |
|-----------------|--------------|
| 01010B/10101B   | Enable       |
| Any other value | MCU reset    |

**Watchdog Timer Function Control**

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





## Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the devices can be set to some predetermined condition irrespective of outside parameters. The most important reset condition is after power is first applied to the microcontrollers. In this case, internal circuitry will ensure that the microcontrollers, 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.

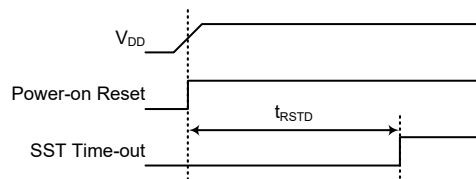
Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All types of reset operations result 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 microcontrollers. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. All the I/O port and port control registers will power up in a high condition ensuring that all pins will be first set to inputs.



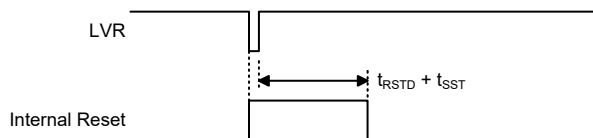
Note:  $t_{RSTD}$  is power-on delay specified in System Start Up Time Characteristics.

**Power-on Reset Timing Chart**

#### Low Voltage Reset – LVR

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

The LVR function is always enabled in the FAST/SLOW mode with a specific LVR voltage  $V_{LVR}$ . If the supply voltage of the devices 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 devices internally and the LVRF bit in the RSTFC register will also be set high. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between  $0.9V \sim V_{LVR}$  must exist for a time greater than that specified by  $t_{LVR}$  in the LVR/LVD Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual  $V_{LVR}$  value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are set to the values specified in the LVRC register, the LVR function is enabled with a fixed LVR voltage of 2.55V. If the LVS7~LVS0 bits are changed to some different values by environmental noise, the LVR will reset the devices after a delay time,  $t_{SRESET}$ . When this happens, the LRF bit in the RSTFC register will be set high. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the devices enter the IDLE or SLEEP mode.



**Low Voltage Reset Timing Chart**

• **LVRC Register**

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

Bit 7~0 **LVS7~LVS0**: LVR voltage select control

01010101: 2.55V

00110011: 2.55V

10011001: 2.55V

10101010: 2.55V

Any other value: MCU reset – register is reset to POR value

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

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

• **RSTFC Register**

| Bit  | 7 | 6 | 5 | 4 | 3 | 2    | 1   | 0   |
|------|---|---|---|---|---|------|-----|-----|
| Name | — | — | — | — | — | LVRF | LRF | WRF |
| R/W  | — | — | — | — | — | R/W  | R/W | R/W |
| POR  | — | — | — | — | — | x    | 0   | 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 high when a specific low voltage reset situation condition occurs. This bit can only be cleared to zero by the application program.

Bit 1 **LRF**: LVR control register software reset flag

0: Not occurred

1: Occurred

This bit is set high if the LVRC register contains any non-defined LVRC register values. This in effect acts like a software-reset function. This bit can only be cleared to zero by the application program.

Bit 0 **WRF**: WDT control register software reset flag

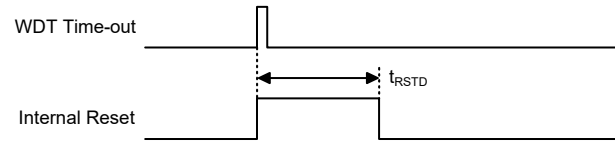
Refer to the Watchdog Timer Control Register section.

**In Application Programming Reset**

The device contains an IAP function, therefore an IAP reset exists, which is caused by writing data 55H to the FC1 register.

### Watchdog Time-out Reset during Normal Operation

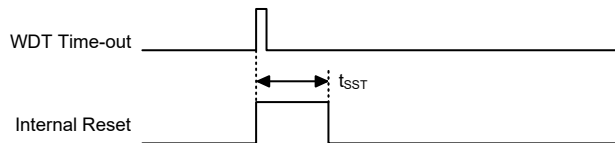
When the Watchdog time-out Reset during normal operations in the FAST or SLOW mode occurs, the Watchdog time-out flag TO will be set to “1”.



**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 Timing Chart**

### Reset Initial Conditions

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

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

“u”: Unchanged

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

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

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

| Register | Reset<br>(Power On) | WDT Time-out<br>(Normal Operation) | WDT Time-out<br>(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     | ---- xxxx           | ---- uuuu                          | ---- uuuu                    |
| 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    | ---- -x00           | ---- -uuu                          | ---- -uuu                    |
| LVRC     | 0101 0101           | 0101 0101                          | uuuu uuuu                    |
| LVDC     | --00 0000           | --00 0000                          | --uu uuuu                    |
| SCC      | 010- --00           | 010- --00                          | uuu- --uu                    |
| HIRCC    | ---- --01           | ---- --01                          | ---- --uu                    |
| PA       | 11-- 1111           | 11-- 1111                          | uu-- uuuu                    |
| PAC      | 11-- 1111           | 11-- 1111                          | uu-- uuuu                    |
| PAPU     | 00-- 0000           | 00-- 0000                          | uu-- uuuu                    |
| PAWU     | 00-- 0000           | 00-- 0000                          | uu-- uuuu                    |
| PB       | 1111 1111           | 1111 1111                          | uuuu uuuu                    |
| PBC      | 1111 1111           | 1111 1111                          | uuuu uuuu                    |
| PBPU     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| PC       | --11 1111           | --11 1111                          | --uu uuuu                    |
| PCC      | --11 1111           | --11 1111                          | --uu uuuu                    |
| PCPU     | --00 0000           | --00 0000                          | --uu uuuu                    |
| PD       | 1111 11--           | 1111 11--                          | uuuu uu--                    |
| PDC      | 1111 11--           | 1111 11--                          | uuuu uu--                    |
| PDPU     | 0000 00--           | 0000 00--                          | uuuu uu--                    |
| PAS0     | 00-- 00--           | 00-- 00--                          | uu-- uu--                    |
| PAS1     | 0000 ----           | 0000 ----                          | uuuu ----                    |
| PBS0     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| PBS1     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| PCS0     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| PCS1     | ---- --00           | ---- --00                          | ---- --uu                    |
| PDS0     | 0000 ----           | 0000 ----                          | uuuu ----                    |
| PDS1     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| WDTC     | 0101 0011           | 0101 0011                          | uuuu uuuu                    |
| CPR      | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| OCVPC    | ---- 1000           | ---- 1000                          | ---- uuuu                    |
| CTM0C0   | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| CTM0C1   | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| CTM0DL   | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| CTM0DH   | ---- --00           | ---- --00                          | ---- --uu                    |

| Register | Reset<br>(Power On)    | WDT Time-out<br>(Normal Operation) | WDT Time-out<br>(IDLE/SLEEP) |
|----------|------------------------|------------------------------------|------------------------------|
| CTM0AL   | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| CTM0AH   | ---- --00              | ---- --00                          | ---- --uu                    |
| CTM1C0   | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| CTM1C1   | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| CTM1DL   | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| CTM1DH   | ---- --00              | ---- --00                          | ---- --uu                    |
| CTM1AL   | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| CTM1AH   | ---- --00              | ---- --00                          | ---- --uu                    |
| PTMC0    | 0000 0---              | 0000 0---                          | uuuu u---                    |
| PTMC1    | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| PTMDL    | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| 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                    |
| SWS0     | ---0 0000              | ---0 0000                          | ---u uuuu                    |
| SADC0    | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| SADC1    | --00 0000              | --00 0000                          | --uu uuuu                    |
| SADOH    | xxxx xxxx<br>(ADRFC=0) | xxxx xxxx<br>(ADRFC=0)             | uuuu uuuu<br>(ADRFC=0)       |
|          | ---- xxxx<br>(ADRFC=1) | ---- xxxx<br>(ADRFC=1)             | ---- uuuu<br>(ADRFC=1)       |
| SADOL    | xxxx ----<br>(ADRFC=0) | xxxx ----<br>(ADRFC=0)             | uuuu ----<br>(ADRFC=0)       |
|          | xxxx xxxx<br>(ADRFC=1) | xxxx xxxx<br>(ADRFC=1)             | uuuu uuuu<br>(ADRFC=1)       |
| INTC0    | -000 0000              | -000 0000                          | -uuu uuuu                    |
| INTC1    | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| INTC2    | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| INTC3    | -000 -000              | -000 -000                          | -uuu -uuu                    |
| MFI      | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| INTEG    | ---- 0000              | ---- 0000                          | ---- uuuu                    |
| PSC0R    | ---- --00              | ---- --00                          | ---- --uu                    |
| PSC1R    | ---- --00              | ---- --00                          | ---- --uu                    |
| PMPS     | ---- 0000              | ---- 0000                          | ---- uuuu                    |
| SLEDC0   | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| SLEDC1   | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| OCP0C0   | 0000 0--0              | 0000 0--0                          | uuuu u--u                    |
| OCP0C1   | --00 0000              | --00 0000                          | --uu uuuu                    |
| OCP0DA   | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| OCP0OCAL | 0010 0000              | 0010 0000                          | uuuu uuuu                    |
| OCP0CCAL | 0001 0000              | 0001 0000                          | uuuu uuuu                    |
| OCP1C0   | 0000 0--0              | 0000 0--0                          | uuuu u--u                    |
| OCP1C1   | --00 0000              | --00 0000                          | --uu uuuu                    |
| OCP1DA   | 0000 0000              | 0000 0000                          | uuuu uuuu                    |
| OCP1OCAL | 0010 0000              | 0010 0000                          | uuuu uuuu                    |
| OCP1CCAL | 0001 0000              | 0001 0000                          | uuuu uuuu                    |

| Register | Reset<br>(Power On) | WDT Time-out<br>(Normal Operation) | WDT Time-out<br>(IDLE/SLEEP) |
|----------|---------------------|------------------------------------|------------------------------|
| OUVPC0   | --00 0000           | --00 0000                          | --uu uuuu                    |
| OUVPC1   | --00 0000           | --00 0000                          | --uu uuuu                    |
| OUVPC2   | 0001 0000           | 0001 0000                          | uuuu uuuu                    |
| OUVPC3   | 0001 0000           | 0001 0000                          | uuuu uuuu                    |
| OVPAH    | ---- 0000           | ---- 0000                          | ---- uuuu                    |
| OVPAH    | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| UVPDAH   | ---- 0000           | ---- 0000                          | ---- uuuu                    |
| UVPDAH   | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| ADUDA0   | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| ADUDA1   | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| ADUC0    | --00 --00           | --00 --00                          | --uu --uu                    |
| ADUC1    | --00 0000           | --00 0000                          | --uu uuuu                    |
| SIMC0    | 1110 0000           | 1110 0000                          | uuuu uuuu                    |
| SIMC1    | 1000 0001           | 1000 0001                          | uuuu uuuu                    |
| UUCR1    | 0000 00x0           | 0000 00x0                          | uuuu uuuu                    |
| SIMD     | xxxx xxxx           | xxxx xxxx                          | uuuu uuuu                    |
| UTXR_RXR | xxxx xxxx           | xxxx xxxx                          | uuuu uuuu                    |
| SIMC2    | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| SIMA     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| UUCR2    | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| UUCR3    | ---- --0            | ---- --0                           | ---- --u                     |
| SIMTOC   | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| UBRG     | xxxx xxxx           | xxxx xxxx                          | uuuu uuuu                    |
| UUSR     | 0000 1011           | 0000 1011                          | uuuu uuuu                    |
| FC0      | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| FC1      | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| FC2      | ---- --0            | ---- --0                           | ---- --u                     |
| FARL     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| FARH     | ---- 0000           | ---- 0000                          | ---- uuuu                    |
| FD0L     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| FD0H     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| FD1L     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| FD1H     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| FD2L     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| FD2H     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| FD3L     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| FD3H     | 0000 0000           | 0000 0000                          | uuuu uuuu                    |
| IFS      | ---- -000           | ---- -000                          | ---- -uuu                    |
| TB0C     | 0--- -000           | 0--- -000                          | u--- -uuu                    |
| TB1C     | 0--- -000           | 0--- -000                          | u--- -uuu                    |

Note: “u” stands for unchanged  
“x” stands for unknown  
“-” stands for unimplemented

## Input/Output Ports

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

The devices provide bidirectional input/output lines labeled with port names PA~PD. 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   | —     | —     | PA3   | PA2   | PA1   | PA0   |
| PAC           | PAC7  | PAC6  | —     | —     | PAC3  | PAC2  | PAC1  | PAC0  |
| PAPU          | PAPU7 | PAPU6 | —     | —     | PAPU3 | PAPU2 | PAPU1 | PAPU0 |
| PAWU          | PAWU7 | PAWU6 | —     | —     | PAWU3 | PAWU2 | PAWU1 | PAWU0 |
| PB            | PB7   | PB6   | PB5   | PB4   | PB3   | PB2   | PB1   | PB0   |
| PBC           | PBC7  | PBC6  | PBC5  | PBC4  | PBC3  | PBC2  | PBC1  | PBC0  |
| PBPU          | PBPU7 | PBPU6 | PBPU5 | PBPU4 | PBPU3 | PBPU2 | PBPU1 | PBPU0 |
| PC            | —     | —     | PC5   | PC4   | PC3   | PC2   | PC1   | PC0   |
| PCC           | —     | —     | PCC5  | PCC4  | PCC3  | PCC2  | PCC1  | PCC0  |
| PCPU          | —     | —     | PCPU5 | PCPU4 | PCPU3 | PCPU2 | PCPU1 | PCPU0 |
| PD            | PD7   | PD6   | PD5   | PD4   | PD3   | PD2   | —     | —     |
| PDC           | PDC7  | PDC6  | PDC5  | PDC4  | PDC3  | PDC2  | —     | —     |
| PDPU          | PDPU7 | PDPU6 | PDPU5 | PDPU4 | PDPU3 | PDPU2 | —     | —     |

“—”: Unimplemented, read as “0”

Note: The I/O lines, PB0~PB3 and PC5, are not connected to the external pins for the BP45FH4NB device.

### 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 the relevant pull-high control registers and are implemented using weak PMOS transistors.

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

#### • PxPU Register

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

**PxPUn:** I/O port x pin pull-high function control

0: Disable

1: Enable

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

For the PB0 and PD3 pins, there is another internal pull-high resistor which is always enabled and connected in parallel with the register controlled pull-high resistor. Care must be taken to the PB1 pin, which has an always enabled internal pull-low resistor, if its pull-high function is enabled, this will lead to some increase in power consumption.

#### Port A Wake-up

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

Note that the wake-up function can be controlled by the wake-up control registers only when the pin is selected as a general purpose input and the MCU enters the IDLE or SLEEP mode.

#### • PAWU Register

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

**PAWUn:** Port A pin 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~PDC, 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 set as a CMOS output. If the pin is currently set as an output, instructions can still be used to read the output register. However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.



• **PxC Register**

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

**PxCn:** I/O port x pin type selection

0: Output

1: Input

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

Care must be taken to the PB0, PB1 and PD3 pins. For the PB0 and PD3 pins, there is another internal pull-high resistor which is always enabled, if they are configured to output low level, this will lead to some increase in power consumption. For the PB1 pin, there is an internal pull-low resistor which is always enabled, if the pin is configured to output high level, this will also lead to some increase in power consumption. For BP45FH4NB device, as the PB0~PB3 and PC5 are not connected to the external pins, it is recommended to set these pins as I/O output high or output low via the related I/O port control bits.

**I/O Port Power Source Control**

This devices support different I/O port power source selections for PB4~PB7 and PD4~PD7 pins. With the exception of RES/OCDS, the multi-power function is only available when the pin function is selected as digital input or output function.

The port power can come from the power pin VDD or VDDIO, which is determined using the PMPS1~PMPS0 bit field in the PMPS register. The VDDIO power pin function should first be selected using the corresponding pin-shared function selection bits if the port power is supposed to come from the VDDIO pin. An important point to know is that the input power voltage on the VDDIO pin should be equal to or less than the devices supply power voltage  $V_{DD}$  when the VDDIO pin is selected as the port power supply pin. However, when either  $V_{DD}$  or  $V_{DDIO}$  is less than 2.2V, it is recommended that the  $V_{DDIO}$  power should be equal to  $V_{DD}$ .

• **PMPS Register**

| Bit  | 7 | 6 | 5 | 4 | 3     | 2     | 1     | 0     |
|------|---|---|---|---|-------|-------|-------|-------|
| Name | — | — | — | — | PMPS3 | PMPS2 | PMPS1 | PMPS0 |
| 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 **PMPS3~PMPS2:** PD4~PD7 pin power supply selection

00:  $V_{DD}$

01: Reserved

10: Reserved

11:  $V_{DDIO}$

Bit 1~0 **PMPS1~PMPS0:** PB4~PB7 pin power supply selection

00:  $V_{DD}$

01: Reserved

10: Reserved

11:  $V_{DDIO}$

If the PD2 pin-shared function is switched to the VDDIO function, and the PMPS3 and PMPS2 bits are set to “11”, the VDDIO pin input voltage can be used for PD4~PD7 pin power; If the PMPS1 and PMPS0 bits are set to “11”, the VDDIO pin input voltage can be used for PB4~PB7 pin power. Note that the input power voltage on the VDDIO pin should be equal to or less than the devices supply power voltage.

## I/O Port Source Current Control

The devices support different source current driving capability for each I/O port. With the corresponding selection registers, SLEDC0 and SLEDC1, each I/O port can support four levels of the source current driving capability. Users should refer to the Input/Output Characteristics section to select the desired source current for different applications.

### • SLEDC0 Register

| Bit  | 7       | 6       | 5       | 4       | 3       | 2       | 1       | 0       |
|------|---------|---------|---------|---------|---------|---------|---------|---------|
| Name | SLEDC07 | SLEDC06 | SLEDC05 | SLEDC04 | SLEDC03 | SLEDC02 | SLEDC01 | SLEDC00 |
| 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 **SLEDC07~SLEDC06**: PB7~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 **SLEDC05~SLEDC04**: PB3~PB2 source current selection

00: Source current=Level 0 (min.)  
 01: Source current=Level 1  
 10: Source current=Level 2  
 11: Source current=Level 3 (max.)

Bit 3~2 **SLEDC03~SLEDC02**: PA7~PA6 source current selection

00: Source current=Level 0 (min.)  
 01: Source current=Level 1  
 10: Source current=Level 2  
 11: Source current=Level 3 (max.)

Bit 1~0 **SLEDC01~SLEDC00**: PA3~PA0 source current selection

00: Source current=Level 0 (min.)  
 01: Source current=Level 1  
 10: Source current=Level 2  
 11: Source current=Level 3 (max.)

### • SLEDC1 Register

| Bit  | 7       | 6       | 5       | 4       | 3       | 2       | 1       | 0       |
|------|---------|---------|---------|---------|---------|---------|---------|---------|
| Name | SLEDC17 | SLEDC16 | SLEDC15 | SLEDC14 | SLEDC13 | SLEDC12 | SLEDC11 | SLEDC10 |
| 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 **SLEDC17~SLEDC16**: PD7~PD4 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 **SLEDC15~SLEDC14**: PD2 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 **SLEDC13~SLEDC12**: PC5~PC4 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      **SLEDC11~SLEDC10**: PC3~PC0 source current selection  
 00: Source current=Level 0 (min.)  
 01: Source current=Level 1  
 10: Source current=Level 2  
 11: Source current=Level 3 (max.)

## Pin-shared Functions

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

## Pin-shared Function Selection Registers

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

The most important point to note is to make sure that the desired pin-shared function is properly selected and also deselected. For most pin-shared functions, to select the desired pin-shared function, the pin-shared function should first be correctly selected using the corresponding pin-shared control register. After that the corresponding peripheral functional setting should be configured and then the peripheral function can be enabled. However, a special point must be noted for digital input pins, such as xTCK<sub>n</sub> and INT<sub>n</sub>, which share the same pin-shared control configuration with their corresponding general purpose I/O functions when setting the relevant pin-shared control bits. To select these pin functions, in addition to the necessary pin-shared control and peripheral functional setup aforementioned, they must also be set 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.

| Register Name | Bit   |       |       |       |       |        |          |            |
|---------------|-------|-------|-------|-------|-------|--------|----------|------------|
|               | 7     | 6     | 5     | 4     | 3     | 2      | 1        | 0          |
| PAS0          | PAS07 | PAS06 | —     | —     | PAS03 | PAS02  | —        | —          |
| PAS1          | PAS17 | PAS16 | PAS15 | PAS14 | —     | —      | —        | —          |
| PBS0          | PBS07 | PBS06 | PBS05 | PBS04 | PBS03 | PBS02  | PBS01    | PBS00      |
| PBS1          | PBS17 | PBS16 | PBS15 | PBS14 | PBS13 | PBS12  | PBS11    | PBS10      |
| PCS0          | PCS07 | PCS06 | PCS05 | PCS04 | PCS03 | PCS02  | PCS01    | PCS00      |
| PCS1          | —     | —     | —     | —     | —     | —      | PCS11    | PCS10      |
| PDS0          | PDS07 | PDS06 | PDS05 | PDS04 | —     | —      | —        | —          |
| PDS1          | PDS17 | PDS16 | PDS15 | PDS14 | PDS13 | PDS12  | PDS11    | PDS10      |
| IFS           | —     | —     | —     | —     | —     | SCSBPS | SCKSCLPS | SDISDARXPS |

**Pin-shared Function Selection Register List**

**• PAS0 Register**

| Bit  | 7     | 6     | 5 | 4 | 3     | 2     | 1 | 0 |
|------|-------|-------|---|---|-------|-------|---|---|
| Name | PAS07 | PAS06 | — | — | PAS03 | PAS02 | — | — |
| R/W  | R/W   | R/W   | — | — | R/W   | R/W   | — | — |
| POR  | 0     | 0     | — | — | 0     | 0     | — | — |

Bit 7~6 **PAS07~PAS06:** PA3 pin-shared function selection

00: PA3

01: AN7

10: PA3

11: PA3

Bit 5~4 Unimplemented, read as “0”

Bit 3~2 **PAS03~PAS02:** PA1 pin-shared function selection

00: PA1

01: AN9/BATV

10: PA1

11: PA1

Bit 1~0 Unimplemented, read as “0”

**• PAS1 Register**

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

Bit 7~6 **PAS17~PAS16:** PA7 pin-shared function selection

00: PA7

01: AN4

10: VREF

11: PA7

Bit 5~4 **PAS15~PAS14:** PA6 pin-shared function selection

00: PA6

01: AN10/OUVPI

10: PA6

11: PA6

Bit 3~0 Unimplemented, read as “0”

**• 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: PTPB

10: CTP1B

11: PB3

Bit 5~4 **PBS05~PBS04:** PB2 pin-shared function selection

00: PB2

01: PTP

10: OCP1COUT

11: PB2

- Bit 3~2     **PBS03~PBS02:** PB1 pin-shared function selection  
                  00: PB1  
                  01: PWML  
                  10: PB1  
                  11: PB1
- Bit 1~0     **PBS01~PBS00:** PB0 pin-shared function selection  
                  00: PB0  
                  01: PWMH  
                  10: PB0  
                  11: PB0

• **PBS1 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PBS17 | PBS16 | PBS15 | PBS14 | PBS13 | PBS12 | PBS11 | PBS10 |
| 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     **PBS17~PBS16:** PB7 pin-shared function selection  
                  00: PB7  
                  01: SCS  
                  10: UVPO  
                  11: PB7
- Bit 5~4     **PBS15~PBS14:** PB6 pin-shared function selection  
                  00: PB6  
                  01: SDO/UTX  
                  10: PB6  
                  11: PB6
- Bit 3~2     **PBS13~PBS12:** PB5 pin-shared function selection  
                  00: PB5  
                  01: SDI/SDA/URX/UTX  
                  10: PB5  
                  11: PB5
- Bit 1~0     **PBS11~PBS10:** PB4 pin-shared function selection  
                  00: PB4  
                  01: SCK/SCL  
                  10: PB4  
                  11: PB4

• **PCS0 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PCS07 | PCS06 | PCS05 | PCS04 | PCS03 | PCS02 | PCS01 | PCS00 |
| R/W  | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   | R/W   |
| POR  | 0     | 0     | 0     | 0     | 0     | 0     | 0     | 0     |

- Bit 7~6     **PCS07~PCS06:** PC3 pin-shared function selection  
                  00: PC3  
                  01: AN6  
                  10: OCP1I  
                  11: CTP0B
- Bit 5~4     **PCS05~PCS04:** PC2 pin-shared function selection  
                  00: PC2  
                  01: OCP1I  
                  10: PC2  
                  11: PC2

Bit 3~2     **PCS03~PCS02**: PC1 pin-shared function selection  
 00: PC1  
 01: AN8  
 10: OCP0I  
 11: PC1

Bit 1~0     **PCS01~PCS00**: PC0 pin-shared function selection  
 00: PC0  
 01: OCP0I  
 10: PC0  
 11: PC0

Note: If PCS0[7:4]=1001B, both the PC2 pin and PC3 pin can be used for OCP1I input at the same time. If PCS0[3:0]=1001B, both PC0 pin and PC1 pin can be used for OCP0I input at the same time. However, when setting the PCS0 register, it is recommended to avoid setting the PCS0[7:0] to OCP0I or OCP1I input at the same time

#### • PCS1 Register

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1     | 0     |
|------|---|---|---|---|---|---|-------|-------|
| Name | — | — | — | — | — | — | PCS11 | PCS10 |
| R/W  | — | — | — | — | — | — | R/W   | R/W   |
| POR  | — | — | — | — | — | — | 0     | 0     |

Bit 7~2     Unimplemented, read as “0”

Bit 1~0     **PCS11~PCS10**: PC4 pin-shared function selection  
 00: PC4/INT0  
 01: AN5  
 10: CTP0  
 11: PC4/INT0

#### • PDS0 Register

| Bit  | 7     | 6     | 5     | 4     | 3 | 2 | 1 | 0 |
|------|-------|-------|-------|-------|---|---|---|---|
| Name | PDS07 | PDS06 | PDS05 | PDS04 | — | — | — | — |
| R/W  | R/W   | R/W   | R/W   | R/W   | — | — | — | — |
| POR  | 0     | 0     | 0     | 0     | — | — | — | — |

Bit 7~6     **PDS07~PDS06**: PD3 pin-shared function selection  
 00: PD3  
 01: CTP1  
 10: OCP0COUT  
 11: PD3

Bit 5~4     **PDS05~PDS02**: PD4 pin-shared function selection  
 00: PD2  
 01: VDDIO  
 10: OVPO  
 11: PD2

Bit 3~0     Unimplemented, read as “0”

• **PDS1 Register**

| Bit  | 7     | 6     | 5     | 4     | 3     | 2     | 1     | 0     |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| Name | PDS17 | PDS16 | PDS15 | PDS14 | PDS13 | PDS12 | PDS11 | PDS10 |
| 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      **PDS17~PDS16:** PD7 pin-shared function selection

00: PD7

01: AN3

10: D2-

11: SCS

Bit 5~4      **PDS15~PDS14:** PD6 pin-shared function selection

00: PD6

01: AN2

10: D2+

11: SDO/UTX

Bit 3~2      **PDS13~PDS12:** PD5 pin-shared function selection

00: PD5

01: AN1

10: D1-

11: SDI/SDA/URX/UTX

Bit 1~0      **PDS11~PDS10:** PD4 pin-shared function selection

00: PD4

01: AN0

10: D1+

11: SCK/SCL

• **IFS Register**

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

Bit 7~3      Unimplemented, read as “0”

Bit 2      **SCSBPS:**  $\overline{\text{SCS}}$  input source pin selection

0: PB7

1: PD7

Bit 1      **SCKSCLPS:** SCK/SCL input source pin selection

0: PB4

1: PD4

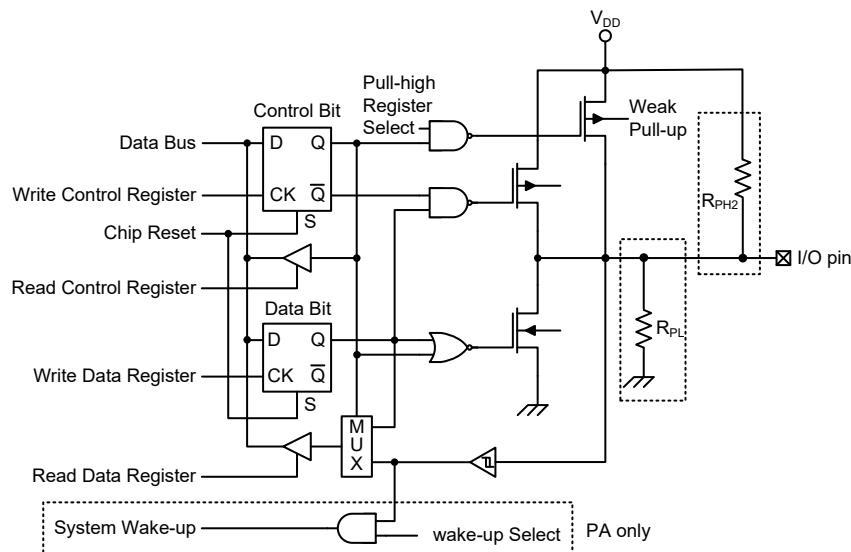
Bit 0      **SDISDARXPS:** SDI/SDA/URX/UTX input source pin selection

0: PB5

1: PD5

## I/O Pin Structure

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.



Note: The  $R_{PH2}$  resistor is only available for the PB0 and PD3 pins, while the  $R_{PL}$  resistor is only available for the PB1 pins.

### Logic Function Input/Output Structure

## Programming Considerations

Within the user program, one of the first things to consider is port initialisation. After a reset, all of the I/O data and port control registers will be set high. This means that all I/O pins will default to an input state, the level of which depends on the other connected circuitry and whether pull-high selections have been chosen. If the port control registers are then programmed to set 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 devices are in the SLEEP or IDLE Mode, various methods are available to wake the devices 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 set to have this function.



## Timer Modules – TM

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

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

### Introduction

The devices contain several Timer Modules and each individual TM can be categorised as a certain type, namely Compact Type TM or Periodic Type TM. Although similar in nature, the different TM types vary in their feature complexity. The common features to all of the Compact and Periodic type TMs will be described in this section and the detailed operation regarding each of the TM types will be described in separate sections. The main features and differences between the two types of TMs are summarised in the accompanying table.

| TM Function                  | CTM            | PTM            |
|------------------------------|----------------|----------------|
| Timer/Counter                | √              | √              |
| Compare Match Output         | √              | √              |
| PWM Output                   | √              | √              |
| Single Pulse Output          | —              | √              |
| PWM Alignment                | Edge           | Edge           |
| PWM Adjustment Period & Duty | Duty or Period | 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 count-up counter whose value is then compared with the value of pre-programmed internal comparators. When the free running count-up counter has the same value as the pre-programmed comparator, known as a compare match situation, a TM interrupt signal will be generated which can clear the counter and perhaps also change the condition of the TM output pin. The internal TM counter is driven by a user selectable clock source, which can be an internal clock or an external pin.

### TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the xTnCK2~xTnCK0 bits in the xTM control registers, where “x” stands for C or P type TM and “n” stands for the specific TM serial number. For the PTM there is no serial number “n” in the relevant pins, registers and control bits since there is only one PTM in the devices. 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 xTCKn pin. The xTCKn pin clock source is used to allow an external signal to drive the TM as an external clock source for event counting.

## TM Interrupts

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

## TM External Pins

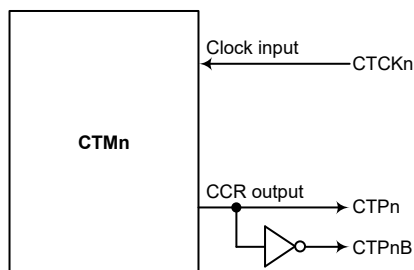
Each of the TMs, irrespective of what type, has one TM input pin, with the label xTCKn. The xTMn input pin, xTCKn, is essentially a clock source for the xTMn and is selected using the xTnCK2~xTnCK0 bits in the xTMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The TM 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 for the PTM.

The TMs each has two output pins with the label xTPn and xTPnB. 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 xTPn and xTPnB output pins are also the pins where the TM generates the PWM output waveform.

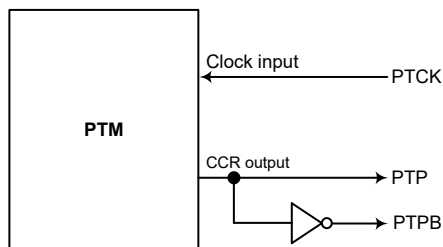
As the TM input/output pins are pin-shared with other functions, the TM input/output function must first be setup using relevant pin-shared function selection registers. The details of the pin-shared function selection are described in the pin-shared function section.

| CTM            |                            | CTM   |           |
|----------------|----------------------------|-------|-----------|
| Input          | Input                      | Input | Output    |
| CTCK0<br>CTCK1 | CTP0, CTP0B<br>CTP1, CTP1B | PTCK  | PTP, PTPB |

**TM External Pins**



**CTM Function Pin Block Diagram (n=0~1)**

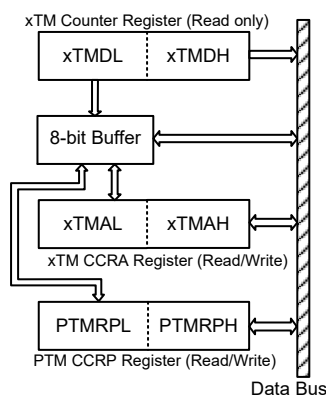


**PTM Function Pin Block Diagram**

## Programming Considerations

The TM Counter Registers and the 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 xTMAL 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.

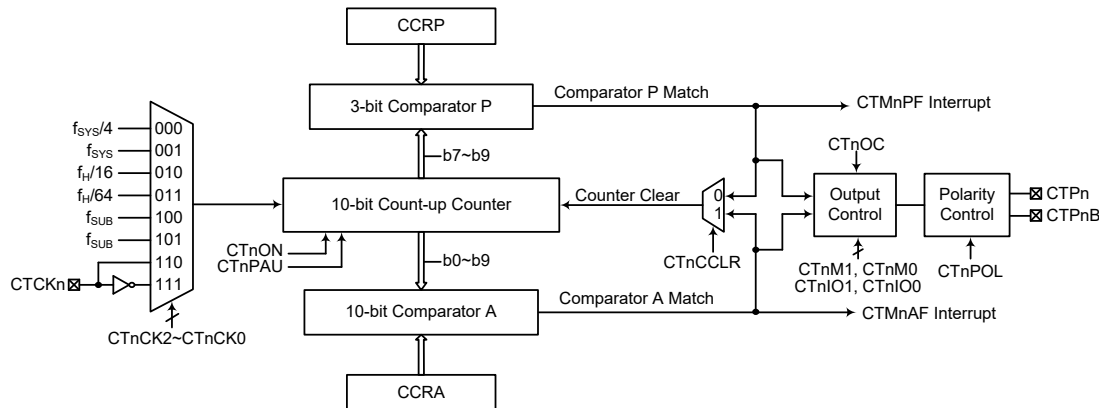


The following steps show the read and write procedures:

- Writing Data to CCRA or CCRP
  - ♦ Step 1. Write data to Low Byte xTMAL or PTMRPL
    - Note that here data is only written to the 8-bit buffer.
  - ♦ Step 2. Write data to High Byte xTMAH 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 CCRA or CCRP
  - ♦ Step 1. Read data from the High Byte xTMDH, xTMAH 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 xTMDL, xTMAL or PTMRPL
    - This step reads data from the 8-bit buffer.

## Compact Type TM – CTM

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



Note: 1. The CTM external pins are pin-shared with other functions, so before using the CTM function, ensure that the pin-shared function register has been set properly to enable the CTM pin function. The CTCKn pins, if used, must also be set as an input by setting the corresponding bits in the port control register.

2. CTPnB is the inverse signal of CTPn.

**10-bit Compact Type TM Block Diagram (n=0~1)**

## Compact Type TM Operation

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

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the CTnON 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 CTMn interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control two output pins. All operating setup conditions are selected using relevant internal registers.

## Compact Type TM Register Description

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

| Register Name | Bit    |        |        |        |       |        |        |         |
|---------------|--------|--------|--------|--------|-------|--------|--------|---------|
|               | 7      | 6      | 5      | 4      | 3     | 2      | 1      | 0       |
| CTMnC0        | CTnPAU | CTnCK2 | CTnCK1 | CTnCK0 | CTnON | CTnRP2 | CTnRP1 | CTnRP0  |
| CTMnC1        | CTnM1  | CTnM0  | CTnIO1 | CTnIO0 | CTnOC | CTnPOL | CTnDPX | CTnCCLR |
| CTMnDL        | D7     | D6     | D5     | D4     | D3    | D2     | D1     | D0      |
| CTMnDH        | —      | —      | —      | —      | —     | —      | D9     | D8      |
| CTMnAL        | D7     | D6     | D5     | D4     | D3    | D2     | D1     | D0      |
| CTMnAH        | —      | —      | —      | —      | —     | —      | D9     | D8      |

**10-bit Compact Type TM Register List (n=0~1)**

• **CTMnC0 Register**

| Bit  | 7      | 6      | 5      | 4      | 3     | 2      | 1      | 0      |
|------|--------|--------|--------|--------|-------|--------|--------|--------|
| Name | CTnPAU | CTnCK2 | CTnCK1 | CTnCK0 | CTnON | CTnRP2 | CTnRP1 | CTnRP0 |
| 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 **CTnPAU**: CTMn 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 CTMn 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 **CTnCK2~CTnCK0**: Select CTMn counter clock

000:  $f_{SYS}/4$

001:  $f_{SYS}$

010:  $f_H/16$

011:  $f_H/64$

100:  $f_{SUB}$

101:  $f_{SUB}$

110: CTCKn rising edge clock

111: CTCKn falling edge clock

These three bits are used to select the clock source for the CTMn. 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 **CTnON**: CTMn counter on/off control

0: Off

1: On

This bit controls the overall on/off function of the CTMn. Setting the bit high enables the counter to run, clearing the bit to 0 disables the CTMn. Clearing this bit to zero will stop the counter from counting and turn off the CTMn 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 CTMn is in the Compare Match Output Mode or the PWM Output Mode then the CTMn output pin will be reset to its initial condition, as specified by the CTnOC bit, when the CTnON bit changes from low to high.

Bit 2~0 **CTnRP2~CTnRP0**: CTMn CCRP 3-bit register, compared with the CTMn Counter bit 9~bit 7

Comparator P Match Period

- 000: 1024 CTMn clocks
- 001: 128 CTMn clocks
- 010: 256 CTMn clocks
- 011: 384 CTMn clocks
- 100: 512 CTMn clocks
- 101: 640 CTMn clocks
- 110: 768 CTMn clocks
- 111: 896 CTMn clocks

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

#### • CTMnC1 Register

| Bit  | 7     | 6     | 5      | 4      | 3     | 2      | 1      | 0       |
|------|-------|-------|--------|--------|-------|--------|--------|---------|
| Name | CTnM1 | CTnM0 | CTnIO1 | CTnIO0 | CTnOC | CTnPOL | CTnDPX | CTnCCLR |
| 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 **CTnM1~CTnM0**: Select CTMn operating mode

- 00: Compare Match Output Mode
- 01: Undefined
- 10: PWM Output Mode
- 11: Timer/Counter Mode

These bits setup the required operating mode for the CTMn. To ensure reliable operation the CTMn should be switched off before any changes are made to the CTnM1 and CTnM0 bits. In the Timer/Counter Mode, the CTMn output pin state is undefined.

Bit 5~4 **CTnIO1~CTnIO0**: Select CTMn external pin function

Compare Match Output Mode

- 00: No change
- 01: Output low
- 10: Output high
- 11: Toggle output

PWM Output Mode

- 00: PWM Output inactive state
- 01: PWM Output active state
- 10: PWM output
- 11: Undefined

Timer/counter Mode

Unused

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

In the Compare Match Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn output pin changes state when a compare match occurs from the Comparator A. The CTMn 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 CTMn

output pin should be setup using the CTnOC bit in the CTMnC1 register. Note that the output level requested by the CTnIO1 and CTnIO0 bits must be different from the initial value setup using the CTnOC bit otherwise no change will occur on the CTMn output pin when a compare match occurs. After the CTMn output pin changes state it can be reset to its initial level by changing the level of the CTnON bit from low to high.

In the PWM Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn 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 CTnIO1 and CTnIO0 bits only after the CTMn has been switched off. Unpredictable PWM outputs will occur if the CTnIO1 and CTnIO0 bits are changed when The CTMn is running.

Bit 3 **CTnOC**: CTPn output control bit

Compare Match Output Mode

0: Initial low

1: Initial high

PWM Output Mode

0: Active low

1: Active high

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

Bit 2 **CTnPOL**: CTPn output polarity control

0: Non-invert

1: Invert

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

Bit 1 **CTnDPX**: CTMn PWM period/duty control

0: CCRP – period; CCRA – duty

1: CCRP – duty; CCRA – period

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

Bit 0 **CTnCCLR**: Select CTMn counter clear condition

0: CTMn Comparator P match

1: CTMn Comparator A match

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

#### • CTMnDL 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**: CTMn counter low byte register bit 7 ~ bit 0

CTMn 10-bit counter bit 7 ~ bit 0

**• CTMnDH 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**: CTMn counter high byte register bit 1 ~ bit 0  
 CTMn 10-bit counter bit 9 ~ bit 8

**• CTMnAL 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**: CTMn CCRA low byte register bit 7 ~ bit 0  
 CTMn 10-bit CCRA bit 7 ~ bit 0

**• CTMnAH 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**: CTMn CCRA high byte register bit 1 ~ bit 0  
 CTMn 10-bit CCRA bit 9 ~ bit 8

**Compact Type TM Operating Modes**

The Compact type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

**Compare Match Output Mode**

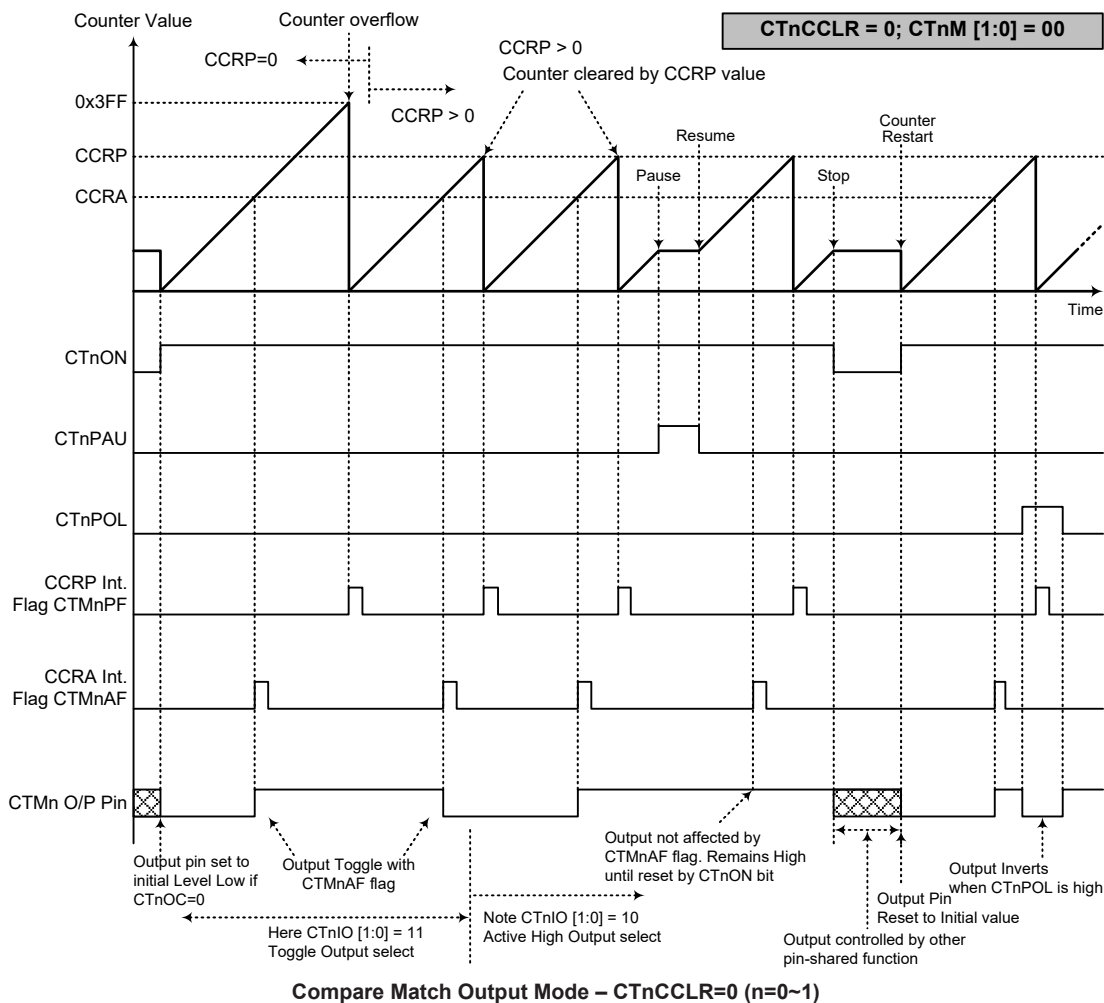
To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTnCCLR 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 CTMnAF and CTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the CTnCCLR bit in the CTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when CTnCCLR is high no CTMnPF interrupt request flag will be generated. If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the CTMnAF interrupt request flag will not be generated.

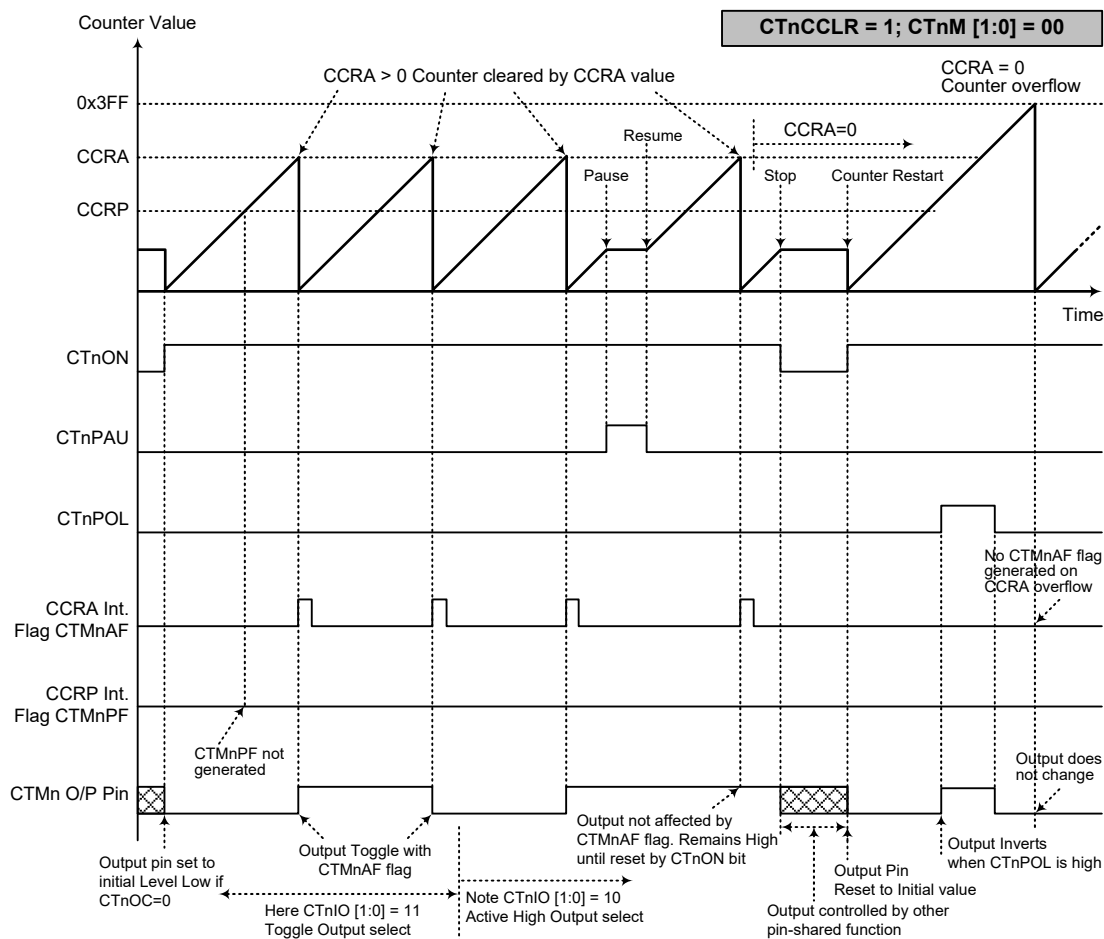
As the name of the mode suggests, after a comparison is made, the CTMn output pin will change state. The CTMn output pin condition however only changes state when a CTMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMnPF interrupt



request flag, generated from a compare match occurs from Comparator P, will have no effect on the CTMn output pin. The way in which the CTMn output pin changes state are determined by the condition of the CTnIO1 and CTnIO0 bits in the CTMnC1 register. The CTMn output pin can be selected using the CTnIO1 and CTnIO0 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 CTMn output pin, which is setup after the CTnON bit changes from low to high, is setup using the CTnOC bit. Note that if the CTnIO1 and CTnIO0 bits are zero then no pin change will take place.



- Note: 1. With CTnCCLR=0, a Comparator P match will clear the counter  
 2. The CTMn output pin controlled only by the CTMnAF flag  
 3. The output pin reset to initial state by a CTnON bit rising edge



**Compare Match Output Mode – CTnCCR=1 (n=0~1)**

- Note:
1. With CTnCCR=1, a Comparator A match will clear the counter
  2. The CTMn output pin controlled only by the CTMnAF flag
  3. The output pin reset to initial state by a CTnON rising edge
  4. The CTMnPF flags is not generated when CTnCCR=1

### Timer/Counter Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 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 CTMn 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 CTMn output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

### PWM Output Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register should be set to 10 respectively. The PWM function within the CTMn 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 CTMn 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 CTnCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CTnDPX bit in the CTMnC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The CTnOC bit In the CTMnC1 register is used to select the required polarity of the PWM waveform while the two CTnIO1 and CTnIO0 bits are used to enable the PWM output or to force the CTMn output pin to a fixed high or low level. The CTnPOL bit is used to reverse the polarity of the PWM output waveform.

#### • 10-bit CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=0

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

If  $f_{SYS}=30\text{MHz}$ , CTMn clock source is  $f_{SYS}/4$ , CCRP=2, CCRA=128,

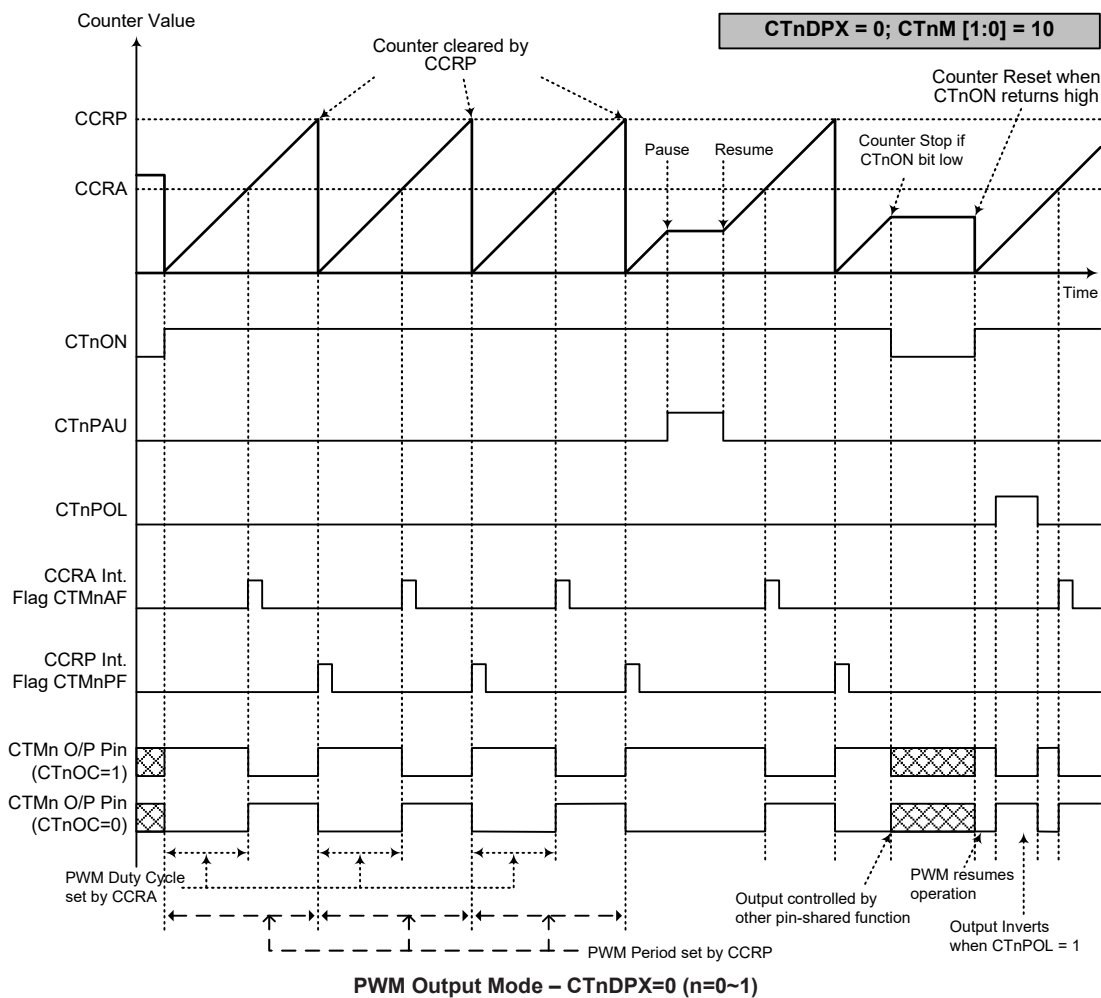
The CTMn PWM output frequency= $(f_{SYS}/4)/(2 \times 128)=f_{SYS}/1024=29.297\text{kHz}$ , duty= $128/(2 \times 128)=50\%$ .

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

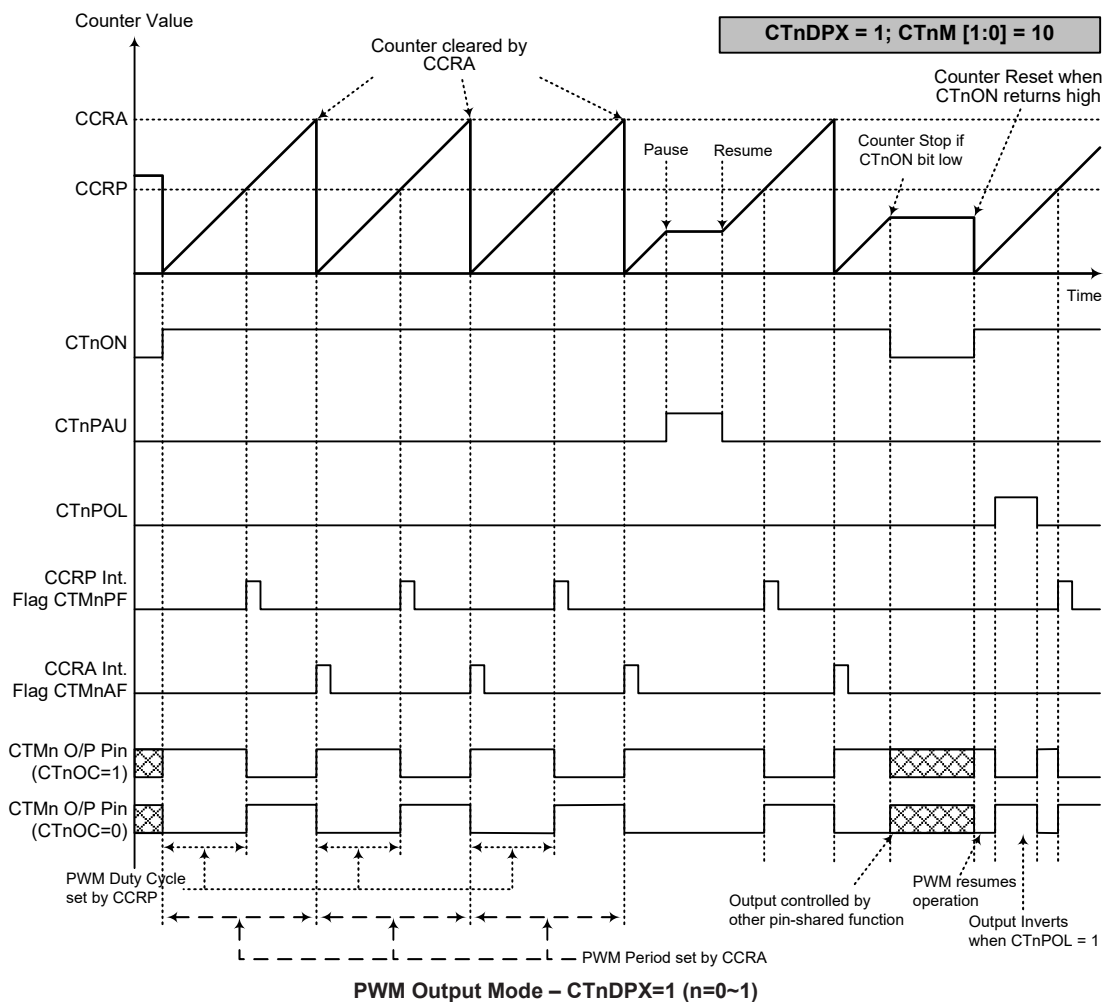
#### • 10-bit CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=1

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

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



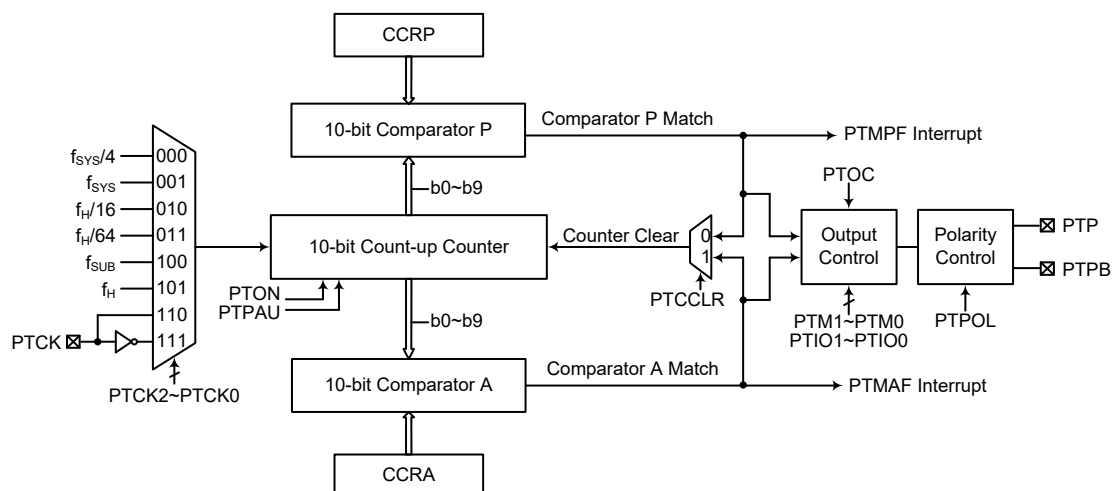
- Note: 1. Here CTnDPX=0 – Counter cleared by CCRP
2. A counter clear sets PWM Period
3. The internal PWM function continues running even when CTnIO[1:0]=00 or 01
4. The CTnCCLR bit has no influence on PWM operation



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

## Periodic Type TM – PTM

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



- Note: 1. The PTM external pins are pin-shared with other functions, so before using the PTM function, ensure that the pin-shared function register has been set properly to enable the PTM pin function. The  $PTCK$  pin, if used, must also be set as an input by setting the corresponding bits in the port control register.
2. The  $PTPB$  is the inverted signal of the  $PTP$ .

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

## Periodic Type TM Operation

The size of Periodic Type TM is 10-bit wide and its core is a 10-bit count-up counter which is driven by a user selectable internal or external clock source. There are also two internal comparators with the names, Comparator A and Comparator P. These comparators will compare the value in the counter with  $CCRP$  and  $CCRA$  registers. The  $CCRP$  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  $CCRA$  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 | D1 | 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        | D7    | D6    | D5    | D4    | D3   | D2    | D1 | D0     |
| PTMRPH        | —     | —     | —     | —     | —    | —     | D9 | D8     |

**10-bit Periodic Type 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:  $f_H$   
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 while clearing the bit to zero 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 | D1  | 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: Undefined
- 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

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 TM output pin changes state when a certain compare match condition occurs. The PTM 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 PTM 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

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 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 **D1**: Reserved, must be fixed at “0”

Bit 0 **PTCCLR**: PTM counter clear condition selection  
 0: Comparator P match  
 1: Comparator A match

This bit is used to select the method which clears the counter. Remember that the Periodic type 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 or Single Pulse Output 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 | 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 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 | — | — | — | — | — | — | D9  | D8  |
| R/W  | — | — | — | — | — | — | R/W | R/W |
| POR  | — | — | — | — | — | — | 0   | 0   |

Bit 7~2      Unimplemented, read as “0”  
Bit 1~0      **D9~D8**: PTM CCRP high byte register bit 1 ~ bit 0  
PTM 10-bit CCRP bit 9 ~ bit 8

## **Periodic Type TM Operation Modes**

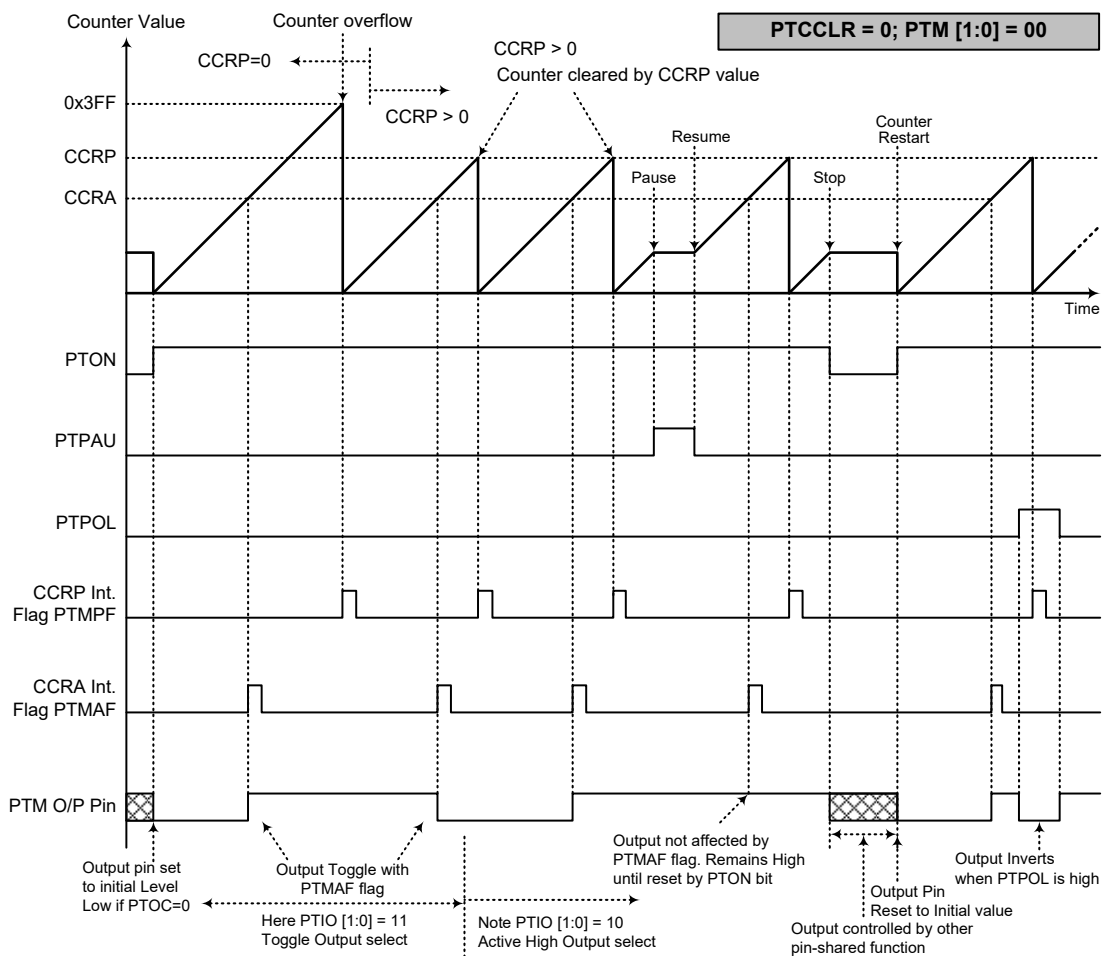
The Periodic Type TM can operate in one of four operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode or Timer/Counter Mode. The operating mode is selected using the PTM1 and PTM0 bits in the PTMC1 register.

### **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.

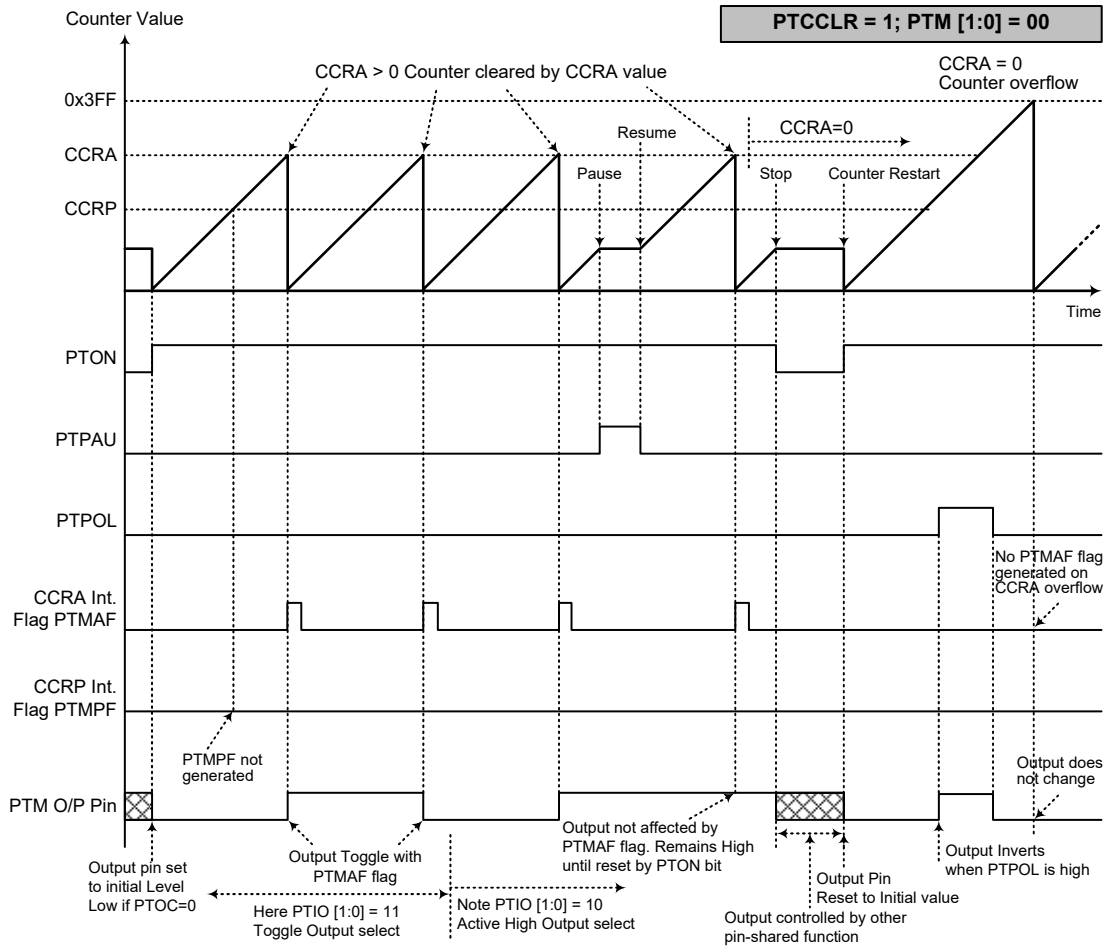
If the PTCCLR bit in the PTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTCCLR is high no PTMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA can not be cleared to “0”. If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the PTMnAF 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 – PTCCLR=0**

- Note: 1. With PTCCLR=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 pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

### PWM Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits 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 as the PWM period. Both of the CCRP and CCRA registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The 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.

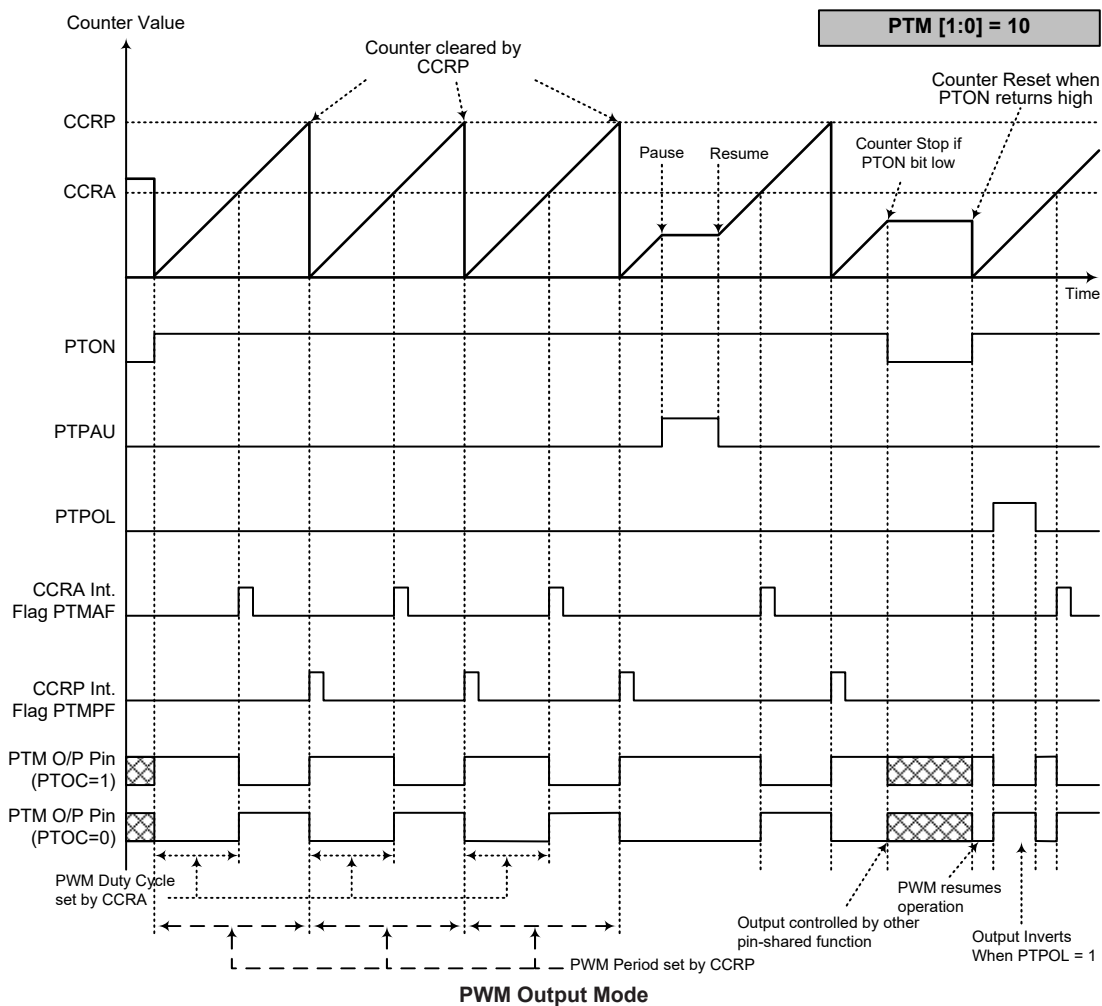
#### • 10-bit PTM, PWM Output Mode, Edge-aligned Mode

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

If  $f_{SYS}=30\text{MHz}$ , PTM clock source select  $f_{SYS}/4$ , CCRP=512 and CCRA=128,

The PTM PWM output frequency= $(f_{SYS}/4)/512=f_{SYS}/2048=14.6484\text{kHz}$ , 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%.



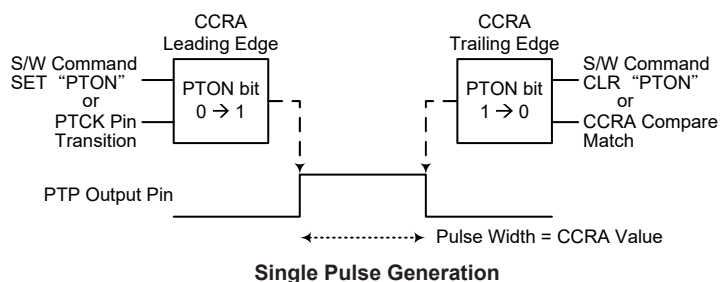
- Note:
1. The counter is cleared by CCRP
  2. A counter clear sets the PWM Period
  3. The internal PWM function continues running even when PTIO [1:0]=00 or 01
  4. The PTCCLR bit has no influence on PWM operation

### Single Pulse Output Mode

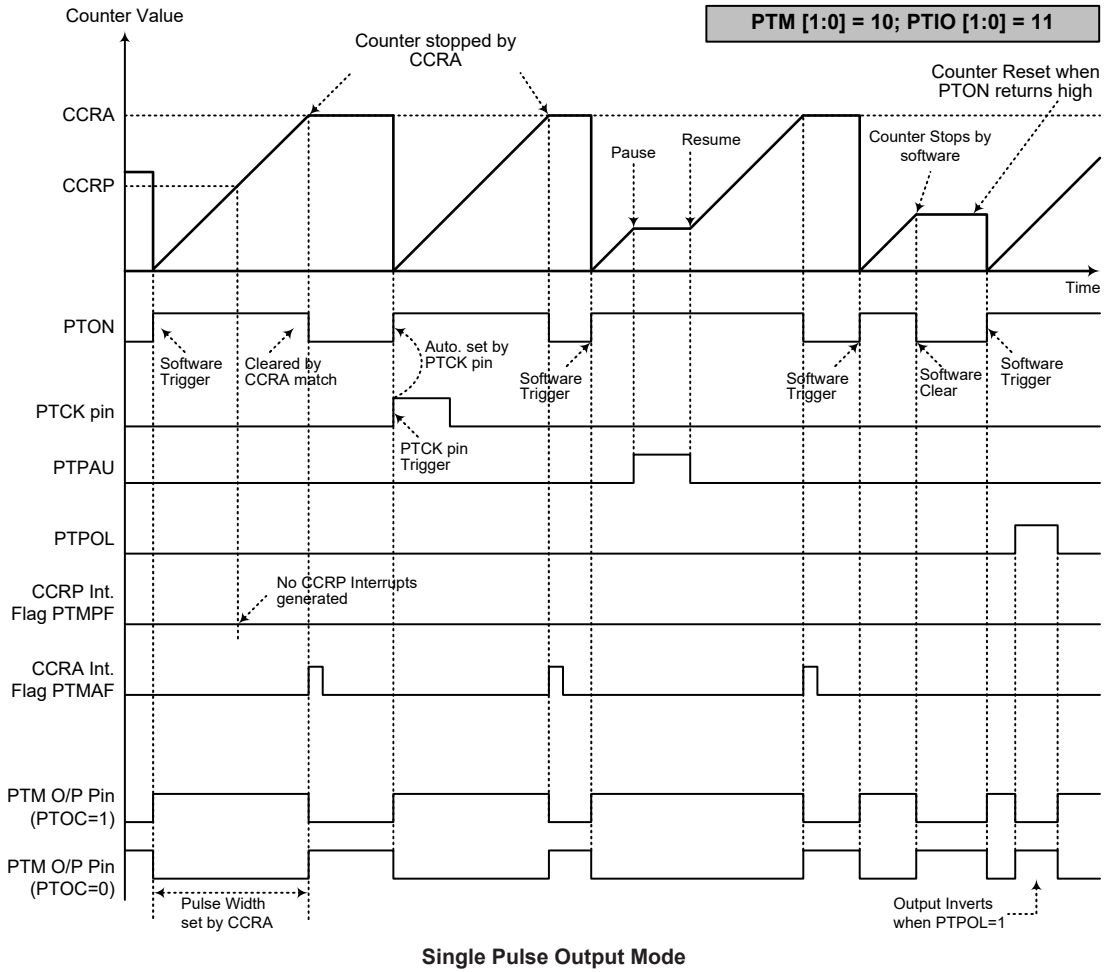
To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTM output pin.

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 is not used in this Mode.



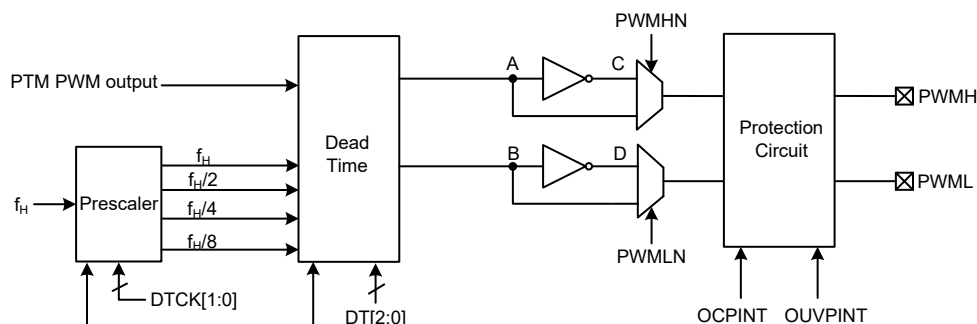




- Note: 1. Counter stopped by CCRA  
 2. CCRP is not used  
 3. The pulse triggered by the PTCK pin or by setting the PTON bit high  
 4. A PTCK pin active edge will automatically set the PTON bit high  
 5. In the Single Pulse Output Mode, PTIO [1:0] must be set to "11" and can not be changed

## Complementary PWM Output with Dead Time

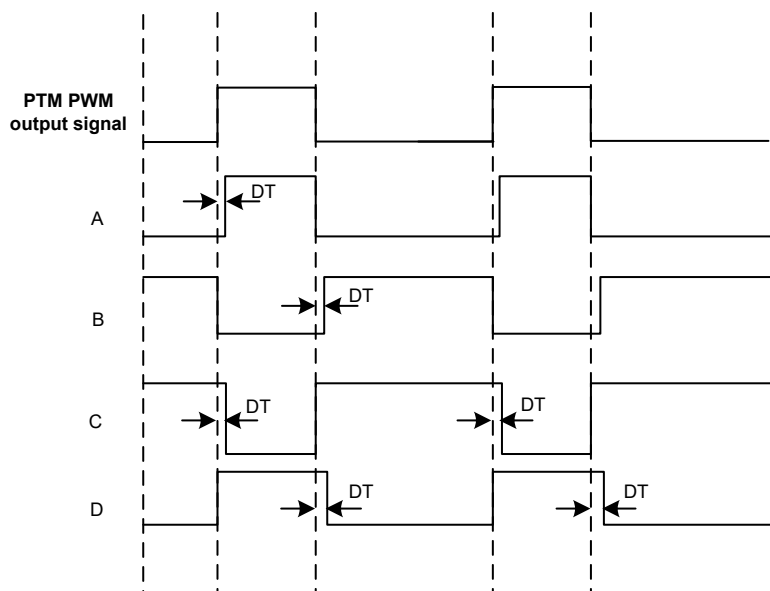
The devices provide a complementary output pair of signals which can be used as a PWM driver signal. The PWM signal is sourced from the PTM PWM output which is an active high signal. A dead time will be inserted into the PTM PWM output signals to prevent excessive DC currents. In addition to register configuration, the complementary PWM output can also be stopped by an OCP or OUVF condition occurrence, when such condition occurs and the corresponding control bit in the OCVPC register is enabled, the PWM output will stop and the PWM output pair status will be forced to certain level determined by the PWMHOPS and PWMLOPS bits.



Complementary PWM Output with Dead Time Block Diagram

## Dead Time Insertion

The complementary PWM output circuit provides a dead time insertion function. By setting the DTEN bit in the CPR register, the dead time generator and prescaler will be enabled. The clock source of the prescaler originates from the internal clock  $f_H$  and the division ratio is determined by the DTCK[1:0] bits. When the related register bits are properly configured, a dead time, which is programmable using the DT[2:0] bits in the CPR register, will be inserted to prevent excessive DC currents. The dead time will be inserted whenever the rising edge of the dead time generator input signal, namely the PTM PWM output signal, occurs.



Complementary PWM Output with Dead Time Control

## Complementary PWM Registers

The complementary PWM output function can be controlled using internal registers. The CPR register is used to control the dead time function enable/disable, PWMH/PWML inverse signal selection, dead time prescaler selection and dead time selection. The OCVPC register is used to control the protection circuit and determine the PWM output pair status when the complementary PWM output circuit is stopped.

### • CPR Register

| Bit  | 7    | 6     | 5     | 4     | 3     | 2   | 1   | 0   |
|------|------|-------|-------|-------|-------|-----|-----|-----|
| Name | DTEN | PWMHN | PWMLN | DTCK1 | DTCK0 | DT2 | DT1 | DT0 |
| 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      **DTEN**: Dead Time On/Off control  
0: Dead time & prescaler off  
1: Dead time & Prescaler on  
When this bit is cleared to zero, the PWMH and PWML status are determined by the PWMHOPS and PWMLOPS bits respectively.
- Bit 6      **PWMHN**: PWMH inverse signal selection  
0: PWMH=A  
1: PWMH=C
- Bit 5      **PWMLN**: PWML inverse signal selection  
0: PWML=B  
1: PWML=D
- Bit 4~3    **DTCK1~DTCK0**: Dead time prescaler selection  
00:  $f_D = f_H$   
01:  $f_D = f_H/2$   
10:  $f_D = f_H/4$   
11:  $f_D = f_H/8$
- Bit 2~0    **DT2~DT0**: Dead time selection  
000:  $[(1/f_D) - (1/f_H)] \sim (1/f_D)$   
001:  $[(2/f_D) - (1/f_H)] \sim (2/f_D)$   
010:  $[(3/f_D) - (1/f_H)] \sim (3/f_D)$   
011:  $[(4/f_D) - (1/f_H)] \sim (4/f_D)$   
100:  $[(5/f_D) - (1/f_H)] \sim (5/f_D)$   
101:  $[(6/f_D) - (1/f_H)] \sim (6/f_D)$   
110:  $[(7/f_D) - (1/f_H)] \sim (7/f_D)$   
111:  $[(8/f_D) - (1/f_H)] \sim (8/f_D)$   
Note:  $t_D = 1/f_D$ .

### • OCVPC Register

| Bit  | 7 | 6 | 5 | 4 | 3       | 2       | 1       | 0       |
|------|---|---|---|---|---------|---------|---------|---------|
| Name | — | — | — | — | PWMHOPS | PWMLOPS | PWMOCEN | PWMOVEN |
| R/W  | — | — | — | — | R/W     | R/W     | R/W     | R/W     |
| POR  | — | — | — | — | 1       | 0       | 0       | 0       |

- Bit 7~4    Unimplemented, read as “0”
- Bit 3      **PWMHOPS**: PWMH output status when complementary PWM output is stopped  
0: Output 0  
1: Output 1  
When the complementary PWM output circuit is stopped by clearing the DTEN bit to zero, or by an OUVF or OCP condition occurrence, the PWMH output status will be forced to output 1 if the PWMHOPS bit is set to “1”, otherwise the output status will

- be forced to output 0 if this bit is cleared to “0”. Note that configuring this bit has no effect when the complementary PWM output circuit is in normal operation.
- Bit 2 **PWMLOPS**: PWML output status when complementary PWM output is stopped  
 0: Output 0  
 1: Output 1  
 When the complementary PWM output circuit is stopped by clearing the DTEN bit to zero, or by an OUVF or OCP condition occurrence, the PWML output status will be forced to output 1 if the PWMLOPS bit is set, otherwise the output status will be forced to output 0 if this bit is cleared to zero. Note that configuring this bit has no effect when the complementary PWM output circuit is in normal operation.
- Bit 1 **PWMOCEN**: PWM over current protection enable control  
 0: Disable  
 1: Enable  
 This bit is used to determine if an OCP condition occurrence will affect the PWM output circuit. If an OCP condition occurs and this bit is set, the DTEN will be automatically cleared to zero by hardware to stop the complementary PWM output circuit. In this case, the PWMH and PWML status will be forced to a fixed high or low level determined by the PWMHOPS and PWMLOPS bits.
- Bit 0 **PWMOVEN**: PWM over voltage protection enable control  
 0: Disable  
 1: Enable  
 This bit is used to determine if an OVP condition occurrence will affect the PWM output circuit. If an OUVF condition occurs and this bit is set, the DTEN will be automatically cleared to zero by hardware to stop the complementary PWM output circuit. In this case, the PWMH and PWML status will be forced to a fixed high or low level determined by the PWMHOPS and PWMLOPS bits.

## 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.

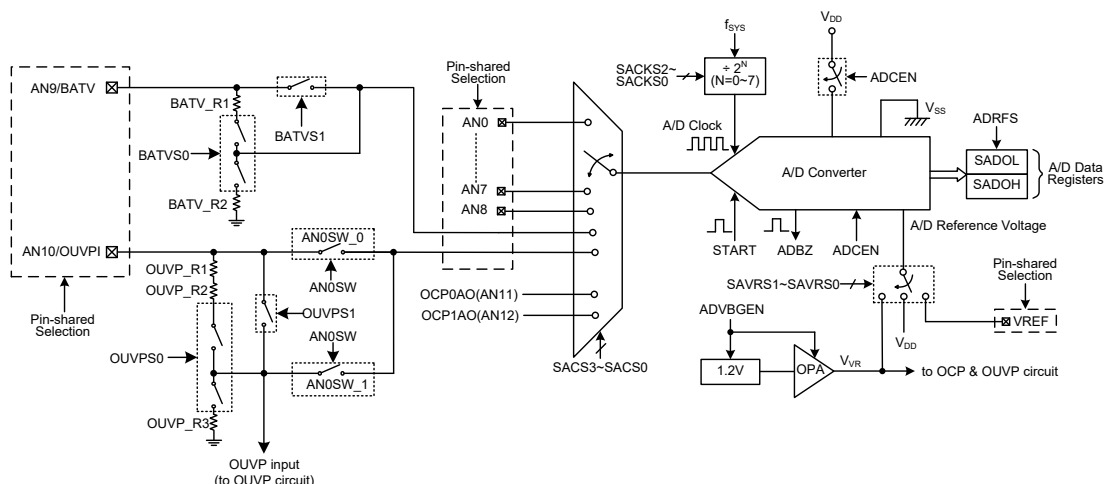
### A/D Converter Overview

The devices contain a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals, or the internal analog signal, such as the OCPn OPA output signal, and convert these signals directly into a 12-bit digital value.

The external or internal analog signal to be converted is determined by the SACS3~SACS0 bits. When the external analog signal channel, AN9 or AN10, is to be converted, the SWS0 register bits together with the SACS3~SACS0 bits should be properly configured to select the required input signals. More detailed information about the A/D input signal selection 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 Selection Bits |
|-------------------------|-----------------|----------------------------|
| 11: AN0~AN10            | OCP0AO, OCP1AO  | SACS3~SACS0                |

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



Note: The A/D Converter OPA output VVR can be used as the DAC reference input of the internal OCPn and OUVp functions.

#### A/D Converter Structure

#### A/D Converter Register Description

Overall operation of the A/D converter is controlled using several 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 configure the operating and control function of the A/D converter. The SWS0 register is used to control the input voltage division circuit.

| Register Name   | Bit   |      |         |        |        |        |        |        |
|-----------------|-------|------|---------|--------|--------|--------|--------|--------|
|                 | 7     | 6    | 5       | 4      | 3      | 2      | 1      | 0      |
| SADOL (ADRFs=0) | D3    | D2   | D1      | D0     | —      | —      | —      | —      |
| SADOL (ADRFs=1) | D7    | D6   | D5      | D4     | D3     | D2     | D1     | D0     |
| SADOH (ADRFs=0) | D11   | D10  | D9      | D8     | D7     | D6     | D5     | D4     |
| SADOH (ADRFs=1) | —     | —    | —       | —      | D11    | D10    | D9     | D8     |
| SADC0           | START | ADBZ | ADCEN   | ADRFs  | SACS3  | SACS2  | SACS1  | SACS0  |
| SADC1           | —     | —    | ADVBGEN | SAVRS1 | SAVRS0 | SACKS2 | SACKS1 | SACKS0 |
| SWS0            | —     | —    | —       | BATVS1 | BATVS0 | AN10SW | OUVPS1 | OUVPS0 |

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 ADRFS bit in the SADC0 register, as shown in the accompanying table. D0~D11 are the conversion result data bits. Any unused bits will be read as zero. Note that A/D data registers contents will be cleared 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, SWS0

To control the function and operation of the A/D converter, three control registers known as SADC0, SADC1 and SWS0 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, etc.. As each device contains only one actual analog to digital converter hardware circuit, each of the analog signal inputs must be routed to the converter. The SACS3~SACS0 bits in the SADC0 register are used to determine which analog input signal is selected to be converted.

The AN9 and AN10 input channel each has an integrated voltage divider circuit which can be connected or disconnected using the SWS0 register control the internal analog switches.

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 | ADRF5 | 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 A/D conversion  
This bit is used to initiate an A/D 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 1 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's 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 cleared to zero.
- Bit 4 **ADRF5**: A/D converter data format selection  
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 analog input channel selection  
 0000: AN0  
 0001: AN1  
 0010: AN2  
 0011: AN3  
 0100: AN4  
 0101: AN5  
 0110: AN6  
 0111: AN7  
 1000: AN8  
 1001: AN9  
 1010: AN10  
 1011: AN11 – from internal OCP0 OPA output  
 1100: AN12 – from internal OCP1 OPA output  
 1101~1111: Undefined, input floating

• **SADC1 Register**

| Bit  | 7 | 6 | 5       | 4      | 3      | 2      | 1      | 0      |
|------|---|---|---------|--------|--------|--------|--------|--------|
| Name | — | — | ADVBGEN | SAVRS1 | SAVRS0 | SACKS2 | SACKS1 | SACKS0 |
| R/W  | — | — | R/W     | R/W    | R/W    | R/W    | R/W    | R/W    |
| POR  | — | — | 0       | 0      | 0      | 0      | 0      | 0      |

- Bit 7~6     Unimplemented, read as “0”
- Bit 5     **ADVBGEN**: A/D converter internal 1.2V bandgap and OPA (Gain=2) enable control  
 0: Disable  
 1: Enable
- Bit 4~3     **SAVRS1~SAVRS0**: A/D converter reference voltage selection  
 00/11: Internal A/D converter power supply,  $V_{DD}$   
 01: External VREF pin  
 10: Internal OPA output voltage,  $V_{VR}$   
 These bits are used to select the A/D converter reference voltage source. The  $V_{VR}$  is the A/D converter internal OPA output voltage. It should be noted that when the internal reference voltage source is selected, the reference voltage derived from the external VREF pin will automatically be switched off.
- Bit 2~0     **SACKS2~SACKS0**: A/D conversion clock source selection  
 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.

• **SWS0 Register**

| Bit  | 7 | 6 | 5 | 4      | 3      | 2      | 1      | 0      |
|------|---|---|---|--------|--------|--------|--------|--------|
| Name | — | — | — | BATVS1 | BATVS0 | AN10SW | OUVPS1 | OUVPS0 |
| 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     **BATVS1**: Integrated voltage division resistor bypass control  
 0: BATVS1 disable  
 1: BATVS1 enable

- This bit will only be enabled when the AN9 channel is selected by the SACS3~SACS0 bits and the BATVS1 bit is set to 1, otherwise this bit will be disabled. When this bit is enabled, the input signal will bypass the integrated voltage division circuit.
- Bit 3 **BATVS0**: Integrated voltage division resistor BATV\_R1 and BATV\_R2 control  
 0: BATVS0 disable  
 1: BATVS0 enable
- This bit will only be enabled when the AN9 channel is selected by the SACS3~SACS0 bits and the BATVS[1:0] is set to “01B”, otherwise this bit will be disabled. When this bit is enabled, the input signal will pass through the integrated voltage division circuit to obtain a divided input voltage.
- Bit 2 **AN10SW**: AN0 input selection  
 0: AN10SW\_0 on and AN10SW\_1 off  
 1: AN10SW\_0 off and AN10SW\_1 on
- This bit only takes effect when the AN10 channel is selected by the SACS3~SACS0 bits, otherwise the two switches will both be off.
- Bit 1 **OUVPS1**: Integrated voltage division resistor bypass control  
 0: OUVPS1 disable  
 1: OUVPS1 enable
- This bit will only be enabled when the AN10 function is selected using the corresponding pin-shared control bits and the OUVPS1 bit is set to 1, otherwise this bit will be disabled.
- Bit 0 **OUVPS0**: Integrated voltage division resistor OUV\_P\_R2 and OUV\_P\_R3 control  
 0: OUVPS0 disable  
 1: OUVPS0 enable
- This bit will only be enabled when the AN10 function is selected using the corresponding pin-shared control bits and the OUVPS[1:0] is set to “01B”, otherwise this bit will be disabled.

## A/D Converter Reference Voltage

The reference voltage supply to the A/D converter can be supplied from the internal A/D converter power supply voltage,  $V_{DD}$ , or internal operational amplifier output voltage,  $V_{VR}$ , or from an external reference source supplied on pin VREF. The desired selection is made using the SAVRS1~SAVRS0 bits. When the SAVRS bit field is set to “00” or “11”, the A/D converter reference voltage will come from the power supply voltage,  $V_{DD}$ . When the SAVRS bit field is set to “10”, the A/D converter reference voltage will come from the internal operational amplifier output voltage,  $V_{VR}$ . Otherwise, if the SAVRS bit field is set to “01”, the A/D converter reference voltage will come from the VREF pin. As the VREF pin is pin-shared with other functions, when the VREF pin is selected as the reference voltage supply pin, the VREF pin-shared function control bit should be properly configured to disable other pin functions. However, if the internal reference signal is selected as the reference voltage, the external reference input from the VREF pin will automatically be switched off by hardware. The analog input values must not be allowed to exceed the selected reference voltage.

| SAVRS[1:0] | Reference Source | Description                                            |
|------------|------------------|--------------------------------------------------------|
| 00, 11     | $V_{DD}$         | From internal A/D converter power supply voltage       |
| 01         | VREF pin         | From external A/D converter reference voltage pin VREF |
| 10         | $V_{VR}$         | From internal operational amplifier output voltage     |

**A/D Converter Reference Voltage Selection**

## A/D Converter Input Signals

All the external A/D converter 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 converter external input pin in the pin-shared function selection register determine whether the input pins are set as A/D converter



analog inputs or whether they have other functions. If the pin is set to be as an A/D converter 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 set through register programming, will be automatically disconnected if the pins are set as A/D inputs. Note that it is not necessary to first set 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 SACS3~SACS0 bits are set to any value except “1011~1100”, the external analog channel input is selected to be converted. If the SACS3~SACS0 bits are set to “1011~1100”, the internal signal OCPnAO is selected to be converted. It should be noted that the AN9 and AN10 channels each has an integrated voltage division circuit to prevent the input signal from exceeding the selected reference voltage. Properly configuring the BATVS0 and BATVS1 bits in the SWS0 register can determine whether the input signal will pass through the integrated voltage division circuit on the AN9 channel or not. For the AN10 channel, the input signal will pass through the integrated voltage division circuit on the AN10 channel by clearing the OUVPS1 bit to zero and setting both the OUVPS0 and AN10SW bits to one. Users can also set the OUVPS1 to high and clear the OUVPS0 and AN10SW bits to zero and use external resistors to achieve voltage division. Special attention must be taken to the configuration of these bits, otherwise it will result in an unnormal operation of the AN10 and OUVPS input. Refer to the “A/D Converter Structure” and “A/D Converter Register Description” for more detailed information.

| SACS[3:0] | Input Signals | Description                                                                |
|-----------|---------------|----------------------------------------------------------------------------|
| 0000~1000 | AN0~AN8       | External channel analog input ANn                                          |
| 1001~1010 | AN9, AN10     | External channel analog input ANn with integrated voltage division circuit |
| 1011~1100 | OCPnAO        | Internal signal derived from the OCPn OPA output                           |

**A/D Converter Input Signal Selection**

## A/D Converter Operation

The START bit in the SADC0 register is used to start the A/D 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 associated 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 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 $\mu$ s to 10 $\mu$ s, care must be taken for system clock frequencies. For example, if 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 greater than the maximum A/D clock period, which may result in inaccurate A/D conversion values. Refer to the following table for examples, special care must be taken to values marked with an asterisk \*, as these values may be less or greater than the specified A/D clock period.

| $f_{sys}$ | A/D Clock Period ( $t_{ADCK}$ )     |                                       |                                       |                                       |                                        |                                        |                                        |                                         |
|-----------|-------------------------------------|---------------------------------------|---------------------------------------|---------------------------------------|----------------------------------------|----------------------------------------|----------------------------------------|-----------------------------------------|
|           | SACKS[2:0]<br>=000<br>( $f_{sys}$ ) | SACKS[2:0]<br>=001<br>( $f_{sys}/2$ ) | SACKS[2:0]<br>=010<br>( $f_{sys}/4$ ) | SACKS[2:0]<br>=011<br>( $f_{sys}/8$ ) | SACKS[2:0]<br>=100<br>( $f_{sys}/16$ ) | SACKS[2:0]<br>=101<br>( $f_{sys}/32$ ) | SACKS[2:0]<br>=110<br>( $f_{sys}/64$ ) | SACKS[2:0]<br>=111<br>( $f_{sys}/128$ ) |
| 1MHz      | 1 $\mu$ s                           | 2 $\mu$ s                             | 4 $\mu$ s                             | 8 $\mu$ s                             | 16 $\mu$ s *                           | 32 $\mu$ s *                           | 64 $\mu$ s *                           | 128 $\mu$ s *                           |
| 2MHz      | 500ns                               | 1 $\mu$ s                             | 2 $\mu$ s                             | 4 $\mu$ s                             | 8 $\mu$ s                              | 16 $\mu$ s *                           | 32 $\mu$ s *                           | 64 $\mu$ s *                            |
| 4MHz      | 250ns *                             | 500ns                                 | 1 $\mu$ s                             | 2 $\mu$ s                             | 4 $\mu$ s                              | 8 $\mu$ s                              | 16 $\mu$ s *                           | 32 $\mu$ s *                            |
| 8MHz      | 125ns *                             | 250ns *                               | 500ns                                 | 1 $\mu$ s                             | 2 $\mu$ s                              | 4 $\mu$ s                              | 8 $\mu$ s                              | 16 $\mu$ s *                            |
| 12MHz     | 83ns *                              | 167ns *                               | 333ns *                               | 667ns                                 | 1.33 $\mu$ s                           | 2.67 $\mu$ s                           | 5.33 $\mu$ s                           | 10.67 $\mu$ s *                         |

**A/D Clock Period Examples**

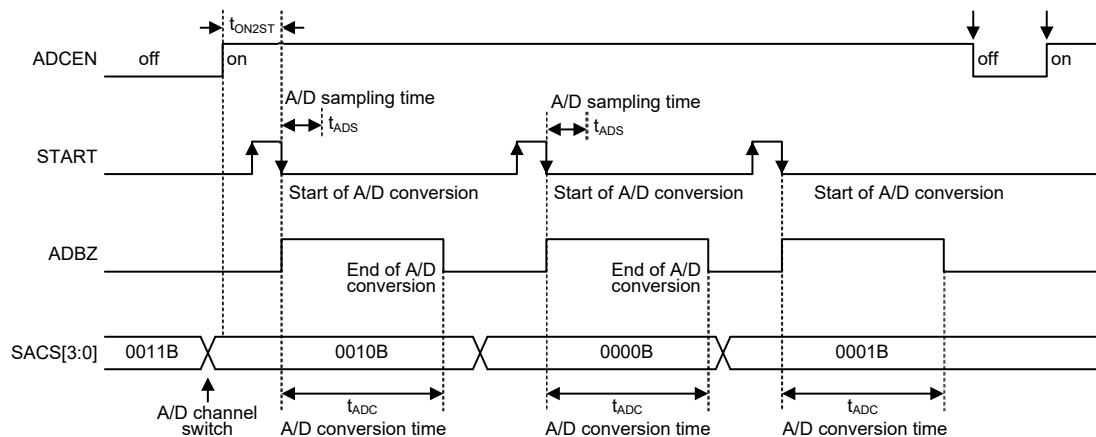
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry a certain delay, as indicated in the 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 cycles and the data conversion takes 12 A/D clock cycles. Therefore a total of 16 A/D clock cycles for an external input A/D conversion which is defined as  $t_{ADC}$  are necessary.

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

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 cycles where  $t_{ADCK}$  is equal to the A/D clock period.



**A/D Conversion Timing**

## 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 converter by setting the ADCEN bit in the SADC0 register to “1”.
- Step 3  
Select which signal is to be connected to the internal A/D converter by correctly configuring the SACS3~SACS0 bits in the SADC0 register.
- Step 4  
Select the reference voltage source by configuring the SAVRS1~SAVRS0 bits in the SADC1 register. If the A/D converter power supply voltage or the operational amplifier output voltage is selected, the external reference voltage input will be automatically switched off.
- Step 5  
Select A/D converter output data format by configuring the ADRFS bit in the SADC0 register.
- Step 6  
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 converter interrupt control bit, ADE, must both be set high in advance.
- Step 7  
The A/D conversion procedure can now be initiated by setting the START bit from low to high and then low again.
- Step 8  
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

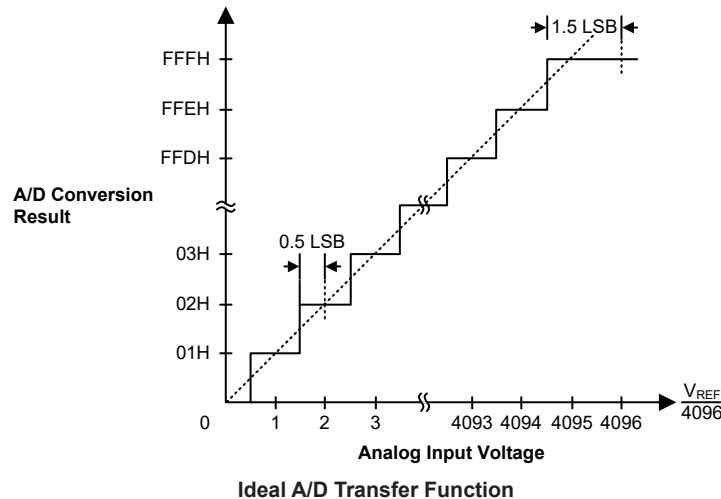
As the devices contain a 12-bit A/D converter, its full-scale converted digitised value is equal to FFFH. Since the full-scale analog input value is equal to the actual A/D converter reference voltage,  $V_{REF}$ , this gives a single bit analog input value of  $V_{REF}$  divided by 4096.

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

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 configure 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              ; select fsys/8 as A/D clock and
mov SADC1,a            ; select internal reference voltage VDD
mov a,01h              ; set PDS1 to configure pin AN0
mov PDS1,a
mov a,20h
mov SADC0,a            ; enable A/D converter 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              ; select fsys/8 as A/D clock and
mov SADC1,a            ; select internal reference voltage VDD
mov a,01h              ; set PDS1 to configure pin AN0
mov PDS1,a
mov a,20h
mov SADC0,a            ; enable A/D converter 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_ISR:               ; ADC interrupt service routine
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 devices contain a Universal Serial Interface Module, which includes the four-line SPI interface, the two-line I<sup>2</sup>C interface and the two-line/single-wire 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, I<sup>2</sup>C or UART based hardware such as sensors or Flash 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 I<sup>2</sup>C type is used is made using the UART mode selection bit, named UMD, and the SPI/I<sup>2</sup>C 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.

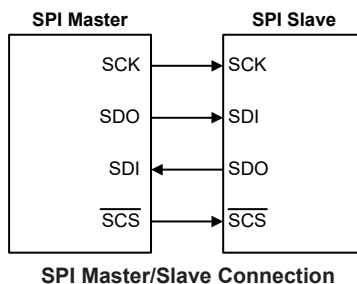
### SPI Interface

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 devices 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  $\overline{\text{SCS}}$  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  $\overline{\text{SCS}}$ . Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, the SCK pin is the Serial Clock line and  $\overline{\text{SCS}}$  is the Slave Select line. As the SPI interface pins are pin-shared with normal I/O pins and with the I<sup>2</sup>C/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  $\overline{\text{SCS}}$  pin only one slave device can be utilized. The  $\overline{\text{SCS}}$  pin is controlled by software, set CSEN bit to 1 to enable  $\overline{\text{SCS}}$  pin function, set CSEN bit to 0 the  $\overline{\text{SCS}}$  pin will be floating state.

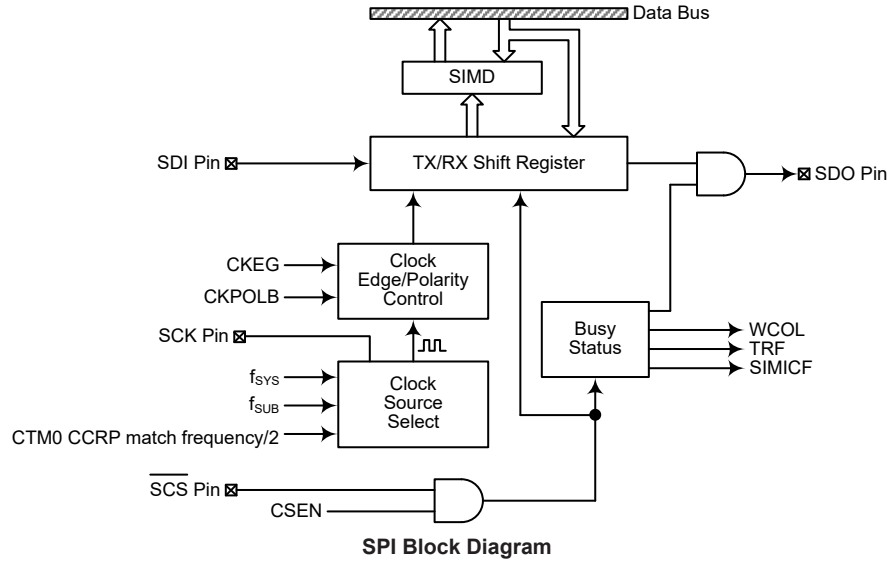


The SPI function offers the following features:

- 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 I<sup>2</sup>C 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/I<sup>2</sup>C 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/I<sup>2</sup>C 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 CTM0 CCRP match frequency/2
- 101: SPI slave mode
- 110: I<sup>2</sup>C slave mode
- 111: Unused mode

When the UMD bit is cleared to zero, these bits setup the SPI or I<sup>2</sup>C operating mode of the USIM function. As well as selecting if the I<sup>2</sup>C 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 CTM0 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 I<sup>2</sup>C mode
- 1: UART mode

This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I<sup>2</sup>C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be cleared to zero for SPI or I<sup>2</sup>C mode.

Bit 3~2 **SIMDEB1~SIMDEB0**: I<sup>2</sup>C Debounce Time Selection

These bits are only available when the USIM is configured to operate in the I<sup>2</sup>C mode. Refer to the I<sup>2</sup>C register section.

Bit 1 **SIMEN**: USIM SPI/I<sup>2</sup>C Enable Control

- 0: Disable
- 1: Enable

The bit is the overall on/off control for the USIM SPI/I<sup>2</sup>C interface. When the SIMEN bit is cleared to zero to disable the USIM SPI/I<sup>2</sup>C interface, the SDI, SDO, SCK and  $\overline{SCS}$ , or SDA and SCL lines will lose their SPI or I<sup>2</sup>C function and the USIM operating current will be reduced to a minimum value. When the bit is high the USIM SPI/I<sup>2</sup>C 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 I<sup>2</sup>C interface via the UMD and SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I<sup>2</sup>C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I<sup>2</sup>C 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 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 active clock 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.

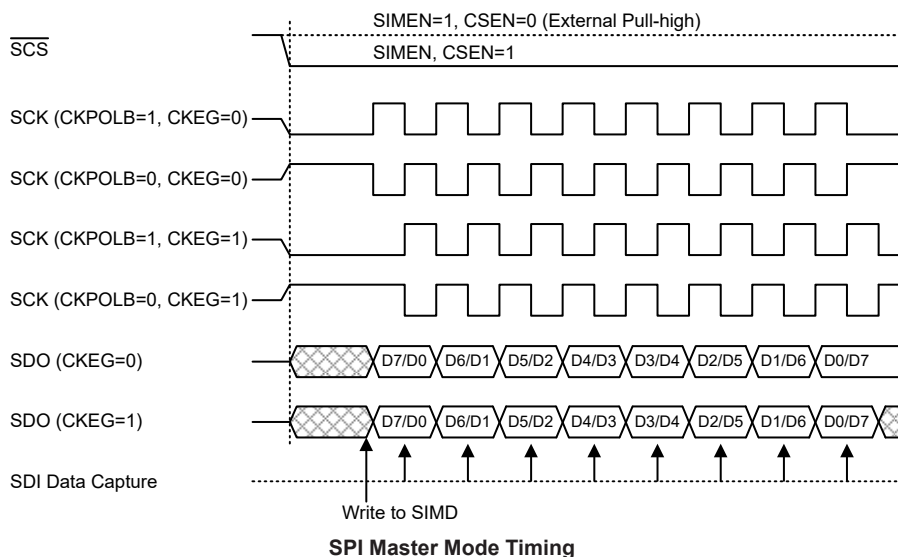
- Bit 3 **MLS**: SPI data shift order  
 0: LSB first  
 1: MSB first  
 This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

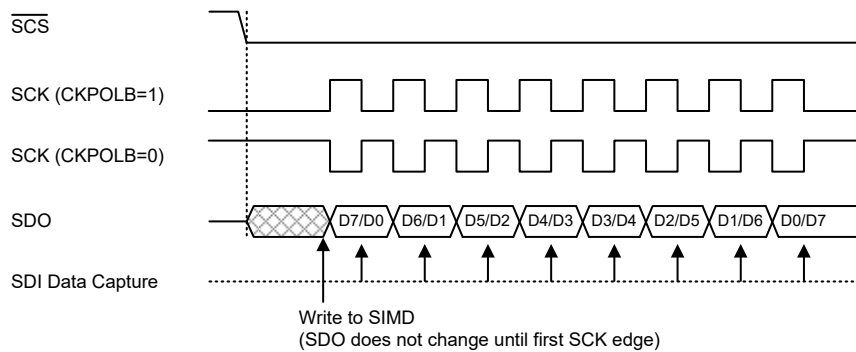
|       |                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Bit 2 | <p><b>CSEN</b>: SPI <math>\overline{\text{SCS}}</math> pin control</p> <p>0: Disable</p> <p>1: Enable</p> <p>The CSEN bit is used as an enable/disable for the <math>\overline{\text{SCS}}</math> pin. If this bit is low, then the <math>\overline{\text{SCS}}</math> pin will be disabled and placed into a floating condition. If the bit is high the <math>\overline{\text{SCS}}</math> pin will be enabled and used as a select pin.</p> |
| Bit 1 | <p><b>WCOL</b>: SPI write collision flag</p> <p>0: No collision</p> <p>1: Collision</p> <p>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.</p>               |
| Bit 0 | <p><b>TRF</b>: SPI Transmit/Receive complete flag</p> <p>0: SPI data is being transferred</p> <p>1: SPI data transmission is completed</p> <p>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.</p>                                                                         |

### SPI Communication

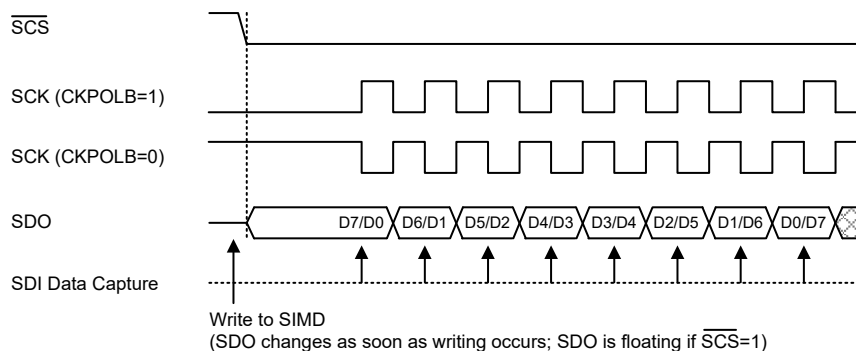
After the SPI interface is enabled by setting the SIMEN bit high, then in the Master Mode, when data is written to the SIMD register, transmission/reception will begin simultaneously. When the data transfer is completed, the TRF flag will be set high automatically, but must be cleared using the application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

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



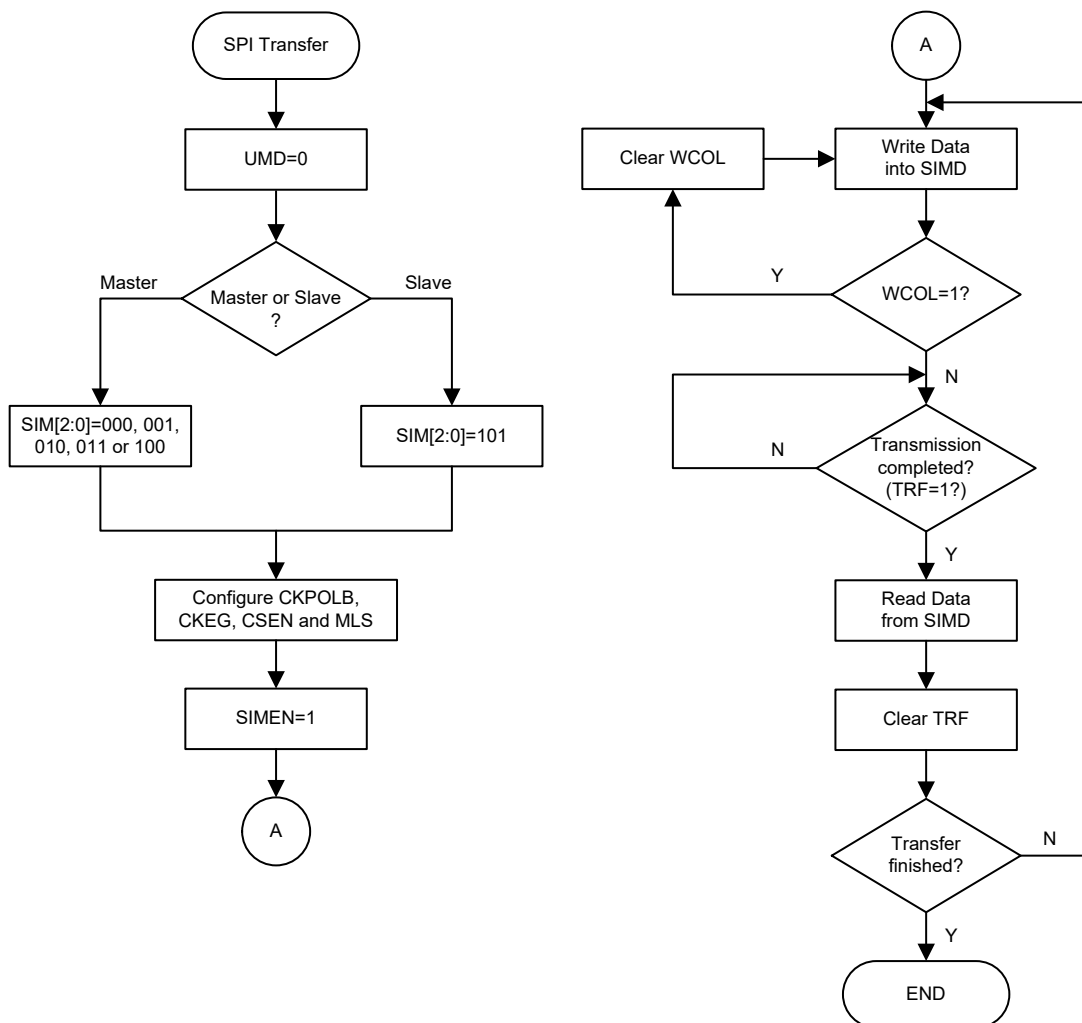


**SPI Slave Mode Timing – CKEG=0**



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  $\overline{SCS}$  pin 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.

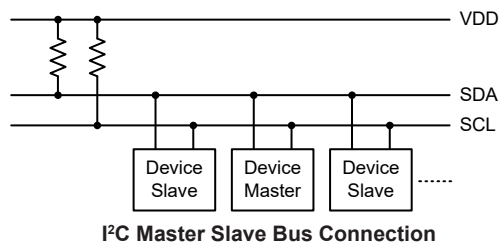
- Step 5  
Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6  
Check the TRF bit or wait for an USIM SPI serial bus interrupt.
- Step 7  
Read data from the SIMD register.
- Step 8  
Clear TRF.
- Step 9  
Go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

### I<sup>2</sup>C Interface

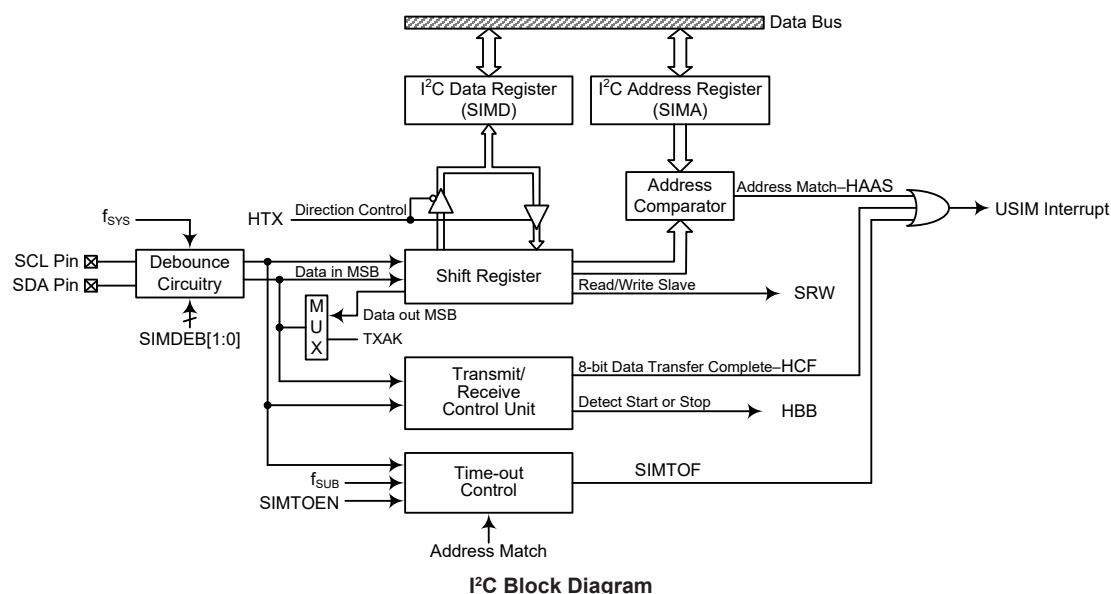
The I<sup>2</sup>C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two-line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple communication protocol and the ability to accommodate multiple devices on the same bus has made it an extremely popular interface type for many applications.



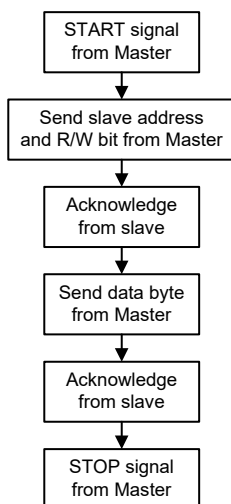
### I<sup>2</sup>C Interface Operation

The I<sup>2</sup>C 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 I<sup>2</sup>C bus is identified by a unique address which will be transmitted and received on the I<sup>2</sup>C bus.

When two devices communicate with each other on the bidirectional I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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<sub>sys</sub>**, and the I²C debounce time. For either the I²C Standard or Fast mode operation, users must take care of the selected system clock frequency and the configured debounce time to match the criterion shown in the following table.

| I²C Debounce Time Selection | I²C Standard Mode (100kHz) | I²C Fast Mode (400kHz)   |
|-----------------------------|----------------------------|--------------------------|
| No Debounce                 | $f_{sys} > 2\text{MHz}$    | $f_{sys} > 5\text{MHz}$  |
| 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<sub>sys</sub>** Frequency Requirements**

## I<sup>2</sup>C Registers

There are three control registers associated with the I<sup>2</sup>C 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 I<sup>2</sup>C 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 |

I<sup>2</sup>C Register List

## I<sup>2</sup>C 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 I<sup>2</sup>C functions. Before the device writes data to the I<sup>2</sup>C bus, the actual data to be transmitted must be placed in the SIMD register. After the data is received from the I<sup>2</sup>C bus, the device can read it from the SIMD register. Any transmission or reception of data from the I<sup>2</sup>C 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/I<sup>2</sup>C data register bit 7 ~ bit 0

## I<sup>2</sup>C Address Register

The SIMA register is also used by the SPI interface but has the name SIMC2. The SIMA register is the location where the 7-bit slave address of the slave device is stored. 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 I<sup>2</sup>C 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:** I<sup>2</sup>C slave address  
SIMA6~SIMA0 is the 7-bit I<sup>2</sup>C slave address.

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

## I<sup>2</sup>C Control Registers

There are three control registers for the I<sup>2</sup>C 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 I<sup>2</sup>C communication status. Another register, SIMTOC, is used to control the I<sup>2</sup>C 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/I<sup>2</sup>C 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 CTM0 CCRP match frequency/2
- 101: SPI slave mode
- 110: I<sup>2</sup>C slave mode
- 111: Unused mode

When the UMD bit is cleared to zero, these bits setup the SPI or I<sup>2</sup>C operating mode of the USIM function. As well as selecting if the I<sup>2</sup>C 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 CTM0 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 I<sup>2</sup>C mode
- 1: UART mode

This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I<sup>2</sup>C mode can be selected using the SIM2~SIM0 bits. Note that the UMD bit must be set low for SPI or I<sup>2</sup>C mode.

Bit 3~2

**SIMDEB1~SIMDEB0:** I<sup>2</sup>C Debounce Time Selection

- 00: No debounce
- 01: 2 system clock debounce
- 10: 4 system clock debounce
- 11: 4 system clock debounce

These bits are used to select the I<sup>2</sup>C debounce time when the USIM is configured as the I<sup>2</sup>C interface function by setting the UMD bit to "0" and SIM2~SIM0 bits to "110".

Bit 1

**SIMEN:** USIM SPI/I<sup>2</sup>C Enable Control

- 0: Disable
- 1: Enable

The bit is the overall on/off control for the USIM SPI/I<sup>2</sup>C interface. When the SIMEN bit is cleared to zero to disable the USIM SPI/I<sup>2</sup>C interface, the SDI, SDO, SCK and  $\overline{SCS}$ , or SDA and SCL lines will lose their SPI or I<sup>2</sup>C function and the USIM operating current will be reduced to a minimum value. When the bit is high the USIM SPI/I<sup>2</sup>C 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 I<sup>2</sup>C interface via the UMD and SIM2~SIM0 bits and the SIMEN bit changes from low to high, the contents of the I<sup>2</sup>C control bits such as HTX and TXAK will remain at the previous settings and should therefore be first initialised by the application program while the relevant I<sup>2</sup>C 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**: I<sup>2</sup>C 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**: I<sup>2</sup>C 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**: I<sup>2</sup>C Bus busy flag  
 0: I<sup>2</sup>C Bus is not busy  
 1: I<sup>2</sup>C Bus is busy  
 The HBB flag is the I<sup>2</sup>C busy flag. This flag will be “1” when the I<sup>2</sup>C 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**: I<sup>2</sup>C slave device is transmitter or receiver selection  
 0: Slave device is the receiver  
 1: Slave device is the transmitter

Bit 3 **TXAK**: I<sup>2</sup>C 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**: I<sup>2</sup>C 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 I<sup>2</sup>C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I<sup>2</sup>C 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**: I<sup>2</sup>C Address Match Wake-up control  
 0: Disable  
 1: Enable  
 This bit should be set high to enable the I<sup>2</sup>C 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 I<sup>2</sup>C address match wake up, then this bit must be cleared to zero by the application program after wake-up to ensure correction device operation.

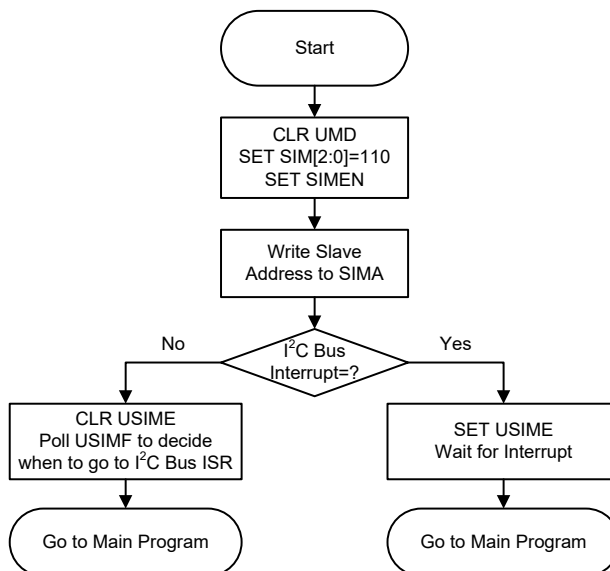
- Bit 0      **RXAK:** I<sup>2</sup>C Bus Receive acknowledge flag  
             0: Slave receive acknowledge flag  
             1: Slave does not receive acknowledge flag

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 I<sup>2</sup>C Bus.

## I<sup>2</sup>C Bus Communication

Communication on the I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C bus.
- Step 2  
Write the slave address of the device to the I<sup>2</sup>C bus address register SIMA.
- Step 3  
Set the USIME interrupt enable bit of the interrupt control register to enable the USIM interrupt.



**I<sup>2</sup>C Bus Initialisation Flow Chart**

### **I<sup>2</sup>C Bus Start Signal**

The START signal can only be generated by the master device connected to the I<sup>2</sup>C bus and not by the slave device. This START signal will be detected by all devices connected to the I<sup>2</sup>C bus. When detected, this indicates that the I<sup>2</sup>C 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.

### **I<sup>2</sup>C Slave Address**

The transmission of a START signal by the master will be detected by all devices on the I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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 I<sup>2</sup>C 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.

### **I<sup>2</sup>C Bus Read/Write Signal**

The SRW bit in the SIMC1 register defines whether the master device wishes to read data from the I<sup>2</sup>C bus or write data to the I<sup>2</sup>C 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 I<sup>2</sup>C bus, therefore the slave device must be setup to send data to the I<sup>2</sup>C bus as a transmitter. If the SRW flag is “0” then this indicates that the master wishes to send data to the I<sup>2</sup>C bus, therefore the slave device must be setup to read data from the I<sup>2</sup>C bus as a receiver.

### **I<sup>2</sup>C Bus Slave Address Acknowledge Signal**

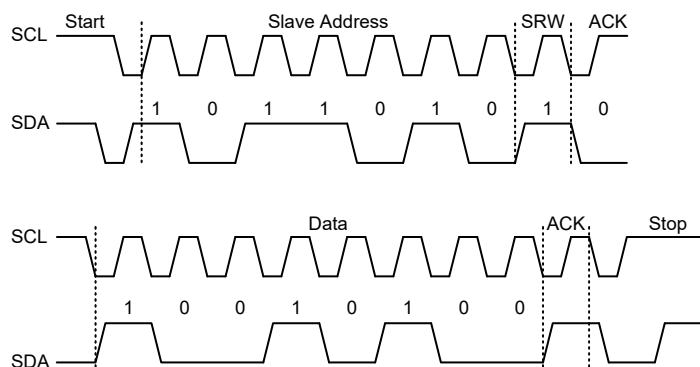
After the master has transmitted a calling address, any slave device on the I<sup>2</sup>C bus, whose own internal address matches the calling address, must generate an acknowledge signal. The acknowledge signal will inform the master that a slave device has accepted its calling address. If no acknowledge 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”.

### **I<sup>2</sup>C Bus Data and Acknowledge Signal**

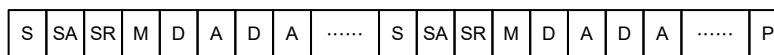
The transmitted data is 8-bit wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8 bits of data, the receiver must transmit an acknowledge signal, level “0”, before it can 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 I<sup>2</sup>C Bus. The corresponding data will be stored in the SIMD register. If setup as a transmitter, the slave device must first write the data to be transmitted into the SIMD register. If setup as a receiver, the slave device must read the transmitted data from the SIMD register.

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

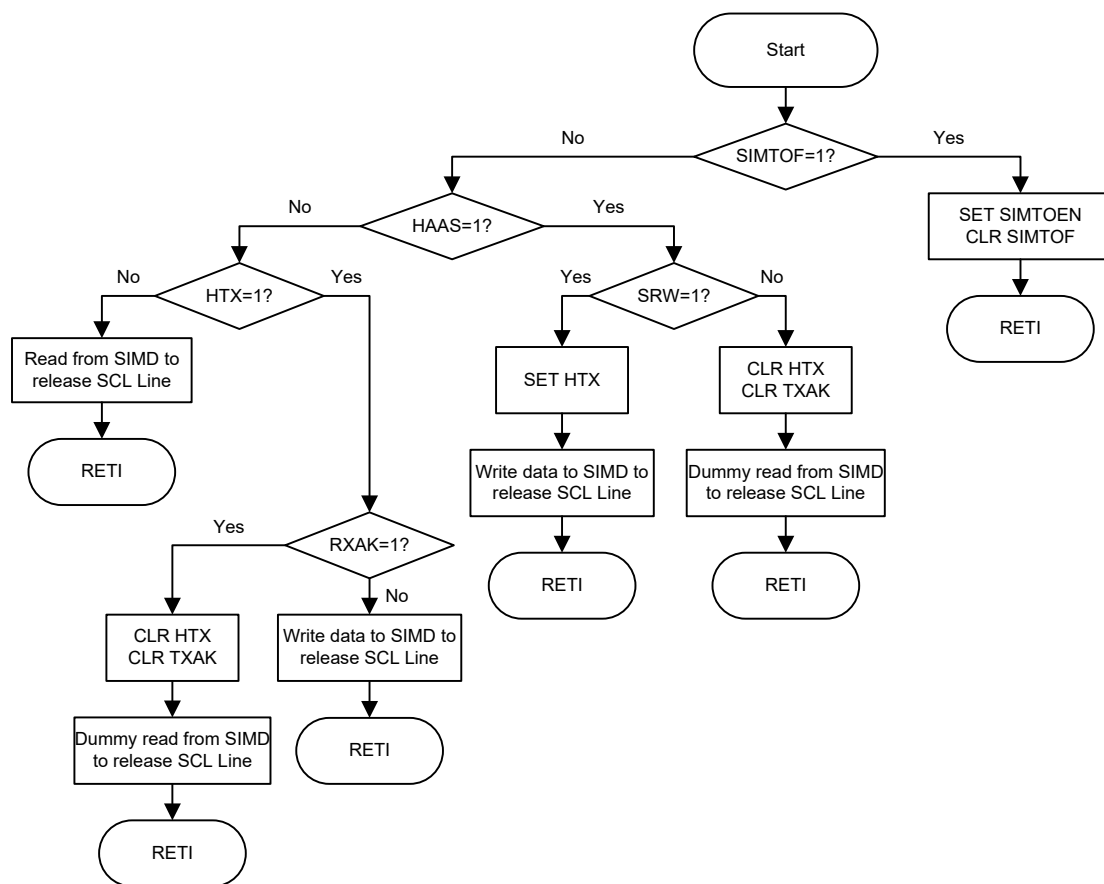


S=Start (1 bit)  
 SA=Slave Address (7 bits)  
 SR=SRW bit (1 bit)  
 M=Slave device send acknowledge bit (1 bit)  
 D=Data (8 bits)  
 A=ACK (RXAK bit for transmitter, TXAK bit for receiver, 1 bit)  
 P=Stop (1 bit)



**I<sup>2</sup>C Communication Timing Diagram**

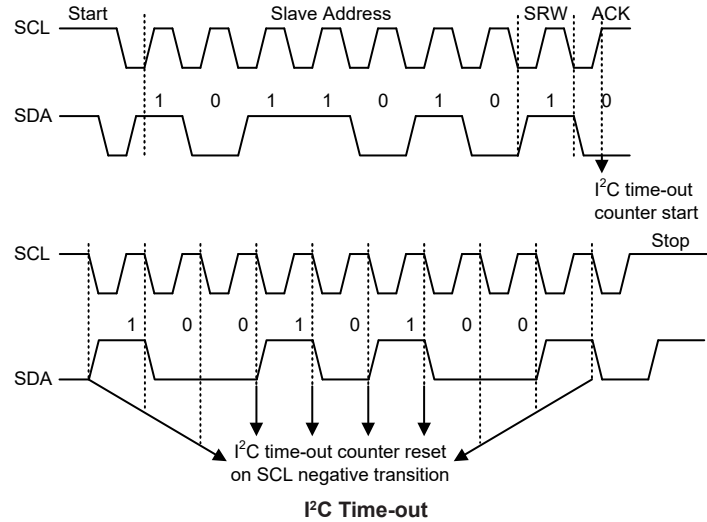
Note: When a slave address is matched, the device must be placed in either the transmit mode and then write data to the SIMD register, or in the receive mode where it must implement a dummy read from the SIMD register to release the SCL line.



**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 I<sup>2</sup>C 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 I<sup>2</sup>C time-out occurs, the I<sup>2</sup>C internal circuitry will be reset and the registers will be reset into the following condition:

| Registers         | After I <sup>2</sup> C Time-out |
|-------------------|---------------------------------|
| SIMD, SIMA, SIMC0 | No change                       |
| SIMC1             | Reset to POR condition          |

**I<sup>2</sup>C 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:  $((1 \sim 64) \times 32) / f_{SUB}$ . 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 I<sup>2</sup>C Time-out control  
0: Disable  
1: Enable

Bit 6 **SIMTOF**: USIM I<sup>2</sup>C 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.

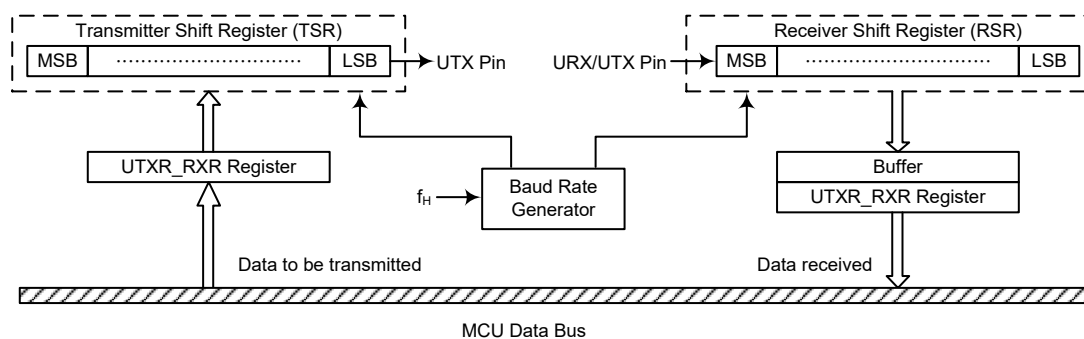
Bit 5~0 **SIMTOS5~SIMTOS0**: USIM I<sup>2</sup>C Time-out period selection  
I<sup>2</sup>C time-out clock source is  $f_{SUB}/32$ .  
I<sup>2</sup>C time-out time is equal to  $(SIMTOS[5:0]+1) \times (32/f_{SUB})$ .

## UART Interface

The devices contain an integrated full-duplex or half-duplex asynchronous serial communication 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 I<sup>2</sup>C interfaces which can be used to indicate when a reception occurs or when a transmission terminates.

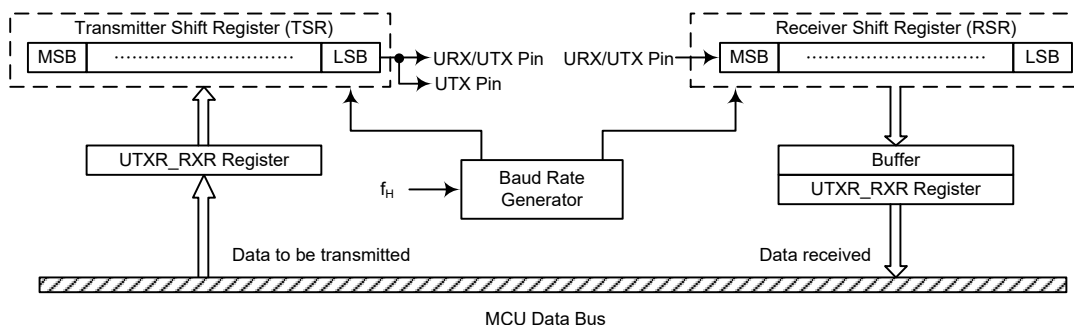
The integrated UART function contains the following features:

- Full-duplex or half-duplex (single wire mode) 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
- URX/UTX 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 – USWM=0**





**UART Data Transfer Block Diagram – USWM=1**

### UART External Pins

To communicate with an external serial interface, the internal UART has two external pins known as UTX pin and URX/UTX pin, which are pin-shared with I/O or other pin functions. The UTX and URX/UTX 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 bit, the UTXEN or URXEN bits, if set, will configure these pins to transmitter output and receiver input conditions. At this time the internal pull-high resistor related to the transmitter output pin will be disabled, while the internal pull-high resistor related to the receiver input pin is controlled by the corresponding I/O pull-high function control bit. When the UTX or URX/UTX pin function is disabled by clearing the UMD, UREN, UTXEN or URXEN bit, the UTX or URX/UTX pin will be set to a floating state. At this time whether the internal pull-high resistor is connected to the UTX or URX/UTX pin or not is determined by the corresponding I/O pull-high function control bit.

### UART Single Wire Mode

The UART function also supports a Single Wire Mode communication which is selected using the USWM bit in the UUCR3 register. When the USWM bit is set high, the UART function will be in the single wire mode. In the single wire mode, a single URX/UTX pin can be used to transmit and receive data depending upon the corresponding control bits. When the URXEN bit is set high, the URX/UTX pin is used as a receiver pin. When the URXEN bit is cleared to zero and the UTXEN bit is set high, the URX/UTX pin will act as a transmitter pin.

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

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

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

### UART Data Transfer Scheme

The UART Data Transfer 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 UTX 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 URX/UTX 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 seven control registers associated with the UART function. The UMD bit in the SIMC0 register can be used to select the UART interface. The USWM bit in the UUCR3 register is used to enable/disable the UART Single Wire 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  |
| UUCR3         | —      | —      | —      | —      | —       | —       | —      | USWM   |
| UTXR_RXR      | UTXRX7 | UTXRX6 | UTXRX5 | UTXRX4 | UTXRX3  | UTXRX2  | UTXRX1 | UTXRX0 |
| 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/I<sup>2</sup>C Operating Mode Control  
 When the UMD bit is cleared to zero, these bits setup the SPI or I<sup>2</sup>C operating mode of the USIM function. Refer to the SPI or I<sup>2</sup>C register section for more details.
- Bit 4 **UMD**: UART mode selection bit  
 0: SPI or I<sup>2</sup>C mode  
 1: UART mode  
 This bit is used to select the UART mode. When this bit is cleared to zero, the actual SPI or I<sup>2</sup>C mode can be selected using the SIM2~SIM0 bits.
- Bit 3~2 **SIMDEB1~SIMDEB0**: I<sup>2</sup>C Debounce Time Selection  
 Refer to the I<sup>2</sup>C register section.
- Bit 1 **SIMEN**: USIM SPI/I<sup>2</sup>C Enable Control  
 This bit is only available when the USIM is configured to operate in an SPI or I<sup>2</sup>C mode with the UMD bit cleared to zero. Refer to the SPI or I<sup>2</sup>C 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 an 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 URX/UTX 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 UTX 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 and UUCR3 registers are the three 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, single wire mode communication 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. UTX and URX/UTX pins are in a floating state

1: Enable UART. UTX and URX/UTX 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 URX/UTX pin as well as the UTX 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 UTX and URX/UTX pins will function as defined by the USWM mode selection bit

together with 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, UUCR3 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  
            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. 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 UTX 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 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 UTX 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 UTX 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 UTX 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 URX/UTX 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 URX/UTX 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 URX/UTX 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**: URX/UTX pin wake-up UART function enable control  
             0: URX/UTX pin wake-up UART function is disabled  
             1: URX/UTX pin wake-up UART function is enabled  
 This bit is used to control the wake-up UART function when a falling edge on the URX/UTX pin occurs. Note that this bit is only available when the UART clock ( $f_{H}$ ) is switched off. There will be no URX/UTX 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 URX/UTX pin occurs. When this request happens and the corresponding interrupt is enabled, an URX/UTX 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 URX/UTX 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.

#### • UUCR3 Register

The UUCR3 register is used to enable the UART Single Wire Mode communication. As the name suggests in the single wire mode the UART communication can be implemented in one single line, URX/UTX, together with the control of the URXEN and UTXEN bits in the UUCR2 register.

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

- Bit 7~1      Unimplemented, read as “0”
- Bit 0      **USWM**: Single Wire Mode enable control  
             0: Disable, the URX/UTX pin is used as UART receiver function only  
             1: Enable, the URX/UTX pin can be used as UART receiver or transmitter function controlled by the URXEN and UTXEN bits  
 Note that when the Single Wire Mode is enabled, if both the URXEN and UTXEN bits are high, the URX/UTX pin will just be used as UART receiver input.



### • UTXR\_RXR Register

The UTXR\_RXR register is the data register which is used to store the data to be transmitted on the UTX pin or being received from the URX/UTX 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)  | $f_H/[64 (N+1)]$ | $f_H/[16 (N+1)]$ |

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  $BR = f_H/[64 (N+1)]$

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 UTX output pin and URX/UTX input pin respectively. If no data is being transmitted on the UTX pin, then it will default to a logic high value.

Clearing the UREN bit will disable the UTX and URX/UTX pin and allow these 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, UUCR3 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.

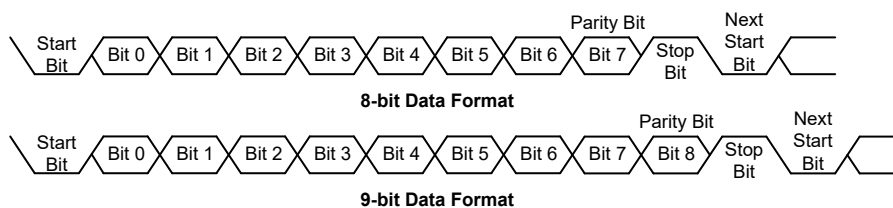
### Data, Parity and Stop Bit Selection

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 |
|--------------------------------------|-----------|-------------|------------|----------|
| <b>Example of 8-bit Data Formats</b> |           |             |            |          |
| 1                                    | 8         | 0           | 0          | 1        |
| 1                                    | 7         | 0           | 1          | 1        |
| 1                                    | 7         | 1           | 0          | 1        |
| <b>Example of 9-bit Data Formats</b> |           |             |            |          |
| 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 UTX 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 UTX 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 ensure that the UTX 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 greater than  $[(BRG+1) \times t_H]$  while UTIDLE=1, then the 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  $N=1, 2$ , 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 URX/UTX pin input 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 URX/UTX 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 URX/UTX pin input is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the URX/UTX 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 URX/UTX pin input, 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 URX/UTX 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 URIF 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**

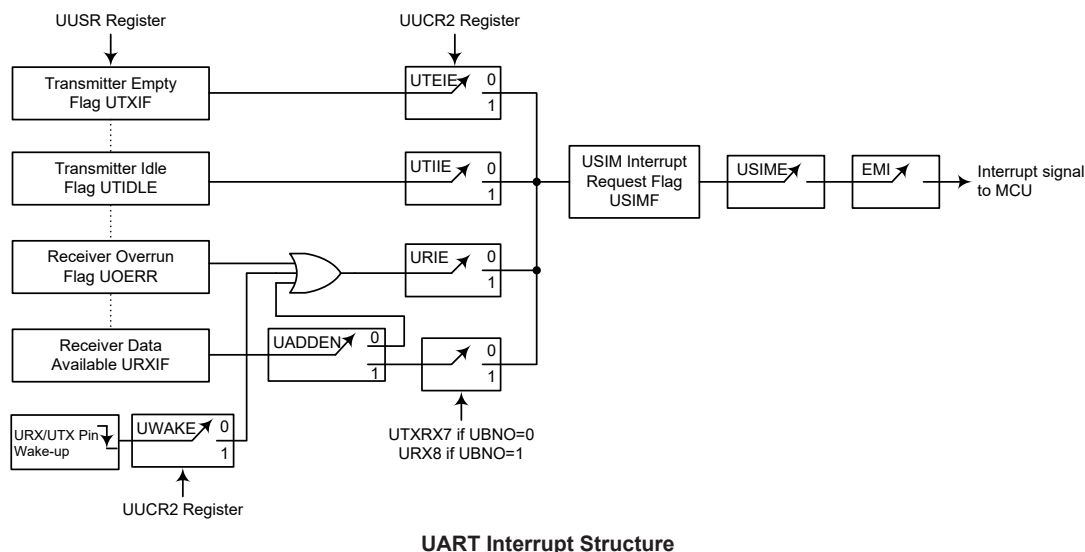
The read only parity error flag, UPERR, in the UUSR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, UPREN=1, and if the parity type, odd or even is selected. The read only UPERR flag and the received data will be recorded 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 URX/UTX 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 URX/UTX 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 ( $f_H$ ) source is switched off and the UWAKE and URUE bits in the UUCR2 register are set when a falling edge on the URX/UTX pin occurs. Note that in the event of an URX/UTX 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.



### 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,<br>8th bit if UBNO=0 | USIM Interrupt<br>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, UUCR3, transmit and receive registers, 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.

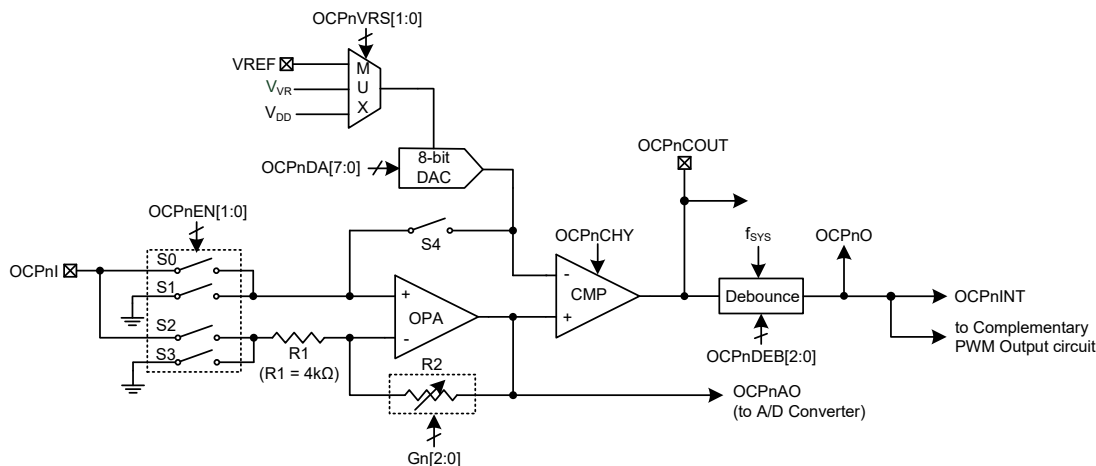
The UART function contains a receiver URX/UTX pin wake-up function, which is enabled or disabled by the UWAKE bit in the UUCR2 register. If this bit, along with the UART mode selection bit, UMD, the UART enable bit, UREN, the receiver enable bit, URXEN and the receiver interrupt bit, URIE, are all set when the UART clock ( $f_{H}$ ) is off, then a falling edge on the URX/UTX pin will trigger an URX/UTX 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 URX/UTX 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.



## Over Current Protection

The devices include an over current protection function which provides a protection mechanism for applications. To prevent the battery charge or load current from exceeding a specific level, the current on the OCPnI pin is converted to a relevant voltage level according to the current value using the OCP operational amplifier. It is then compared with a reference voltage generated by an 8-bit D/A converter. When an over current event occurs, an OCP interrupt will be generated if the corresponding interrupt control bit is enabled.



Note: The  $V_{VR}$  is from the A/D converter OPA output and the OCPnAO can be selected as the A/D converter input signals.

**Over Current Protection Circuit (n=0~1)**

## Over Current Protection Operation

The illustrated OCP circuit is used to prevent the input current from exceeding a reference level. The current on the OCPnI pin is converted to a voltage and then amplified by the OCP operational amplifier with a programmable gain from 1 to 50 selected by the Gn2~Gn0 bits in the OCPnC1 register. This is known as a Programmable Gain Amplifier or PGA. This PGA can also be configured to operate in the non-invert, invert or input offset calibration mode determined by the OCPnEN1 and OCPnEN0 bits in the OCPnC0 register. After the current is converted and amplified to a specific voltage level, it will be compared with a reference voltage provided by an 8-bit D/A converter. The 8-bit D/A converter reference voltage can be supplied from the internal power supply voltage,  $V_{DD}$ , or A/D converter internal operational amplifier output voltage,  $V_{VR}$ , or from an external reference source supplied on pin VREF, selected by the OCPnVRS[1:0] bits in the OCPnC0 register. The comparator output, OCPnCHY, will first be filtered with a certain de-bounce time period selected by the OCPnDEB2~OCPnDEB0 bits in the OCPnC1 register. Then a filtered OCPn digital comparator output, OCPnO, is obtained to indicate whether an over current condition occurs or not. The OCPnO bit will be set to 1 if an over current condition occurs. Otherwise, the OCPnO bit is zero. Once an over current event occurs, i.e., the converted voltage of the OCPn input current is greater than the reference voltage, the corresponding interrupt will be generated if the relevant interrupt control bit is enabled.

## Over Current Protection Registers

Overall operation of the over current protection is controlled using several registers. One register is used to provide the reference voltages for the over current protection circuit. There are two registers used to cancel out the operational amplifier and comparator input offset. Two control registers are



used to control the OCPn function, D/A converter reference voltage selection, PGA gain selection, comparator de-bounce time together with the hysteresis function.

| Register Name | Bit      |          |          |          |          |          |          |          |
|---------------|----------|----------|----------|----------|----------|----------|----------|----------|
|               | 7        | 6        | 5        | 4        | 3        | 2        | 1        | 0        |
| OCPnC0        | OCPnEN1  | OCPnEN0  | OCPnVRS1 | OCPnVRS0 | OCPnCHY  | —        | —        | OCPnO    |
| OCPnC1        | —        | —        | Gn2      | Gn1      | Gn0      | OCPnDEB2 | OCPnDEB1 | OCPnDEB0 |
| OCPnDA        | D7       | D6       | D5       | D4       | D3       | D2       | D1       | D0       |
| OCPnOCAL      | OCPnOOFM | OCPnORSP | OCPnOOF5 | OCPnOOF4 | OCPnOOF3 | OCPnOOF2 | OCPnOOF1 | OCPnOOF0 |
| OCPnCCAL      | OCPnCOU  | OCPnCOFM | OCPnCRSP | OCPnCOF4 | OCPnCOF3 | OCPnCOF2 | OCPnCOF1 | OCPnCOF0 |

**OCP Register List (n=0~1)**

• **OCPnC0 Register**

| Bit  | 7       | 6       | 5        | 4        | 3       | 2 | 1 | 0     |
|------|---------|---------|----------|----------|---------|---|---|-------|
| Name | OCPnEN1 | OCPnEN0 | OCPnVRS1 | OCPnVRS0 | OCPnCHY | — | — | OCPnO |
| R/W  | R/W     | R/W     | R/W      | R/W      | R/W     | — | — | R     |
| POR  | 0       | 0       | 0        | 0        | 0       | — | — | 0     |

Bit 7~6 **OCPnEN1~OCPnEN0**: OCPn function operating mode selection

- 00: OCPn function is disabled; S1 and S3 on, S0 and S2 off
- 01: Non-invert mode; S0 and S3 on, S1 and S2 off
- 10: Invert mode; S1 and S2 on, S0 and S3 off
- 11: Calibration mode; S1 and S3 on, S0 and S2 off

Bit 5~4 **OCPnVRS1~OCPnVRS0**: OCPn D/A converter reference voltage selection

- 00/11: From  $V_{DD}$
- 01: From external VREF pin
- 10: From  $V_{VR}$

When setting these bits to “10” to select the  $V_{VR}$  as the OCPn D/A converter reference voltage, care must be taken that as the  $V_{VR}$  signal is from the A/D converter OPA output, so the OPA must first be enabled by setting the ADVBGEN bit high.

Bit 3 **OCPnCHY**: OCPn comparator hysteresis function control

- 0: Disable
- 1: Enable

Bit 2~1 Unimplemented, read as “0”

Bit 0 **OCPnO**: OCPn digital output bit

- 0: No over current condition occurs in the monitored source current
- 1: Over current condition occurs in the monitored source current

• **OCPnC1 Register**

| Bit  | 7 | 6 | 5   | 4   | 3   | 2        | 1        | 0        |
|------|---|---|-----|-----|-----|----------|----------|----------|
| Name | — | — | Gn2 | Gn1 | Gn0 | OCPnDEB2 | OCPnDEB1 | OCPnDEB0 |
| R/W  | — | — | R/W | R/W | R/W | R/W      | R/W      | R/W      |
| POR  | — | — | 0   | 0   | 0   | 0        | 0        | 0        |

Bit 7~6 Unimplemented, read as “0”

Bit 5~3 **Gn2~Gn0**: R2/R1 ratio selection

- 000: Unity gain buffer (non-invert mode) or R2/R1=1(invert mode)
- 001: R2/R1=5
- 010: R2/R1=10
- 011: R2/R1=15
- 100: R2/R1=20
- 101: R2/R1=30
- 110: R2/R1=40
- 111: R2/R1=50

These bits are used to select the R2/R1 ratio to obtain various gain values for invert and non-invert mode. The calculating formula of the OCPn PGA gain for the invert and non-invert mode is described in the “Input Voltage Range” section.

Bit 2~0 **OCPnDEB2~OCPnDEB0**: OCPn output filter debounce time selection

000: Bypass, without debounce

001:  $(1\sim2) \times t_{DEB}$

010:  $(3\sim4) \times t_{DEB}$

011:  $(7\sim8) \times t_{DEB}$

100:  $(15\sim16) \times t_{DEB}$

101:  $(31\sim32) \times t_{DEB}$

110:  $(63\sim64) \times t_{DEB}$

111:  $(127\sim128) \times t_{DEB}$

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

#### • OCPnDA 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**: OCPn D/A converter output voltage control bits

OCPn D/A converter output  $V_{OUT} = (D/A \text{ converter reference voltage}/256) \times D[7:0]$

#### • OCPnOCAL Register

| Bit  | 7        | 6        | 5        | 4        | 3        | 2        | 1        | 0        |
|------|----------|----------|----------|----------|----------|----------|----------|----------|
| Name | OCPnOOFM | OCPnORSP | OCPnOOF5 | OCPnOOF4 | OCPnOOF3 | OCPnOOF2 | OCPnOOF1 | OCPnOOF0 |
| 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 **OCPnOOFM**: OCPn operational amplifier operating mode selection

0: Normal operation mode

1: Input Offset Calibration Mode

This bit is used to control the OCPn operational amplifier input offset calibration function. The OCPnEN1 and OCPnEN0 bits must first be set to “11” and then the OCPnOOFM bit must be set to 1 followed by the OCPnCOFM bit being cleared to 0, then the operational amplifier input offset calibration mode will be enabled. Refer to the “Operational Amplifier Input Offset Calibration” section for the detailed offset calibration procedures.

Bit 6 **OCPnORSP**: OCPn operational amplifier input offset voltage calibration reference selection

0: Select negative input as the reference input

1: Select positive input as the reference input

Bit 5~0 **OCPnOOF5~OCPnOOF0**: OCPn operational amplifier input offset voltage calibration value

This 6-bit field is used to perform the operational amplifier input offset calibration operation and the value for the OCPn operational amplifier input offset calibration can be restored into this bit field. More detailed information is described in the “Operational Amplifier Input Offset Calibration” section.

• **OCPnCCAL Register**

| Bit  | 7         | 6        | 5        | 4        | 3        | 2        | 1        | 0        |
|------|-----------|----------|----------|----------|----------|----------|----------|----------|
| Name | OCPnCOOUT | OCPnCOFM | OCPnCRSP | OCPnCOF4 | OCPnCOF3 | OCPnCOF2 | OCPnCOF1 | OCPnCOF0 |
| R/W  | R         | R/W      | R/W      | R/W      | R/W      | R/W      | R/W      | R/W      |
| POR  | 0         | 0        | 0        | 1        | 0        | 0        | 0        | 0        |

- Bit 7 OCPnOUT:** OCPn comparator output bit, positive logic (read only)  
 0: Positive input voltage < Negative input voltage  
 1: Positive input voltage > Negative input voltage  
 This bit is used to indicate whether the positive input voltage is greater than the negative input voltage when the OCPn operates in the input offset calibration mode. If the OCPnCOOUT is set to 1, the positive input voltage is greater than the negative input voltage. Otherwise, the positive input voltage is less than the negative input voltage.
- Bit 6 OCPnCOFM:** OCPn comparator operating mode selection  
 0: Normal operation  
 1: Input Offset Calibration Mode  
 This bit is used to control the OCPn comparator input offset calibration function. The OCPnEN1 and OCPnEN0 bits must first be set to “11” and then the OCPnCOFM bit must be set to 1 followed by the OCPnCOFM bit being cleared to 0, then the comparator input offset calibration mode will be enabled. Refer to the “Comparator Input Offset Calibration” section for the detailed offset calibration procedures.
- Bit 5 OCPnCRSP:** OCPn comparator input offset calibration reference input selection  
 0: Select negative input as the reference input  
 1: Select positive input as the reference input
- Bit 4~0 OCPnCOF4~OCPnCOF0:** OCPn comparator input offset calibration value  
 This 5-bit field is used to perform the comparator input offset calibration operation and the value for the OCPn comparator input offset calibration can be restored into this bit field. More detailed information is described in the “Comparator Input Offset Calibration” section.

## Input Voltage Range

Together with different PGA operating modes, the input voltage on the OCPnI pin can be positive or negative for flexible operation. The PGA output for the positive or negative input voltage is calculated based on different formulas and described by the following.

For input voltages  $V_{IN} > 0$ , the PGA operates in the non-invert mode and the PGA output is obtained using the formula below:

$$V_{OUT} = (1 + \frac{R_2}{R_1}) \times V_{IN}$$

When the PGA operates in the non-invert mode by setting the OCPnEN[1:0] to “01” with unity gain select by setting the Gn[2:0] to “000”, the PGA will act as a unit-gain buffer whose output is equal to  $V_{IN}$ .

$$V_{OUT} = V_{IN}$$

For input voltages  $0 > V_{IN} > -0.2V$ , the PGA operates in the invert mode and the PGA output is obtained using the formula below. Note that if the input voltage is negative, it cannot be lower than -0.2V which will result in current leakage.

$$V_{OUT} = -\frac{R_2}{R_1} \times V_{IN}$$

## OCP OPA and Comparator Offset Calibration

The OCPn circuit has four operating modes controlled by OCPnEN[1:0], one of them is calibration mode. In calibration mode, Operational amplifier and comparator offset can be calibrated. The procedures and settings of the operational amplifier and comparator input offset calibration are shown as follows.

### Operational Amplifier Input Offset Calibration

- Step 1. Set OCPnEN[1:0]=11, OCPnOOFM=1, OCPnCOFM=0 and OCPnORSP=1, the OCPn will operate in the operational amplifier input offset calibration mode. In this mode operation, the S4 is off, the OPA output to the OCPnCOOUT will bypass the comparator.
- Step 2. Set OCPnOOF[5:0]=000000 and then read the OCPnCOOUT bit.
- Step 3. Increase the OCPnOOF[5:0] value by 1 and then read the OCPnCOOUT bit.
- If the OCPnCOOUT bit state has not changed, then repeat Step 3 until the OCPnCOOUT bit state has changed.
- If the OCPnCOOUT bit state has changed, record the OCPnOOF value as  $V_{OOS1}$  and then go to Step 4.
- Step 4. Set OCPnOOF[5:0]=111111 and read the OCPnCOOUT bit.
- Step 5. Decrease the OCPnOOF[5:0] value by 1 and then read the OCPnCOOUT bit.
- If the OCPnCOOUT bit state has not changed, then repeat Step 5 until the OCPnCOOUT bit state has changed.
- If the OCPnCOOUT bit state has changed, record the OCPnOOF value as  $V_{OOS2}$  and then go to Step 6.
- Step 6. Restore the operational amplifier input offset calibration value  $V_{OOS}$  into the OCPnOOF[5:0] bit field. The offset Calibration procedure is now finished.

$$\text{Where } V_{OOS} = \frac{V_{OOS1} + V_{OOS2}}{2}$$

### Comparator Input Offset Calibration

- Step 1. Set OCPnEN[1:0]=11, OCPnCOFM=1 and OCPnOOFM=0, the OCPn is now in the comparator input offset calibration mode in which the S4 is on and the D/A converter is off (S4 is used only for comparator calibration mode, in other operation modes, it is off).
- Step 2. Set OCPnCOF[4:0] = 00000 and read the OCPnCOOUT bit.
- Step 3. Increase the OCPnCOF[4:0] value by 1 and then read the OCPnCOOUT bit.
- If the OCPnCOOUT bit state has not changed, then repeat Step 3 until the OCPnCOOUT bit state has changed.
- If the OCPnCOOUT bit state has changed, record the OCPnCOF value as  $V_{COS1}$  and then go to Step 4.
- Step 4. Set OCPnCOF[4:0] = 11111 and then read the OCPnCOOUT bit.
- Step 5. Decrease the OCPnCOF[4:0] value by 1 and then read the OCPnCOOUT bit.
- If the OCPnCOOUT bit state has not changed, then repeat Step 5 until the OCPnCOOUT bit state has changed.
- If the OCPnCOOUT bit state has changed, record the OCPnCOF value as  $V_{COS2}$  and then go to Step 6.
- Step 6. Restore the comparator input offset calibration value  $V_{COS}$  into the OCPnCOF[4:0] bit field. The offset Calibration procedure is now finished.

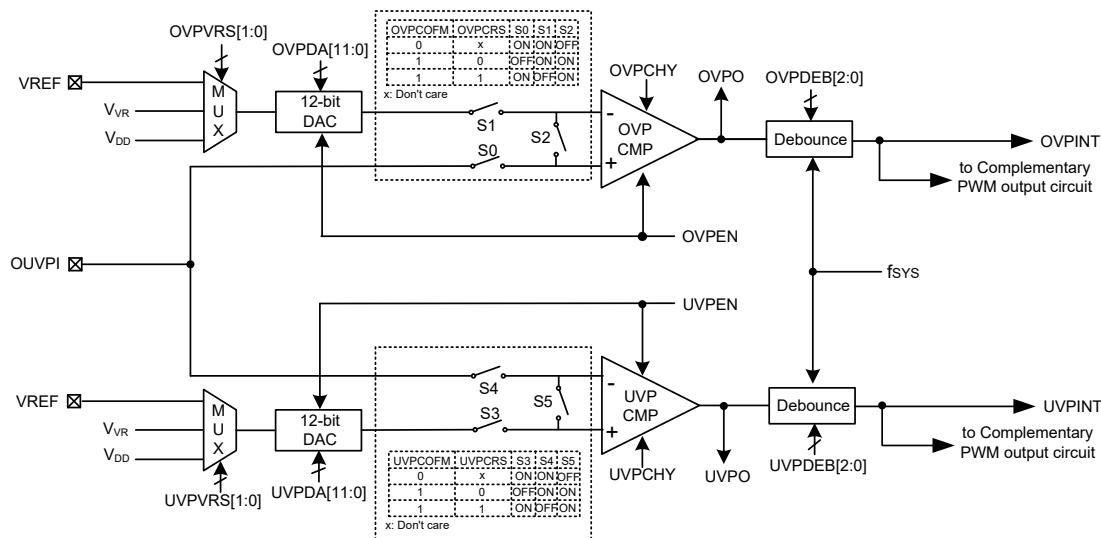
$$\text{Where } V_{COS} = \frac{V_{COS1} + V_{COS2}}{2}$$

## Over/Under Voltage Protection

The devices include an over/under voltage protection (OUVP) function which can be used for the application of battery charge/discharge.

### OUVP Circuit Operation

The OUVP circuit is built-in with the Over Voltage Protection (OVP) and the Under Voltage Protection (UVP) functions.



Note: V<sub>VR</sub> is from the A/D Converter OPA output and the OUVPI input voltage will pass through a voltage division circuit, refer to the A/D converter section for more detailed information.

**Over/Under Voltage Protection Block Diagram**

### Over Voltage Protection function

To prevent the output voltage from exceeding the specific voltage level, the OVP input voltage is compared with a reference voltage generated by a 12-bit D/A converter. The 12-bit D/A converter reference input signal range can come from V<sub>DD</sub>, V<sub>VR</sub>, or external VREF pin which is selected by the OVPVRS[1:0] bits. Once the OVP input voltage is greater than the reference voltage, the OVPO will change from “0” to “1”. The OVPINT is the de-bounce version of OVPO and used to indicate that the source voltage coming from OUVPI input is over the specification or not. OVPO is defined as OVP output and OVPINT is OVP interrupt trigger. The comparator of the OVP also has a hysteresis function controlled by OVPCHY bit.

### Under Voltage Protection function

To prevent the output voltage from being less than the specific voltage, the UVP input voltage is compared with a reference voltage generated by a 12-bit D/A converter. The 12-bit D/A converter reference input signal range can come from V<sub>DD</sub>, V<sub>VR</sub>, or external VREF pin which is selected by the UVPVRS[1:0] bits. Once the UVP input voltage is lower than the reference voltage, the UVPO will change from “0” to “1”. The UVPINT is the de-bounce version of UVPO and used to indicate that the source voltage coming from the OUVPI input is under the specification or not. UVPO is defined as UVP output and UVPINT is UVP interrupt trigger. The comparator of the UVP also has a hysteresis function controlled by UVPCHY bit.

## OUVP Register Description

The overall operation of the voltage protection and under voltage protection is controlled using several registers.

| Register Name | Bit  |         |        |         |         |         |         |         |
|---------------|------|---------|--------|---------|---------|---------|---------|---------|
|               | 7    | 6       | 5      | 4       | 3       | 2       | 1       | 0       |
| OUVPC0        | —    | —       | OVPEN  | OVPCHY  | OVPVRS1 | OVPVRS0 | OVPDEB1 | OVPDEB0 |
| OUVPC1        | —    | —       | UVPEN  | UVPCHY  | UVPVRS1 | UVPVRS0 | UVPDEB1 | UVPDEB0 |
| OUVPC2        | OVPO | OVPCOFM | OVPCRS | OVPCOF4 | OVPCOF3 | OVPCOF2 | OVPCOF1 | OVPCOF0 |
| OUVPC3        | UVPO | UVPCOFM | UVPCRS | UVPCOF4 | UVPCOF3 | UVPCOF2 | UVPCOF1 | UVPCOF0 |
| OVPDAH        | —    | —       | —      | —       | D11     | D10     | D9      | D8      |
| OVPDAL        | D7   | D6      | D5     | D4      | D3      | D2      | D1      | D0      |
| UVPDAH        | —    | —       | —      | —       | D11     | D10     | D9      | D8      |
| UVPDAL        | D7   | D6      | D5     | D4      | D3      | D2      | D1      | D0      |

OUVP Register List

### • OVPDAH & OVPDAL Registers

| Register | OVPDAH |   |   |   |     |     |     |     | OVPDAL |     |     |     |     |     |     |     |
|----------|--------|---|---|---|-----|-----|-----|-----|--------|-----|-----|-----|-----|-----|-----|-----|
| Bit      | 7      | 6 | 5 | 4 | 3   | 2   | 1   | 0   | 7      | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| Name     | —      | — | — | — | D11 | D10 | D9  | D8  | D7     | D6  | D5  | D4  | D3  | D2  | D1  | D0  |
| R/W      | —      | — | — | — | R/W | R/W | R/W | R/W | R/W    | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR      | —      | — | — | — | 0   | 0   | 0   | 0   | 0      | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

“—”: Unimplemented, read as “0”

**D11~D8:** OVP DAC reference level selection high byte

**D7~D0:** OVP DAC reference level selection low byte

$$\text{OVP DAC Output} = (\text{OVP DAC VREF}/4096) \times D[11:0]$$

Note: Write OVPDAL register only write to shadow buffer, and until write OVPDAH register will also copy the shadow buffer data to OVPDAL register.

### • UVPDAH & UVPDAL Registers

| Register | UVPDAH |   |   |   |     |     |     |     | UVPDAL |     |     |     |     |     |     |     |
|----------|--------|---|---|---|-----|-----|-----|-----|--------|-----|-----|-----|-----|-----|-----|-----|
| Bit      | 7      | 6 | 5 | 4 | 3   | 2   | 1   | 0   | 7      | 6   | 5   | 4   | 3   | 2   | 1   | 0   |
| Name     | —      | — | — | — | D11 | D10 | D9  | D8  | D7     | D6  | D5  | D4  | D3  | D2  | D1  | D0  |
| R/W      | —      | — | — | — | R/W | R/W | R/W | R/W | R/W    | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| POR      | —      | — | — | — | 0   | 0   | 0   | 0   | 0      | 0   | 0   | 0   | 0   | 0   | 0   | 0   |

“—”: Unimplemented, read as “0”

**D11~D8:** UVP DAC reference level selection high byte

**D7~D0:** UVP DAC reference level selection low byte

$$\text{UVP DAC Output} = (\text{UVP DAC V}_{\text{REF}}/4096) \times D[11:0]$$

Note: Write UVPDAL register only write to shadow buffer, and until write UVPDAH register will also copy the shadow buffer data to UVPDAL register.

### • OUVPC0 Register

| Bit  | 7 | 6 | 5     | 4      | 3       | 2       | 1       | 0       |
|------|---|---|-------|--------|---------|---------|---------|---------|
| Name | — | — | OVPEN | OVPCHY | OVPVRS1 | OVPVRS0 | OVPDEB1 | OVPDEB0 |
| R/W  | — | — | R/W   | R/W    | R/W     | R/W     | R/W     | R/W     |
| POR  | — | — | 0     | 0      | 0       | 0       | 0       | 0       |

Bit 7~6 Unimplemented, read as “0”

- Bit 5      **OVPEN**: Over Voltage Protection n function Enable control  
0: Disable  
1: Enable  
If the OVPEN bit is cleared to 0, the over voltage protection function is disabled and no power will be consumed. This results in the comparator and D/A converter of OVP all being switched off.
- Bit 4      **OVPCY**: Over Voltage Protection n Comparator Hysteresis Enable control  
0: Disable  
1: Enable
- Bit 3~2    **OVVRS1~OVVRS0**: OVP DAC reference voltage selection  
00: From  $V_{DD}$   
01: From external VREF pin  
10: From  $V_{VR}$   
11: From  $V_{DD}$   
Note: when setting these bits to “10” to select the  $V_{VR}$  as the OVP DAC reference voltage, care must be taken that as the  $V_{VR}$  signal is from the A/D Converter OPA output, so the OPA must first be enabled by setting the ADVBGEN bit high.
- Bit 1~0    **OVDEB1~OVDEB0**: Over Voltage Protection n comparator debounce time selection  
00: No debounce  
01:  $(7\sim8) \times 1/f_{SYS}$   
10:  $(15\sim16) \times 1/f_{SYS}$   
11:  $(31\sim32) \times 1/f_{SYS}$

• **OUVPC1 Register**

| Bit  | 7 | 6 | 5     | 4      | 3       | 2       | 1       | 0       |
|------|---|---|-------|--------|---------|---------|---------|---------|
| Name | — | — | UVPEN | UVPCHY | UVPVRS1 | UVPVRS0 | UVPDEB1 | UVPDEB0 |
| R/W  | — | — | R/W   | R/W    | R/W     | R/W     | R/W     | R/W     |
| POR  | — | — | 0     | 0      | 0       | 0       | 0       | 0       |

- Bit 7~6    Unimplemented, read as “0”
- Bit 5      **UVPEN**: Under Voltage Protection n function Enable control  
0: Disable  
1: Enable  
If the UVPEN bit is cleared to 0, the under voltage protection function is disabled and no power will be consumed. This results in the comparator and D/A converter of UVP all being switched off.
- Bit 4      **UVPCHY**: Under Voltage Protection n Comparator Hysteresis Enable control  
0: Disable  
1: Enable
- Bit 3~2    **UVPVRS1~UVPVRS0**: UVP DAC reference voltage selection  
00: From  $V_{DD}$   
01: From external VREF pin  
10: From  $V_{VR}$   
11: From  $V_{DD}$   
Note: when setting these bits to “10” to select the  $V_{VR}$  as the UVP DAC reference voltage, care must be taken that as the  $V_{VR}$  signal is from the A/D Converter OPA output, so the OPA must first be enabled by setting the ADVBGEN bit high.
- Bit 1~0    **UVPDEB1~UVPDEB0**: Under Voltage Protection n comparator debounce time selection  
00: No debounce  
01:  $(7\sim8) \times 1/f_{SYS}$   
10:  $(15\sim16) \times 1/f_{SYS}$   
11:  $(31\sim32) \times 1/f_{SYS}$

**• OUVPC2 Register**

| Bit  | 7    | 6       | 5      | 4       | 3       | 2       | 1       | 0       |
|------|------|---------|--------|---------|---------|---------|---------|---------|
| Name | OVPO | OVPCOFM | OVPCRS | OVPCOF4 | OVPCOF3 | OVPCOF2 | OVPCOF1 | OVPCOF0 |
| R/W  | R    | R/W     | R/W    | R/W     | R/W     | R/W     | R/W     | R/W     |
| POR  | 0    | 0       | 0      | 1       | 0       | 0       | 0       | 0       |

- Bit 7      **OVPO**: OVP comparator output bit  
0: Positive input voltage < negative input voltage  
1: Positive input voltage > negative input voltage
- Bit 6      **OVPCOFM**: OVP comparator normal operation or input offset voltage cancellation mode selection bit  
0: Normal operation  
1: Input offset voltage calibration mode
- Bit 5      **OVPCRS**: OVP comparator input offset voltage calibration reference selection bit  
0: Input reference voltage comes from negative input  
1: Input reference voltage comes from positive input  
This bit is used to select that the reference input voltage comes from the OVP D/A converter or external input. Note that this bit is only available when the OVP comparator input offset voltage calibration mode is selected by setting the OVPCOFM bit to 1.
- Bit 4~0    **OVPCOF4~OVPCOF0**: OVP comparator input offset voltage calibration control bits

**• OUVPC3 Register**

| Bit  | 7    | 6       | 5      | 4       | 3       | 2       | 1       | 0       |
|------|------|---------|--------|---------|---------|---------|---------|---------|
| Name | UVPO | UVPCOFM | UVPCRS | UVPCOF4 | UVPCOF3 | UVPCOF2 | UVPCOF1 | UVPCOF0 |
| R/W  | R    | R/W     | R/W    | R/W     | R/W     | R/W     | R/W     | R/W     |
| POR  | 0    | 0       | 0      | 1       | 0       | 0       | 0       | 0       |

- Bit 7      **UVPO**: UVP comparator output bit  
0: Positive input voltage < negative input voltage  
1: Positive input voltage > negative input voltage
- Bit 6      **UVPCOFM**: UVP comparator normal operation or input offset voltage cancellation mode selection bit  
0: Normal operation  
1: Input offset voltage calibration mode
- Bit 5      **UVPCRS**: UVP comparator input offset voltage calibration reference selection bit  
0: Input reference voltage comes from negative input  
1: Input reference voltage comes from positive input  
This bit is used to select that the reference input voltage comes from the UVP D/A converter or external input. Note that this bit is only available when the UVP comparator input offset voltage calibration mode is selected by setting the UVPCOFM bit to 1.
- Bit 4~0    **UVPCOF4~UVPCOF0**: UVP comparator input offset voltage calibration control bits

**OVP and UVP Comparator Offset calibration**

The OVP and UVP circuits provide comparator offset calibration function. Before offset calibration, the hysteresis voltage should be zero by clearing the OVPCHY or UVPCHY bit to zero. As the OUVV input pins are pin-shared with other functions, the OUVV pin function must first be setup as comparator input using the corresponding pin-shared function selection register bits. The following content are the steps for the OVP or UVP comparator calibration.



**OVP Comparator calibration:**

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

Step2: Set OVPCOF[4:0] = 00000 and then read the OVPO bit

Step3: Increase the OVPCOF[4:0] by 1 and then read the OVPO bit.

If the OVPO bit state has not changed, then repeat Step 3 until the OVPO bit state has changed.

If the OVPO bit state has changed, record the OVPCOF[4:0] value as  $V_{OS1}$  and then go to Step 4.

Step4: Set OVPCOF[4:0] = 11111 and then read the OVPO bit

Step5: Decrease the OVPCOF[4:0] value by 1 and then read the OVPO bit.

If the OVPO bit state has not changed, then repeat Step 5 until the OVPO bit state has changed.

If the OVPO bit state has changed, record the OVPCOF[4:0] value as  $V_{OS2}$  and then go to Step 6.

Step6: Restore the  $V_{OS} = \frac{V_{OS1} + V_{OS2}}{2}$  to OVPCOF[4:0] bit field, the calibration procedure is now finished.

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

$$\text{Residue } V_{OS} = V_{OUT} - V_{IN} \quad (1)$$

**UVP Comparator calibration:**

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

Step2: Set UVPCOF[4:0] = 00000 and then read the UVPO bit

Step3: Increase the UVPCOF[4:0] by 1 and then read the UVPO bit.

If the UVPO bit state has not changed, then repeat Step 3 until the UVPO bit state has changed.

If the UVPO bit state has changed, record the UVPCOF[4:0] value as  $V_{OS1}$  and then go to Step 4.

Step4: Set UVPCOF[4:0] = 11111 and then read the UVPO bit

Step5: Decrease the UVPCOF[4:0] value by 1 and then read the UVPO bit.

If the UVPO bit state has not changed, then repeat Step 5 until the UVPO bit state has changed.

If the UVPO bit state has changed, record the UVPCOF[4:0] value as  $V_{OS2}$  and then go to Step 6.

Step6: Restore the  $V_{OS} = \frac{V_{OS1} + V_{OS2}}{2}$  to UVPCOF[4:0] bit field, the calibration procedure is now finished.

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

$$\text{Residue } V_{OS} = V_{OUT} - V_{IN} \quad (2)$$

The devices include two USB ports named D1+/D1- and D2+/D2- to implement the Charge/Discharge Devices Auto Detection function. Users can distinguish the devices connected to the USB ports is a dedicated charger, portable device, general USB interface or charging device with USB interface by monitoring the voltage and current of the connected USB lines.



There two 8-bit D/A Converters, DACn, which is enabled by the DACnEN bits in the ADUC0 register. The D/A Converter output signal is controlled by the ADUDAn register value and the reference voltage which is selected by the D1VRS[1:0] bit in the ADUC0 register. There is an analog switch connected between the D1+ and D1- lines, which is controlled by the USW1 bit. Similarly, there is an analog switch connected between the D2+ and D2- lines, which is controlled by the USW2 bit. But it needs to note that only when one of the D1+ and D1- or D2+ and D2- pins is in the analog or digital input type by setting the Pin-shared Function register, and then set the USW1or USW2 bit high, the switch can be on. The D1+/D1- and D2+/D2- lines are individually connected a pull low resistor respectively to VSS which are controlled by the D1NPLEN/D1PPLEN and D2NPLEN/D2PPLEN bits in the ADUC1 register.

## USB Auto Detection Registers

Overall operation of the USB auto detection function is controlled using several registers.

| Register Name | Bit |    |         |         |         |         |        |        |
|---------------|-----|----|---------|---------|---------|---------|--------|--------|
|               | 7   | 6  | 5       | 4       | 3       | 2       | 1      | 0      |
| ADUC0         | —   | —  | D1VRS1  | D1VRS0  | —       | —       | DAC1EN | DAC0EN |
| ADUC1         | —   | —  | D2NPLEN | D2PPLEN | D1NPLEN | D1PPLEN | USW2   | USW1   |
| ADUDA0        | D7  | D6 | D5      | D4      | D3      | D2      | D1     | D0     |
| ADUDA1        | D7  | D6 | D5      | D4      | D3      | D2      | D1     | D0     |

USB Auto Detection Register List

### • ADUC0 Register

| Bit  | 7 | 6 | 5      | 4      | 3 | 2 | 1      | 0      |
|------|---|---|--------|--------|---|---|--------|--------|
| Name | — | — | D1VRS1 | D1VRS0 | — | — | DAC1EN | DAC0EN |
| R/W  | — | — | R/W    | R/W    | — | — | R/W    | R/W    |
| POR  | — | — | 0      | 0      | — | — | 0      | 0      |

- Bit 7~6 Unimplemented, read as “0”.
- Bit 5~4 **D1VRS1~D1VRS0**: DAC1 and DAC0 reference voltage selection  
 00/11: From VDD pin  
 01: From VREF pin  
 10: From V<sub>VR</sub>
- Bit 3~2 Unimplemented, read as “0”.
- Bit 1 **DAC1EN**: DAC1 enable Control  
 0: Disable  
 1: Enable
- Bit 0 **DAC0EN**: DAC0 enable Control  
 0: Disable  
 1: Enable

### • ADUC1 Register

| Bit  | 7 | 6 | 5       | 4       | 3       | 2       | 1    | 0    |
|------|---|---|---------|---------|---------|---------|------|------|
| Name | — | — | D2NPLEN | D2PPLEN | D1NPLEN | D1PPLEN | USW2 | USW1 |
| R/W  | — | — | R/W     | R/W     | R/W     | R/W     | R/W  | R/W  |
| POR  | — | — | 0       | 0       | 0       | 0       | 0    | 0    |

- Bit 7~6 Unimplemented, read as “0”.
- Bit 5 **D2NPLEN**: D2- pin Pull-Low Control  
 0: Disable  
 1: Enable
- Bit 4 **D2PPLEN**: D2+ pin Pull-Low Control  
 0: Disable  
 1: Enable
- Bit 3 **D1NPLEN**: D1- pin Pull-Low Control  
 0: Disable  
 1: Enable
- Bit 2 **D1PPLEN**: D1+ pin Pull-Low Control  
 0: Disable  
 1: Enable

- Bit 1      **USW2:** USW2 switch control  
             0: Switch off  
             1: Switch on  
             This bit controls the USW2 switch on/off. But only when one of the D2+ and D2- pins is used as the analog or digital input pin by setting the Pin-shared Function register, and then set the USW2 bit high, can the switch be on.
- Bit 0      **USW1:** USW1 switch control  
             0: Switch off  
             1: Switch on  
             This bit controls the USW1 switch on/off. But only when one of the D1+ and D1- pins is used as the analog or digital input pin by setting the Pin-shared Function register, and then set the USW1 bit high, can the switch be on.

• **ADUDA0 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:** 8-bit DAC0 Output Control Data Bits  
              $\text{DAC0 Output} = (\text{DAC0 Reference Voltage}) \times (\text{ADUDA0 [7:0]}) / 256$

• **ADUDA1 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:** 8-bit DAC1 Output Control Data Bits  
              $\text{DAC1 Output} = (\text{DAC1 Reference Voltage}) \times (\text{ADUDA1 [7:0]}) / 256$

## Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a TM Comparator P, Comparator A match, requires microcontroller attention, its corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to its needs. The devices contain several external interrupt and internal interrupt functions. The external interrupts are generated by the action of the external INT0~INT1 pins, while the internal interrupts are generated by internal functions including the TMs, A/D converter, LVD, OCP, OVP and Time Bases.

### 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~INTC3 registers which set the primary interrupts, the second is an INTEG register to setup the external interrupts trigger edge type. Finally there is MFI register which setup the Multi-function interrupts

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

| Function      | Enable Bit | Request Flag | Notes |
|---------------|------------|--------------|-------|
| Global        | EMI        | —            | —     |
| INTn Pin      | INTnE      | INTnF        | n=0~1 |
| A/D Converter | ADE        | ADF          | —     |
| Time Bases    | TBnE       | TBnF         | n=0~1 |
| OVP           | OVPE       | OVPF         | —     |
| UVP           | UVPE       | UVPF         | —     |
| OCP           | OCPnE      | OCPnF        | n=0~1 |
| LVD           | LVE        | LVF          | —     |
| CTM           | CTMnPE     | CTMnPF       | n=0~1 |
|               | CTMnAE     | CTMnAF       | n=0~1 |
| PTM           | PTMPE      | PTMPF        | —     |
|               | PTMAE      | PTMAF        | —     |
| USIM          | USIME      | USIMF        | —     |

**Interrupt Register Bit Naming Conventions**

| Register Name | Bit    |        |        |        |        |        |        |        |
|---------------|--------|--------|--------|--------|--------|--------|--------|--------|
|               | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
| INTEG         | —      | —      | —      | —      | INT1S1 | INT1S0 | INT0S1 | INT0S0 |
| INTC0         | —      | OVPF   | OCP1F  | OCPOF  | OVPE   | OCP1E  | OCP0E  | EMI    |
| INTC1         | ADF    | INT1F  | INT0F  | UVPF   | ADE    | INT1E  | INT0E  | UVPE   |
| INTC2         | LVF    | PTMAF  | PTMPF  | MFF    | LVE    | PTMAE  | PTMPE  | MFE    |
| INTC3         | —      | USIMF  | TB1F   | TB0F   | —      | USIME  | TB1E   | TB0E   |
| MFI           | CTM1AF | CTM1PF | CTM0AF | CTM0PF | CTM1AE | CTM1PE | CTM0AE | CTM0PE |

**Interrupt Register List**

**• INTEG Register**

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

- Bit 7~4      Unimplemented, read as “0”
- Bit 3~2      **INT1S1~INT1S0**: Interrupt edge control for INT1 pin  
                  00: Disable  
                  01: Rising edge  
                  10: Falling edge  
                  11: Rising and falling edges
- Bit 1~0      **INT0S1~INT0S0**: Interrupt edge control for INT0 pin  
                  00: Disable  
                  01: Rising edge  
                  10: Falling edge  
                  11: Rising and falling edges

**• INTC0 Register**

| Bit  | 7 | 6    | 5     | 4     | 3    | 2     | 1     | 0   |
|------|---|------|-------|-------|------|-------|-------|-----|
| Name | — | OVPF | OCP1F | OCP0F | OVPE | OCP1E | OCP0E | 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      **OVPF**: OVP interrupt request flag  
                  0: No request  
                  1: Interrupt request
- Bit 5      **OCP1F**: OCP1 interrupt request flag  
                  0: No request  
                  1: Interrupt request
- Bit 4      **OCP0F**: OCP0 interrupt request flag  
                  0: No request  
                  1: Interrupt request
- Bit 3      **OVPE**: OVP interrupt control  
                  0: Disable  
                  1: Enable
- Bit 2      **OCP1E**: OCP1 interrupt control  
                  0: Disable  
                  1: Enable
- Bit 1      **OCP0E**: OCP0 interrupt control  
                  0: Disable  
                  1: Enable
- Bit 0      **EMI**: Global interrupt control  
                  0: Disable  
                  1: Enable

• **INTC1 Register**

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

- Bit 7      **ADF**: A/D converter interrupt request flag  
0: No request  
1: Interrupt request
- Bit 6      **INT1F**: INT1 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5      **INT0F**: INT0 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **UVPF**: UVP interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3      **ADE**: A/D converter interrupt control  
0: Disable  
1: Enable
- Bit 2      **INT1E**: INT1 interrupt control  
0: Disable  
1: Enable
- Bit 1      **INT0E**: INT0 interrupt control  
0: Disable  
1: Enable
- Bit 0      **UVPE**: UVP interrupt control  
0: Disable  
1: Enable

• **INTC2 Register**

| Bit  | 7   | 6     | 5     | 4   | 3   | 2     | 1     | 0   |
|------|-----|-------|-------|-----|-----|-------|-------|-----|
| Name | LVF | PTMAF | PTMPF | MFF | LVE | PTMAE | PTMPE | MFE |
| R/W  | R/W | R/W   | R/W   | R/W | R/W | R/W   | R/W   | R/W |
| POR  | 0   | 0     | 0     | 0   | 0   | 0     | 0     | 0   |

- Bit 7      **LVF**: LVD interrupt request flag  
0: No request  
1: Interrupt request
- Bit 6      **PTMAF**: PTM comparator A match interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5      **PTMPF**: PTM comparator P match interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **MFF**: Multi-function interrupt 0 request flag  
0: No request  
1: Interrupt request
- Bit 3      **LVE**: LVD interrupt control  
0: Disable  
1: Enable

- Bit 2      **PTMAE**: PTM comparator A match interrupt control  
0: Disable  
1: Enable
- Bit 1      **PTMPE**: PTM comparator P match interrupt control  
0: Disable  
1: Enable
- Bit 0      **MFE**: Multi-function interrupt control  
0: Disable  
1: Enable

**• INTC3 Register**

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

- Bit 7      Unimplemented, read as “0”
- Bit 6      **USIMF**: USIM interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5      **TB1F**: Time Base 1 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **TB0F**: Time Base 0 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3      Unimplemented, read as “0”
- Bit 2      **USIME**: USIM interrupt control  
0: Disable  
1: Enable
- Bit 1      **TB1E**: Time Base 1 interrupt control  
0: Disable  
1: Enable
- Bit 0      **TB0E**: Time Base 0 interrupt control  
0: Disable  
1: Enable

**• MFI Register**

| Bit  | 7      | 6      | 5      | 4      | 3      | 2      | 1      | 0      |
|------|--------|--------|--------|--------|--------|--------|--------|--------|
| Name | CTM1AF | CTM1PF | CTM0AF | CTM0PF | CTM1AE | CTM1PE | CTM0AE | CTM0PE |
| 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      **CTM1AF**: CTM1 CCRA comparator interrupt request flag  
0: No request  
1: Interrupt request
- Bit 6      **CTM1PF**: CTM1 CCRP comparator interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5      **CTM0AF**: CTM0 CCRA comparator interrupt request flag  
0: No request  
1: Interrupt request



|       |                                                                                                      |
|-------|------------------------------------------------------------------------------------------------------|
| Bit 4 | <b>CTM0PF</b> : CTM0 CCRP comparator interrupt request flag<br>0: No request<br>1: Interrupt request |
| Bit 3 | <b>CTM1AE</b> : CTM1 CCRA comparator interrupt control<br>0: Disable<br>1: Enable                    |
| Bit 2 | <b>CTM1PE</b> : CTM1 CCRP comparator interrupt control<br>0: Disable<br>1: Enable                    |
| Bit 1 | <b>CTM0AE</b> : CTM0 CCRA comparator interrupt control<br>0: Disable<br>1: Enable                    |
| Bit 0 | <b>CTM0PE</b> : CTM0 CCRP comparator interrupt control<br>0: Disable<br>1: Enable                    |

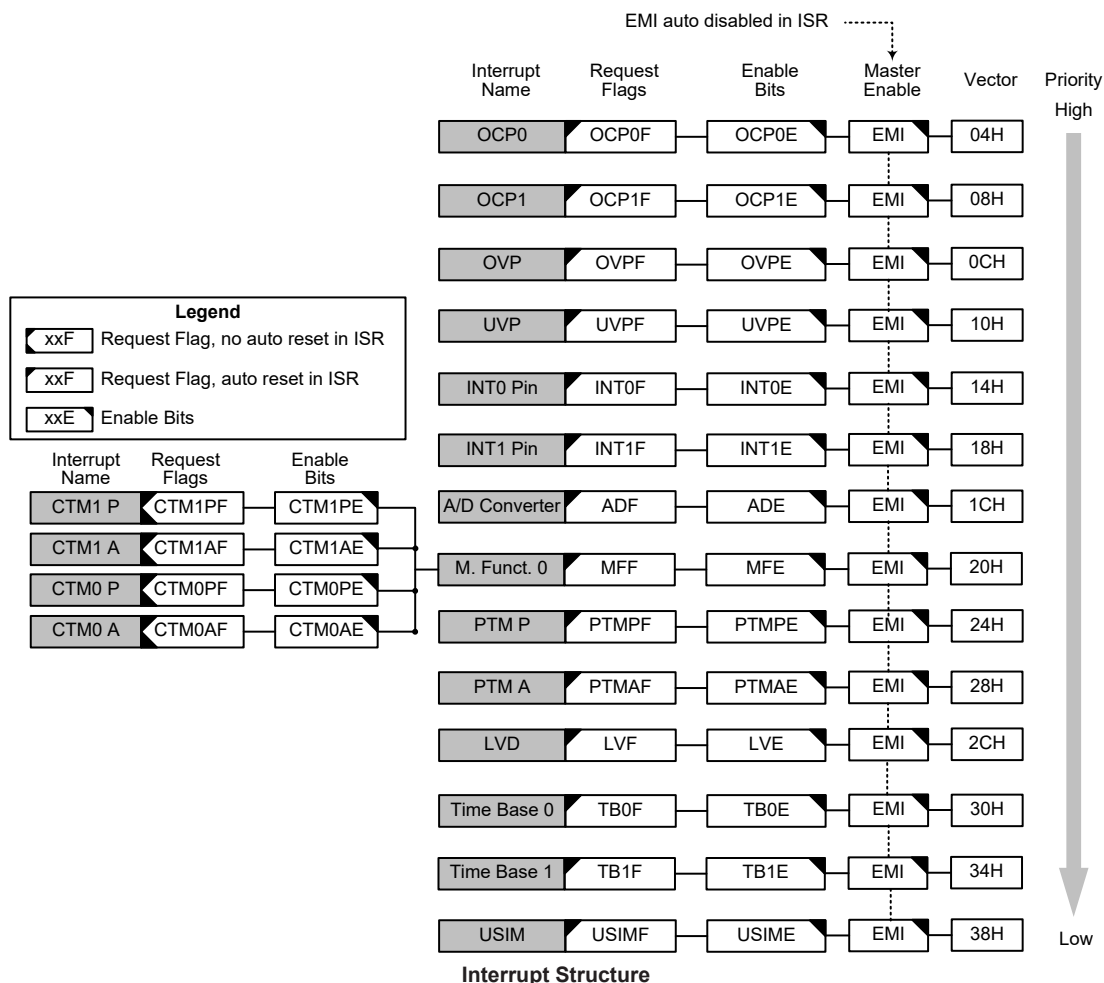
### Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P, Comparator A match, 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 an “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 diagram with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector. Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

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



## External Interrupts

The external interrupts are controlled by signal transitions on the INTn pins. An external interrupt request will take place when the external interrupt request flag, INTnF, is set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the external interrupt enable bit, INTnE, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bits in the corresponding interrupt registers has been set and the external interrupt pins are selected by the corresponding pin-shared function selection bits. The pins must also be set 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 pins, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flag, INTnF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any 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 respective interrupt vector address, the global interrupt enable bit, EMI, and A/D converter 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 converter interrupt flag, ADF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

### **Over Current Protection Interrupt**

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

### **Over Voltage Protection Interrupt**

The OVP interrupt is controlled by detecting the OVP input voltage. An OVP interrupt request will take place when the OVP interrupt request flag, OVPF, is set, which occurs when the over voltage protection circuit detects an over voltage condition. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and OVP interrupt enable bit, OVPE, must first be set. When the interrupt is enabled, the stack is not full and an over voltage condition is detected, a subroutine call to the OVP interrupt vector, will take place. When the interrupt is serviced, the OVP interrupt flag, OVPF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

### **Under Voltage Protection Interrupts**

The UVPn Interrupt is controlled by detecting the UVP input voltage. An UVP Interrupt request will take place when the UVP Interrupt request flag, UVPF, is set, which occurs when the Under Voltage Protection circuit detects an under voltage condition. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and UVPn Interrupt enable bit, UVPE, must first be set. When the interrupt is enabled, the stack is not full and an under voltage is detected, a subroutine call to the UVP Interrupt vector, will take place. When the interrupt is serviced, the UVP Interrupt flag, UVF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

### **Multi-function Interrupts**

Within the devices there is a Multi-function interrupt. Unlike the other independent interrupts, this interrupt has no independent source, but rather are formed from other existing interrupt sources, namely the CTMn Interrupts.

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

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

### **Timer Module Interrupts**

The Compact and Periodic TMs each have two interrupts, one comes from the comparator A match situation and the other comes from the comparator P match situation. The CTMn interrupts are contained within the Multi-function Interrupt while the PTM interrupt sources has their own individual vector. For all of the TM types there are two interrupt request flags and two enable control bits. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

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

To allow the program to branch to the PTM interrupt vector addresses, the global interrupt enable bit, EMI, PTM Interrupt enable bit, PTME must first be set. When the interrupt is enabled, the stack is not full and the PTM overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, PTMF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

### **USIM 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, I<sup>2</sup>C mode and UART mode, the USIMF flag can be set by different conditions depending on the selected interface mode.

If the SPI or I<sup>2</sup>C mode is selected, the USIM interrupt can be triggered when a byte of data has been received or transmitted by the USIM SPI or I<sup>2</sup>C interface, or an I<sup>2</sup>C slave address match occurs, or an I<sup>2</sup>C 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 URX/UTX pin wake-up, can generate an USIM interrupt with the USIMF flag bit set high.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and the Universal Serial Interface Module 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,

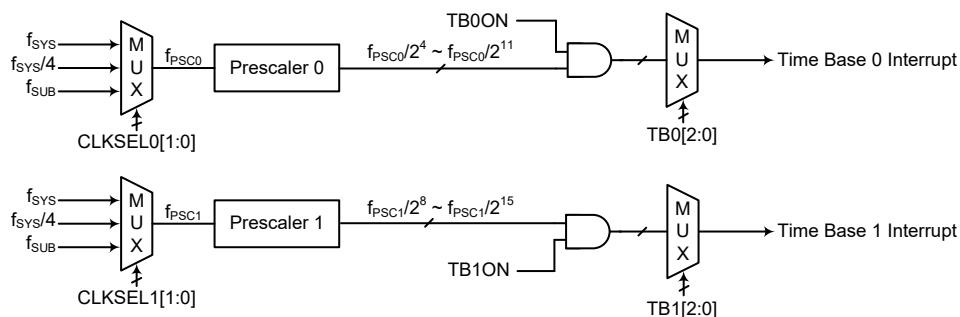
a subroutine call to the respective interrupt vector, will take place. When the interrupt is serviced, the Universal Serial Interface Module interrupt flag, USIMF, will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts.

Note that if the USIM interrupt is triggered by the UART interface, after the interrupt has been serviced, the USR register flags will be cleared automatically

## Time Base Interrupts

The function of the Time Base interrupts is to provide regular time signal in the form of an internal interrupt. It is controlled by the overflow signals from the timer function. When this happens its interrupt request flag TBnF will be set. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI and Time Base enable bit, TBnE, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to its interrupt vector location will take place. When the interrupt is serviced, the interrupt request flag, TBnF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base interrupts is to provide an interrupt signal at fixed time periods. Its clock source,  $f_{PSCn}$ , 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 TBnC register to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSELn[1:0] bits in the PSCnR register.



Time Base Interrupts

### • PSCnR Register (n=0~1)

| Bit  | 7 | 6 | 5 | 4 | 3 | 2 | 1        | 0        |
|------|---|---|---|---|---|---|----------|----------|
| Name | — | — | — | — | — | — | CLKSELn1 | CLKSELn0 |
| R/W  | — | — | — | — | — | — | R/W      | R/W      |
| POR  | — | — | — | — | — | — | 0        | 0        |

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **CLKSELn1~CLKSELn0**: Prescaler n clock source  $f_{PSCn}$  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**: Time Base 0 time-out period selection  
000:  $2^4/f_{PSC0}$   
001:  $2^5/f_{PSC0}$   
010:  $2^6/f_{PSC0}$   
011:  $2^7/f_{PSC0}$   
100:  $2^8/f_{PSC0}$   
101:  $2^9/f_{PSC0}$   
110:  $2^{10}/f_{PSC0}$   
111:  $2^{11}/f_{PSC0}$

**• 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**: Time Base 1 time-out period selection  
000:  $2^8/f_{PSC1}$   
001:  $2^9/f_{PSC1}$   
010:  $2^{10}/f_{PSC1}$   
011:  $2^{11}/f_{PSC1}$   
100:  $2^{12}/f_{PSC1}$   
101:  $2^{13}/f_{PSC1}$   
110:  $2^{14}/f_{PSC1}$   
111:  $2^{15}/f_{PSC1}$

**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 respective 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 LVD 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 devices are in the SLEEP or IDLE Mode and its system oscillator stopped, situations such as external edge transitions on the external interrupt pin 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 devices enter 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 an RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

## Low Voltage Detector – LVD

The devices have a Low Voltage Detector function, also known as LVD. This enables the devices to monitor the power supply voltage,  $V_{DD}$ , and provides a warning signal should it fall below a certain level. This function may be especially useful in battery applications where the supply voltage will gradually reduce as the battery ages, as it allows an early warning battery low signal to be generated. The Low Voltage Detector also has the capability of generating an interrupt signal.

### LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, VLVD2~VLVD0, are used to select one of five 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

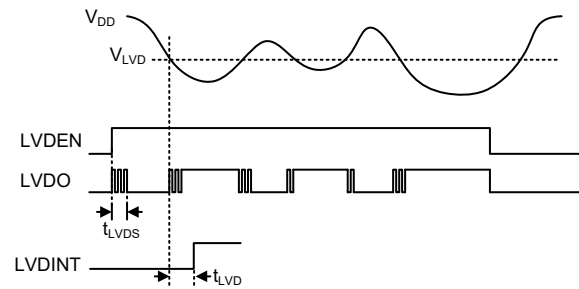
Note that the Bandgap circuit is enabled when the LVD or LVR function is enabled or when the VBGEN bit is set to 1.

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



## LVD Operation

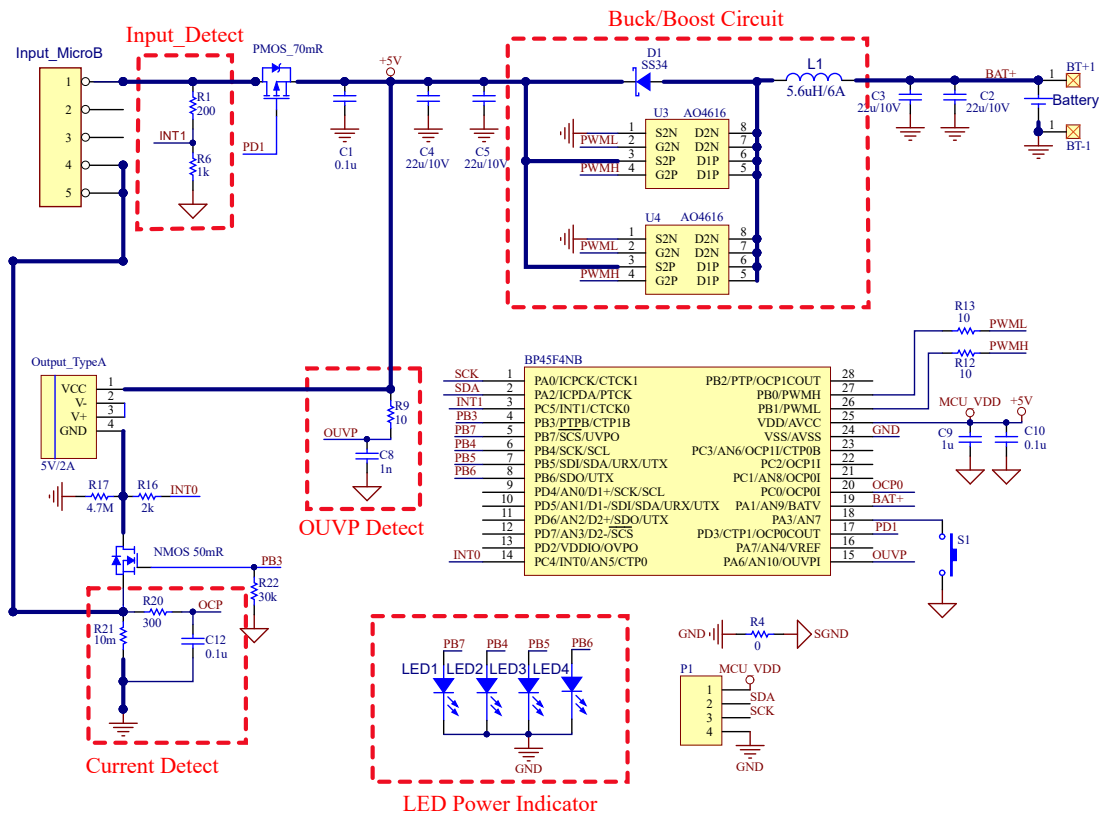
The Low Voltage Detector function operates by comparing the power supply voltage,  $V_{DD}$ , with a pre-specified voltage level stored in the LVDC register. This has a range of between 2.7V and 4.0V. When the power supply voltage,  $V_{DD}$ , falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. When the devices enter the SLEEP mode, the low voltage detector will be automatically disabled even if the LVDEN bit is set high. After enabling the Low Voltage Detector, a time delay  $t_{LVDS}$  should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the  $V_{DD}$  voltage may rise and fall rather slowly, at the voltage nears that of  $V_{LVD}$ , there may be multiple bit LVDO transitions.



**LVD Operation**

The Low Voltage Detector also has its own interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of  $t_{LVD}$  after the LVDO bit has been set high by a low voltage condition. 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 devices 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 devices enter the IDLE Mode.

## Application Circuits



## **Instruction Set**

### **Introduction**

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

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

### **Instruction Timing**

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5 $\mu$ s and branch or call instructions would be implemented within 1 $\mu$ 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        |
|----------------------------------|-----------------------------------------------------------------|-------------------|----------------------|
| <b>Arithmetic</b>                |                                                                 |                   |                      |
| ADD A,[m]                        | Add Data Memory to ACC                                          | 1                 | Z, C, AC, OV, SC     |
| ADDM A,[m]                       | Add ACC to Data Memory                                          | 1 <sup>Note</sup> | 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 <sup>Note</sup> | 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 <sup>Note</sup> | 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 <sup>Note</sup> | Z, C, AC, OV, SC, CZ |
| DAA [m]                          | Decimal adjust ACC for Addition with result in Data Memory      | 1 <sup>Note</sup> | C                    |
| <b>Logic Operation</b>           |                                                                 |                   |                      |
| 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 <sup>Note</sup> | Z                    |
| ORM A,[m]                        | Logical OR ACC to Data Memory                                   | 1 <sup>Note</sup> | Z                    |
| XORM A,[m]                       | Logical XOR ACC to Data Memory                                  | 1 <sup>Note</sup> | 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 <sup>Note</sup> | Z                    |
| CPLA [m]                         | Complement Data Memory with result in ACC                       | 1                 | Z                    |
| <b>Increment &amp; Decrement</b> |                                                                 |                   |                      |
| INCA [m]                         | Increment Data Memory with result in ACC                        | 1                 | Z                    |
| INC [m]                          | Increment Data Memory                                           | 1 <sup>Note</sup> | Z                    |
| DECA [m]                         | Decrement Data Memory with result in ACC                        | 1                 | Z                    |
| DEC [m]                          | Decrement Data Memory                                           | 1 <sup>Note</sup> | Z                    |
| <b>Rotate</b>                    |                                                                 |                   |                      |
| RRA [m]                          | Rotate Data Memory right with result in ACC                     | 1                 | None                 |
| RR [m]                           | Rotate Data Memory right                                        | 1 <sup>Note</sup> | None                 |
| RRCA [m]                         | Rotate Data Memory right through Carry with result in ACC       | 1                 | C                    |
| RRC [m]                          | Rotate Data Memory right through Carry                          | 1 <sup>Note</sup> | C                    |
| RLA [m]                          | Rotate Data Memory left with result in ACC                      | 1                 | None                 |
| RL [m]                           | Rotate Data Memory left                                         | 1 <sup>Note</sup> | None                 |
| RLCA [m]                         | Rotate Data Memory left through Carry with result in ACC        | 1                 | C                    |
| RLC [m]                          | Rotate Data Memory left through Carry                           | 1 <sup>Note</sup> | C                    |

| Mnemonic                    | Description                                                                               | Cycles            | Flag Affected |
|-----------------------------|-------------------------------------------------------------------------------------------|-------------------|---------------|
| <b>Data Move</b>            |                                                                                           |                   |               |
| MOV A,[m]                   | Move Data Memory to ACC                                                                   | 1                 | None          |
| MOV [m],A                   | Move ACC to Data Memory                                                                   | 1 <sup>Note</sup> | None          |
| MOV A,x                     | Move immediate data to ACC                                                                | 1                 | None          |
| <b>Bit Operation</b>        |                                                                                           |                   |               |
| CLR [m].i                   | Clear bit of Data Memory                                                                  | 1 <sup>Note</sup> | None          |
| SET [m].i                   | Set bit of Data Memory                                                                    | 1 <sup>Note</sup> | None          |
| <b>Branch Operation</b>     |                                                                                           |                   |               |
| JMP addr                    | Jump unconditionally                                                                      | 2                 | None          |
| SZ [m]                      | Skip if Data Memory is zero                                                               | 1 <sup>Note</sup> | None          |
| SZA [m]                     | Skip if Data Memory is zero with data movement to ACC                                     | 1 <sup>Note</sup> | None          |
| SZ [m].i                    | Skip if bit i of Data Memory is zero                                                      | 1 <sup>Note</sup> | None          |
| SNZ [m]                     | Skip if Data Memory is not zero                                                           | 1 <sup>Note</sup> | None          |
| SNZ [m].i                   | Skip if bit i of Data Memory is not zero                                                  | 1 <sup>Note</sup> | None          |
| SIZ [m]                     | Skip if increment Data Memory is zero                                                     | 1 <sup>Note</sup> | None          |
| SDZ [m]                     | Skip if decrement Data Memory is zero                                                     | 1 <sup>Note</sup> | None          |
| SIZA [m]                    | Skip if increment Data Memory is zero with result in ACC                                  | 1 <sup>Note</sup> | None          |
| SDZA [m]                    | Skip if decrement Data Memory is zero with result in ACC                                  | 1 <sup>Note</sup> | 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          |
| <b>Table Read Operation</b> |                                                                                           |                   |               |
| TABRD [m]                   | Read table (specific page) to TBLH and Data Memory                                        | 2 <sup>Note</sup> | None          |
| TABRDL [m]                  | Read table (last page) to TBLH and Data Memory                                            | 2 <sup>Note</sup> | None          |
| ITABRD [m]                  | Increment table pointer TBLP first and Read table (specific page) to TBLH and Data Memory | 2 <sup>Note</sup> | None          |
| ITABRDL [m]                 | Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory     | 2 <sup>Note</sup> | None          |
| <b>Miscellaneous</b>        |                                                                                           |                   |               |
| NOP                         | No operation                                                                              | 1                 | None          |
| CLR [m]                     | Clear Data Memory                                                                         | 1 <sup>Note</sup> | None          |
| SET [m]                     | Set Data Memory                                                                           | 1 <sup>Note</sup> | None          |
| CLR WDT                     | Clear Watchdog Timer                                                                      | 1                 | TO, PDF       |
| SWAP [m]                    | Swap nibbles of Data Memory                                                               | 1 <sup>Note</sup> | 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        |
|----------------------------------|-----------------------------------------------------------------|-------------------|----------------------|
| <b>Arithmetic</b>                |                                                                 |                   |                      |
| LADD A,[m]                       | Add Data Memory to ACC                                          | 2                 | Z, C, AC, OV, SC     |
| LADDM A,[m]                      | Add ACC to Data Memory                                          | 2 <sup>Note</sup> | 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 <sup>Note</sup> | 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 <sup>Note</sup> | 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 <sup>Note</sup> | Z, C, AC, OV, SC, CZ |
| LDAA [m]                         | Decimal adjust ACC for Addition with result in Data Memory      | 2 <sup>Note</sup> | C                    |
| <b>Logic Operation</b>           |                                                                 |                   |                      |
| 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 <sup>Note</sup> | Z                    |
| LORM A,[m]                       | Logical OR ACC to Data Memory                                   | 2 <sup>Note</sup> | Z                    |
| LXORM A,[m]                      | Logical XOR ACC to Data Memory                                  | 2 <sup>Note</sup> | Z                    |
| LCPL [m]                         | Complement Data Memory                                          | 2 <sup>Note</sup> | Z                    |
| LCPLA [m]                        | Complement Data Memory with result in ACC                       | 2                 | Z                    |
| <b>Increment &amp; Decrement</b> |                                                                 |                   |                      |
| LINCA [m]                        | Increment Data Memory with result in ACC                        | 2                 | Z                    |
| LINC [m]                         | Increment Data Memory                                           | 2 <sup>Note</sup> | Z                    |
| LDECA [m]                        | Decrement Data Memory with result in ACC                        | 2                 | Z                    |
| LDEC [m]                         | Decrement Data Memory                                           | 2 <sup>Note</sup> | Z                    |
| <b>Rotate</b>                    |                                                                 |                   |                      |
| LRRA [m]                         | Rotate Data Memory right with result in ACC                     | 2                 | None                 |
| LRR [m]                          | Rotate Data Memory right                                        | 2 <sup>Note</sup> | None                 |
| LRRCA [m]                        | Rotate Data Memory right through Carry with result in ACC       | 2                 | C                    |
| LRRC [m]                         | Rotate Data Memory right through Carry                          | 2 <sup>Note</sup> | C                    |
| LRLA [m]                         | Rotate Data Memory left with result in ACC                      | 2                 | None                 |
| LRL [m]                          | Rotate Data Memory left                                         | 2 <sup>Note</sup> | None                 |
| LRLCA [m]                        | Rotate Data Memory left through Carry with result in ACC        | 2                 | C                    |
| LRLC [m]                         | Rotate Data Memory left through Carry                           | 2 <sup>Note</sup> | C                    |
| <b>Data Move</b>                 |                                                                 |                   |                      |
| LMOV A,[m]                       | Move Data Memory to ACC                                         | 2                 | None                 |
| LMOV [m],A                       | Move ACC to Data Memory                                         | 2 <sup>Note</sup> | None                 |
| <b>Bit Operation</b>             |                                                                 |                   |                      |
| LCLR [m].i                       | Clear bit of Data Memory                                        | 2 <sup>Note</sup> | None                 |
| LSET [m].i                       | Set bit of Data Memory                                          | 2 <sup>Note</sup> | None                 |

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

|                   |                                                                                                                                             |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <b>ADC A,[m]</b>  | Add Data Memory to ACC with Carry                                                                                                           |
| Description       | The contents of the specified Data Memory, Accumulator and the carry flag are added.<br>The result is stored in the Accumulator.            |
| Operation         | $ACC \leftarrow ACC + [m] + C$                                                                                                              |
| Affected flag(s)  | OV, Z, AC, C, SC                                                                                                                            |
| <b>ADCM A,[m]</b> | Add ACC to Data Memory with Carry                                                                                                           |
| Description       | The contents of the specified Data Memory, Accumulator and the carry flag are added.<br>The result is stored in the specified Data Memory.  |
| Operation         | $[m] \leftarrow ACC + [m] + C$                                                                                                              |
| Affected flag(s)  | OV, Z, AC, C, SC                                                                                                                            |
| <b>ADD A,[m]</b>  | Add Data Memory to ACC                                                                                                                      |
| Description       | The contents of the specified Data Memory and the Accumulator are added.<br>The result is stored in the Accumulator.                        |
| Operation         | $ACC \leftarrow ACC + [m]$                                                                                                                  |
| Affected flag(s)  | OV, Z, AC, C, SC                                                                                                                            |
| <b>ADD A,x</b>    | Add immediate data to ACC                                                                                                                   |
| Description       | The contents of the Accumulator and the specified immediate data are added.<br>The result is stored in the Accumulator.                     |
| Operation         | $ACC \leftarrow ACC + x$                                                                                                                    |
| Affected flag(s)  | OV, Z, AC, C, SC                                                                                                                            |
| <b>ADDM A,[m]</b> | Add ACC to Data Memory                                                                                                                      |
| Description       | The contents of the specified Data Memory and the Accumulator are added.<br>The result is stored in the specified Data Memory.              |
| Operation         | $[m] \leftarrow ACC + [m]$                                                                                                                  |
| Affected flag(s)  | OV, Z, AC, C, SC                                                                                                                            |
| <b>AND A,[m]</b>  | 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                                                                                                                                           |
| <b>AND A,x</b>    | 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                                                                                                                                           |
| <b>ANDM A,[m]</b> | 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                                                                                                                                           |

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>CALL addr</b> | 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 $\leftarrow$ Program Counter + 1<br>Program Counter $\leftarrow$ addr                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Affected flag(s) | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>CLR [m]</b>   | Clear Data Memory                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Description      | Each bit of the specified Data Memory is cleared to 0.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Operation        | [m] $\leftarrow$ 00H                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Affected flag(s) | None                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>CLR [m].i</b> | 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                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| <b>CLR WDT</b>   | Clear Watchdog Timer                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Description      | The TO, PDF flags and the WDT are all cleared.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Operation        | WDT cleared<br>TO $\leftarrow$ 0<br>PDF $\leftarrow$ 0                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Affected flag(s) | TO, PDF                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| <b>CPL [m]</b>   | 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                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>CPLA [m]</b>  | 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                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>DAA [m]</b>   | 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<br>[m] $\leftarrow$ ACC + 06H or<br>[m] $\leftarrow$ ACC + 60H or<br>[m] $\leftarrow$ ACC + 66H                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Affected flag(s) | C                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

|                  |                                                                                                                                                                                                                                                            |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>DEC [m]</b>   | Decrement Data Memory                                                                                                                                                                                                                                      |
| Description      | Data in the specified Data Memory is decremented by 1.                                                                                                                                                                                                     |
| Operation        | $[m] \leftarrow [m] - 1$                                                                                                                                                                                                                                   |
| Affected flag(s) | Z                                                                                                                                                                                                                                                          |
| <b>DECA [m]</b>  | 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                                                                                                                                                                                                                                                          |
| <b>HALT</b>      | 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        | $TO \leftarrow 0$<br>$PDF \leftarrow 1$                                                                                                                                                                                                                    |
| Affected flag(s) | TO, PDF                                                                                                                                                                                                                                                    |
| <b>INC [m]</b>   | Increment Data Memory                                                                                                                                                                                                                                      |
| Description      | Data in the specified Data Memory is incremented by 1.                                                                                                                                                                                                     |
| Operation        | $[m] \leftarrow [m] + 1$                                                                                                                                                                                                                                   |
| Affected flag(s) | Z                                                                                                                                                                                                                                                          |
| <b>INCA [m]</b>  | 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                                                                                                                                                                                                                                                          |
| <b>JMP addr</b>  | 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                                                                                                                                                                                                                                                       |
| <b>MOV A,[m]</b> | 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                                                                                                                                                                                                                                                       |
| <b>MOV A,x</b>   | Move immediate data to ACC                                                                                                                                                                                                                                 |
| Description      | The immediate data specified is loaded into the Accumulator.                                                                                                                                                                                               |
| Operation        | $ACC \leftarrow x$                                                                                                                                                                                                                                         |
| Affected flag(s) | None                                                                                                                                                                                                                                                       |
| <b>MOV [m],A</b> | 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                                                                                                                                                                                                                                                       |

|                  |                                                                                                                                                                                                                                                                                                                  |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>NOP</b>       | No operation                                                                                                                                                                                                                                                                                                     |
| Description      | No operation is performed. Execution continues with the next instruction.                                                                                                                                                                                                                                        |
| Operation        | No operation                                                                                                                                                                                                                                                                                                     |
| Affected flag(s) | None                                                                                                                                                                                                                                                                                                             |
| <b>OR A,[m]</b>  | 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                                                                                                                                                                                                                                                                                                                |
| <b>OR A,x</b>    | 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        | $ACC \leftarrow ACC \text{ "OR" } x$                                                                                                                                                                                                                                                                             |
| Affected flag(s) | Z                                                                                                                                                                                                                                                                                                                |
| <b>ORM A,[m]</b> | 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                                                                                                                                                                                                                                                                                                                |
| <b>RET</b>       | Return from subroutine                                                                                                                                                                                                                                                                                           |
| Description      | The Program Counter is restored from the stack. Program execution continues at the restored address.                                                                                                                                                                                                             |
| Operation        | Program Counter $\leftarrow$ Stack                                                                                                                                                                                                                                                                               |
| Affected flag(s) | None                                                                                                                                                                                                                                                                                                             |
| <b>RET A,x</b>   | 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 $\leftarrow$ Stack<br>$ACC \leftarrow x$                                                                                                                                                                                                                                                         |
| Affected flag(s) | None                                                                                                                                                                                                                                                                                                             |
| <b>RETI</b>      | 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 $\leftarrow$ Stack<br>$EMI \leftarrow 1$                                                                                                                                                                                                                                                         |
| Affected flag(s) | None                                                                                                                                                                                                                                                                                                             |
| <b>RL [m]</b>    | 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        | $[m].(i+1) \leftarrow [m].i; (i=0\sim6)$<br>$[m].0 \leftarrow [m].7$                                                                                                                                                                                                                                             |
| Affected flag(s) | None                                                                                                                                                                                                                                                                                                             |

|                  |                                                                                                                                                                                                                                                                           |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>RLA [m]</b>   | 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        | $ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$<br>$ACC.0 \leftarrow [m].7$                                                                                                                                                                                                      |
| Affected flag(s) | None                                                                                                                                                                                                                                                                      |
| <b>RLC [m]</b>   | 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        | $[m].(i+1) \leftarrow [m].i; (i=0\sim6)$<br>$[m].0 \leftarrow C$<br>$C \leftarrow [m].7$                                                                                                                                                                                  |
| Affected flag(s) | C                                                                                                                                                                                                                                                                         |
| <b>RLCA [m]</b>  | 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        | $ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$<br>$ACC.0 \leftarrow C$<br>$C \leftarrow [m].7$                                                                                                                                                                                  |
| Affected flag(s) | C                                                                                                                                                                                                                                                                         |
| <b>RR [m]</b>    | Rotate Data Memory right                                                                                                                                                                                                                                                  |
| Description      | The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.                                                                                                                                                                       |
| Operation        | $[m].i \leftarrow [m].(i+1); (i=0\sim6)$<br>$[m].7 \leftarrow [m].0$                                                                                                                                                                                                      |
| Affected flag(s) | None                                                                                                                                                                                                                                                                      |
| <b>RRA [m]</b>   | Rotate Data Memory right with result in ACC                                                                                                                                                                                                                               |
| Description      | Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.                                                                          |
| Operation        | $ACC.i \leftarrow [m].(i+1); (i=0\sim6)$<br>$ACC.7 \leftarrow [m].0$                                                                                                                                                                                                      |
| Affected flag(s) | None                                                                                                                                                                                                                                                                      |
| <b>RRC [m]</b>   | Rotate Data Memory right through Carry                                                                                                                                                                                                                                    |
| Description      | The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.                                                                                                  |
| Operation        | $[m].i \leftarrow [m].(i+1); (i=0\sim6)$<br>$[m].7 \leftarrow C$<br>$C \leftarrow [m].0$                                                                                                                                                                                  |
| Affected flag(s) | C                                                                                                                                                                                                                                                                         |

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>RRCA [m]</b>   | Rotate Data Memory right through Carry with result in ACC                                                                                                                                                                                                                                                                                                                                                                                 |
| Description       | Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.                                                                                                                                                                    |
| Operation         | $ACC.i \leftarrow [m].(i+1); (i=0\sim6)$<br>$ACC.7 \leftarrow C$<br>$C \leftarrow [m].0$                                                                                                                                                                                                                                                                                                                                                  |
| Affected flag(s)  | C                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| <b>SBC A,[m]</b>  | 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         | $ACC \leftarrow ACC - [m] - \overline{C}$                                                                                                                                                                                                                                                                                                                                                                                                 |
| Affected flag(s)  | OV, Z, AC, C, SC, CZ                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>SBC A, x</b>   | 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         | $ACC \leftarrow ACC - [m] - \overline{C}$                                                                                                                                                                                                                                                                                                                                                                                                 |
| Affected flag(s)  | OV, Z, AC, C, SC, CZ                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>SBCM A,[m]</b> | 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         | $[m] \leftarrow ACC - [m] - \overline{C}$                                                                                                                                                                                                                                                                                                                                                                                                 |
| Affected flag(s)  | OV, Z, AC, C, SC, CZ                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>SDZ [m]</b>    | 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         | $[m] \leftarrow [m] - 1$<br>Skip if $[m]=0$                                                                                                                                                                                                                                                                                                                                                                                               |
| Affected flag(s)  | None                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| <b>SDZA [m]</b>   | 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         | $ACC \leftarrow [m] - 1$<br>Skip if $ACC=0$                                                                                                                                                                                                                                                                                                                                                                                               |
| Affected flag(s)  | None                                                                                                                                                                                                                                                                                                                                                                                                                                      |

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SET [m]</b>   | Set Data Memory                                                                                                                                                                                                                                                                                                                                                                                                                          |
| Description      | Each bit of the specified Data Memory is set to 1.                                                                                                                                                                                                                                                                                                                                                                                       |
| Operation        | $[m] \leftarrow FFH$                                                                                                                                                                                                                                                                                                                                                                                                                     |
| Affected flag(s) | None                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>SET [m].i</b> | 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                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>SIZ [m]</b>   | 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        | $[m] \leftarrow [m] + 1$<br>Skip if $[m]=0$                                                                                                                                                                                                                                                                                                                                                                                              |
| Affected flag(s) | None                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>SIZA [m]</b>  | 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        | $ACC \leftarrow [m] + 1$<br>Skip if $ACC=0$                                                                                                                                                                                                                                                                                                                                                                                              |
| Affected flag(s) | None                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>SNZ [m].i</b> | 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                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>SNZ [m]</b>   | 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                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| <b>SUB A,[m]</b> | 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                                                                                                                                                                                                                                                                                                                                                                                                                     |

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>SUBM A,[m]</b> | 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                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>SUB A,x</b>    | 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                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>SWAP [m]</b>   | 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                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>SWAPA [m]</b>  | Swap nibbles of Data Memory with result in ACC                                                                                                                                                                                                                                                                                                                                                                    |
| Description       | The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.                                                                                                                                                                                                                                    |
| Operation         | $ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$<br>$ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$                                                                                                                                                                                                                                                                                                                  |
| Affected flag(s)  | None                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>SZ [m]</b>     | 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                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>SZA [m]</b>    | Skip if Data Memory is 0 with data movement to ACC                                                                                                                                                                                                                                                                                                                                                                |
| Description       | The contents of the specified Data Memory are copied to the Accumulator. If the value is zero, the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.                                                                      |
| Operation         | $ACC \leftarrow [m]$<br>Skip if $[m]=0$                                                                                                                                                                                                                                                                                                                                                                           |
| Affected flag(s)  | None                                                                                                                                                                                                                                                                                                                                                                                                              |
| <b>SZ [m].i</b>   | 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                                                                                                                                                                                                                                                                                                                                                                                                              |



|                    |                                                                                                                                                                                                                  |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>TABRD [m]</b>   | 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)<br>TBLH ← program code (high byte)                                                                                                                                                 |
| Affected flag(s)   | None                                                                                                                                                                                                             |
| <b>TABRDL [m]</b>  | Read table (last page) to TBLH and Data Memory                                                                                                                                                                   |
| Description        | The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.                                                        |
| Operation          | [m] ← program code (low byte)<br>TBLH ← program code (high byte)                                                                                                                                                 |
| Affected flag(s)   | None                                                                                                                                                                                                             |
| <b>ITABRD [m]</b>  | Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory                                                                                                                    |
| Description        | Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.    |
| Operation          | [m] ← program code (low byte)<br>TBLH ← program code (high byte)                                                                                                                                                 |
| Affected flag(s)   | None                                                                                                                                                                                                             |
| <b>ITABRDL [m]</b> | Increment table pointer low byte first and read table (last page) to TBLH and Data Memory                                                                                                                        |
| Description        | Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH. |
| Operation          | [m] ← program code (low byte)<br>TBLH ← program code (high byte)                                                                                                                                                 |
| Affected flag(s)   | None                                                                                                                                                                                                             |
| <b>XOR A,[m]</b>   | 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                                                                                                                                                                                                                |
| <b>XORM A,[m]</b>  | 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                                                                                                                                                                                                                |
| <b>XOR A,x</b>     | 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.

|                    |                                                                                                                                            |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------|
| <b>LADC A,[m]</b>  | Add Data Memory to ACC with Carry                                                                                                          |
| Description        | The contents of the specified Data Memory, Accumulator and the carry flag are added.<br>The result is stored in the Accumulator.           |
| Operation          | $ACC \leftarrow ACC + [m] + C$                                                                                                             |
| Affected flag(s)   | OV, Z, AC, C, SC                                                                                                                           |
| <b>LADCM A,[m]</b> | Add ACC to Data Memory with Carry                                                                                                          |
| Description        | The contents of the specified Data Memory, Accumulator and the carry flag are added.<br>The result is stored in the specified Data Memory. |
| Operation          | $[m] \leftarrow ACC + [m] + C$                                                                                                             |
| Affected flag(s)   | OV, Z, AC, C, SC                                                                                                                           |
| <b>LADD A,[m]</b>  | Add Data Memory to ACC                                                                                                                     |
| Description        | The contents of the specified Data Memory and the Accumulator are added.<br>The result is stored in the Accumulator.                       |
| Operation          | $ACC \leftarrow ACC + [m]$                                                                                                                 |
| Affected flag(s)   | OV, Z, AC, C, SC                                                                                                                           |
| <b>LADDM A,[m]</b> | Add ACC to Data Memory                                                                                                                     |
| Description        | The contents of the specified Data Memory and the Accumulator are added.<br>The result is stored in the specified Data Memory.             |
| Operation          | $[m] \leftarrow ACC + [m]$                                                                                                                 |
| Affected flag(s)   | OV, Z, AC, C, SC                                                                                                                           |
| <b>LAND A,[m]</b>  | 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                                                                                                                                          |
| <b>LANDM A,[m]</b> | 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                                                                                                                                          |
| <b>LCLR [m]</b>    | Clear Data Memory                                                                                                                          |
| Description        | Each bit of the specified Data Memory is cleared to 0.                                                                                     |
| Operation          | $[m] \leftarrow 00H$                                                                                                                       |
| Affected flag(s)   | None                                                                                                                                       |
| <b>LCLR [m].i</b>  | 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                                                                                                                                       |

|                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>LCPL [m]</b>  | 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                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>LCPLA [m]</b> | 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                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>LDAA [m]</b>  | 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<br>$[m] \leftarrow ACC + 06H$ or<br>$[m] \leftarrow ACC + 60H$ or<br>$[m] \leftarrow ACC + 66H$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Affected flag(s) | C                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>LDEC [m]</b>  | Decrement Data Memory                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Description      | Data in the specified Data Memory is decremented by 1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Operation        | $[m] \leftarrow [m] - 1$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Affected flag(s) | Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>LDECA [m]</b> | 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                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>LINC [m]</b>  | Increment Data Memory                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| Description      | Data in the specified Data Memory is incremented by 1.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| Operation        | $[m] \leftarrow [m] + 1$                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Affected flag(s) | Z                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| <b>LINCA [m]</b> | 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                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

|                   |                                                                                                                                                                                                                                                                           |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>LMOV A,[m]</b> | 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                                                                                                                                                                                                                                                                      |
| <b>LMOV [m],A</b> | 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                                                                                                                                                                                                                                                                      |
| <b>LOR A,[m]</b>  | 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                                                                                                                                                                                                                                                                         |
| <b>LORM A,[m]</b> | 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                                                                                                                                                                                                                                                                         |
| <b>LRL [m]</b>    | 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         | $[m].(i+1) \leftarrow [m].i; (i=0\sim6)$<br>$[m].0 \leftarrow [m].7$                                                                                                                                                                                                      |
| Affected flag(s)  | None                                                                                                                                                                                                                                                                      |
| <b>LRLA [m]</b>   | 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         | $ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$<br>$ACC.0 \leftarrow [m].7$                                                                                                                                                                                                      |
| Affected flag(s)  | None                                                                                                                                                                                                                                                                      |
| <b>LRLC [m]</b>   | 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         | $[m].(i+1) \leftarrow [m].i; (i=0\sim6)$<br>$[m].0 \leftarrow C$<br>$C \leftarrow [m].7$                                                                                                                                                                                  |
| Affected flag(s)  | C                                                                                                                                                                                                                                                                         |
| <b>LRLCA [m]</b>  | 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         | $ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$<br>$ACC.0 \leftarrow C$<br>$C \leftarrow [m].7$                                                                                                                                                                                  |
| Affected flag(s)  | C                                                                                                                                                                                                                                                                         |

|                    |                                                                                                                                                                                                                                                                                                                             |
|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>LRR [m]</b>     | Rotate Data Memory right                                                                                                                                                                                                                                                                                                    |
| Description        | The contents of the specified Data Memory are rotated right by 1 bit with bit 0 rotated into bit 7.                                                                                                                                                                                                                         |
| Operation          | $[m].i \leftarrow [m].(i+1); (i=0\sim6)$<br>$[m].7 \leftarrow [m].0$                                                                                                                                                                                                                                                        |
| Affected flag(s)   | None                                                                                                                                                                                                                                                                                                                        |
| <b>LRRA [m]</b>    | Rotate Data Memory right with result in ACC                                                                                                                                                                                                                                                                                 |
| Description        | Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.                                                                                                                            |
| Operation          | $ACC.i \leftarrow [m].(i+1); (i=0\sim6)$<br>$ACC.7 \leftarrow [m].0$                                                                                                                                                                                                                                                        |
| Affected flag(s)   | None                                                                                                                                                                                                                                                                                                                        |
| <b>LRRC [m]</b>    | Rotate Data Memory right through Carry                                                                                                                                                                                                                                                                                      |
| Description        | The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.                                                                                                                                                    |
| Operation          | $[m].i \leftarrow [m].(i+1); (i=0\sim6)$<br>$[m].7 \leftarrow C$<br>$C \leftarrow [m].0$                                                                                                                                                                                                                                    |
| Affected flag(s)   | C                                                                                                                                                                                                                                                                                                                           |
| <b>LRRCA [m]</b>   | Rotate Data Memory right through Carry with result in ACC                                                                                                                                                                                                                                                                   |
| Description        | Data in the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.                                                      |
| Operation          | $ACC.i \leftarrow [m].(i+1); (i=0\sim6)$<br>$ACC.7 \leftarrow C$<br>$C \leftarrow [m].0$                                                                                                                                                                                                                                    |
| Affected flag(s)   | C                                                                                                                                                                                                                                                                                                                           |
| <b>LSBC A,[m]</b>  | 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          | $ACC \leftarrow ACC - [m] - \overline{C}$                                                                                                                                                                                                                                                                                   |
| Affected flag(s)   | OV, Z, AC, C, SC, CZ                                                                                                                                                                                                                                                                                                        |
| <b>LSBCM A,[m]</b> | 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          | $[m] \leftarrow ACC - [m] - \overline{C}$                                                                                                                                                                                                                                                                                   |
| Affected flag(s)   | OV, Z, AC, C, SC, CZ                                                                                                                                                                                                                                                                                                        |

|                   |                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>LSDZ [m]</b>   | 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         | $[m] \leftarrow [m] - 1$<br>Skip if $[m]=0$                                                                                                                                                                                                                                                                                                                                                                                                 |
| Affected flag(s)  | None                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>LSDZA [m]</b>  | 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         | $ACC \leftarrow [m] - 1$<br>Skip if $ACC=0$                                                                                                                                                                                                                                                                                                                                                                                                 |
| Affected flag(s)  | None                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>LSET [m]</b>   | Set Data Memory                                                                                                                                                                                                                                                                                                                                                                                                                             |
| Description       | Each bit of the specified Data Memory is set to 1.                                                                                                                                                                                                                                                                                                                                                                                          |
| Operation         | $[m] \leftarrow FFH$                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Affected flag(s)  | None                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>LSET [m].i</b> | 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                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>LSIZ [m]</b>   | 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         | $[m] \leftarrow [m] + 1$<br>Skip if $[m]=0$                                                                                                                                                                                                                                                                                                                                                                                                 |
| Affected flag(s)  | None                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>LSIZA [m]</b>  | 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         | $ACC \leftarrow [m] + 1$<br>Skip if $ACC=0$                                                                                                                                                                                                                                                                                                                                                                                                 |
| Affected flag(s)  | None                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| <b>LSNZ [m].i</b> | 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                                                                                                                                                                                                                                                                                                                                                                                                                                        |

|                    |                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>LSNZ [m]</b>    | 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] $\neq$ 0                                                                                                                                                                                                                                                                                                                                                                                           |
| Affected flag(s)   | None                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>LSUB A,[m]</b>  | 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                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>LSUBM A,[m]</b> | 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                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>LSWAP [m]</b>   | 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                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>LSWAPA [m]</b>  | Swap nibbles of Data Memory with result in ACC                                                                                                                                                                                                                                                                                                                                                                 |
| Description        | The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.                                                                                                                                                                                                                                 |
| Operation          | $ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$<br>$ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$                                                                                                                                                                                                                                                                                                               |
| Affected flag(s)   | None                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>LSZ [m]</b>     | 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                                                                                                                                                                                                                                                                                                                                                                                                           |
| <b>LSZA [m]</b>    | 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          | $ACC \leftarrow [m]$<br>Skip if [m]=0                                                                                                                                                                                                                                                                                                                                                                          |
| Affected flag(s)   | None                                                                                                                                                                                                                                                                                                                                                                                                           |

|                     |                                                                                                                                                                                                                                                                                              |
|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>LSZ [m].i</b>    | 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                                                                                                                                                                                                                                                                                         |
| <b>LTABRD [m]</b>   | Read table (specific page) to TBLH and Data Memory                                                                                                                                                                                                                                           |
| Description         | The low byte of the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.                                                                                                                       |
| Operation           | [m] ← program code (low byte)<br>TBLH ← program code (high byte)                                                                                                                                                                                                                             |
| Affected flag(s)    | None                                                                                                                                                                                                                                                                                         |
| <b>LTABRDL [m]</b>  | Read table (last page) to TBLH and Data Memory                                                                                                                                                                                                                                               |
| Description         | The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.                                                                                                                                    |
| Operation           | [m] ← program code (low byte)<br>TBLH ← program code (high byte)                                                                                                                                                                                                                             |
| Affected flag(s)    | None                                                                                                                                                                                                                                                                                         |
| <b>LITABRD [m]</b>  | Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory                                                                                                                                                                                                |
| Description         | Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.                                                                                |
| Operation           | [m] ← program code (low byte)<br>TBLH ← program code (high byte)                                                                                                                                                                                                                             |
| Affected flag(s)    | None                                                                                                                                                                                                                                                                                         |
| <b>LITABRDL [m]</b> | Increment table pointer low byte first and read table (last page) to TBLH and Data Memory                                                                                                                                                                                                    |
| Description         | Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.                                                                             |
| Operation           | [m] ← program code (low byte)<br>TBLH ← program code (high byte)                                                                                                                                                                                                                             |
| Affected flag(s)    | None                                                                                                                                                                                                                                                                                         |
| <b>LXOR A,[m]</b>   | 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                                                                                                                                                                                                                                                                                            |
| <b>LXORM A,[m]</b>  | 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                                                                                                                                                                                                                                                                                            |

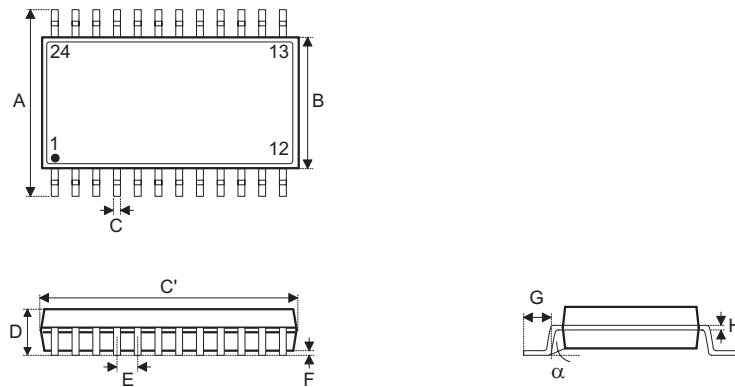


## Package Information

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.

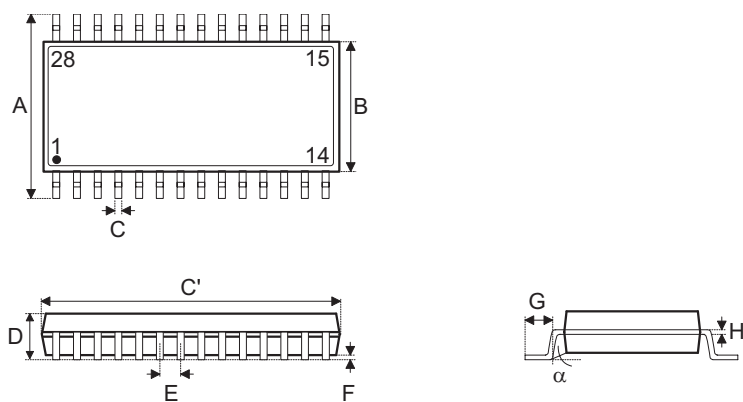
- [Package Information \(include Outline Dimensions, Product Tape and Reel Specifications\)](#)
- [The Operation Instruction of Packing Materials](#)
- [Carton information](#)

**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°    |

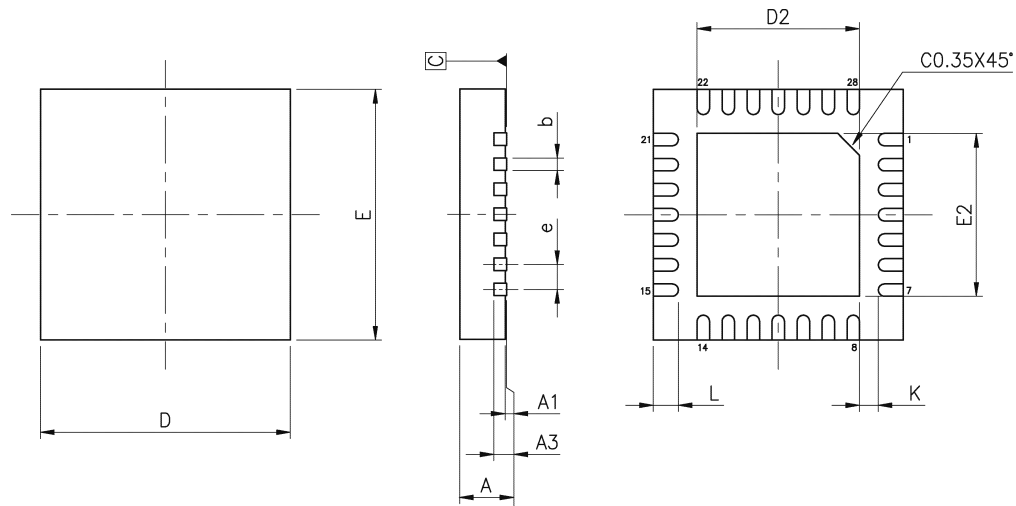
| Symbol | Dimensions in mm |           |      |
|--------|------------------|-----------|------|
|        | Min.             | Nom.      | Max. |
| A      | —                | 6.00 BSC  | —    |
| B      | —                | 3.90 BSC  | —    |
| C      | 0.20             | —         | 0.30 |
| C'     | —                | 8.66 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 |
| $\alpha$ | 0°                 | —         | 8°    |

| Symbol   | Dimensions in mm |           |      |
|----------|------------------|-----------|------|
|          | Min.             | Nom.      | Max. |
| A        | —                | 6.00 BSC  | —    |
| B        | —                | 3.90 BSC  | —    |
| C        | 0.20             | —         | 0.30 |
| C'       | —                | 9.90 BSC  | —    |
| D        | —                | —         | 1.75 |
| E        | —                | 0.635 BSC | —    |
| F        | 0.10             | —         | 0.25 |
| G        | 0.41             | —         | 1.27 |
| H        | 0.10             | —         | 0.25 |
| $\alpha$ | 0°               | —         | 8°   |

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


| Symbol | Dimensions in inch |           |       |
|--------|--------------------|-----------|-------|
|        | Min.               | Nom.      | Max.  |
| A      | 0.028              | 0.030     | 0.031 |
| A1     | 0.000              | 0.001     | 0.002 |
| A3     | —                  | 0.008 REF | —     |
| b      | 0.006              | 0.008     | 0.010 |
| D      | —                  | 0.157 BSC | —     |
| E      | —                  | 0.157 BSC | —     |
| e      | —                  | 0.016 BSC | —     |
| D2     | 0.091              | —         | 0.104 |
| E2     | 0.091              | —         | 0.104 |
| L      | 0.012              | 0.016     | 0.020 |

| Symbol | Dimensions in mm |           |      |
|--------|------------------|-----------|------|
|        | Min.             | Nom.      | Max. |
| A      | 0.70             | 0.75      | 0.80 |
| A1     | 0.00             | 0.02      | 0.05 |
| A3     | —                | 0.203 REF | —    |
| b      | 0.15             | 0.20      | 0.25 |
| D      | —                | 4.00 BSC  | —    |
| E      | —                | 4.00 BSC  | —    |
| e      | —                | 0.40 BSC  | —    |
| D2     | 2.30             | —         | 2.65 |
| E2     | 2.30             | —         | 2.65 |
| L      | 0.30             | 0.40      | 0.50 |

Copyright© 2023 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

The information provided in this document has been produced with reasonable care and attention before publication, however, HOLTEK does not guarantee that the information is completely accurate. The information contained in this publication is provided for reference only and may be superseded by updates. HOLTEK disclaims any expressed, implied or statutory warranties, including but not limited to suitability for commercialization, satisfactory quality, specifications, characteristics, functions, fitness for a particular purpose, and non-infringement of any third-party's rights. HOLTEK disclaims all liability arising from the information and its application. In addition, HOLTEK does not recommend the use of HOLTEK's products where there is a risk of personal hazard due to malfunction or other reasons. HOLTEK hereby declares that it does not authorise the use of these products in life-saving, life-sustaining or safety critical components. Any use of HOLTEK's products in life-saving/sustaining or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold HOLTEK harmless from any damages, claims, suits, or expenses resulting from such use. The information provided in this document, including but not limited to the content, data, examples, materials, graphs, and trademarks, is the intellectual property of HOLTEK (and its licensors, where applicable) and is protected by copyright law and other intellectual property laws. No license, express or implied, to any intellectual property right, is granted by HOLTEK herein. HOLTEK reserves the right to revise the information described in the document at any time without prior notice. For the latest information, please contact us.