



---

Impedance & Electrochemical LCD Flash MCU

**BH67F2495**

Revision: V1.51 Date: April 22, 2024

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

## Table of Contents

<b>Features</b> .....	<b>8</b>
CPU Features .....	8
Peripheral Features.....	8
<b>Development Tools</b> .....	<b>9</b>
<b>General Description</b> .....	<b>10</b>
<b>Block Diagram</b> .....	<b>11</b>
<b>Pin Assignment</b> .....	<b>12</b>
<b>Pin Descriptions</b> .....	<b>14</b>
<b>Absolute Maximum Ratings</b> .....	<b>19</b>
<b>D.C. Characteristics</b> .....	<b>20</b>
Operating Voltage Characteristics.....	20
Operating Current Characteristics.....	20
Standby Current Characteristics .....	21
<b>A.C. Characteristics</b> .....	<b>22</b>
Internal High Speed Oscillator – HIRC – Frequency Accuracy .....	22
Internal Low Speed Oscillator Characteristics – LIRC .....	22
External Low Speed Crystal Oscillator Characteristics – LXT .....	22
Operating Frequency Characteristic Curves .....	23
System Start Up Time Characteristics .....	23
<b>Input/Output Characteristics</b> .....	<b>24</b>
Input/Output (without Multi-power) D.C. Characteristics .....	24
Input/Output (with Multi-power) D.C. Characteristics .....	25
<b>Memory Characteristics</b> .....	<b>25</b>
<b>LVD/LVR Electrical Characteristics</b> .....	<b>26</b>
<b>Analog Front End Circuit Characteristics</b> .....	<b>27</b>
LDO Electrical Characteristics .....	27
Operational Amplifier Electrical Characteristics .....	28
Internal Reference Voltage Characteristics.....	28
Analog Switch Electrical Characteristics .....	28
12-bit D/A Converter Electrical Characteristics .....	29
A/D Converter Electrical Characteristics.....	29
PGA Electrical Characteristics .....	30
Effective Number of Bits – ENOB .....	30
<b>LCD Electrical Characteristics</b> .....	<b>31</b>
<b>USB Electrical Characteristics</b> .....	<b>32</b>
<b>I<sup>2</sup>C Electrical Characteristics</b> .....	<b>33</b>
<b>Power-on Reset Characteristics</b> .....	<b>34</b>

<b>System Architecture .....</b>	<b>34</b>
Clocking and Pipelining.....	34
Program Counter.....	35
Stack .....	36
Arithmetic and Logic Unit – ALU .....	38
<b>Flash Program Memory .....</b>	<b>39</b>
Structure.....	39
Special Vectors .....	40
Look-up Table.....	40
Table Program Example.....	40
In Circuit Programming – ICP .....	41
On-Chip Debug Support – OCDS .....	42
In Application Programming – IAP .....	42
In System Programming – ISP.....	57
<b>Data Memory .....</b>	<b>58</b>
Structure.....	58
Data Memory Addressing.....	59
General Purpose Data Memory .....	59
Special Purpose Data Memory .....	60
<b>Special Function Register Description.....</b>	<b>61</b>
Indirect Addressing Registers – IAR0, IAR1, IAR2 .....	61
Memory Pointers – MP0, MP1H/MP1L, MP2H/MP2L.....	61
Program Memory Bank Pointer – PBP.....	63
Accumulator – ACC.....	63
Program Counter Low Byte Register – PCL.....	63
Look-up Table Registers – TBLP, TBHP, TBLH.....	63
Option Memory Mapping Register – ORMC .....	64
Status Register – STATUS .....	64
<b>EEPROM Data Memory.....</b>	<b>66</b>
EEPROM Data Memory Structure .....	66
EEPROM Registers .....	66
Read Operation from the EEPROM.....	68
Page Erase Operation to the EEPROM .....	69
Write Operation to the EEPROM .....	69
Write Protection.....	71
EEPROM Interrupt.....	71
Programming Considerations.....	71
<b>Oscillators .....</b>	<b>74</b>
Oscillator Overview .....	74
System Clock Configurations.....	74
Internal PLL Frequency Generator.....	75
External Crystal/Ceramic Oscillator – HXT .....	76
Internal High Speed RC Oscillator – HIRC .....	77

Internal 32kHz Oscillator – LIRC .....	77
External 32.768kHz Crystal Oscillator – LXT .....	77
<b>Operating Modes and System Clocks .....</b>	<b>79</b>
System Clocks .....	79
System Operation Modes.....	80
Control Registers .....	81
Operating Mode Switching .....	84
Standby Current Considerations .....	87
Wake-up .....	88
<b>Watchdog Timer .....</b>	<b>89</b>
Watchdog Timer Clock Source.....	89
Watchdog Timer Control Register .....	89
Watchdog Timer Operation .....	90
<b>Reset and Initialisation.....</b>	<b>91</b>
Reset Functions .....	91
Reset Initial Conditions .....	95
<b>Input/Output Ports .....</b>	<b>100</b>
Pull-high Resistors .....	101
Port A Wake-up .....	101
I/O Port Control Registers .....	102
I/O Port Power Source Control.....	102
Pin-shared Functions .....	103
I/O Pin Structures .....	111
READ PORT function.....	111
Programming Considerations.....	112
<b>Timer Modules – TM .....</b>	<b>113</b>
Introduction .....	113
TM Operation .....	113
TM Clock Source.....	113
TM Interrupts.....	114
TM External Pins.....	114
Programming Considerations.....	115
<b>Standard Type TM – STM .....</b>	<b>116</b>
Standard Type TM Operation.....	116
Standard Type TM Register Description .....	117
Standard Type TM Operation Modes .....	121
<b>Periodic Type TM – PTM.....</b>	<b>131</b>
Periodic Type TM Operation.....	131
Periodic Type TM Register Description .....	131
Periodic Type TM Operation Modes.....	136
<b>Audio Type TM – ATM .....</b>	<b>145</b>
Audio Type TM Operation .....	145
Audio Type TM Register Description.....	146

Audio Type TM Operation Modes.....	150
Audio PWM Output Usage Description .....	159
<b>Internal Power Supply .....</b>	<b>163</b>
<b>Internal Reference Voltage Generator .....</b>	<b>164</b>
Internal Reference Voltage Register Description .....	164
<b>D/A Converter – DAC .....</b>	<b>165</b>
D/A Converter Register Description .....	165
<b>Operational Amplifier – OPA.....</b>	<b>168</b>
OPA Register Description.....	168
Input Offset Calibration .....	172
<b>Bio-impedance Analysis Function .....</b>	<b>173</b>
Sine Wave Generator.....	173
Bio-impedance Analysis Register Description.....	175
<b>Analog to Digital Converter – ADC.....</b>	<b>178</b>
A/D Converter Overview .....	178
A/D Converter Register Description .....	179
A/D Converter Rate Definition.....	184
A/D Converter Operation.....	185
Summary of A/D Conversion Steps.....	186
Programming Considerations.....	187
A/D Converter Transfer Function .....	187
A/D Converted Data .....	188
A/D Converted Data to Voltage .....	188
<b>Serial Interface Module – SIM .....</b>	<b>189</b>
SPI Interface .....	189
I <sup>2</sup> C Interface .....	197
<b>UART Interface.....</b>	<b>207</b>
UART External Pins .....	208
UART Single Wire Mode .....	208
UART Data Transfer Scheme.....	208
UART Status and Control Registers.....	209
Baud Rate Generator .....	216
UART Setup and Control.....	217
UART Transmitter.....	219
UART Receiver .....	220
Managing Receiver Errors .....	222
UART Interrupt Structure.....	223
UART Power Down and Wake-up.....	224
<b>USB Interface .....</b>	<b>225</b>
USB Interface Operation.....	225
USB Interface Registers.....	225
USB Suspend Mode and Wake-Up.....	234
USB Interrupts.....	235

<b>LCD Driver .....</b>	<b>236</b>
LCD Display Memory .....	236
LCD Clock Source.....	237
LCD Registers.....	238
LCD Voltage Source and Biasing.....	240
LCD Reset Status .....	243
LCD Driver Output.....	243
LCD Charge Pump.....	252
Programming Considerations.....	252
<b>Low Voltage Detector – LVD .....</b>	<b>253</b>
LVD Register .....	253
LVD Operation.....	254
<b>16-bit Multiplication Division Unit – MDU.....</b>	<b>255</b>
MDU Registers.....	255
MDU Operation .....	256
<b>Cyclic Redundancy Check – CRC .....</b>	<b>258</b>
CRC Registers .....	258
CRC Operation.....	259
CRC Computation.....	259
<b>Interrupts .....</b>	<b>261</b>
Interrupt Registers.....	261
Interrupt Operation.....	269
External Interrupts.....	269
A/D Converter Interrupt.....	270
Multi-function Interrupts.....	270
Timer Module Interrupts .....	270
EEPROM Interrupt.....	271
LVD Interrupt.....	271
Time Base Interrupts.....	271
USB Interrupt .....	273
USB Start of Frame Interrupt .....	273
Serial Interface Module Interrupt.....	273
UART Interrupts .....	273
Interrupt Wake-up Function.....	274
Programming Considerations.....	274
<b>Application Circuits.....</b>	<b>275</b>
<b>Instruction Set.....</b>	<b>276</b>
Introduction .....	276
Instruction Timing .....	276
Moving and Transferring Data.....	276
Arithmetic Operations.....	276
Logical and Rotate Operation .....	277
Branches and Control Transfer .....	277

---

Bit Operations .....	277
Table Read Operations .....	277
Other Operations.....	277
<b>Instruction Set Summary .....</b>	<b>278</b>
Table Conventions.....	278
Extended Instruction Set.....	280
<b>Instruction Definition.....</b>	<b>282</b>
Extended Instruction Definition .....	291
<b>Package Information .....</b>	<b>298</b>
64-pin LQFP (7mm×7mm) Outline Dimensions .....	299
80-pin LQFP (10mm×10mm) Outline Dimensions .....	300

## Features

### CPU Features

- Operating Voltage
  - ♦  $V_{DD}$  (MCU):
    - $f_{SYS}=6\text{MHz}$ : 2.2V~5.5V
    - $f_{SYS}=12\text{MHz}$ : 2.7V~5.5V
    - $f_{SYS}=16\text{MHz}$ : 3.3V~5.5V
  - ♦  $V_{DD}$  (USB mode):
    - $f_{SYS}=6\text{MHz}/12\text{MHz}/16\text{MHz}$ : 3.6V~5.5V
- Up to 0.25 $\mu\text{s}$  instruction cycle with 16MHz system clock at  $V_{DD}=5\text{V}$
- Power down and wake-up functions to reduce power consumption
- Oscillator Types
  - ♦ External High Speed Crystal – HXT
  - ♦ Internal High Speed 12MHz RC – HIRC
  - ♦ External Low Speed 32.768kHz Crystal – LXT
  - ♦ Internal Low Speed 32kHz RC – LIRC
- Fully integrated internal oscillators require no external components
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 109 powerful instructions
- 16-level subroutine nesting
- Bit manipulation instruction

### Peripheral Features

- Flash Program Memory: 64K $\times$ 16
- Data Memory: 4096 $\times$ 8
- True EEPROM Memory: 4096 $\times$ 8
- In Application Programming function – IAP
- In System Programming function – ISP
- Internal On-Chip Debug Support function – OCDS
- Watchdog Timer function
- 49 bidirectional I/O lines
- 6 external interrupt lines shared with I/O pins
- Multiple Timer Modules for time measure, input capture, compare match output, PWM output function or single pulse output function
  - ♦ One 16-bit Standard Type TM – STM
  - ♦ Two 10-bit Periodic Type TMs – PTM0~PTM1
  - ♦ One 10-bit Audio Type TM – ATM
- Dual Time Base functions for generation of fixed time interrupt signals



- Glucose Meter and Body Fat Scale AFE
  - ♦ Multi-channel 24-bit resolution Delta Sigma A/D Converter
  - ♦ Internal LDO
  - ♦ Internal Reference Voltage Generator
  - ♦ Two 12-bit D/A Converters
  - ♦ 2 internal Operational Amplifiers
  - ♦ Bio-impedance Analysis Function
- LCD driver function
  - ♦ SEGs×COMs: 36×4, 34×6, 32×8
  - ♦ Duty type: 1/4 duty, 1/6 duty or 1/8 duty
  - ♦ Bias level: 1/3 bias or 1/4 bias
  - ♦ Bias type: R type or C type
  - ♦ Waveform type: type A or type B
- USB interface
  - ♦ USB 2.0 Full Speed compatible
  - ♦ 4 endpoints supported including endpoint 0
  - ♦ All endpoints except endpoint 0 can support interrupt and bulk transfer
  - ♦ All endpoints except endpoint 0 have a fixed FIFO size of 16, 32 and 64 bytes respectively
  - ♦ Endpoint 0 support control transfer
  - ♦ Endpoint 0 has 8 byte FIFO
  - ♦ Support 3.3V LDO and internal UDP 1.5kΩ pull-up resistor
  - ♦ Internal 12MHz RC oscillator with ±0.25% accuracy for all USB modes
- Serial Interface Module – SIM for SPI or I<sup>2</sup>C interface
- Fully-duplex / Half-duplex Universal Asynchronous Receiver and Transmitter Interface – UART
- Integrated Multiplier/Divider Unit – MDU
- Integrated 16-bit Cyclic Redundancy Check function – CRC
- Low voltage reset function
- Low voltage detect function
- Package types: 64/80-pin LQFP

## Development Tools

For rapid product development and to simplify device parameter setting, Holtek has provided relevant development tools which users can download from the following link:

[https://www.holtek.com/page/detail/dev\\_plat/voice\\_mcu\\_workshop](https://www.holtek.com/page/detail/dev_plat/voice_mcu_workshop)

## General Description

The device is a Flash Memory 8-bit high performance RISC architecture microcontroller device with Glucose Meter and Body Fat Scale AFE module, USB interface and LCD driver that specifically designed for USB Glucose Meter and Body Fat Scale with LCD display applications.

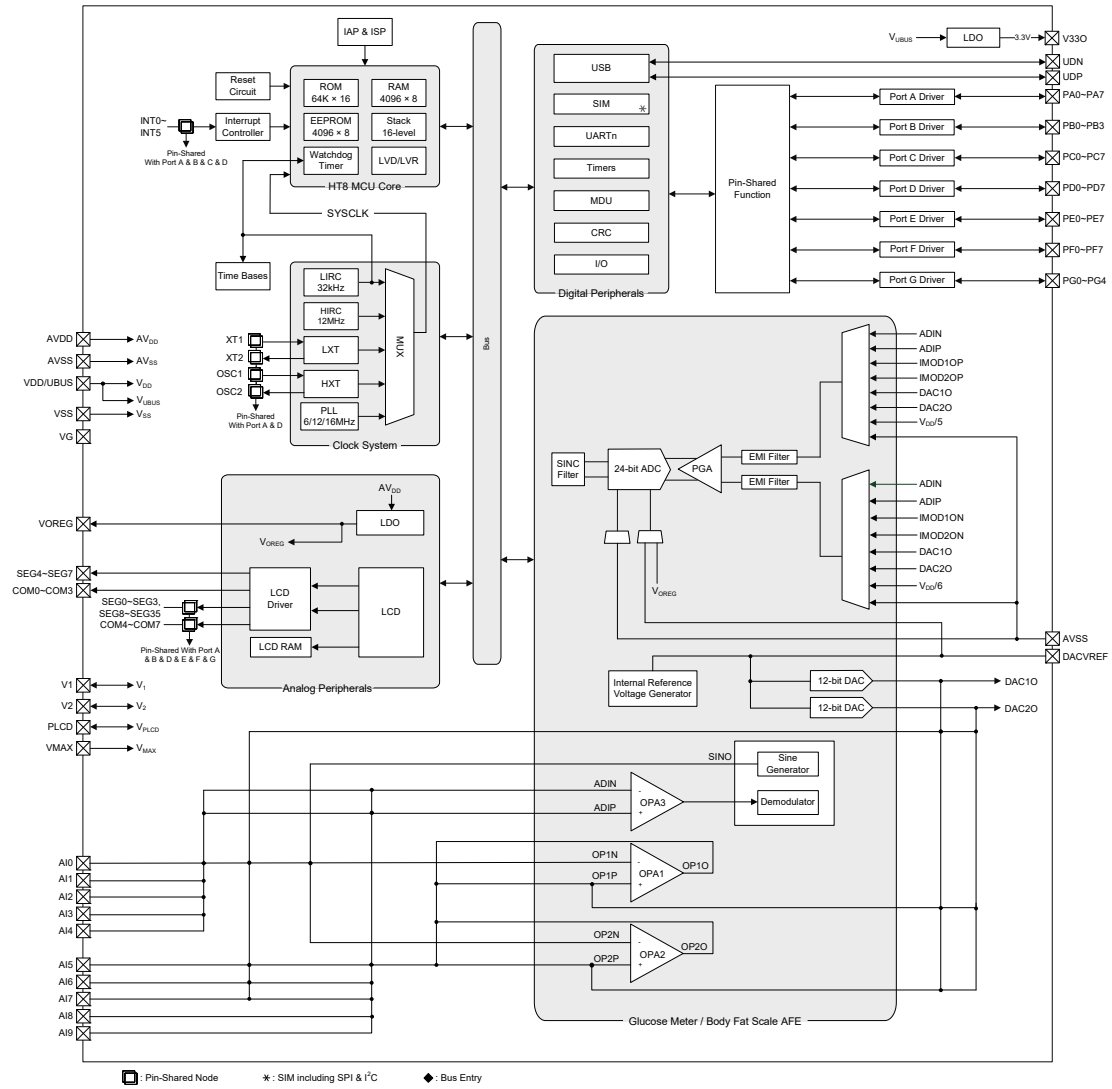
For memory features, the Flash Memory offers users the convenience of Flash Memory multi-programming features. Other memory includes an area of RAM Data Memory as well as an area of true EEPROM memory for storage of non-volatile data such as serial numbers, calibration data etc. By using the In Application 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 24-bit Delta Sigma A/D converter, two 12-bit D/A converters, two internal operational amplifiers and a fully integrated LCD driver. 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, UART and USB interface functions, four 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 Detect coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

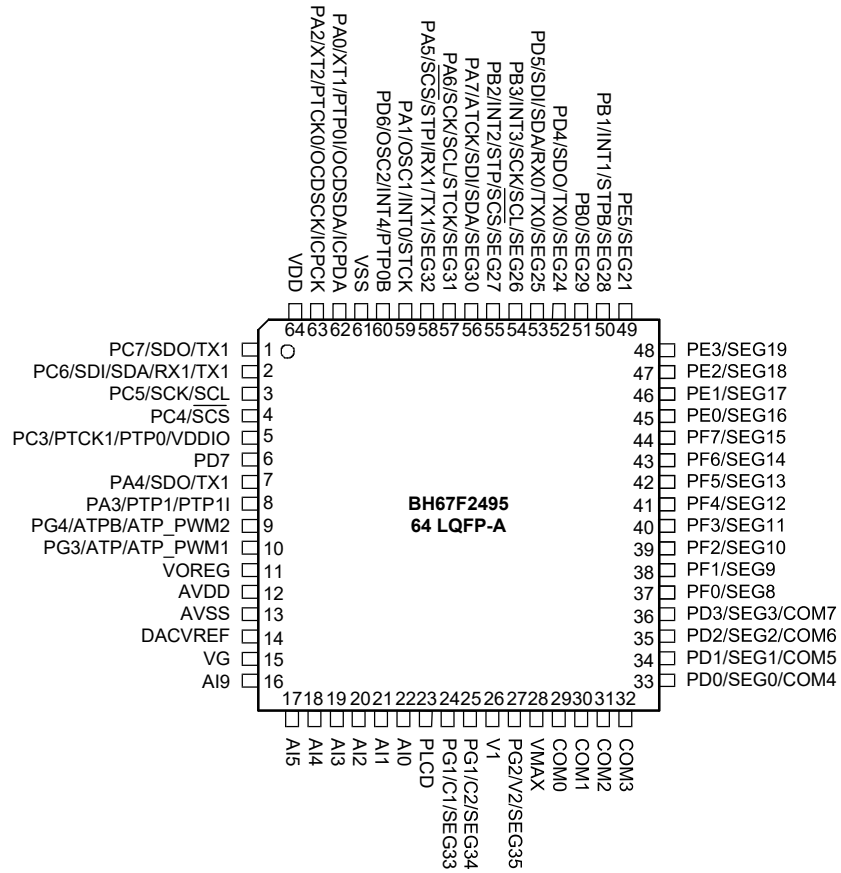
A full choice of external, internal high and low oscillator functions is provided including two fully integrated system oscillators which require no external components for their implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources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

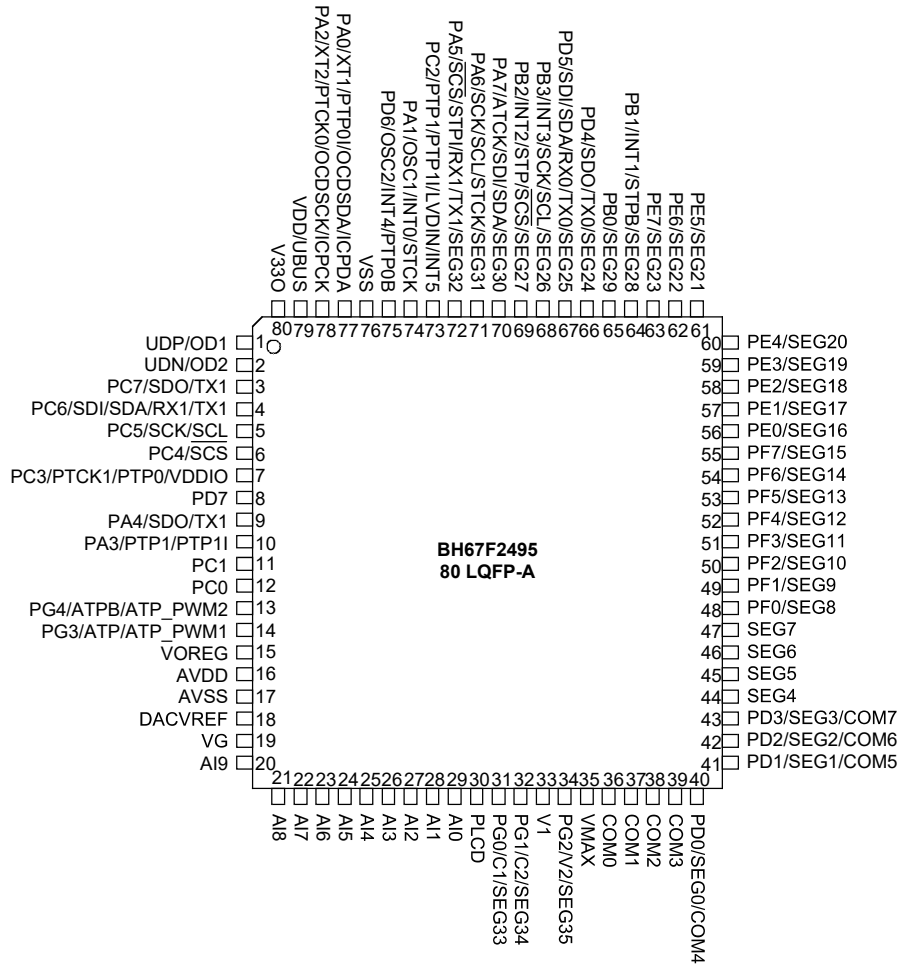
With regard to USB Glucose Meter and Body Fat Scale applications, the device has integrated many of the functions required by these products. These include functions such as Internal Reference Voltage generator, Internal LDO, 24-bit Delta Sigma A/D converter, 12-bit D/A Converters, operational amplifiers, LCD driver and USB interface, etc. The inclusion of flexible I/O programming features, 16-bit Cyclic Redundancy Check function, Multiplier/Divider Unit, Time Base functions along with many other features enhance the versatility of the device to suit for USB Glucose Meter and Body Fat Scale applications which require an LCD display.

**Block Diagram**



### Pin Assignment





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

## Pin Descriptions

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

Pin Name	Function	OPT	I/T	O/T	Description
PA0/XT1/PTP0I/ OCSDSA/ICPDA	PA0	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	XT1	PAS0	AN	—	LXT oscillator pin
	PTP0I	PAS0	ST	—	PTM0 capture input
	OCSDSA	—	ST	CMOS	OCDS data/address pin
	ICPDA	—	ST	CMOS	ICP data/address pin
PA1/OSC1/INT0/STCK	PA1	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OSC1	PAS0	AN	—	HXT oscillator pin
	INT0	PAS0 INTEG0 INTC0	ST	—	External interrupt
	STCK	PAS0 IFS0	ST	—	STM clock input
PA2/XT2/PTCK0/ OCDSCK/ICPCK	PA2	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	XT2	PAS0	—	AN	LXT oscillator pin
	PTCK0	PAS0	ST	—	PTM0 clock input or capture input
	OCDSCK	—	ST	—	OCDS clock pin
PA3/PTP1/PTP11	PA3	PAPU PAWU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	PTP1	PAS0	—	CMOS	PTM1 output
	PTP11	PAS0 IFS0	ST	—	PTM1 capture input
PA4/SDO/TX1	PA4	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SDO	PAS1	—	CMOS	SPI serial data output
	TX1	PAS1	—	CMOS	UART1 serial data output
PA5/ $\overline{SCS}$ /STPI/RX1/ TX1/SEG32	PA5	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	$\overline{SCS}$	PAS1 IFS0	ST	CMOS	SPI slave select pin
	STPI	PAS1	ST	—	STM capture input
	RX1/TX1	PAS1 IFS1	ST	CMOS	UART1 serial data input in full-duplex communication or UART1 serial data input/output in single wire mode communication
	SEG32	PAS1	—	AN	LCD SEG output

Pin Name	Function	OPT	I/T	O/T	Description
PA6/SCK/SCL/STCK/SEG31	PA6	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SCK	PAS1 IFS0	ST	CMOS	SPI serial clock
	SCL	PAS1 IFS0	ST	NMOS	I <sup>2</sup> C clock line
	STCK	PAS1 IFS0	ST	—	STM clock input
	SEG31	PAS1	—	AN	LCD SEG output
PA7/ATCK/SDI/SDA/SEG30	PA7	PAPU PAWU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	ATCK	PAS1	ST	—	ATM clock input
	SDI	PAS1 IFS0	ST	—	SPI serial data input
	SDA	PAS1 IFS0	ST	NMOS	I <sup>2</sup> C data line
	SEG30	PAS1	—	AN	LCD SEG output
PB0/SEG29	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG29	PBS0	—	AN	LCD SEG output
PB1/INT1/STPB/SEG28	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	INT1	PBS0 INTEG0 INTC0	ST	—	External interrupt
	STPB	PBS0	—	CMOS	STM inverted output
	SEG28	PBS0	—	AN	LCD SEG output
PB2/INT2/STP/SCS/SEG27	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	INT2	PBS0 INTEG0 MFI3	ST	—	External interrupt
	STP	PBS0	—	CMOS	STM output
	SCS	PBS0 IFS0	ST	CMOS	SPI slave select pin
	SEG27	PBS0	—	AN	LCD SEG output
PB3/INT3/SCK/SCL/SEG26	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	INT3	PBS0 INTEG0 MFI3	ST	—	External interrupt
	SCK	PBS0 IFS0	ST	CMOS	SPI serial clock
	SCL	PBS0 IFS0	ST	NMOS	I <sup>2</sup> C clock line
	SEG26	PBS0	—	AN	LCD SEG output
PC0	PC0	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-high
PC1	PC1	PCPU	ST	CMOS	General purpose I/O. Register enabled pull-high

Pin Name	Function	OPT	I/T	O/T	Description
PC2/PTP1/PTP11/ LVDIN/INT5	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTP1	PCS0	—	CMOS	PTM1 output
	PTP11	PCS0 IFS0	ST	—	PTM1 capture input
	LVDIN	PCS0	AN	—	LVD input
	INT5	PCS0 INTEG1 MFI3	ST	—	External interrupt
PC3/PTCK1/PTP0/ VDDIO	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	PTCK1	PCS0	ST	—	PTM1 clock input
	PTP0	PCS0	—	CMOS	PTM0 output
	VDDIO	PCS0	PWR	—	PC4~PC7 pin power supply
PC4/ $\overline{SCS}$	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	$\overline{SCS}$	PCS1 IFS0	ST	CMOS	SPI slave select pin
PC5/SCK/SCL	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SCK	PCS1 IFS0	ST	CMOS	SPI serial clock
	SCL	PCS1 IFS0	ST	NMOS	I <sup>2</sup> C clock line
PC6/SDI/SDA/RX1/TX1	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SDI	PCS1 IFS0	ST	—	SPI serial data input
	SDA	PCS1 IFS0	ST	NMOS	I <sup>2</sup> C data line
	RX1/TX1	PCS1 IFS1	ST	CMOS	UART1 serial data input in full-duplex communication or UART1 serial data input/output in single wire mode communication
PC7/SDO/TX1	PC7	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SDO	PCS1	—	CMOS	SPI serial data output
	TX1	PCS1	—	CMOS	UART1 serial data output
PD0/SEG0/COM4	PD0	PDP PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG0	PDS0	—	AN	LCD SEG output
	COM4	PDS0	—	AN	LCD COM output
PD1/SEG1/COM5	PD1	PDP PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG1	PDS0	—	AN	LCD SEG output
	COM5	PDS0	—	AN	LCD COM output
PD2/SEG2/COM6	PD2	PDP PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG2	PDS0	—	AN	LCD SEG output
	COM6	PDS0	—	AN	LCD COM output
PD3/SEG3/COM7	PD3	PDP PDS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG3	PDS0	—	AN	LCD SEG output
	COM7	PDS0	—	AN	LCD COM output



Pin Name	Function	OPT	I/T	O/T	Description
PD4/SDO/TX0/SEG24	PD4	PDP PDS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SDO	PDS1	—	CMOS	SPI serial data output
	TX0	PDS1	—	CMOS	UART0 serial data output
	SEG24	PDS1	—	AN	LCD SEG output
PD5/SDI/SDA/RX0/ TX0/SEG25	PD5	PDP PDS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SDI	PDS1 IFS0	ST	—	SPI serial data input
	SDA	PDS1 IFS0	ST	NMOS	I <sup>2</sup> C data line
	RX0/TX0	PDS1	ST	CMOS	UART0 serial data input in full-duplex communication or UART0 serial data input/output in single wire mode communication
	SEG25	PDS1	—	AN	LCD SEG output
PD6/OSC2/INT4/ PTP0B	PD6	PDP PDS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	OSC2	PDS1	—	AN	HXT oscillator pin
	INT4	PDS1 INTEG1 MFI3	ST	—	External interrupt
	PTP0B	PDS1	—	CMOS	PTM0 inverted output
PD7	PD7	PDP	ST	CMOS	General purpose I/O. Register enabled pull-high
PE0/SEG16	PE0	PEP PES0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG16	PES0	—	AN	LCD SEG output
PE1/SEG17	PE1	PEP PES0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG17	PES0	—	AN	LCD SEG output
PE2/SEG18	PE2	PEP PES0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG18	PES0	—	AN	LCD SEG output
PE3/SEG19	PE3	PEP PES0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG19	PES0	—	AN	LCD SEG output
PE4/SEG20	PE4	PEP PES1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG20	PES1	—	AN	LCD SEG output
PE5/SEG21	PE5	PEP PES1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG21	PES1	—	AN	LCD SEG output
PE6/SEG22	PE6	PEP PES1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG22	PES1	—	AN	LCD SEG output
PE7/SEG23	PE7	PEP PES1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG23	PES1	—	AN	LCD SEG output
PF0/SEG8	PF0	PFP PFS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG8	PFS0	—	AN	LCD SEG output

Pin Name	Function	OPT	I/T	O/T	Description
PF1/SEG9	PF1	PFFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG9	PFS0	—	AN	LCD SEG output
PF2/SEG10	PF2	PFFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG10	PFS0	—	AN	LCD SEG output
PF3/SEG11	PF3	PFFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG11	PFS0	—	AN	LCD SEG output
PF4/SEG12	PF4	PFFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG12	PFS1	—	AN	LCD SEG output
PF5/SEG13	PF5	PFFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG13	PFS1	—	AN	LCD SEG output
PF6/SEG14	PF6	PFFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG14	PFS1	—	AN	LCD SEG output
PF7/SEG15	PF7	PFFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	SEG15	PFS1	—	AN	LCD SEG output
PG0/C1/SEG33	PG0	PGPU PGS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	C1	PGS0	AN	AN	LCD voltage pump
	SEG33	PGS0	—	AN	LCD SEG output
PG1/C2/SEG34	PG1	PGPU PGS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	C2	PGS0	AN	AN	LCD voltage pump
	SEG34	PGS0	—	AN	LCD SEG output
PG2/V2/SEG35	PG2	PGPU PGS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	V2	PGS0	PWR	AN	LCD voltage pump
	SEG35	PGS0	—	AN	LCD SEG output
PG3/ATP/ATP_PWM1	PG3	PGPU PGS0	ST	CMOS	General purpose I/O. Register enabled pull-high
	ATP	PGS0	—	CMOS	ATM output
	ATP_PWM1	PGS0	—	CMOS	ATM Audio PWM Mode output 1
PG4/ATPB/ATP_PWM2	PG4	PGPU PGS1	ST	CMOS	General purpose I/O. Register enabled pull-high
	ATPB	PGS1	—	CMOS	ATM inverted output
	ATP_PWM2	PGS1	—	CMOS	ATM Audio PWM Mode output 2
COM0~COM3	COM0~COM3	—	—	AN	LCD COM output
SEG4~SEG7	SEG4~SEG7	—	—	AN	LCD SEG output
VMAX	VMAX	—	PWR	—	IC maximum voltage, connected to VDD or V1
PLCD	PLCD	—	PWR	AN	LCD power supply
V1	V1	—	PWR	AN	LCD voltage pump
AI0~AI9	AI0~AI9	—	AN	AN	AFE analog I/O
VG	VG	—	AN	—	Virtual ground
DACVREF	DACVREF	—	AN	—	D/A Converter reference voltage
VOREG	VOREG	—	—	AN	LDO regulator output
		—	AN	—	Positive power supply for D/A converter, OPA

Pin Name	Function	OPT	I/T	O/T	Description
UDP/OD1	UDP	—	ST	CMOS	USB D+ line
	OD1	—	ST	NMOS	NMOS Open Drain I/O pin
UDN/OD2	UDN	—	ST	CMOS	USB D- line
	OD2	—	ST	NMOS	NMOS Open Drain I/O pin
V33O	V33O	—	—	PWR	USB 3.3V regulator output
VDD/UBUS	VDD	—	PWR	—	Digital positive power supply
	UBUS	—	PWR	—	USB positive power supply
VSS	VSS	—	PWR	—	Digital negative power supply
AVDD	AVDD	—	PWR	—	Analog positive power supply
AVSS	AVSS	—	PWR	—	Analog negative power supply

Legend: I/T: Input type;

OPT: Optional by register option;

CMOS: CMOS output;

AN: Analog signal;

O/T: Output type;

ST: Schmitt Trigger input;

NMOS: NMOS output;

PWR: Power

## Absolute Maximum Ratings

Supply Voltage .....	$V_{SS}-0.3V$ to $6.0V$
Input Voltage .....	$V_{SS}-0.3V$ to $V_{DD}+0.3V$
Storage Temperature.....	$-60^{\circ}C$ 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 device. Functional operation of the device at other conditions beyond those listed in the specification is not implied and prolonged exposure to extreme conditions may affect device reliability.

## D.C. Characteristics

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

### Operating Voltage Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit	
V <sub>DD</sub>	Operating Voltage – HXT	f <sub>sys</sub> =6MHz	2.2	—	5.5	V	
		f <sub>sys</sub> =12MHz	2.7	—	5.5		
		f <sub>sys</sub> =16MHz	3.3	—	5.5		
	Operating Voltage – HIRC	f <sub>sys</sub> =12MHz	2.7	—	5.5	V	
	Operating Voltage – LXT	f <sub>sys</sub> =32768Hz	2.2	—	5.5	V	
	Operating Voltage – LIRC	f <sub>sys</sub> =32kHz	2.2	—	5.5	V	
	Operating Voltage – PLL		f <sub>sys</sub> =6MHz, PLEN=1	2.2	—	5.5	V
			f <sub>sys</sub> =12MHz, PLEN=1	2.7	—	5.5	
			f <sub>sys</sub> =16MHz, PLEN=1	3.3	—	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	3V	f <sub>sys</sub> =32kHz	—	16	30	μA
		5V		—	28	50	
	SLOW Mode – LXT	3V	f <sub>sys</sub> =32.768kHz	—	17	30	μA
		5V		—	30	50	
	FAST Mode – HIRC	3V	f <sub>sys</sub> =12MHz	—	1.5	2.8	mA
		5V		—	3.0	4.5	
	FAST Mode – HXT	3V	f <sub>sys</sub> =6MHz	—	0.7	1.5	mA
		5V		—	1.6	3.0	
		3V	f <sub>sys</sub> =12MHz	—	1.3	3.0	
		5V		—	2.7	6.0	
		3.3V	f <sub>sys</sub> =16MHz	—	3	6	
		5V		—	5.0	9.5	
	FAST Mode – PLL	3V	f <sub>sys</sub> =6MHz, PLEN=1	—	1.5	2.0	mA
		5V		—	3	4	
		3V	f <sub>sys</sub> =12MHz, PLEN=1	—	2.1	3.5	
		5V		—	3.8	7.0	
		3.3V	f <sub>sys</sub> =16MHz, PLEN=1	—	3.2	6.5	
		5V		—	4.5	9.0	

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

**Standby Current Characteristics**

Ta=25°C, unless otherwise specified

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V <sub>DD</sub>	Conditions					
I <sub>STB</sub>	SLEEP Mode	3V	WDT off	—	0.45	0.90	8.00	μA
		5V		—	0.5	2.0	10.0	
		3V	WDT on, f <sub>SUB</sub> =f <sub>LIRC</sub>	—	1.5	3.0	3.7	μA
		5V		—	3	5	6	
		3V	WDT on, f <sub>SUB</sub> =f <sub>LXT</sub>	—	1.5	3.0	3.7	μA
		5V		—	3	5	6	
	IDLE0 Mode – LIRC	3V	f <sub>SUB</sub> on	—	3.0	5.0	5.7	μA
		5V		—	5	10	11	
	IDLE0 Mode – LXT	3V	f <sub>SUB</sub> on	—	3.0	5.0	5.7	μA
		5V		—	5	10	11	
	IDLE1 Mode – HIRC	3V	f <sub>SUB</sub> on, f <sub>SYS</sub> =12MHz	—	0.65	1.40	1.40	mA
		5V		—	1.3	3.0	3.0	
	IDLE1 Mode – HXT	3V	f <sub>SUB</sub> on, f <sub>SYS</sub> =12MHz	—	0.9	1.5	1.5	mA
		5V		—	1.4	3.0	3.0	
	IDLE1 Mode – PLL	3V	f <sub>SUB</sub> on, f <sub>SYS</sub> =6MHz, PLEN=1	—	1.0	2.5	2.5	mA
		5V		—	2.0	3.5	3.5	
3V		f <sub>SUB</sub> on, f <sub>SYS</sub> =12MHz, PLEN=1	—	1.5	3.0	3.0	mA	
5V			—	2.5	4.0	4.0		
3.3V		f <sub>SUB</sub> on, f <sub>SYS</sub> =16MHz, PLEN=1	—	1.7	3.5	3.5	mA	
5V			—	3	5	5		

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.

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

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
f <sub>HIRC</sub>	12MHz Writer Trimmed HIRC Frequency	4.4V~5.25V	USB mode and CLKADJ=1, UDP/UDN plug in the HOST, Ta=-40°C~85°C	-0.25%	12	+0.25%	MHz
		5V	Ta=25°C	-2%	12	+2%	
		5V	Ta=0°C~70°C	-3%	12	+3%	
		5V	Ta=-40°C~85°C	-3.5%	12	+3.5%	
		2.7V~5.5V	Ta=0°C~70°C	-4%	12	+4%	
		2.7V~5.5V	Ta=-40°C~85°C	-4.5%	12	+4.5%	

- 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 rows below the 5V trim voltage rows are provided to show the values for the full V<sub>DD</sub> range operating voltage.

### Internal Low Speed Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Temp.				
f <sub>LIRC</sub>	LIRC Frequency	3V	25°C	-2%	32	+2%	kHz
		2.2V~5.5V	-40°C~85°C	-7%	32	+7%	
t <sub>START</sub>	LIRC Start Up Time	—	-40°C~85°C	—	—	100	μs

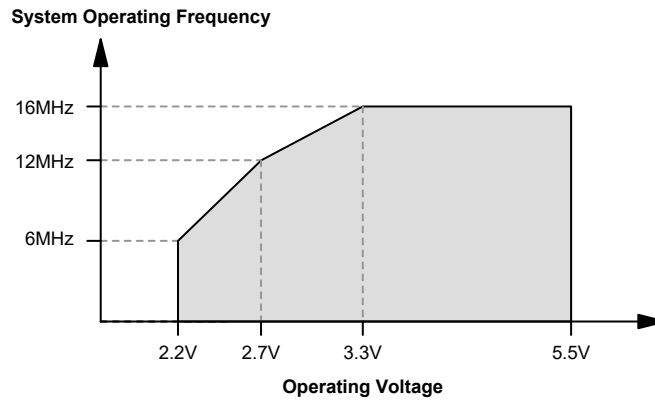
### External Low Speed Crystal Oscillator Characteristics – LXT

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
f <sub>LXT</sub>	LXT Frequency	2.2V~5.5V	—	—	32768	—	Hz
t <sub>START</sub>	LXT Start Up Time	3V	—	—	—	1000	ms
		5V	—	—	—	1000	ms
Duty Cycle	Duty Cycle	—	—	40	—	60	%
R <sub>NEG</sub>	Negative Resistance	2.2V	—	3×ESR	—	—	Ω

Note: C<sub>1</sub>, C<sub>2</sub> and R<sub>P</sub> are external components. C<sub>1</sub>=C<sub>2</sub>=10pF. R<sub>P</sub>=10MΩ. C<sub>L</sub>=7pF, ESR=30kΩ.

### 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 (Wake-up from Conditions where f <sub>sys</sub> is off)	—	f <sub>sys</sub> =f <sub>LXT</sub>	—	1024	—	t <sub>sys</sub>
		—	f <sub>sys</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HXT</sub>	—	128	—	t <sub>sys</sub>
		—	f <sub>sys</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HIRC</sub>	—	16	—	t <sub>sys</sub>
		—	f <sub>PLL</sub> off → on (PLL <sub>F</sub> =1)	—	2560	—	t <sub>PLL</sub> (48MHz)
		—	f <sub>sys</sub> =f <sub>LIRC</sub>	—	2	—	t <sub>sys</sub>
	System Start-up Time (Wake-up from Conditions where f <sub>sys</sub> is on)	—	f <sub>sys</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HXT</sub> or f <sub>HIRC</sub> or f <sub>PLL</sub>	—	2	—	t <sub>sys</sub>
		—	f <sub>sys</sub> =f <sub>LXT</sub> or f <sub>LIRC</sub>	—	2	—	t <sub>sys</sub>
	System Speed Switch Time (FAST to Slow Mode or SLOW to FAST Mode)	—	f <sub>HXT</sub> off → on (HXTF=1)	—	1024	—	t <sub>HXT</sub>
		—	f <sub>HIRC</sub> off → on (HIRCF=1)	—	16	—	t <sub>HIRC</sub>
		—	f <sub>PLL</sub> off → on (PLL <sub>F</sub> =1)	—	2560	—	t <sub>PLL</sub> (48MHz)
—		f <sub>LXT</sub> off → on (LXTF=1)	—	1024	—	t <sub>LXT</sub>	
t <sub>RSTD</sub>	System Reset Delay Time (Reset Source from Power-on Reset or LVR Hardware Reset)	—	RR <sub>POR</sub> =5V/ms	14	16	18	ms
	System Reset Delay Time (LVRC/WDT/RSTC Software Reset)	—	—	—	—	—	
	System Reset Delay Time (Reset Source from WDT Overflow)	—	—	14	16	18	
t <sub>SRESET</sub>	Minimum Software Reset Width to Reset	—	—	45	90	120	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

Ta=-40°C~85°C

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

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 PC4~PC7 Pins	5V	—	0	—	1.5	V
		—	—	0	—	0.2V <sub>DD</sub>	V
V <sub>IH</sub>	Input High Voltage for I/O Ports Except PC4~PC7 Pins	5V	—	3.5	—	5.0	V
		—	—	0.8V <sub>DD</sub>	—	V <sub>DD</sub>	V
I <sub>OL</sub>	Sink Current for I/O Ports Except PC4~PC7 Pins	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	16	32	—	mA
		5V		32	65	—	mA
I <sub>OH</sub>	Source Current for I/O Ports Except PC4~PC7 Pins	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-4	-8	—	mA
		5V		-8	-16	—	mA
R <sub>PH</sub>	Pull-high Resistance Except PC4~PC7 Pins <sup>(1)</sup>	3V	—	20	60	100	kΩ
		5V	—	10	30	50	
I <sub>LEAK</sub>	Input Leakage Current Except PC4~PC7 Pins	5V	V <sub>IN</sub> =V <sub>DD</sub> or V <sub>IN</sub> =V <sub>SS</sub>	—	—	±1	μA
t <sub>INT</sub>	External Interrupt Input Minimum Pulse Width	—	—	10	—	—	μs
t <sub>TCK</sub>	xTM Clock Input Minimum Pulse Width	—	—	0.3	—	—	μs
t <sub>TPI</sub>	STM, PTMn Capture Input Minimum Pulse Width	—	—	0.3	—	—	μs
f <sub>TMCLK</sub>	STM, PTMn Maximum Timer Clock Source Frequency	5V	—	—	—	1	f <sub>sys</sub>
t <sub>CPW</sub>	STM, PTMn Minimum Capture Pulse Width	—	—	t <sub>CPW</sub> <sup>(2)</sup>	—	—	μs

Note: 1. The R<sub>PH</sub> internal pull-high resistance value is calculated by connecting to ground and enabling the input pin with a pull-high resistor and then measuring the pin current at the specified supply voltage level. Dividing the voltage by this measured current provides the R<sub>PH</sub> value.

2. For STM

$$t_{CPW} = \max(2 \times t_{TMCLK}, t_{TPI})$$

Ex1: If f<sub>TMCLK</sub>=16MHz, t<sub>TPI</sub>=0.3μs, then t<sub>CPW</sub>=max(0.125μs, 0.3μs)=0.3μs

Ex2: If f<sub>TMCLK</sub>=8MHz, t<sub>TPI</sub>=0.3μs, then t<sub>CPW</sub>=max(0.25μs, 0.3μs)=0.3μs

Ex3: If f<sub>TMCLK</sub>=4MHz, t<sub>TPI</sub>=0.3μs, then t<sub>CPW</sub>=max(0.5μs, 0.3μs)=0.5μs

For PTM

If PTnCAPTS=0, then t<sub>CPW</sub>=max(2×t<sub>TMCLK</sub>, t<sub>TPI</sub>)

If PTnCAPTS=1, then t<sub>CPW</sub>=max(2×t<sub>TMCLK</sub>, t<sub>TCK</sub>)

Ex1: If PTnCAPTS=0, f<sub>TMCLK</sub>=16MHz, t<sub>TPI</sub>=0.3μs, then t<sub>CPW</sub>=max(0.125μs, 0.3μs)=0.3μs

Ex2: If PTnCAPTS=1, f<sub>TMCLK</sub>=16MHz, t<sub>TCK</sub>=0.3μs, then t<sub>CPW</sub>=max(0.125μs, 0.3μs)=0.3μs

Ex3: If PTnCAPTS=0, f<sub>TMCLK</sub>=8MHz, t<sub>TPI</sub>=0.3μs, then t<sub>CPW</sub>=max(0.25μs, 0.3μs)=0.3μs

Ex4: If PTnCAPTS=0, f<sub>TMCLK</sub>=4MHz, t<sub>TPI</sub>=0.3μs, then t<sub>CPW</sub>=max(0.5μs, 0.3μs)=0.5μs

Where t<sub>TMCLK</sub>=1/f<sub>TMCLK</sub>.



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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	V <sub>DD</sub> Power Supply for PC4~PC7 Pins	—	—	2.2	5.0	5.5	V
V <sub>DDIO</sub>	V <sub>DDIO</sub> Power Supply for PC4~PC7 Pins	—	—	2.2	—	V <sub>DD</sub>	V
V <sub>IL</sub>	Input Low Voltage for PC4~PC7 Pins	5V	Pin power=V <sub>DD</sub> or V <sub>DDIO</sub> , 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 (V <sub>DD</sub> / V <sub>DDIO</sub> )	V
V <sub>IH</sub>	Input High Voltage for PC4~PC7 Pins	5V	Pin power=V <sub>DD</sub> or V <sub>DDIO</sub> , 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 (V <sub>DD</sub> / V <sub>DDIO</sub> )	—	V <sub>DD</sub> / V <sub>DDIO</sub>	V
I <sub>OL</sub>	Sink Current for PC4~PC7 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>DDIO</sub> =V <sub>DD</sub>	32	65	—	mA
		5V	V <sub>OL</sub> =0.1(V <sub>DD</sub> /V <sub>DDIO</sub> ), V <sub>DDIO</sub> =3V	20	40	—	mA
I <sub>OH</sub>	Source Current for PC4~PC7 Pins	3V	V <sub>OH</sub> =0.9(V <sub>DD</sub> /V <sub>DDIO</sub> ), V <sub>DDIO</sub> =V <sub>DD</sub>	-4	-8	—	mA
		5V	V <sub>DDIO</sub> =V <sub>DD</sub>	-8	-16	—	mA
		5V	V <sub>OH</sub> =0.9(V <sub>DD</sub> /V <sub>DDIO</sub> ), V <sub>DDIO</sub> =3V	-2.5	-5.0	—	mA
R <sub>PH</sub>	Pull-high Resistance for PC4~PC7 Pins <sup>(Note)</sup>	3V	V <sub>DDIO</sub> =V <sub>DD</sub>	20	60	100	kΩ
		5V	V <sub>DDIO</sub> =V <sub>DD</sub>	10	30	50	
		5V	V <sub>DDIO</sub> =3V	36	110	180	
I <sub>LEAK</sub>	Input Leakage Current for PC4~PC7 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 a pull-high resistor and then measuring the pin current at the specified supply voltage level. Dividing the voltage by this measured current provides the R<sub>PH</sub> value.

**Memory Characteristics**

T<sub>a</sub>=-40°C~85°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
<b>Flash Program Memory</b>							
t <sub>FER</sub>	Erase Time	—	FWERTS=0	—	4.70	5.65	ms
		—	FWERTS=1	—	5.20	6.25	
t <sub>FWR</sub>	Write Time	—	FWERTS=0	—	2.7	3.3	ms
		—	FWERTS=1	—	3.25	3.90	
E <sub>P</sub>	Cell Endurance	—	—	100K	—	—	E/W
t <sub>RETD</sub>	ROM Data Retention Time	—	T <sub>a</sub> =25°C	—	40	—	Year
t <sub>ACTV</sub>	ROM Activation Time – Wake-up from IDLE/SLEEP Mode	—	—	32	—	64	μs
<b>Data EEPROM Memory</b>							
V <sub>DD</sub>	V <sub>DD</sub> for Read / Write	—	—	2.2	—	5.5	V
t <sub>EERD</sub>	EEPROM Read Time	—	—	—	—	4	t <sub>sys</sub>
t <sub>EEER</sub>	EEPROM Erase Time	—	EWERTS=0	—	3.2	3.9	ms
		—	EWERTS=1	—	3.7	4.5	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
t <sub>EEWR</sub>	EEPROM Write Time (Byte Mode)	—	EWERTS=0	—	5.4	6.6	ms
		—	EWERTS=1	—	6.7	8.1	
	EEPROM Write Time (Page Mode)	—	EWERTS=0	—	2.2	2.7	
		—	EWERTS=1	—	3.0	3.6	
E <sub>P</sub>	Cell Endurance	—	—	100K	—	—	E/W
t <sub>RETD</sub>	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: 1. “E/W” means Erase/Write times.

2. The ROM activation time t<sub>ACTV</sub> should be added when calculating the total system start-up time of a wake-up from the IDLE/SLEEP mode.

## LVD/LVR Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	Operating Voltage	—	—	2.2	—	5.5	V
V <sub>LVR</sub>	Low Voltage Reset Voltage	—	LVR enable, voltage select 2.1V	-5%	2.1	+5%	V
		—	LVR enable, voltage select 2.55V		2.55		
		—	LVR enable, voltage select 3.15V		3.15		
		—	LVR enable, voltage select 3.8V		3.8		
V <sub>LVD</sub>	Low Voltage Detector Voltage	—	LVD enable, voltage select 1.23V	-10%	1.23	+10%	V
		—	LVD enable, voltage select 2.2V	-5%	2.2	+5%	
		—	LVD enable, voltage select 2.4V		2.4		
		—	LVD enable, voltage select 2.7V		2.7		
		—	LVD enable, voltage select 3.0V		3.0		
		—	LVD enable, voltage select 3.3V		3.3		
		—	LVD enable, voltage select 3.6V		3.6		
		—	LVD enable, voltage select 4.0V		4.0		
I <sub>LVR/LVDBG</sub>	Operating Current	3V	LVD enable, LVR enable,		—		—
		5V	VBGEN=0	—	20	25	μA
		3V	LVD enable, LVR enable,	—	—	150	μA
		5V	VBGEN=1	—	180	200	μA
t <sub>LVDS</sub>	LVDO Stable Time	—	For LVR enable, VBGEN=0, LVD off → on	—	—	18	μs
t <sub>LVR</sub>	Minimum Low Voltage Width to Reset	—	TLVR[1:0]=00B	120	240	480	μs
		—	TLVR[1:0]=01B	0.5	1.0	2.0	ms
		—	TLVR[1:0]=10B	1	2	4	
		—	TLVR[1:0]=11B	2	4	8	
t <sub>LVD</sub>	Minimum Low Voltage Width to Interrupt	—	TLVD[1:0]=00B/11B	50	80	140	μs
		—	TLVD[1:0]=01B	80	140	260	
		—	TLVD[1:0]=10B	140	260	500	

Note: If V<sub>LVD</sub>=1.23V, it is used to detect the LVDIN pin input voltage. Other V<sub>LVD</sub> choices are used to detect the power supply V<sub>DD</sub>.

## Analog Front End Circuit Characteristics

### LDO Electrical Characteristics

$V_{DD}=V_{IN}$ ,  $T_a=25^{\circ}\text{C}$ , unless otherwise specified  
 LDO test conditions: MCU enters SLEEP mode, other functions disabled

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		$V_{DD}$	Conditions				
$V_{IN}$	LDO Input Voltage	—	—	2.2	—	5.5	V
$I_Q$	LDO Quiescent Current	—	LDOVS[1:0]=00B, $V_{IN}=3.6\text{V}$ , no load	—	600	720	$\mu\text{A}$
$V_{OUT\_LDO}$	LDO Output Voltage	—	LDOVS[1:0]=00B, $V_{IN}=3.6\text{V}$ , $I_{LOAD}=0.1\text{mA}$	-5%	2.4	+5%	V
		—	LDOVS[1:0]=01B, $V_{IN}=3.6\text{V}$ , $I_{LOAD}=0.1\text{mA}$		2.6		
		—	LDOVS[1:0]=10B, $V_{IN}=3.6\text{V}$ , $I_{LOAD}=0.1\text{mA}$		2.2		
		—	LDOVS[1:0]=11B, $V_{IN}=3.6\text{V}$ , $I_{LOAD}=0.1\text{mA}$		2.3		
$\Delta V_{LOAD}$	LDO Load Regulation <sup>(1)</sup>	—	LDOVS[1:0]=00B, $V_{IN}=V_{OUT\_LDO}+0.2\text{V}$ , $0\text{mA}\leq I_{LOAD}\leq 10\text{mA}$	—	0.105	0.210	%/mA
$V_{DROPO\_LDO}$	LDO Dropout Voltage <sup>(2)</sup>	—	LDOVS[1:0]=00B, $V_{IN}=3.6\text{V}$ , $I_{LOAD}=10\text{mA}$ , $\Delta V_{OUT\_LDO}=2\%$	—	—	220	mV
		—	LDOVS[1:0]=01B, $V_{IN}=3.6\text{V}$ , $I_{LOAD}=10\text{mA}$ , $\Delta V_{OUT\_LDO}=2\%$	—	—	200	
		—	LDOVS[1:0]=10B, $V_{IN}=3.6\text{V}$ , $I_{LOAD}=10\text{mA}$ , $\Delta V_{OUT\_LDO}=2\%$	—	—	240	
		—	LDOVS[1:0]=11B, $V_{IN}=3.6\text{V}$ , $I_{LOAD}=10\text{mA}$ , $\Delta V_{OUT\_LDO}=2\%$	—	—	230	
$TC_{LDO}$	LDO Temperature Coefficient	—	$T_a=-40^{\circ}\text{C}\sim 85^{\circ}\text{C}$ , LDOVS[1:0]=00B, $V_{IN}=3.6\text{V}$ , $I_{LOAD}=100\mu\text{A}$	—	—	$\pm 200$	ppm/ $^{\circ}\text{C}$
$\Delta V_{LINE\_LDO}$	LDO Line Regulation	—	LDOVS[1:0]=00B, $2.6\text{V}\leq V_{IN}\leq 5.5\text{V}$ , $I_{LOAD}=100\mu\text{A}$	—	—	0.7	%/V
		—	LDOVS[1:0]=00B, $2.6\text{V}\leq V_{IN}\leq 3.6\text{V}$ , $I_{LOAD}=100\mu\text{A}$	—	—	0.2	%/V

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

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

**Operational Amplifier Electrical Characteristics**

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	Operating Voltage	—	—	2.2	—	5.5	V
I <sub>OPA</sub>	Additional Current for OPA1	3V	—	—	200	320	μA
	Additional Current for OPA2	3V	—	—	—	1300	μA
A <sub>OL</sub>	OPA Open Loop Gain	3V	—	80	100	—	dB
R <sub>o</sub>	Output Resistance	2.4V~3.6V	Ta=0°C~50°C, R <sub>LOAD</sub> =50kΩ, 0.2V<V <sub>OP</sub> <V <sub>DD</sub> -1.4V (Voltage follower configuration)	—	—	260	Ω
I <sub>OS</sub>	Input Offset Current	2.4V~3.6V	Ta=0°C~50°C	-5	—	5	nA
TC	Temperature Coefficient of Offset Voltage	3V	Ta=0°C~50°C	—	—	±20	μV/°C
GBW	Gain Bandwidth for OPA1	3V	Ta=25°C, R <sub>L</sub> =1MΩ, C <sub>L</sub> =60pF, V <sub>IN</sub> =V <sub>CM</sub> /2	—	2	—	MHz
	Gain Bandwidth for OPA2	3V	Ta=25°C, R <sub>L</sub> =1MΩ, C <sub>L</sub> =60pF, V <sub>IN</sub> =V <sub>CM</sub> /2	—	7	—	MHz
V <sub>OS</sub>	Input Offset Voltage	3V	Without calibration (OPAxOF[4:0]=10000B)	-15	—	15	mV
V <sub>CM</sub>	OPA Common Mode Voltage Range	3V	—	0.1	—	V <sub>DD</sub> -1.4	V
V <sub>OR</sub>	OPA Maximum Output Voltage Range	3V	—	V <sub>SS</sub> +0.1	—	V <sub>DD</sub> -0.1	V

**Internal Reference Voltage Characteristics**

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>IREF</sub>	Internal Reference Voltage	3V	IREFEN=1, PVREF=10000000B	-3%	2.0	+3%	V
I <sub>IREF</sub>	Additional Current for Internal Reference Voltage	2.5V~3.3V	IREFEN=1	—	650	1000	μA
T <sub>CIREF</sub>	Temperature Coefficient of Internal Reference Voltage	3V	Ta=10°C~40°C	—	±40	—	ppm/°C

**Analog Switch Electrical Characteristics**

Ta=25°C, unless otherwise specified

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
R <sub>OP</sub>	On Resistance for the Pins expect VG Pin	3V	Ta=10°C~40°C	—	—	1300	Ω
R <sub>VG</sub>	On Resistance for VG Pin	3V	Ta=10°C~40°C	—	—	20	Ω
I <sub>LEAK</sub>	Leakage Current for VG Pin	3V	Ta=10°C~40°C	-5	±0.5	5	nA

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

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>DD</sub>	Operating Voltage	—	—	2.4	—	3.6	V
DNL	Differential Non-linearity	3V	DACVRS[1:0]=00B	—	—	±8	LSB
INL	Integral Non-linearity	3V	DACVRS[1:0]=00B	—	—	±20	LSB
V <sub>DACO</sub>	Output Voltage Range	—	—	0.00	—	1.00	V <sub>DACVREF</sub>
V <sub>DACO_RIPPLE</sub>	Output Voltage Ripple	3V	DACVRS[1:0]=10B	-0.6	—	+0.6	mV

### A/D Converter Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
AV <sub>DD</sub>	Supply Voltage for A/D, PGA	—	—	2.4	—	3.3	V
I <sub>ADC</sub>	Additional Current for ADC Enable	—	—	—	600	750	μA
I <sub>ADSTB</sub>	Standby Current	—	MCU enter SLEEP mode, no load	—	—	1	μA
N <sub>R</sub>	Resolution	—	—	—	—	24	Bit
INL	Integral Non-linearity	—	AV <sub>DD</sub> =3.3V, V <sub>REF</sub> =1.25V, ΔSI=±450mV, PGAGN=1	—	±50	±200	ppm
NFB	Noise Free Bits	—	f <sub>MCLK</sub> =4MHz, FLMS[2:0]=000B, AV <sub>DD</sub> =3.3V, V <sub>REF</sub> =1.25V, PGAGN=128, OSR=16384	—	16	—	Bit
ENOB	Effective Number of Bits	—	f <sub>MCLK</sub> =4MHz, FLMS[2:0]=000B, AV <sub>DD</sub> =3.3V, V <sub>REF</sub> =1.25V, PGAGN=128, OSR=16384	—	18.7	—	Bit
f <sub>ADCK</sub>	ADC Clock Frequency	—	—	40.0	409.6	440.0	kHz
f <sub>ADO</sub>	ADC Output Data Rate	—	f <sub>MCLK</sub> =4MHz, FLMS[2:0]=000B, R_FILSEL=1, R_CKCHOP=1	4	—	1042	Hz
		—	f <sub>MCLK</sub> =4MHz, FLMS[2:0]=010B, R_FILSEL=1, R_CKCHOP=1	10	—	2604	Hz
V <sub>REFP</sub>	Reference Input Voltage	—	—	V <sub>REFN</sub> +0.8	—	AV <sub>DD</sub>	V
V <sub>REFN</sub>		—	—	0	—	V <sub>REFP</sub> -0.8	V
V <sub>REF</sub>		—	V <sub>REF</sub> =(V <sub>REFP</sub> -V <sub>REFN</sub> )×VREFGN	0.80	—	1.75	V

**PGA Electrical Characteristics**

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>CM_PGA</sub>	Common Mode Voltage Range	—	—	0.40	—	A <sub>VDD</sub> -0.95	V
ΔD <sub>i</sub>	Differential input voltage range	—	Gain=PGAGN×ADGN	-V <sub>REF</sub> /Gain	—	+V <sub>REF</sub> /Gain	V

**Effective Number of Bits – ENOB**

A<sub>VDD</sub>=V<sub>OREG</sub>=3.3V, V<sub>REF</sub>=1.65V, f<sub>ADCK</sub>=133kHz

Data Rate (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
4	21.5	21.5	21.4	21.3	20.9	20.2	19.4	18.7
8	21.0	20.9	20.9	20.8	20.5	19.9	19.0	18.3
16	20.5	20.5	20.5	20.3	20.2	19.6	18.7	18.0
33	19.8	19.7	19.7	19.6	19.6	19.5	18.4	17.5
65	19.2	19.0	18.8	18.6	18.6	18.4	17.9	17.1
130	18.8	18.7	18.6	18.6	18.4	18.0	17.1	16.5
260	18.3	18.3	18.3	18.2	17.5	17.2	16.8	16.0
521	17.7	17.6	17.6	17.5	17.4	16.9	16.3	15.5

A<sub>VDD</sub>=V<sub>OREG</sub>=3.3V, V<sub>REF</sub>=1.65V, f<sub>ADCK</sub>=333kHz

Data Rate (SPS)	PGA Gain							
	1	2	4	8	16	32	64	128
10	21.7	21.3	21.2	21.0	20.6	19.9	19.1	18.3
20	21.2	21.2	21.0	20.9	19.6	19.5	18.6	17.9
41	20.5	20.4	20.3	20.1	19.5	18.6	18.3	17.4
81	20.0	19.8	19.7	19.6	19.3	18.6	17.8	17.0
163	19.5	19.3	19.2	19.0	18.7	18.2	17.4	16.4
326	19.0	18.7	18.6	18.5	18.2	17.7	16.9	16.0
651	18.4	18.3	18.2	18.1	17.8	17.2	16.4	15.5
1302	17.9	17.8	17.7	17.6	17.3	16.7	15.9	15.0

**LCD Electrical Characteristics**

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>IN</sub>	LCD Operating Voltage	—	R type, power supply from PLCD pin	3.0	—	5.5	V
		—	C type, power supply from PLCD pin	2.0	—	3.7	
		—	C type, power supply from V1 pin	3.0	—	5.5	
		—	C type, power supply from V2 pin	1.0	—	1.8	
		—	C type, power supply from V <sub>A</sub>	3.0	—	5.5	
		3.3V~5.5V	C type, power supply from V <sub>B</sub>	-10%	3.0	+10%	
		2.2V~5.5V	C type, power supply from V <sub>C</sub> , no load	-10%	1.0	+10%	
I <sub>LCD</sub>	Additional Current for LCD Enable (R type), LCD Clock=4kHz	5V	No load, V <sub>A</sub> =V <sub>PLCD</sub> =V <sub>DD</sub> , 1/3 bias, RCT=0, LCDPR=0, LCDIS[1:0]=00B	—	25.0	37.5	μA
			No load, V <sub>A</sub> =V <sub>PLCD</sub> =V <sub>DD</sub> , 1/4 bias, RCT=0, LCDPR=0, LCDIS[1:0]=00B	—	18	28	μA
			No load, V <sub>A</sub> =V <sub>PLCD</sub> =V <sub>DD</sub> , 1/3 bias, RCT=0, LCDPR=0, LCDIS[1:0]=01B	—	50	75	μA
			No load, V <sub>A</sub> =V <sub>PLCD</sub> =V <sub>DD</sub> , 1/4 bias, RCT=0, LCDPR=0, LCDIS[1:0]=01B	—	37.5	56.0	μA
			No load, V <sub>A</sub> =V <sub>PLCD</sub> =V <sub>DD</sub> , 1/3 bias, RCT=0, LCDPR=0, LCDIS[1:0]=10B	—	100	150	μA
			No load, V <sub>A</sub> =V <sub>PLCD</sub> =V <sub>DD</sub> , 1/4 bias, RCT=0, LCDPR=0, LCDIS[1:0]=10B	—	75.0	112.5	μA
			No load, V <sub>A</sub> =V <sub>PLCD</sub> =V <sub>DD</sub> , 1/3 bias, RCT=0, LCDPR=0, LCDIS[1:0]=11B	—	200	300	μA
			No load, V <sub>A</sub> =V <sub>PLCD</sub> =V <sub>DD</sub> , 1/4 bias, RCT=0, LCDPR=0, LCDIS[1:0]=11B	—	150	225	μA
I <sub>LCDOL</sub>	LCD Commom and Segment Sink Current	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	210	420	—	μA
		5V		350	700	—	μA
I <sub>LCDOH</sub>	LCD Commom and Segment Source Current	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-80	-160	—	μA
		5V		-180	-360	—	μA
V <sub>LCD</sub>	PLCD Comes from Charge Pump	2.2V~5.5V	RCT=0, LCDPR=1, CPVS[1:0]=00B, LCDIS[1:0]=11B	-10%	3.3	+10%	V
			RCT=0, LCDPR=1, CPVS[1:0]=01B, LCDIS[1:0]=11B		3.0		V
			RCT=0, LCDPR=1, CPVS[1:0]=10B, LCDIS[1:0]=11B		2.7		V
		2.7V~5.5V	RCT=0, LCDPR=1, CPVS[1:0]=11B, LCDIS[1:0]=11B	-10%	4.5	+10%	V

## USB Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
I <sub>DD</sub>	Operating Current (USB)	3V	f <sub>SYS</sub> =f <sub>PLL</sub> =6MHz, no load, USB and PLL on, other peripherals off	—	4.5	9.0	mA
		5V	f <sub>SYS</sub> =f <sub>PLL</sub> =12MHz, no load, USB and PLL on, other peripherals off	—	9.5	15.0	mA
		3V	f <sub>SYS</sub> =f <sub>PLL</sub> =12MHz, no load, USB and PLL on, other peripherals off	—	5	10	mA
		5V	f <sub>SYS</sub> =f <sub>PLL</sub> =12MHz, no load, USB and PLL on, other peripherals off	—	10.5	16.0	mA
		5V	f <sub>SYS</sub> =f <sub>PLL</sub> =16MHz, no load, USB and PLL on, other peripherals off	—	11	18	mA
I <sub>SUS</sub>	Suspend Current (USB) (IDLE0 Mode)	5V	f <sub>H</sub> off, f <sub>SUB</sub> =f <sub>LIRC</sub> or f <sub>LXT</sub> , no load, CPU off, USB transceiver and 3.3V Regulator on, other peripherals off, SUSP2=0, RCTRL=0	—	360	450	μA
		5V	f <sub>H</sub> off, f <sub>SUB</sub> =f <sub>LIRC</sub> or f <sub>LXT</sub> , no load, CPU off, USB transceiver and 3.3V Regulator off, other peripherals off, SUSP2=1, RCTRL=1	—	240	330	μA
V <sub>V33O</sub>	3.3V Regulator Output Voltage	5V	I <sub>V33O</sub> =70mA	3.0	3.3	3.6	V
R <sub>UDP</sub>	Pull-high Resistance of UDP to V33O	3.3V	—	-5%	1.5	+5%	kΩ
	Pull-high Resistance of UDP to UBUS	5V	RCTRL=1	4.9	7.8	12.0	kΩ
R <sub>UDPN</sub>	Pull-high Resistance of UDP/UDN to UBUS	5V	PU=1	300	650	1000	kΩ
R <sub>PL</sub>	Pull-low Resistance of UBUS	5V	SUSP2=1, RUBUS=0	0.5	1.0	1.5	MΩ
R <sub>OD</sub>	Pull-high Resistance of OD0/OD1 to VDD	5V	UMS[2:0]=001B	2.0	4.7	8.0	kΩ
		5V	UMS[2:0]=100B~111B	10	30	50	kΩ
I <sub>OL_OD</sub>	Sink Current of OD0/OD1	5V	UMS[2:0]=001B, 100B~111B, V <sub>OL</sub> =0.1V <sub>DD</sub>	8	12	—	mA
I <sub>OH_OD</sub>	Source Current of OD0/OD1	5V	UMS[2:0]=100B~111B, V <sub>OH</sub> =0.9V <sub>DD</sub>	-4	-8	—	mA
V <sub>IH</sub>	Input High Voltage of OD0/OD1	5V	UMS[2:0]=001B, 100B~111B, OD1O/OD0O=11B	2	—	5	V
V <sub>IL</sub>	Input Low Voltage of OD0/OD1	5V	UMS[2:0]=001B, 100B~111B, OD1O/OD0O=11B	0	—	0.8	V

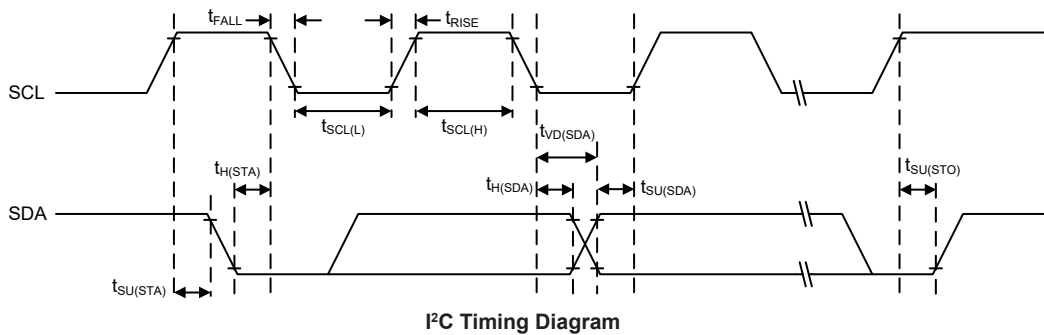


## I<sup>2</sup>C Electrical Characteristics

Ta=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
f <sub>I2C</sub>	I <sup>2</sup> C Standard Mode (100kHz) f <sub>SYS</sub> Frequency <small>(Note)</small>	—	No clock debounce	2	—	—	MHz
			2 system clock debounce	4	—	—	
			4 system clock debounce	4	—	—	
	I <sup>2</sup> C Fast Mode (400kHz) f <sub>SYS</sub> Frequency <small>(Note)</small>	—	No clock debounce	4	—	—	MHz
			2 system clock debounce	8	—	—	
			4 system clock debounce	8	—	—	
f <sub>SCL</sub>	SCL Clock Frequency	3V/5V	Standard mode	—	—	100	kHz
			Fast mode	—	—	400	
t <sub>SCL(H)</sub>	SCL Clock High Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
t <sub>SCL(L)</sub>	SCL Clock Low Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.9	—	—	
t <sub>FALL</sub>	SCL and SDA Fall Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
t <sub>RISE</sub>	SCL and SDA Rise Time	3V/5V	Standard mode	—	—	1.3	μs
			Fast mode	—	—	0.34	
t <sub>SU(SDA)</sub>	SDA Data Setup Time	3V/5V	Standard mode	0.25	—	—	μs
			Fast mode	0.1	—	—	
t <sub>H(SDA)</sub>	SDA Data Hold Time	3V/5V	—	0.1	—	—	μs
t <sub>VD(SDA)</sub>	SDA Data Valid Time	3V/5V	—	—	—	0.6	μs
t <sub>SU(STA)</sub>	Start Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	
t <sub>H(STA)</sub>	Start Condition Hold Time	3V/5V	Standard mode	4.0	—	—	μs
			Fast mode	0.6	—	—	
t <sub>SU(STO)</sub>	Stop Condition Setup Time	3V/5V	Standard mode	3.5	—	—	μs
			Fast mode	0.6	—	—	

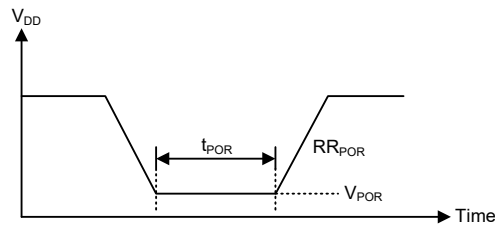
Note: Using the debounce function can make the transmission more stable and reduce the probability of communication failure due to interference.



## Power-on Reset Characteristics

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

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



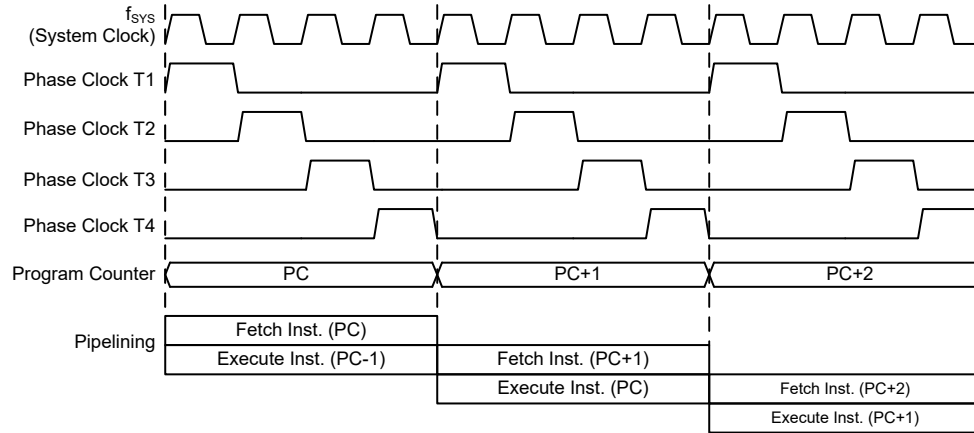
## System Architecture

A key factor in the high-performance features of the Holtek range of microcontrollers is attributed to their internal system architecture. The device takes advantage of the usual features found within RISC microcontrollers providing increased speed of operation and enhanced performance. The pipelining scheme is implemented in such a way that instruction fetching and instruction execution are overlapped, hence instructions are effectively executed in one or two cycles for most of the standard or extended instructions respectively. The exceptions to these are branch or call instructions which need one more cycle. An 8-bit wide ALU is used in practically all instruction set operations, which carries out arithmetic operations, logic operations, rotation, increment, decrement, branch decisions, etc. The internal data path is simplified by moving data through the Accumulator and the ALU. Certain internal registers are implemented in the Data Memory and can be directly or indirectly addressed. The simple addressing methods of these registers along with additional architectural features ensure that a minimum of external components is required to provide a functional I/O and A/D control system with maximum reliability and flexibility. This makes the device suitable for low-cost, high-volume production for controller applications.

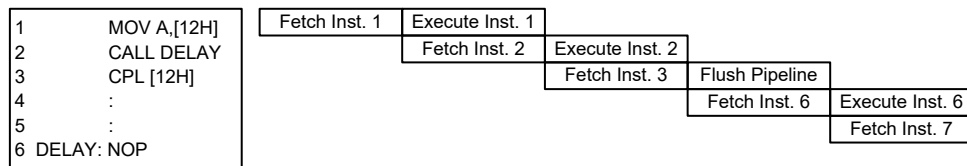
### Clocking and Pipelining

The main system clock, derived from either an HIRC, HXT, LIRC or LXT 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. For the device with a Program Memory capacity in excess of 8K words, the Program Memory address may be located in a certain program memory bank which is selected by the program memory bank pointer bits, PBP2~PBP0. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

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

Program Counter	
High Byte	Low Byte (PCL)
PBP2~PBP0, PC12~PC8	PCL7~PCL0

**Program Counter**

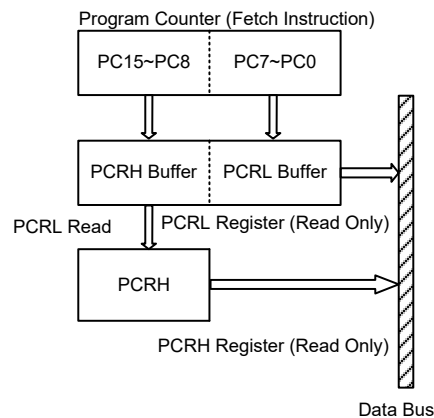
The lower byte of the Program Counter, known as the Program Counter Low register or PCL, is available for program control and is a readable and writeable register. By transferring data directly into this register, a short program jump can be executed directly; however, as only this low byte is available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

### Program Counter Read Register

The Program Counter Read registers are a read only register pair for reading the program counter value which indicates the current program execution address. Read the low byte register first then the high byte register. Reading the low byte register, PCRL, will read the low byte data of the current program execution address, and place the high byte data of the program counter into the 8-bit PCRH buffer. Then reading the PCRH register will read the corresponding data from the 8-bit PCRH buffer.

The following example shows how to read the current program execution address. When the current program execution address is 123H, the steps to execute the instructions are as follows:

- (1) MOV A, PCRL → the ACC value is 23H, and the PCRH value is 01H;  
     MOV A, PCRH → the ACC value is 01H.
- (2) LMOV A, PCRL → the ACC value is 23H, and the PCRH value is 01H;  
     LMOV A, PCRH → the ACC value is 01H.



#### • PCRL 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**: Program Counter Read Low byte register bit 7 ~ bit 0

#### • PCRH Register

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

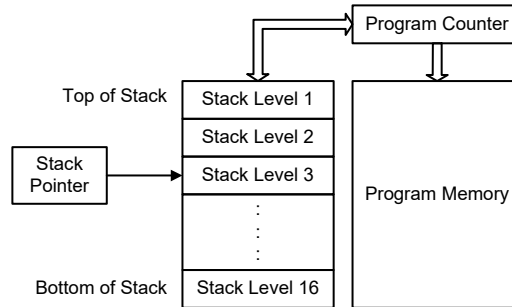
Bit 7~0      **D15~D8**: Program Counter Read High byte register bit 7 ~ bit 0

### Stack

This is a special part of the memory which is used to save the contents of the Program Counter only. The stack is organized into 16 levels and is neither part of the data nor part of the program space, and is neither readable nor writeable. The activated level is indexed by the Stack Pointer, STKPTR[3:0]. 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.



• **STKPTR Register**

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

Bit 7 **OSF**: Stack overflow flag  
 0: No stack overflow occurred  
 1: Stack overflow occurred

When the stack is full and a CALL instruction is executed or when the stack is empty and a RET instruction is executed, the OSF bit will be set high. The OSF bit is cleared only by software and cannot be reset automatically by hardware.

Bit 6~4 Unimplemented, read as “0”

Bit 3~0 **D3~D0**: Stack pointer register bit 3 ~ bit 0

The following example shows how the Stack Pointer and Stack Overflow Flag change when program branching conditions occur.

- (1) When the CALL subroutine instruction is executed 17 times continuously and the RET instruction is not executed during the period, the corresponding changes of the STKPTR[3:0] and OSF bits are as follows:

CALL Execution Times	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
STKPTR[3:0] Bit Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	1
OSF Bit Value	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

- (2) When the OSF bit is set high and not cleared, it will remain high no matter how many times the RET instruction is executed.

- (3) When the stack is empty, the RET instruction is executed 16 times continuously, the corresponding changes of the STKPTR[3:0] and OSF bits are as follows:

RET Execution Times	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
STKPTR[3:0] Bit Value	0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSF Bit Value	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

### Arithmetic and Logic Unit – ALU

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

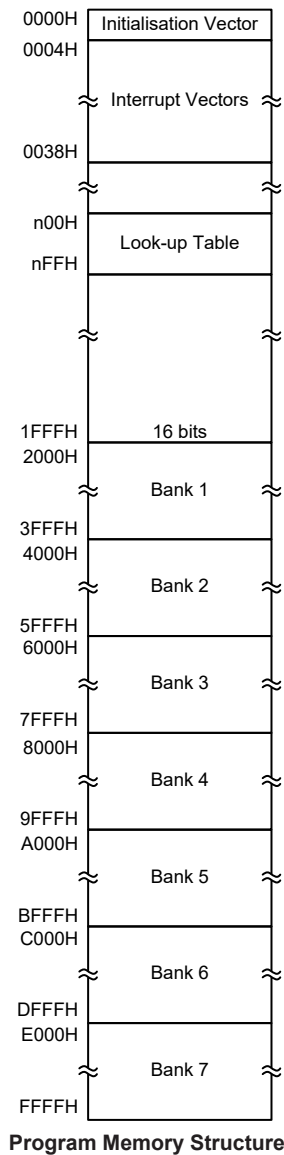
- Arithmetic operations:  
ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,  
LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:  
AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,  
LAND, LOR, LXOR, LANDM, LORM, LXORM, LCPL, LCPLA
- Rotation:  
RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,  
LRRR, LRR, LRRCA, LRRC, LRLA, LRL, LRLCA, LRLC
- Increment and Decrement:  
INCA, INC, DECA, DEC,  
LINCA, LINC, LDECA, LDEC
- Branch decision:  
JMP, SZ, SZA, SNZ, SIZ, SDZ, SIZA, SDZA, CALL, RET, RETI,  
LSZ, LSZA, LSNZ, LSIZ, LSDZ, LSIZA, LSDZA

## Flash Program Memory

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

### Structure

The Program Memory has a capacity of 64K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupt entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer registers.



**Special Vectors**

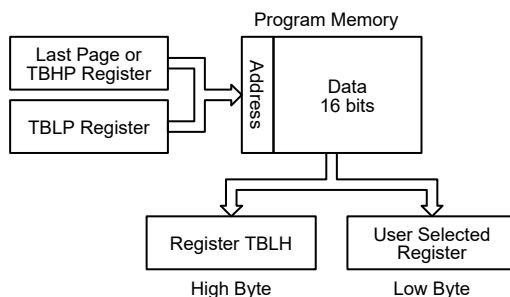
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 0000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

**Look-up Table**

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

After setting up the table pointer, the table data can be retrieved from the Program Memory using the “TABRD [m]” instruction when the memory [m] is located in sector 0. If the memory [m] is located in other sectors except sector 0, the data can be retrieved from the program memory using the corresponding extended table read instruction “LTABRD [m]”. When the instruction is executed, the lower order table byte from the Program Memory will be transferred to the user defined Data Memory register [m] as specified in the instruction. The higher order table data byte from the Program Memory will be transferred to the TBLH special register.

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



**Table Program Example**

The accompanying example shows how the table pointer and table data is defined and retrieved from the device. This example uses raw table data located in the last page which is stored there using the ORG statement. The value at this ORG statement is “1F00H” which is located in ROM Bank 7 and refers to the start address of the last page within the 64K words Program Memory. The table pointer low byte register is setup here to have an initial value of “06H”. This will ensure that the first data read from the data table will be at the Program Memory address “FF06H” 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 TBHP and TBLP 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**

```
Rombank7 code7
ds .section 'data'
tempreg1 db?      ; temporary register #1
tempreg2 db?      ; temporary register #2
code0 .section 'code'
mov a,06H         ; initialise table pointer - note that this address is referenced
mov tblp,a        ; to the last page or the page that tbhp pointed
mov a,FFH         ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1    ; transfers value in table referenced by table pointer data at
                  ; program memory address "FF06H" 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 "FF05H" transferred to tempreg2 and TBLH
                  ; in this example the data "1AH" is transferred to tempreg1 and
                  ; data "0FH" to tempreg2 the value "00H" will be
                  ; transferred to the high byte register TBLH
:
:
code7 .section 'code'
org 1F00H         ; sets initial address of last page
dc 00Ah,00Bh,00Ch,00Dh,00Eh,00Fh,01Ah,01Bh
```

**In Circuit Programming – ICP**

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

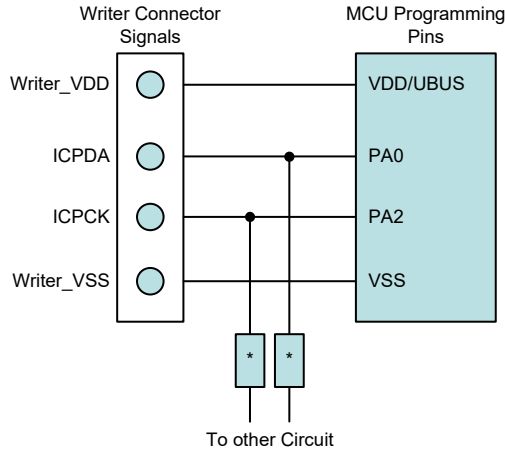
As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller, and then programming or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

The Flash MCU to Writer Programming Pin 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/UBUS	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 device are 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

The device also provides an “On-Chip Debug” function to debug the real MCU device during the development process. Users can use the OCDS function to emulate the device behavior by connecting the OCSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the OCDS function for debugging, other functions which are shared with the OCSDA and OCDSCK pins in the device will have no effect. 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	MCU Chip Pins	Pin Description
OCSDA	OCSDA	On-Chip Debug Support Data/Address input/output
OCDSCK	OCDSCK	On-Chip Debug Support Clock input
VDD	VDD/UBUS	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 not only an the IAP function but also an additional ISP 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 128 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 program 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.

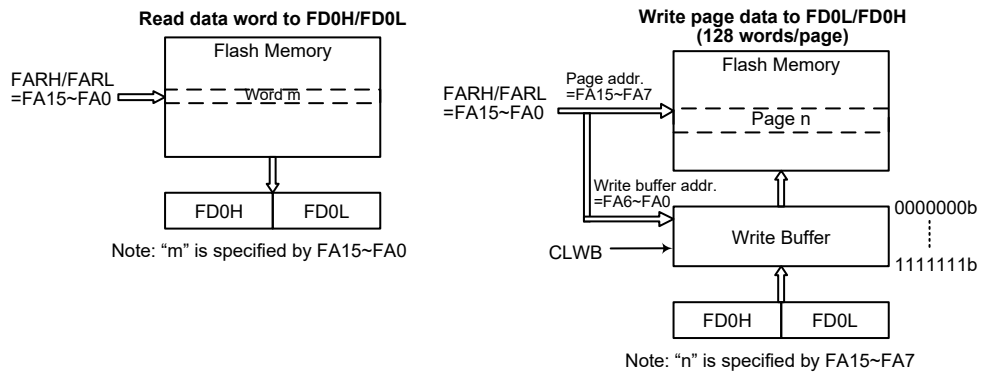
Operations	Format
Erase	128 words/page
Write	128 words/time
Read	1 word/time

Note: Page size=Write buffer size=128 words.

**IAP Operation Format**

Page	FARH	FARL[7]	FARL[6:0]
0	0000 0000	0	Tag Address
1	0000 0000	1	
2	0000 0001	0	
3	0000 0001	1	
4	0000 0010	0	
⋮	⋮	⋮	
⋮	⋮	⋮	
510	1111 1111	0	
511	1111 1111	1	

**Page Number and Address 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 FRCR 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 zero by 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 128 words corresponding to a page. The write buffer address is mapped to a specific Flash memory page specified by the memory address bits, FA15~FA7. 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, 1111111b of a page with 128 words, the address will now not be incremented but 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 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 pairs of 16-bit data registers and two control registers, which are all located in Sector 2. Read and Write operations to the Flash memory are carried out using 16-bit data operations using the address and data registers and the control registers. 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, where n is equal to 0~3, and the control registers are named FCR and FRCR. As all the IAP related registers are located in Sector 2, they can be addressed directly only using the corresponding extended instructions or can be read from or written to indirectly using the MP1H/MP1L or MP2H/MP2L Memory Pointer pairs and Indirect Addressing Register, IAR1 or IAR2.

Register Name	Bit							
	7	6	5	4	3	2	1	0
FCR	CFWEN	FMOD2	FMOD1	FMOD0	BWT	FWT	FRDEN	FRD
FRCR	D7	D6	—	D4	—	—	FWERTS	CLWB
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH	FA15	FA14	FA13	FA12	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**

• **FCR Register**

Bit	7	6	5	4	3	2	1	0
Name	CFWEN	FMOD2	FMOD1	FMOD0	BWT	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 0 by application program, the Flash memory erase/write function is disabled. Note that writing a “1” into this bit results in no action. This bit is used to indicate the Flash memory erase/write function status. When this bit is set to 1 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 is zero.
- Bit 6~4**    **FMOD2~FMOD0:** Flash memory Mode selection  
 000: Write Mode  
 001: Page Erase Mode  
 011: Read Mode  
 110: Flash memory Erase/Write function Enable Mode  
 Other values: 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**      **BWT:** 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 by 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 BWT 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 Flash memory write process  
 This bit is set by software and cleared by hardware when the Flash memory write process has completed.
- Bit 1**      **FRDEN:** Flash memory read enable control  
 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 initiate control  
 0: Do not initiate Flash memory read or indicating that a Flash memory read process has completed  
 1: Initiate Flash memory read process  
 This bit is set by software and cleared by hardware when the Flash memory read process has completed.

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, write or erase operation is successfully activated.
4. Ensure that the read, erase or write operation is totally complete before executing other operations.

• **FRCR Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	—	D4	—	—	FWERTS	CLWB
R/W	R/W	R	—	R/W	—	—	R/W	R/W
POR	0	0	—	0	—	—	0	0

- Bit 7      **D7:** Reserved bit, cannot be used and must be fixed at “0”
- Bit 6      **D6:** Reserved bit, cannot be used
- Bit 5      Unimplemented, read as “0”
- Bit 4      **D4:** Reserved bit, cannot be used and must be fixed at “0”
- Bit 3~2    Unimplemented, read as “0”
- Bit 1      **FWERTS:** Erase time and Write time selection  
             0: Erase time is 4.7ms ( $t_{FER}$ ) / Write time is 2.7ms ( $t_{FWR}$ )  
             1: Erase time is 5.2ms ( $t_{FER}$ ) / Write time is 3.25ms ( $t_{FWR}$ )
- 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 Write Buffer Clear process  
             This bit is set by software and cleared by hardware when the Write Buffer Clear process has completed.

• **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	FA15	FA14	FA13	FA12	FA11	FA10	FA9	FA8
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    **FA15~FA8:** Flash Memory Address bit 15 ~ 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 word 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 word 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 word 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 word 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 word bit 7 ~ bit 0

• **FD2H 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 third Flash Memory data word bit 15 ~ bit 8

• **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 word 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 word bit 15 ~ bit 8

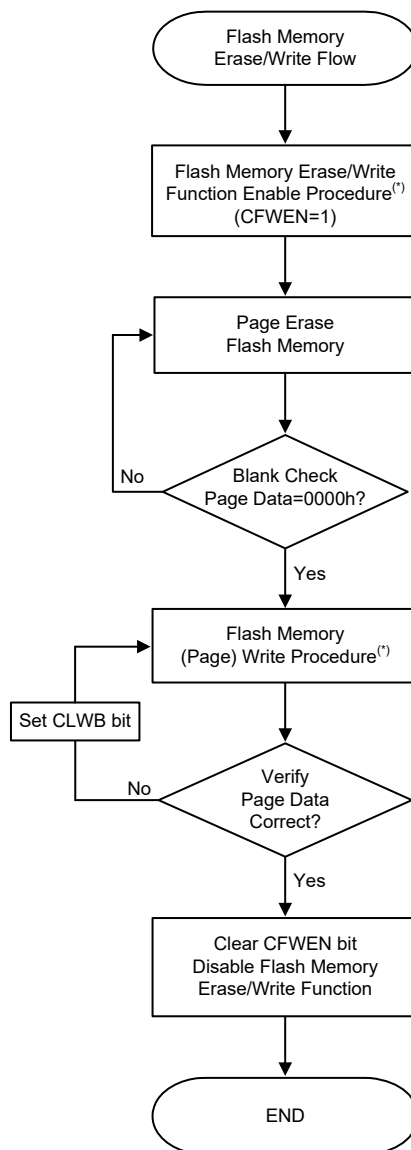
**Flash Memory Erase/Write Flow**

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 FCR 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, tag address and then erase this page.  
  
For a page erase operation, set the FARL and FARH registers to specify the start address of the erase page, then write dummy data into the FD0H register to tag address. The current address will be internally incremented by one after each dummy data is written into the FD0H register. When the address reaches the page boundary, 111111b, the address will not be further incremented but stop at the last address of the page. Note that the write operation to the FD0H register is used to tag address, it must be implemented to determine which addresses to be erased.
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 complete if 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.

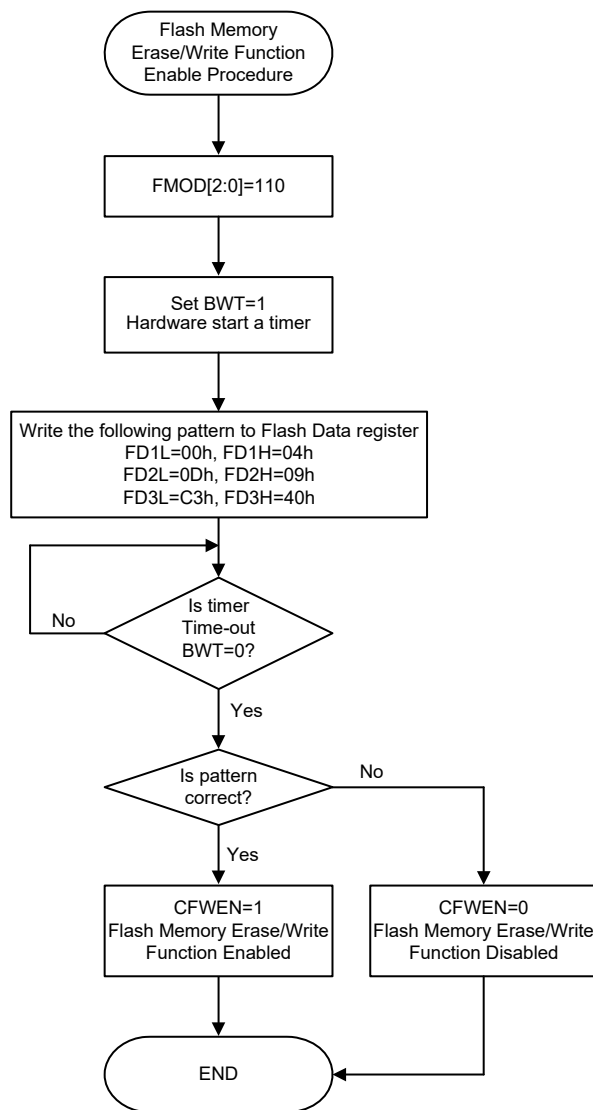
**Flash Memory Erase/Write Function Enable Procedure**

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 FCR register to select the Flash Memory Erase/Write Function Enable Mode.
2. Set the BWT bit in the FCR 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 of FD1L, FD2L, FD3L, FD1H, FD2H and FD3H, successively and as soon as possible after the BWT 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 BWT bit will automatically be cleared to 0 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 FCR register can be cleared. There is no need to execute the above procedure.



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

**Flash Memory Write Procedure**

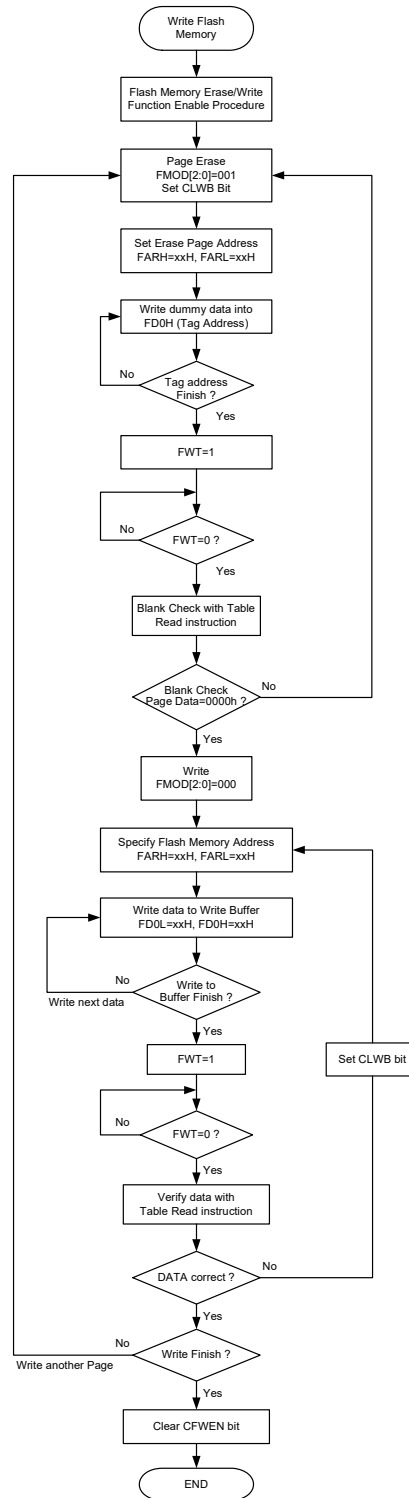
After the Flash memory erase/write function has been successfully enabled as the CFWEN bit is set high, the data to be written into the Flash memory can be loaded into the write buffer. The selected Flash memory page data should be erased by properly configuring the IAP control registers before the data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

The write buffer size is 128 words, known as a page, whose address is mapped to a specific Flash memory page specified by the memory address bits, FA15~FA7. It is important to ensure that the page where the write buffer data is located is the same one which the memory address bits, FA15~FA7, specify.

### Flash Memory Consecutive Write Description

The maximum amount of write data is 128 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 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 FMOD2~FMOD0 to “001” to select the erase operation and set the CLWB bit high to clear the write buffer. Set the FWT bit high to erase the desired page which is specified by the FARH and FARL registers and has been tagged address. 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 FMOD2~FMOD0 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 128 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**

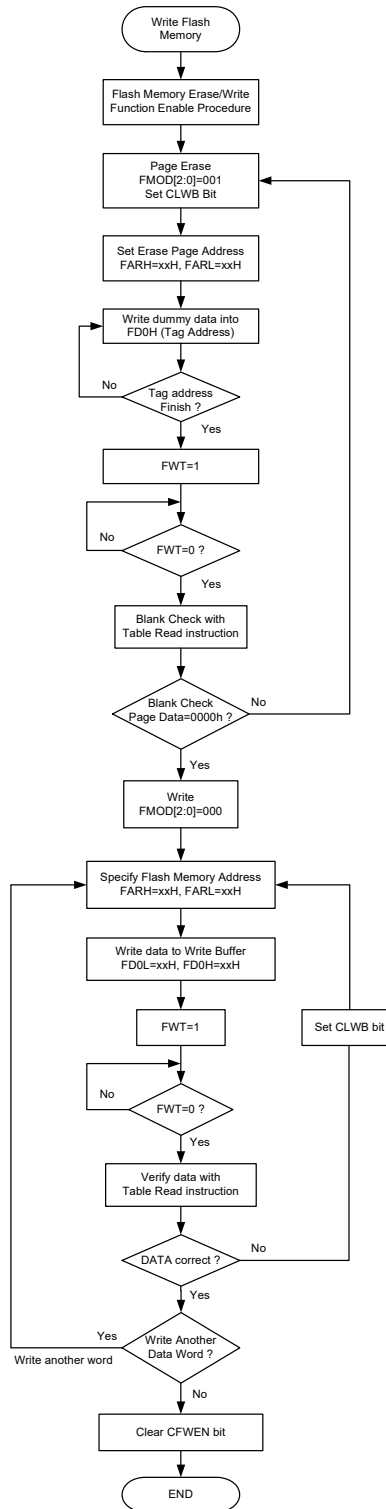
- Note: 1. When the erase or write operation is successfully activated, all CPU operations will temporarily cease.  
 2. It will take certain time for the FWT bit state changing from high to low in the erase or write operation, which can be selected by the FWERTS bit in the FRCR register.

**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 FMOD2~FMOD0 to “001” to select the erase operation and set the CLWB bit high to clear the write buffer. Set the FWT bit high to erase the desired page which is specified by the FARH and FARL registers and has been tagged address. 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 FMOD2~FMOD0 to “000” to select the write operation.
5. Setup the desired address ADDR1 in the FARH and FRARL 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 certain time for the FWT bit state changing from high to low in the erase or write operation, which can be selected by the FWERTS bit in the FRCR register.

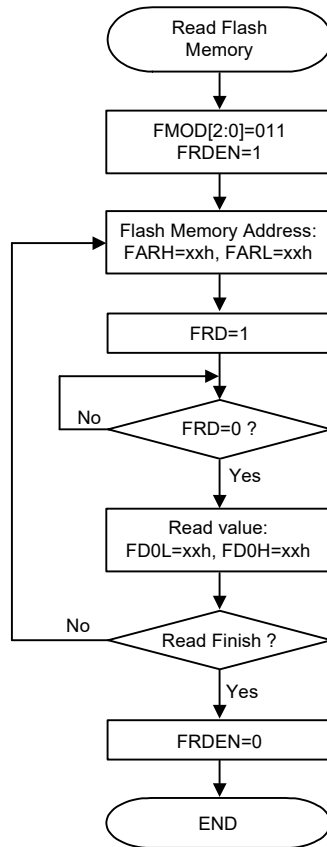
**Important Points to Note for Flash Memory Write Operations**

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. Bit 7 ~ Bit 2 in the FRCR register must remain at “0” to avoid unpredictable errors during the IAP supported operations.
5. 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 writing 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.
6. 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 FMOD2~FMOD0 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.

### **In System Programming – ISP**

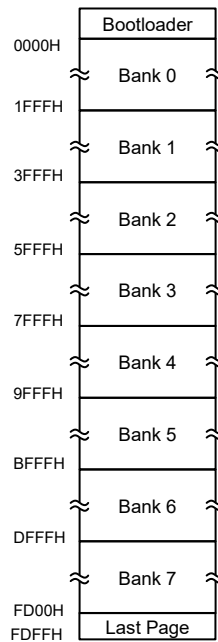
As an additional convenience, Holtek has provided a means of programming the microcontroller in-system using a two-line USB interface. This provides manufacturers with the possibility of manufacturing their circuit boards complete with a programmed or un-programmed microcontroller and then programming 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 microcontroller device.

The Program Memory can be programmed serially in-system using the USB interface, namely using the UDN and UDP pins. The power is supplied by the UBUS pin. The technical details regarding the in-system programming are beyond the scope of this document and will be supplied in supplementary literature. The Flash Program Memory Read/Write function is implemented using a series of registers.

### **ISP Bootloader**

An ISP Bootloader function is provided to upgrade the software in the Flash memory. The user can utilise either the ISP Bootloader application software provided by the Holtek IDE tools or to create their own Bootloader software. When the Holtek Bootloader software is selected note that it will

occupy an area of 0.5K capacity area in the Flash memory. The accompanying diagram illustrates the Flash memory structure including the Holtek Bootloader software.



**Flash Program Memory Structure including Bootloader**

## Data Memory

The Data Memory is an 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 device. Many of these registers can be read from and written to directly under program control, however, some remain protected from user manipulation. The second area of Data Memory is known as the General Purpose Data Memory, which is reserved for general purpose use. All locations within this area are read and write accessible under program control.

The device also provides a dedicated memory area for the LCD display data storage. In this chapter, only the General Purpose Data Memory and the Special Function Register Data Memory are introduced. More information about the LCD Display Data Memory can be obtained in its corresponding chapter.

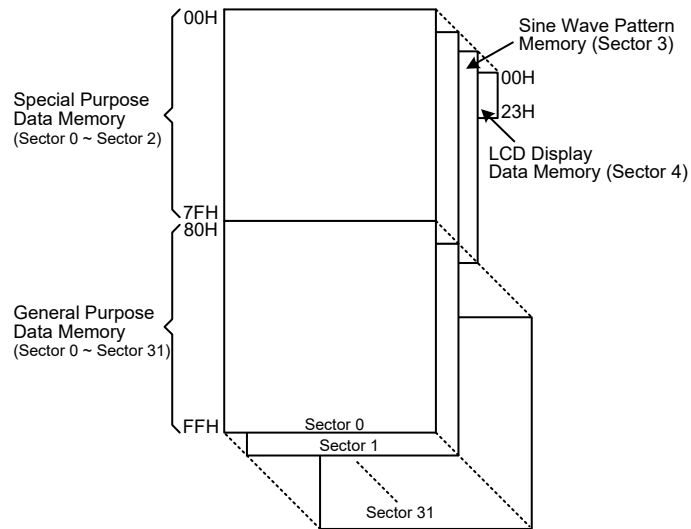
Switching between the different Data Memory sectors is achieved by properly setting the Memory Pointers to correct value when using the indirectly accessing method.

### Structure

The Data Memory is subdivided into several sectors, all of which are implemented in 8-bit wide Memory. Each of the Data Memory Sector is categorized into two types, the Special Purpose Data Memory and the General Purpose Data Memory. The address range of the Special Purpose Data Memory for the device is from 00H to 7FH while the General Purpose Data Memory address range is from 80H to FFH. The Sine Wave Pattern Memory is located from 00H to 7FH in Sector 3. The LCD Display Data Memory is located from 00H to 23H in Sector 4.

Special Purpose Data Memory	General Purpose Data Memory		Sine Wave Pattern Memory		LCD Display Data Memory	
Located Sectors	Capacity	Sector: Address	Capacity	Sector: Address	Capacity	Sector: Address
Sector 0: 00H~7FH Sector 1: 00H~7FH Sector 2: 00H~7FH	4096×8	0: 80H~FFH 1: 80H~FFH : 30: 80H~FFH 31: 80H~FFH	128×8	3: 00H~7FH	36×8	4: 00H~23H

**Data Memory Summary**



**Data Memory Structure**

### Data Memory Addressing

For device that supports the extended instructions, there is no Bank Pointer for Data Memory. The Bank Pointer, PBP, is only available for Program 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 corresponding instruction 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 13 valid bits for this device, the high byte indicates a sector and the low byte indicates a specific address.

### General Purpose Data Memory

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

### Special Purpose Data Memory

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

Sector 0			Sector 1			Sector 2		
00H	IAR0	U0SR						
01H	MP0	U0CR1						
02H	IAR1	U0CR2						
03H	MP1L	U0CR3						
04H	MP1H	BRDH0						
05H	ACC	BRDL0						
06H	PCL	UFCR0						
07H	TBLP	TXR_RXR0						
08H	TBLH	RxCNT0						
09H	TBHP	U1SR						
0AH	STATUS	U1CR1						
0BH	PBP	U1CR2						
0CH	IAR2	U1CR3						
0DH	MP2L	BRDH1						
0EH	MP2H	BRDL1						
0FH	RSTFC	UFCR1						
10H	SCC	TXR_RXR1						
11H	HIRCC	RxCNT1						
12H	HXTC	SIMC0						
13H	LXTC	SIMC1						
14H	PA	SIMD						
15H	PAC	SIM/SIMC2						
16H	PAPU	SIMTOC						
17H	PAWU							
18H	LVRC							
19H	LVDC							
1AH	TLVRC							
1BH	MF10							
1CH	MF11							
1DH	MF12							
1EH	MF13							
1FH	WDTC							
20H	INTEG0	SGC						
21H	INTC0	SGN						
22H	INTC1	SGDN						
23H	INTC2	OPA3C						
24H	INTC3	OPA2C						
25H	PB	ASWAI0						
26H	PBC	ASWAI1						
27H	PBPJ	ASWAI2						
28H	PC	ASWAI3						
29H	PCC	ASWAI4						
2AH	PCPU	ASWAI5						
2BH	PSCR	ASWAI6						
2CH	TB0C	ASWAI7						
2DH	TB1C	ASWAI8						
2EH	RSTC	ASWAI9						
2FH	INTEG1	ASWDAC						
30H								
31H								
32H								
33H								
34H								
35H								
36H								
37H								
38H	PWRC							
39H	PGAC0							
3AH	PGAC1							
3BH	PGACS							
3CH	ADRL							
3DH	ADRM							
3EH	ADRH							
3FH	ADCR0							
40H	EEAL	EEC						
41H	EEAH							
42H	EED							
43H	ADCR1							
44H	ADCS							
45H								
46H	CHOP							
47H	IREFC							
48H	PVREF							
49H	OPA1C							
4AH	AFEDA1C							
4BH	AFEDA1L							
4CH	AFEDA1H							
4DH	AFEDA2C							
4EH	AFEDA2L							
4FH	AFEDA2H							
50H	ATMC0	STMC0						
51H	ATMC1	STMC1						
52H	ATMDL	STMDL						
53H	ATMDH	STMDH						
54H	ATMAL	STMAL						
55H	ATMAH	STMAH						
56H	ATMBL	STMRL						
57H	ATMBH	PTM0C0						
58H	ATMRP	PTM0C1						
59H	MDUWR0	PTM0DL						
5AH	MDUWR1	PTM0DH						
5BH	MDUWR2	PTM0AL						
5CH	MDUWR3	PTM0AH						
5DH	MDUWR4	PTM0RPL						
5EH	MDUWR5	PTM0RPH						
5FH	MDUWCTRL	PTM1C0						
60H	SYSC	PTM1C1						
61H	USB_STAT	PTM1DL						
62H	UINT	PTM1DH						
63H	USC	PTM1AL						
64H	UESR	PTM1AH						
65H	UCC	PTM1RPL						
66H	AWR	PTM1RPH						
67H	STL							
68H	SIES							
69H	MISC							
6AH	SETIO							
6BH	FIFO3							
6CH	FIFO2							
6DH	FIFO1							
6EH	FIFO0							
6FH	FRNUM0							
70H	FRNUM1	STKPTR						
71H	PLLC	IECC						
72H	LCDC0	PCRL						
73H	LCDCP	PCRH						
74H	LCDC2	CRCCR						
75H		CRCIN						
76H		CRCDL						
77H		CRCDH						
78H								
79H								
7AH								
7BH								
7CH								
7DH								
7EH								
7FH	ORMC							

□ : Unused, read as 00H

▣ : Reserved, cannot be changed

**Special Purpose Data Memory Structure**

## Special Function Register Description

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

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

The Indirect Addressing Registers, IAR0, IAR1 and IAR2, although having their locations in normal RAM register space, do not actually physically exist as normal registers. The method of indirect addressing for RAM data manipulation uses these Indirect Addressing Registers and Memory Pointers, in contrast to direct memory addressing, where the actual memory address is specified. Actions on the IAR0, IAR1 and IAR2 registers will result in no actual read or write operation to these registers but rather to the memory location specified by their corresponding Memory Pointers, MP0, MP1L/MP1H or MP2L/MP2H. Acting as a pair, IAR0 and MP0 can together access data only from Sector 0 while the IAR1 register together with MP1L/MP1H register pair and IAR2 register together with MP2L/MP2H register pair can access data from any Data Memory sector. As the Indirect Addressing Registers are not physically implemented, reading the Indirect Addressing Registers indirectly will return a result of “00H” and writing to the registers indirectly will result in no operation.

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

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

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

### Indirect Addressing Program Example

#### Example 1

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block db ?
code .section at 0 code
org 00h
start:
    mov a,04h          ; setup size of block
    mov block,a
    mov a,offset adres1 ; Accumulator loaded with first RAM address
    mov mp0,a         ; setup memory pointer with first RAM address
loop:
    clr IAR0          ; clear the data at address defined by MP0
    inc mp0           ; increment memory pointer
```

```

        sdz block          ; check if last memory location has been cleared
        jmp loop
continue:

```

### Example 2

```

data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block db ?
code .section at 0 'code'
org 00h
start:
    mov a,04h            ; setup size of block
    mov block,a
    mov a,01h           ; setup the memory sector
    mov mplh,a
    mov a,offset adres1 ; Accumulator loaded with first RAM address
    mov mp1l,a          ; setup memory pointer with first RAM address
loop:
    clr IAR1           ; clear the data at address defined by MP1L
    inc mp1l           ; increment memory pointer MP1L
    sdz block          ; check if last memory location has been cleared
    jmp loop
continue:
:

```

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

### Program Memory Bank Pointer – PBP

For the device the Program Memory is divided into several banks. Selecting the required Program Memory area is achieved using the Program Memory Bank Pointer, PBP. The PBP register should be properly configured before the device executes the “Branch” operation using the “JMP” or “CALL” instruction. After that a jump to a non-consecutive Program Memory address which is located in a certain bank selected by the program memory bank pointer bits will occur.

#### • PBP Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	PBP2	PBP1	PBP0
R/W	—	—	—	—	—	R/W	R/W	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as “0”

Bit 2~0 **PBP2~PBP0**: Program Memory Bank Selection  
 000: Bank 0  
 001: Bank 1  
 010: Bank 2  
 011: Bank 3  
 100: Bank 4  
 101: Bank 5  
 110: Bank 6  
 111: Bank 7

### Accumulator – ACC

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

### Program Counter Low Byte Register – PCL

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

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

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

### Option Memory Mapping Register – ORMC

The ORMC register is used to enable Option Memory Mapping function. The Option Memory capacity is 64 words. When a specific pattern of 55H and AAH is consecutively written into this register, the Option Memory Mapping function will be enabled and then the Option Memory code can be read by using the table read instruction. The Option Memory addresses 00H~3FH will be mapped to Program Memory last page addresses C0H~FFH.

To successfully enable the Option Memory Mapping function, the specific pattern of 55H and AAH must be written into the ORMC register in two consecutive instruction cycles. It is therefore recommended that the global interrupt bit EMI should first be cleared before writing the specific pattern, and then set high again at a proper time according to users' requirements after the pattern is successfully written. An internal timer will be activated when the pattern is successfully written. The mapping operation will be automatically finished after a period of  $4 \times t_{LIRC}$ . Therefore, users should read the data in time, otherwise the Option Memory Mapping function needs to be restarted. After the completion of each consecutive write operation to the ORMC register, the timer will recount.

When the table read instructions are used to read the Option Memory code, only "TABRD [m]" instruction can be used. However, care must be taken if the "TABRD [m]" instruction is used, the table pointer defined by the TBHP register must be referenced to the last page. Refer to corresponding sections about the table read instruction for more details.

#### • ORMC Register

Bit	7	6	5	4	3	2	1	0
Name	ORMC7	ORMC6	ORMC5	ORMC4	ORMC3	ORMC2	ORMC1	ORMC0
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 **ORMC7~ORMC0**: Option Memory Mapping specific pattern

When a specific pattern of 55H and AAH is written into this register, the Option Memory Mapping function will be enabled. Note that the register content will be cleared after the MCU is woken up from the IDLE/SLEEP mode.

### Status Register – STATUS

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

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

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

- C is set if an operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation; otherwise C is cleared. C is also affected by a rotate through carry instruction.
- AC is set if an operation results in a carry out of the low nibbles in addition, or no borrow from the high nibble into the low nibble in subtraction; otherwise AC is cleared.
- Z is set if the result of an arithmetic or logical operation is zero; otherwise Z is cleared.



- OV is set if an operation results in a carry into the highest-order bit but not a carry out of the highest-order bit, or vice versa; otherwise OV is cleared.
- PDF is cleared by a system power-up or executing the “CLR WDT” instruction. PDF is set by executing the “HALT” instruction.
- TO is cleared by a system power-up or executing the “CLR WDT” or “HALT” instruction. TO is set by a WDT time-out.
- SC is the result of the “XOR” operation which is performed by the OV flag and the MSB of the current instruction operation result.
- CZ is the operational result of different flags for different instructions. Refer to register definitions for more details.

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

• **STATUS Register**

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

“x”: unknown

- Bit 7      **SC**: The result of the “XOR” operation which is performed by the OV flag and the MSB of the instruction operation result
- Bit 6      **CZ**: The operational result of different flags for different instructions  
 For SUB/SUBM/LSUB/LSUBM instructions, the CZ flag is equal to the Z flag.  
 For SBC/SBCM/LSBC/LSBCM instructions, the CZ flag is the “AND” operation result which is performed by the previous operation CZ flag and current operation Z flag. For other instructions, the CZ flag will not be affected.
- Bit 5      **TO**: Watchdog Time-out flag  
 0: After power up or executing the “CLR WDT” or “HALT” instruction  
 1: A watchdog time-out occurred
- Bit 4      **PDF**: Power down flag  
 0: After power up or executing the “CLR WDT” instruction  
 1: By executing the “HALT” instruction
- Bit 3      **OV**: Overflow flag  
 0: No overflow  
 1: An operation results in a carry into the highest-order bit but not a carry out of the highest-order bit or vice versa
- Bit 2      **Z**: Zero flag  
 0: The result of an arithmetic or logical operation is not zero  
 1: The result of an arithmetic or logical operation is zero
- Bit 1      **AC**: Auxiliary flag  
 0: No auxiliary carry  
 1: An operation results in a carry out of the low nibbles, in addition, or no borrow from the high nibble into the low nibble in subtraction
- Bit 0      **C**: Carry flag  
 0: No carry-out  
 1: An operation results in a carry during an addition operation or if a borrow does not take place during a subtraction operation  
 The “C” flag is also affected by a rotate through carry instruction.

## EEPROM Data Memory

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

### EEPROM Data Memory Structure

The EEPROM Data Memory capacity is 4096×8 bits for this device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in either the byte mode or page mode determined by the mode selection bit, MODE, in the control register, EEC.

### EEPROM Registers

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
EEAL	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
EEAH	—	—	—	—	EEAH3	EEAH2	EEAH1	EEAH0
EED	D7	D6	D5	D4	D3	D2	D1	D0
EEC	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD

**EEPROM Register List**

#### • EEAL Register

Bit	7	6	5	4	3	2	1	0
Name	EEAL7	EEAL6	EEAL5	EEAL4	EEAL3	EEAL2	EEAL1	EEAL0
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      **EEAL7~EEAL0**: Data EEPROM address low byte bit 7 ~ bit 0

#### • EEAH Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	EEAH3	EEAH2	EEAH1	EEAH0
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      **EEAH3~EEAH0**: Data EEPROM address high byte bit 3 ~ bit 0

• **EED Register**

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

Bit 7~0      **D7~D0**: Data EEPROM data bit 7 ~ bit 0

• **EEC Register**

Bit	7	6	5	4	3	2	1	0
Name	EWERTS	EREN	ER	MODE	WREN	WR	RDEN	RD
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      **EWERTS**: Data EEPROM Erase time and Write time select  
 0: Erase time is 3.2ms ( $t_{EEER}$ ) / Write time is 2.2ms ( $t_{EEWR}$ )  
 1: Erase time is 3.7ms ( $t_{EEER}$ ) / Write time is 3.0ms ( $t_{EEWR}$ )

Bit 6      **EREN**: Data EEPROM erase enable  
 0: Disable  
 1: Enable

This bit is used to enable Data EEPROM erase function and must be set high before Data EEPROM erase operations are carried out. This bit will be automatically reset to zero by the hardware after the erase cycle has finished. Clearing this bit to zero will inhibit data EEPROM erase operations.

Bit 5      **ER**: Data EEPROM erase control  
 0: Erase cycle has finished  
 1: Activate an erase cycle

This is the Data EEPROM Erase Control Bit. When this bit is set high by the application program, an erase cycle will be activated. This bit will be automatically reset to zero by hardware after the erase cycle has finished. Setting this bit high will have no effect if the EREN bit has not first been set high.

Bit 4      **MODE**: Data EEPROM operation mode selection  
 0: Byte operation mode  
 1: Page operation mode

This is the EEPROM operation mode selection bit. When the bit is set high by the application program, the Page write, erase or read function will be selected. Otherwise, the byte write or read function will be selected. The EEPROM page buffer size is 16 bytes.

Bit 3      **WREN**: Data EEPROM write enable  
 0: Disable  
 1: Enable

This is the Data EEPROM Write Enable bit which must be set high before Data EEPROM write operations are carried out. Clearing this bit to zero will inhibit Data EEPROM write operations. Note that the WREN bit will automatically be cleared to zero after the write operation is finished.

Bit 2      **WR**: Data EEPROM write control  
 0: Write cycle has finished  
 1: Activate a write cycle

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

- Bit 1      **RDEN**: Data EEPROM read enable  
            0: Disable  
            1: Enable  
            This is the Data EEPROM Read Enable Bit, which must be set high before Data EEPROM read operations are carried out. Clearing this bit to zero will inhibit Data EEPROM read operations.
- Bit 0      **RD**: Data EEPROM read control  
            0: Read cycle has finished  
            1: Activate a read cycle  
            This is the Data EEPROM Read Control Bit. When this bit is set high by the application program, a read cycle will be activated. This bit will be automatically reset to zero by hardware after the read cycle has finished. Setting this bit high will have no effect if the RDEN has not first been set high.

- Note: 1. The EREN, ER, WREN, WR, RDEN and RD cannot be set to “1” at the same time in one instruction.  
2. Ensure that the  $f_{SUB}$  clock is stable before executing the erase or write operation.  
3. Ensure that the erase or write operation is totally complete before changing contents of the EEPROM related registers or activating the IAP function.

## Read Operation from the EEPROM

Reading data from the EEPROM can be implemented by two modes for this device, byte read mode or page read mode, which is controlled by the EEPROM operation mode selection bit, MODE, in the EEC register.

### Byte Read Mode

The EEPROM byte read operation can be executed when the mode selection bit, MODE, is cleared to zero. For a byte read operation the desired EEPROM address should first be placed in the EEAH and EEAL registers, as well as the read enable bit, RDEN, in the EEC register should be set high to enable the read function. Then setting the RD bit high will initiate the EEPROM byte read operation. Note that setting only the RD bit high will not initiate a read operation if the RDEN bit is not set high. When the read cycle terminates, the RD bit will automatically be cleared to zero and the EEPROM data can be read from the EED register. The data will remain in the EED register until another read or write operation is executed. The application program can poll the RD bit to determine when the data is valid for reading.

### Page Read Mode

The EEPROM page read operation can be executed when the mode selection bit, MODE, is set high. The page size can be up to 16 bytes for the page read operation. For a page read operation the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers, as well as the read enable bit, RDEN, in the EEC register should be set high to enable the read function. Then setting the RD bit high will initiate the EEPROM page read operation. Note that setting only the RD bit high will not initiate a read operation if the RDEN bit is not set high. When the current byte read cycle terminates, the RD bit will automatically be cleared to zero indicating that the EEPROM data can be read from the EED register and then the current address will be incremented by one by hardware. The data which is stored in the next EEPROM address can continuously be read out when the RD bit is set high again without reconfiguring the EEPROM address and RDEN control bit. The application program can poll the RD bit to determine when the data is valid for reading.

The EEPROM address higher 8 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page read operation mode the lower 4-bit address value will automatically be incremented by one. However, the higher 8-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally

incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”.

### **Page Erase Operation to the EEPROM**

The EEPROM page erase operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page erase. The internal page buffer will be cleared by hardware after power on reset. When the EEPROM erase enable control bit, namely EREN, is changed from “1” to “0”, the internal page buffer will also be cleared. Note that when the EREN bit is changed from “0” to “1”, the internal page buffer will not be cleared. The EEPROM address higher 8 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page erase operation mode the lower 4-bit address value will automatically be incremented by one after each dummy data byte is written into the EED register. However, the higher 8-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”.

For page erase operations the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers and the dummy data to be written should be placed in the EED register. The maximum data length for a page is 16 bytes. Note that the write operation to the EED register is used to tag address, it must be implemented to determine which addresses to be erased. When the page dummy data is completely written, then the EREN bit in the EEC register should be set high to enable erase operations and the ER bit must be immediately set high to initiate the EEPROM erase process. These two instructions must be executed in two consecutive instruction cycles to activate an erase operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing an erase operation and then set again after a valid erase activation procedure has completed.

As the EEPROM erase cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been erased from the EEPROM. Detecting when the erase cycle has finished can be implemented either by polling the ER bit in the EEC register or by using the EEPROM interrupt. When the erase cycle terminates, the ER bit will be automatically cleared to zero by the microcontroller, indicating that the page data has been erased. The application program can therefore poll the ER bit to determine when the erase cycle has ended. After the erase operation is finished, the EREN bit will be cleared to zero by hardware. The Data EEPROM erased page content will all be zero after a page erase operation.

### **Write Operation to the EEPROM**

Writing data to the EEPROM can be implemented by two modes for this device, byte write mode or page write mode, which is controlled by the EEPROM operation mode selection bit, MODE, in the EEC register.

#### **Byte Write Mode**

The EEPROM byte write operation can be executed when the mode selection bit, MODE, is cleared to zero. For byte write operations the desired EEPROM address should first be placed in the EEAH and EEAL registers and the data to be written should be placed in the EED register. To write data to the EEPROM, the write enable bit, WREN, in the EEC register must first be set high to enable the write function. After this, the WR bit in the EEC register must be immediately set high to initiate a write cycle. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt bit EMI should also first be cleared before implementing any write operations, and then set high again after a valid write activation

procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not set.

As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, indicating that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended. After the write operation is finished, the WREN bit will be cleared to zero by hardware. Note that a byte erase operation will automatically be executed before a byte write operation is successfully activated.

### **Page Write Mode**

Before a page write operation is executed, it is important to ensure that a relevant page erase operation has been successfully executed. The EEPROM page write operation can be executed when the mode selection bit, MODE, is set high. The EEPROM is capable of a 16-byte page write. The internal page buffer will be cleared by hardware after power on reset. When the EEPROM write enable control bit, namely WREN, is changed from “1” to “0”, the internal page buffer will also be cleared. Note that when the WREN bit is changed from “0” to “1”, the internal page buffer will not be cleared. A page write is initiated in the same way as a byte write initiation except that the EEPROM data can be written up to 16 bytes. The EEPROM address higher 8 bits are used to specify the desired page location while the lower 4 bits are used to point to the actual address. In the page write operation mode the lower 4-bit address value will automatically be incremented by one after each data byte is written into the EED register. However, the higher 8-bit address value will not be incremented by hardware. When the EEPROM address lower 4-bit value which is internally incremented by one in the page mode reaches the page boundary, known as 0FH, the EEPROM address lower 4-bit value will stop at 0FH. The EEPROM address will not “roll over”. At this point any data write operations to the EED register will be invalid.

For page write operations the start address of the desired EEPROM page should first be placed in the EEAH and EEAL registers and the data to be written should be placed in the EED register. The maximum data length for a page is 16 bytes. Note that when a data byte is written into the EED register, then the data in the EED register will be loaded into the internal page buffer and the current address value will automatically be incremented by one. When the page data is completely written into the page buffer, then the WREN bit in the EEC register should be set high to enable write operations and the WR bit must be immediately set high to initiate the EEPROM write process. These two instructions must be executed in two consecutive instruction cycles to activate a write operation successfully. The global interrupt enable bit EMI should also first be cleared before implementing any write operations, and then set high again after a valid write activation procedure has completed. Note that setting the WR bit high only will not initiate a write cycle if the WREN bit is not set.

As the EEPROM write cycle is controlled using an internal timer whose operation is asynchronous to microcontroller system clock, a certain time will elapse before the data will have been written into the EEPROM. Detecting when the write cycle has finished can be implemented either by polling the WR bit in the EEC register or by using the EEPROM interrupt. When the write cycle terminates, the WR bit will be automatically cleared to zero by the microcontroller, indicating that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended. After the write operation is finished, the WREN bit will be cleared to zero by hardware.

## Write Protection

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

## EEPROM Interrupt

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

## Programming Considerations

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

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

## Programming Examples

### Reading a Data Byte from the EEPROM – polling method

```
MOV A, 40H           ; setup memory pointer lower byte MP1L
MOV MP1L, A         ; MP1L points to EEC register
MOV A, 01H          ; setup memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4          ; clear MODE bit, select byte operation mode
MOV A, EEPROM_ADRES_H ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L ; user defined low byte address
MOV EEAL, A
SET IAR1.1          ; set RDEN bit, enable read operations
SET IAR1.0          ; start Read Cycle - set RD bit
```

```

BACK:
SZ   IAR1.0           ; check for read cycle end
JMP  BACK
CLR  IAR1             ; disable EEPROM read function
CLR  MP1H
MOV  A, EED           ; move read data to register
MOV  READ_DATA, A

```

#### Reading a Data Page from the EEPROM – polling method

```

MOV  A, 40H           ; set memory pointer low byte MP1L
MOV  MP1L, A         ; MP1 points to EEC register
MOV  A, 01H          ; set memory pointer high byte MP1H
MOV  MP1H, A
SET  IAR1.4          ; set MODE bit, select page operation mode
MOV  A, EEPROM_ADRES_H ; user defined high byte address
MOV  EEAH, A
MOV  A, EEPROM_ADRES_L ; user defined low byte address
MOV  EEAL, A
SET  IAR1.1          ; set RDEN bit, enable read operations
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL READ
CALL READ
:
:
JMP  PAGE_READ_FINISH
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
READ:
SET  IAR1.0          ; start Read Cycle - set RD bit
BACK:
SZ   IAR1.0           ; check for read cycle end
JMP  BACK
MOV  A, EED           ; move read data to register
MOV  READ_DATA, A
RET
:
PAGE_READ_FINISH:
CLR  IAR1             ; disable EEPROM read function
CLR  MP1H

```

#### Erasing a Data Page to the EEPROM – polling method

```

MOV  A, 40H           ; set memory pointer low byte MP1L
MOV  MP1L, A         ; MP1 points to EEC register
MOV  A, 01H          ; set memory pointer high byte MP1H
MOV  MP1H, A
SET  IAR1.4          ; set MODE bit, select page operation mode
MOV  A, EEPROM_ADRES_H ; user defined high byte address
MOV  EEAH, A
MOV  A, EEPROM_ADRES_L ; user defined low byte address
MOV  EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL  WRITE_BUF
CALL  WRITE_BUF
:
:
JMP  Erase_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~

```



```
WRITE_BUF:
MOV A, EEPROM_DATA      ; user defined data, erase mode don't care data value
MOV EED, A
RET
:
Erase_START:
CLR EMI
SET IAR1.6              ; set EREN bit, enable write operations
SET IAR1.5              ; start Write Cycle - set ER bit - executed immediately
                        ; after setting EREN bit

SET EMI
BACK:
SZ IAR1.5              ; check for write cycle end
JMP BACK
CLR MP1H
```

#### **Writing a Data Byte to the EEPROM – polling method**

```
MOV A, 40H              ; set memory pointer low byte MP1L
MOV MP1L, A             ; MP1 points to EEC register
MOV A, 01H              ; set memory pointer high byte MP1H
MOV MP1H, A
CLR IAR1.4             ; clear MODE bit, select byte write mode
MOV A, EEPROM_ADRES    ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES    ; user defined low byte address
MOV EEAL, A
MOV A, EEPROM_DATA     ; user defined data
MOV EED, A
CLR EMI
SET IAR1.3             ; set WREN bit, enable write operations
SET IAR1.2             ; start Write Cycle - set WR bit - executed immediately
                        ; after setting WREN bit

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

#### **Writing a Data Page to the EEPROM – polling method**

```
MOV A, 40H              ; set memory pointer low byte MP1L
MOV MP1L, A             ; MP1 points to EEC register
MOV A, 01H              ; set memory pointer high byte MP1H
MOV MP1H, A
SET IAR1.4             ; set MODE bit, select page operation mode
MOV A, EEPROM_ADRES_H  ; user defined high byte address
MOV EEAH, A
MOV A, EEPROM_ADRES_L  ; user defined low byte address
MOV EEAL, A
; ~~~~ The data length can be up to 16 bytes (Start) ~~~~
CALL WRITE_BUF
CALL WRITE_BUF
:
:
JMP WRITE_START
; ~~~~ The data length can be up to 16 bytes (End) ~~~~
```

```

WRITE_BUF:
MOV  A, EEPROM_DATA      ; user define data
MOV  EED, A
RET
:
WRITE_START:
CLR  EMI
SET  IAR1.3              ; set WREN bit, enable write operations
SET  IAR1.2              ; start Write Cycle - set WR bit - executed immediately
                               ; after setting WREN bit

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

## Oscillators

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

### Oscillator Overview

In addition to being the source of the main system clock the oscillators also provide clock sources for the Watchdog Timer and Time Base Interrupts. External oscillators requiring some external components as well as 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 device has the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

For USB applications, the HXT pins must be connected to a 6MHz or 12MHz crystal if the HXT oscillator is selected to be used.

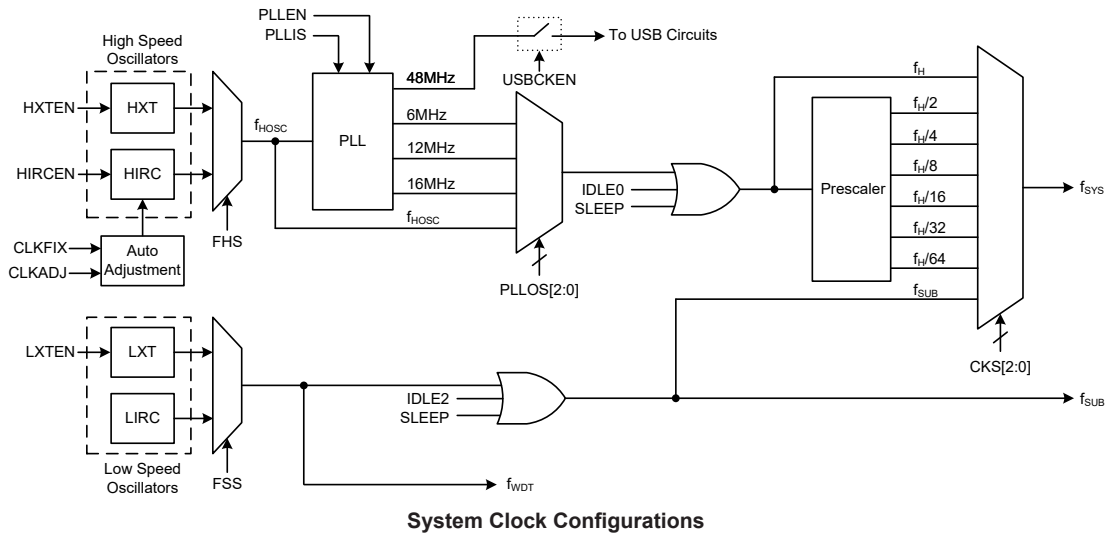
Type	Name	Frequency	Pins
External High Speed Crystal	HXT	6MHz or 12MHz	OSC1/OSC2
Internal High Speed RC Oscillator	HIRC	12MHz	—
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2
Internal Low Speed RC	LIRC	32kHz	—

**Oscillator Types**

### System Clock Configurations

There are several oscillator sources, two high speed oscillators and two low speed oscillators. The high speed oscillator is the internal 12MHz RC oscillator, HIRC, and the external crystal oscillator, HXT. The low speed oscillator is the internal 32kHz RC oscillator, LIRC, and the external 32.768kHz crystal oscillator, LXT. Selecting whether the low or high speed oscillator is used as the system oscillator is implemented using the CKS2~CKS0 bits in the SCC register and as the system clock can be dynamically selected.

The actual source clock used for the low speed oscillator is chosen via the FSS bit in the SCC register while for the high speed oscillator the source clock is selected by the FHS bit in the SCC register. 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. In addition, the internal PLL frequency generator can be enabled by a software control bit to generate various frequencies for the USB interface and system clock.



### Internal PLL Frequency Generator

The internal PLL frequency generator is used to generate the frequency for the USB interface and the system clock. This PLL generator can be enabled or disabled by the PLL control bit PLEN in the PLLC register. After a power on reset, the PLL control bit will be set to 1 to turn on the PLL generator. The PLL generator will provide a fixed 48MHz frequency for the USB operating frequency and another frequency for the system clock source which can be 6MHz, 12MHz or 16MHz. The selection of this system frequency is implemented using the PLEN and PLLOS2~PLLOS0 bits in the PLLC register.

The following table illustrates the high frequency system clock  $f_H$  selected by the related control bits.

PLEN	PLLOS2	PLLOS1	PLLOS0	$f_H$
0	x	x	x	$f_{HOSC}$ – HXT or HIRC, depending on the FHS bit in the SCC register.
1	0	x	x	$f_{HOSC}$ – HXT or HIRC, depending on the FHS bit in the SCC register.
1	1	0	0	$f_{PLL}=6\text{MHz}$
1	1	0	1	$f_{PLL}=12\text{MHz}$
1	1	1	0	$f_{PLL}=16\text{MHz}$
1	1	1	1	Reserved

“x”: don't care

• **PLL Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	—	PLLIS	PLLOS2	PLLOS1	PLLOS0	PLLF	PLLEN
R/W	R/W	—	R/W	R/W	R/W	R/W	R	R/W
POR	0	—	0	0	0	0	0	1

Bit 7        **D7:** Reserved bit, cannot be used and must be fixed at 0

Bit 6        Unimplemented, read as “0”

Bit 5        **PLLIS:** PLL input clock source selection  
               0: 12MHz clock  
               1: 6MHz clock

If FHS=1, when a 12MHz crystal or resonator is used, the HXT frequency is divided by 2 and then multiplied by 8 using the internal PLL circuit, when a 6MHz crystal or resonator is used, the HXT frequency is directly multiplied by 8 using the internal PLL circuit.

If FHS=0, the 12MHz HIRC is selected, this bit will be automatically cleared to zero by hardware.

Bit 4~2     **PLLOS2~PLLOS0:**  $f_{HOSC}$  or  $f_{PLL}$  for  $f_H$  clock source selection  
               0xx:  $f_{HOSC}$   
               100:  $f_{PLL}=6\text{MHz}$   
               101:  $f_{PLL}=12\text{MHz}$   
               110:  $f_{PLL}=16\text{MHz}$   
               111: Reserved

Bit 1        **PLLF:** PLL clock stable flag  
               0: Unstable  
               1: Stable

This bit is used to indicate whether the PLL clock is stable or not. When the PLLEN bit is set to 1 to enable the PLL clock, the PLLF bit will first be cleared to 0 and then set to 1 after the PLL clock is stable.

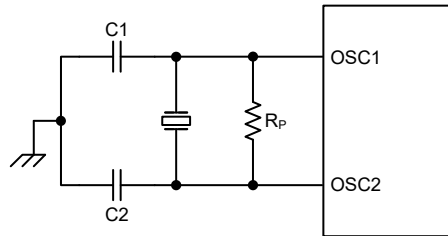
Bit 0        **PLLEN:** PLL enable control  
               0: Disable  
               1: Enable

**External Crystal/Ceramic Oscillator – HXT**

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

For USB applications, the HXT pins must be connected to a 6MHz or 12MHz crystal if the HXT oscillator is selected to be used.

For oscillator stability and to minimise the effects of noise and crosstalk, it is important to ensure that the crystal and any associated resistors and capacitors along with interconnecting lines are all located as close to the MCU as possible.



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

**Crystal/Resonator Oscillator – HXT**

HXT Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
16MHz	0pF	0pF
12MHz	0pF	0pF
8MHz	0pF	0pF
6MHz	0pF	0pF
4MHz	0pF	0pF
1MHz	100pF	100pF

Note: C1 and C2 values are for guidance only.

**Crystal Recommended Capacitor Values**

**Internal High Speed RC Oscillator – HIRC**

The high speed internal RC oscillator is one of the high frequency oscillator choices, which is selected by the FHS bit in the SCC register. It is a fully integrated system oscillator requiring no external components. The internal RC oscillator has a fixed frequency of 12MHz. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised.

**Internal 32kHz Oscillator – LIRC**

The Internal 32kHz System Oscillator is one of the low frequency oscillator choices, which is selected via a software control bit, FSS. It is a fully integrated 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.

**External 32.768kHz Crystal Oscillator – LXT**

The External 32.768kHz Crystal System Oscillator is one of the low frequency oscillator choices, which is selected via a software control bit, FSS. This clock source has a fixed frequency of 32.768kHz and requires a 32.768kHz crystal to be connected between pins XT1 and XT2. The external resistor and capacitor components connected to the 32.768kHz crystal are necessary to provide oscillation. For applications where precise frequencies are essential, these components may be required to provide frequency compensation due to different crystal manufacturing tolerances. After the LXT oscillator is enabled by setting the LXTEN bit to 1, there is a time delay associated with the LXT oscillator waiting for it to start-up.

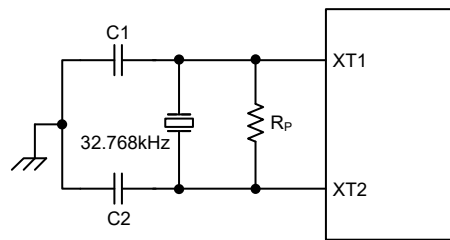
When the microcontroller enters the SLEEP or IDLE Mode, the system clock is switched off to stop microcontroller activity and to conserve power. However, in many microcontroller applications it may be necessary to keep the internal timers operational even when the microcontroller is in the SLEEP or IDLE Mode. To do this, another clock, independent of the system clock, must be provided.

However, for some crystals, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. The external parallel feedback resistor, R<sub>p</sub>, is required.

The pin-shared software control bits determine if the XT1/XT2 pins are used for the LXT oscillator or as I/O or other pin-shared functional pins.

- If the LXT oscillator is not used for any clock source, the XT1/XT2 pins can be used as normal I/O or other pin-shared functional pins.
- If the LXT oscillator is used for any clock source, the 32.768kHz crystal should be connected to the XT1/XT2 pins.

For oscillator stability and to minimise the effects of noise and crosstalk, it is important to ensure that the crystal and any associated resistors and capacitors along with interconnecting lines are all located as close to the MCU as possible.



- Note: 1. R<sub>p</sub>, C1 and C2 are required.  
2. Although not shown XT1/XT2 pins have a parasitic capacitance of around 7pF.

#### External LXT Oscillator

LXT Oscillator C1 and C2 Values		
Crystal Frequency	C1	C2
32.768kHz	10pF	10pF
Note: 1. C1 and C2 values are for guidance only. 2. R <sub>p</sub> =5MΩ~10MΩ is recommended.		

#### 32.768kHz Crystal Recommended Capacitor Values

#### LXT Oscillator Low Power Function

The LXT oscillator can function in one of two modes, the Speed Up Mode and the Low Power Mode. The mode selection is executed using the LXTSP bit in the register.

LXTSP	LXT Mode
0	Low Power
1	Speed Up

When the LXTSP bit is set to high, the LXT Speed Up Mode will be enabled. In the Speed Up Mode the LXT oscillator will power up and stabilise quickly. However, after the LXT oscillator has fully powered up, it can be placed into the Low Power Mode by clearing the LXTSP bit to zero and the oscillator will continue to run but with reduced current consumption. It is important to note that the LXT operating mode switching must be properly controlled before the LXT oscillator clock is

selected as the system clock source. Once the LXT oscillator clock is selected as the system clock source using the CKS2~CKS0 bits and FSS bit in the SCC register, the LXT oscillator operating mode cannot be changed.

It should be noted that, no matter what condition the LXTSP bit is set to, the LXT oscillator will always function normally, the only difference is that it will take more time to start up if it is in the low power mode.

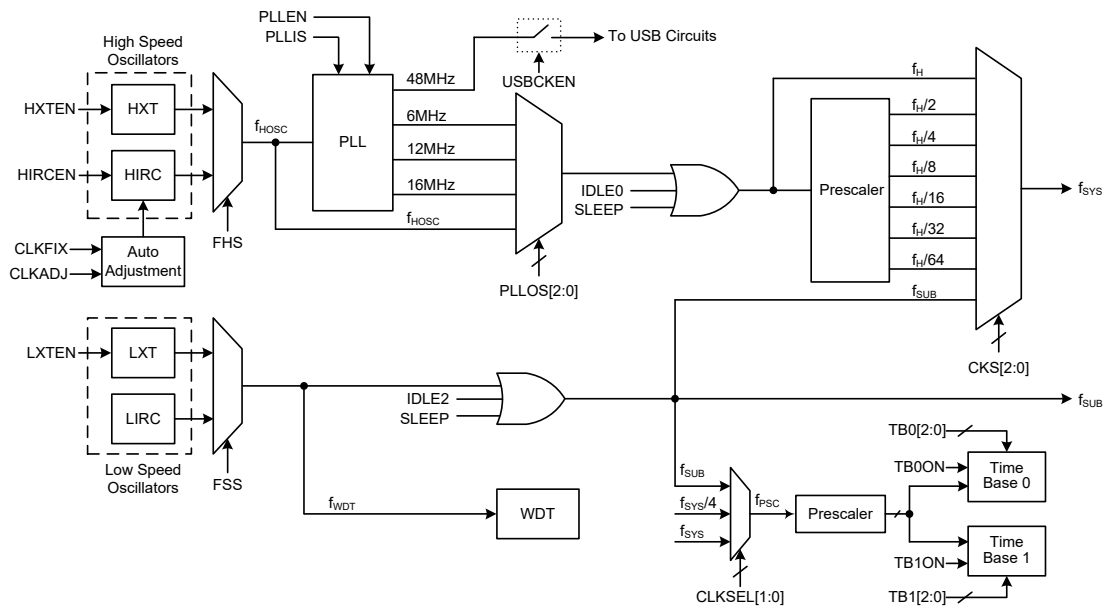
## Operating Modes and System Clocks

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

### System Clocks

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

The main system clock, can come from either a high frequency,  $f_H$ , or low frequency,  $f_{SUB}$ , source, and is selected using the CKS2~CKS0 bits in the SCC register. The high speed system clock can be sourced from an HXT or HIRC oscillator or the PLL output clock, selected via configuring the FHS bit in the SCC register together with the PLEN and PLLS2~PLLS0 bits in the PLLC register. The low speed system clock source can be sourced from the internal clock  $f_{SUB}$ . If  $f_{SUB}$  is selected then it can be sourced by either the LXT or LIRC oscillator, selected via configuring the FSS bit in the SCC register. The other choice, which is a divided version of the high speed system oscillator has a range of  $f_H/2 \sim f_H/64$ .



**Device Clock Configurations**

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

### System Operation Modes

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

Operation Mode	CPU	Register Setting			$f_{SYS}$	$f_H$	$f_{SUB}$	$f_{WDT}$	$f_{PLL}$
		FHIDEN	FSIDEN	CKS2~CKS0					
FAST	On	x	x	000~110	$f_H \sim f_H/64$	On	On	On	On
SLOW	On	x	x	111	$f_{SUB}$	On/Off <sup>(1)</sup>	On	On	On/Off <sup>(3)</sup>
IDLE0	Off	0	1	000~110	Off	Off	On	On	Off
				111	On				
IDLE1	Off	1	1	xxx	On	On	On	On	On
IDLE2	Off	1	0	000~110	On	On	Off	On/Off <sup>(2)</sup>	On
				111	Off				
SLEEP	Off	0	0	xxx	Off	Off	Off	On/Off <sup>(2)</sup>	Off

“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  $f_{WDT}$  clock will be switched on or off which is controlled by the WDT function being enabled or disabled in the IDLE2 and SLEEP mode.
  3. The  $f_{PLL}$  clock can be switched on or off which is controlled by the PLEN bit in the PLLC register in the SLOW mode.
  4. The USB function is inactive in the IDLE0 and SLEEP modes.

#### FAST Mode

This is one of the main operating modes where the microcontroller has all of its functions operational and where the system clock is provided by the internal high speed oscillator. This mode operates allowing the microcontroller to operate normally with a clock source which will come from one of the high speed oscillators, either the HXT or HIRC oscillator, selected via configuring the FHS bit in the SCC register. The high speed oscillator will however first be divided by a ratio ranging from 1 to 64, the actual ratio being selected by the CKS2~CKS0 bits in the SCC register. Although a high speed oscillator is used, running the microcontroller at a divided clock ratio reduces the operating current. In the USB mode, the PLL circuit of which the clock source can be derived from the HXT or HIRC oscillator can generate a clock with a frequency of 6MHz, 12MHz or 16MHz as the device system clock.

#### 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 LXT or LIRC oscillator, selected via configuring the FSS bit in the SCC register.

#### SLEEP Mode

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



### IDLE0 Mode

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

### IDLE1 Mode

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

### IDLE2 Mode

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

## Control Registers

The registers, SCC, HIRCC, HXTC, LXTC and PLLC, 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	—	FHS	FSS	FHIDEN	FSIDEN
HIRCC	CLKADJ	CLKADJF	CLKFIX	—	—	—	HIRCF	HIRCEN
HXTC	—	—	—	—	—	HXTM	HXTF	HXTEN
LXTC	—	—	—	—	—	LXTSP	LXTF	LXTEN
PLLC	D7	—	PL LIS	PLLOS2	PLLOS1	PLLOS0	PLL F	PLLEN

**System Operating Mode Control Register List**

### • SCC Register

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

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

- 000:  $f_H$
- 001:  $f_H/2$
- 010:  $f_H/4$
- 011:  $f_H/8$
- 100:  $f_H/16$
- 101:  $f_H/32$
- 110:  $f_H/64$
- 111:  $f_{SUB}$

These three bits are used to select which clock is used as the system clock source. In addition to the system clock source directly derived from  $f_H$  or  $f_{SUB}$ , a divided version of the high speed system oscillator can also be chosen as the system clock source.

Bit 4 Unimplemented, read as “0”

Bit 3 **FHS**: High Frequency clock,  $f_{HOSC}$ , selection

- 0: HIRC
- 1: HXT

- Bit 2     **FSS**: Low Frequency clock selection  
           0: LIRC  
           1: LXT
- 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, FHS bit or FSS bit. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target clock source.

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

• **HIRCC Register**

Bit	7	6	5	4	3	2	1	0
Name	CLKADJ	CLKADJF	CLKFIX	—	—	—	HIRCF	HIRCEN
R/W	R/W	R/W	R/W	—	—	—	R	R/W
POR	0	0	0	—	—	—	0	1

- Bit 7     **CLKADJ**: HIRC clock automatic adjustment function control in the USB mode  
           0: Disable  
           1: Enable
- Note that if the user selects the HIRC as the system clock, the CLKADJ bit must be set to 1 to adjust the PLL frequency automatically.
- Bit 6     **CLKADJF**: HIRC clock automatic adjustment stable flag  
           0: Unstable  
           1: Stable
- The CLKADJF bit indicates whether the HIRC frequency adjusting operation is completed or not when the CLKADJ bit is set to 1. Users can continuously monitor the CLKADJF bit by application programs to make sure that the HIRC frequency accuracy is stably adjusted in the range of  $\pm 0.25\%$ . The CLKADJF bit can be cleared by the application program.
- Bit 5     **CLKFIX**: clock fix automatic adjustment function control  
           0: Disable  
           1: Enable
- Note that when CLKADJF=1, the CLKFIX bit can be set high to fix the HIRC frequency accuracy in the range of  $\pm 0.25\%$ .
- Bit 4~2   Unimplemented, read as “0”
- Bit 1     **HIRCF**: HIRC oscillator stable flag  
           0: Unstable  
           1: Stable
- This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.
- Bit 0     **HIRCEN**: HIRC oscillator enable control  
           0: Disable  
           1: Enable

• **HXTC Register**

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

Bit 7~3 Unimplemented, read as “0”

Bit 2 **HXTM**: HXT mode selection

0: HXT frequency  $\leq$  10MHz (sink/source current is smaller)

1: HXT frequency > 10MHz (sink/source current is larger)

Note that this bit should be configured correctly according to the used HXT frequency. If HXTM=0 while the HXT frequency is larger than 10MHz, the oscillation performance at a low voltage condition may be not well. If HXTM=1 while the HXT frequency is less than 10MHz, the oscillator frequency and the current may be abnormal.

This bit must be properly configured before the HXT is enabled. When the OSC1 and OSC2 pin functions have been enabled using relevant pin-shared control bits and the HXTEN bit has been set to 1 to enable the HXT oscillator, it is invalid to change the value of the HXTM bit. When the OSC1 or OSC2 pin function is disabled, then the HXTM bit can be changed by software, regardless of the HXTEN bit value.

Bit 1 **HXTF**: HXT oscillator stable flag

0: HXT unstable

1: HXT stable

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

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

0: Disable

1: Enable

• **LXTC Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	LXTSP	LXTF	LXTEN
R/W	—	—	—	—	—	R/W	R	R/W
POR	—	—	—	—	—	0	0	0

Bit 7~3 Unimplemented, read as “0”

Bit 2 **LXTSP**: LXT Speed up control

0: Disable – Low power

1: Enable – Speed up

This bit is used to control whether the LXT oscillator is operating in the low power or Speed-Up mode. When the LXTSP bit is set high, the LXT oscillator will oscillate quickly but consume more power. If the LXTSP bit is cleared to zero, the LXT oscillator will consume less power but take longer time to stabilise. It is important to note that this bit cannot be changed after the LXT oscillator is selected as the system clock source using the CKS2~CKS0 and FSS bits in the SCC register.

Bit 1 **LXTF**: LXT oscillator stable flag

0: LXT unstable

1: LXT stable

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

Bit 0 **LXTEN**: LXT oscillator enable control

0: Disable

1: Enable

• **PLL Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	—	PLLIS	PLLOS2	PLLOS1	PLLOS0	PLLF	PLEN
R/W	R/W	—	R/W	R/W	R/W	R/W	R	R/W
POR	0	—	0	0	0	0	0	1

Bit 7      **D7:** Reserved bit, cannot be used and must be fixed at 0

Bit 6      Unimplemented, read as “0”

Bit 5      **PLLIS:** PLL input clock source selection  
             0: 12MHz clock  
             1: 6MHz clock

If FHS=1, when a 12MHz crystal or resonator is used, the HXT frequency is divided by 2 and then multiplied by 8 using the internal PLL circuit, when a 6MHz crystal or resonator is used, the HXT frequency is directly multiplied by 8 using the internal PLL circuit.

If FHS=0, the 12MHz HIRC is selected, this bit will be automatically cleared to zero by hardware.

Bit 4~2    **PLLOS2~PLLOS0:**  $f_{HOSC}$  or  $f_{PLL}$  for  $f_H$  clock source selection  
             0xx:  $f_{HOSC}$   
             100:  $f_{PLL}=6\text{MHz}$   
             101:  $f_{PLL}=12\text{MHz}$   
             110:  $f_{PLL}=16\text{MHz}$   
             111: Reserved

Bit 1      **PLLF:** PLL clock stable flag  
             0: Unstable  
             1: Stable

This bit is used to indicate whether the PLL clock is stable or not. When the PLEN bit is set to 1 to enable the PLL clock, the PLLF bit will first be cleared to 0 and then set to 1 after the PLL clock is stable.

Bit 0      **PLEN:** PLL enable control  
             0: Disable  
             1: Enable

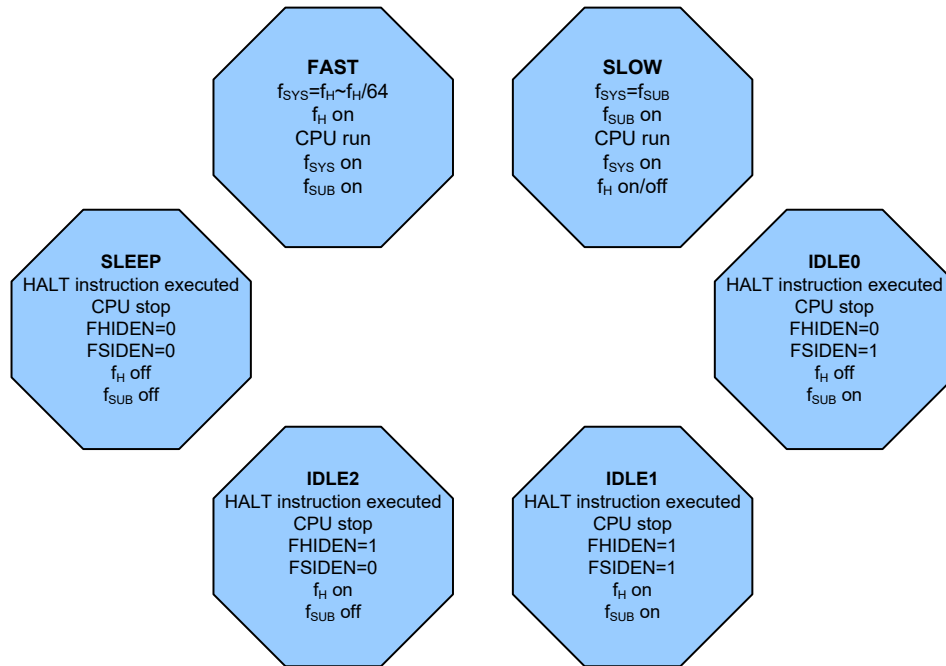
Note: A certain delay is required before the relevant clock is successfully switched to the target clock source after any clock switching setup using the PLLOS2~ PLLOS0 bits. A proper delay time must be arranged before executing the following operations which require immediate reaction with the target 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.

### Operating Mode Switching

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

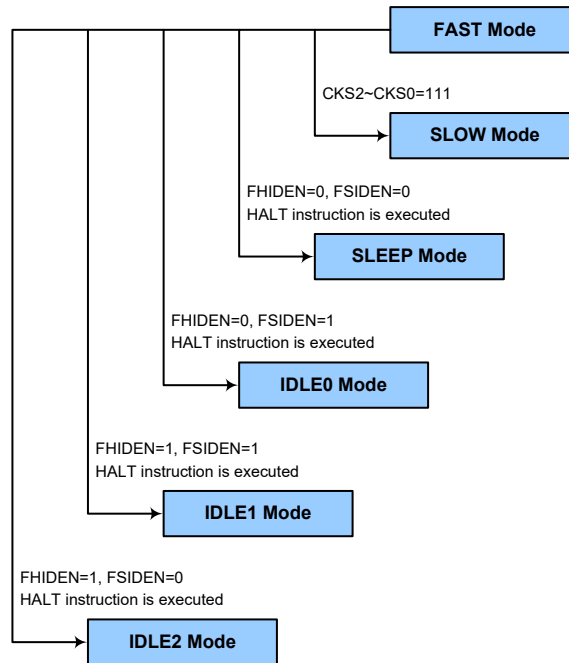
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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



**FAST Mode to SLOW Mode Switching**

When running in the FAST Mode, which uses the high speed system oscillator, and therefore consumes more power, the system clock can switch to run in the SLOW Mode by setting the CKS2~CKS0 bits to “111” in the SCC register. This will then use the low speed system oscillator which will consume less power. Users may decide to do this for certain operations which do not require high performance and can subsequently reduce power consumption.

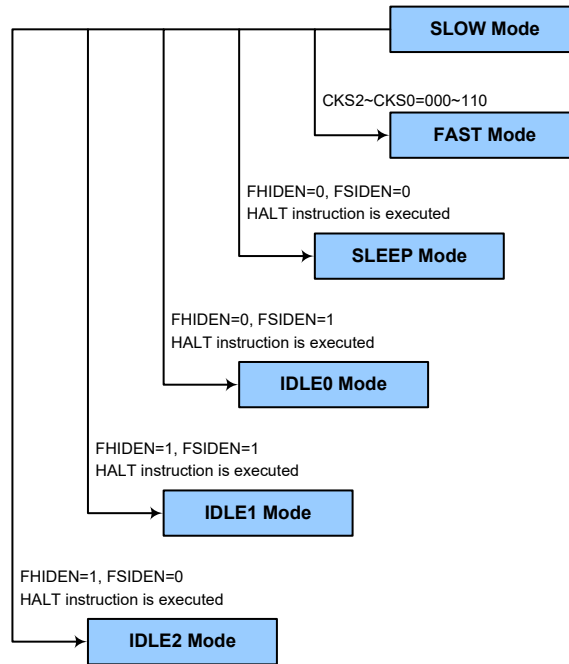
The SLOW Mode is sourced from the LXT or LIRC oscillator determined by the FSS bit in the SCC register and therefore requires this oscillator to be stable before full mode switching occurs.



### SLOW Mode to FAST Mode Switching

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

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



### Entering the SLEEP Mode

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

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

### Entering the IDLE0 Mode

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

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

### Entering the IDLE1 Mode

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

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

### Entering the IDLE2 Mode

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

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

### Standby Current Considerations

As the main reason for entering the SLEEP or IDLE Mode is to keep the current consumption of the device to as low a value as possible, perhaps only in the order of several micro-amps except in the IDLE1 and IDLE2 Mode, there are other considerations which must also be taken into account by

the circuit designer if the power consumption is to be minimised. Special attention must be made to the I/O pins on the device. All high-impedance input pins must be connected to either a fixed high or low level as any floating input pins could create internal oscillations and result in increased current consumption. This also applies to the device which has different package types, as there may be unbonded pins. These must either be setup as outputs or if setup as inputs must have pull-high resistors connected.

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

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

### **Wake-up**

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

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

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

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

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



## 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,  $f_{WDT}$ , which is sourced from either the LIRC or LXT oscillator determined by the FSS bit in the SCC register. 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 LXT oscillator is supplied by an external 32.768kHz crystal. The Watchdog Timer source clock is then subdivided by a ratio of  $2^8$  to  $2^{18}$  to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

### Watchdog Timer Control Register

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

#### • WDTC Register

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

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

10101: Disable

01010: Enable

Other values: Reset MCU

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

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

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

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

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

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

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

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

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

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

These three bits determine the division ratio of the watchdog timer source clock, which in turn determines the time-out period.

#### • RSTFC Register

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

“x”: unknown

Bit 7~4 Unimplemented, read as “0”

Bit 3 **RSTF**: Reset control register software reset flag

Refer to the Internal Reset Control section.

- Bit 2      **LVRF**: LVR function reset flag  
Refer to the Low Voltage Reset section.
- Bit 1      **LRF**: LVRC register software reset flag  
Refer to the Low Voltage Reset section.
- Bit 0      **WRF**: WDT control register software reset flag  
0: Not occurred  
1: Occurred  
  
This bit is set to 1 by the WDT control register software reset and cleared by the application program. This bit can only be cleared to zero by application program.

### Watchdog Timer Operation

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

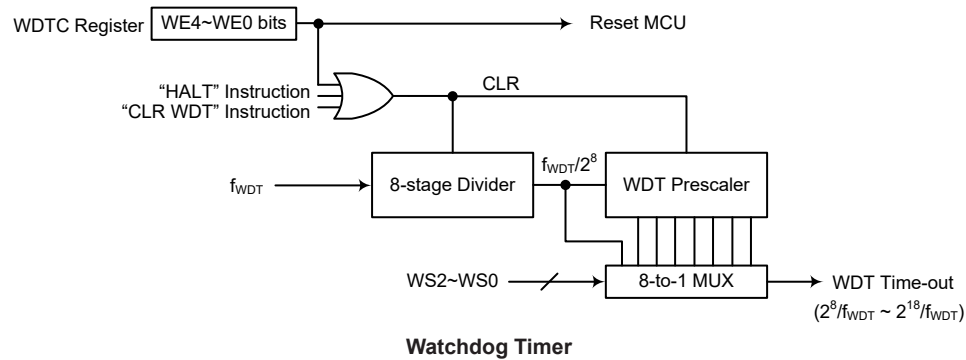
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other value	Reset MCU

**Watchdog Timer Function Control**

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

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

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



## Reset and Initialisation

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

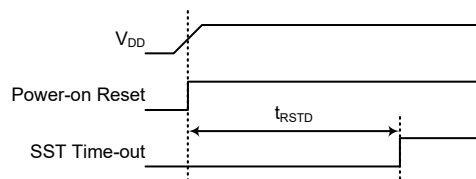
In addition to the power-on reset, another reset exists in the form of a Low Voltage Reset, LVR, where a full reset is implemented in situations where the power supply voltage falls below a certain threshold. Another type of reset is when the Watchdog Timer overflows and resets the microcontroller. All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

## Reset Functions

There are several ways in which a microcontroller reset can occur, through events occurring internally.

### Power-on Reset

The most fundamental and unavoidable reset is the one that occurs after power is first applied to the microcontroller. As well as ensuring that the Program Memory begins execution from the first memory address, a power-on reset also ensures that certain other registers are preset to known conditions. All the I/O port and port control registers will power up in a high condition ensuring that all pins will be first set to inputs.



**Power-On Reset Timing Chart**

### Internal Reset Control

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

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

**Internal Reset Function Control**

• **RSTC Register**

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

Bit 7~0 **RSTC7~RSTC0**: Reset function control  
 01010101: No operation  
 10101010: No operation  
 Other values: Reset MCU

If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time,  $t_{\text{RESET}}$  and the RSTF bit in the RSTFC register will be set to 1. All resets will reset this register to POR value except the WDT time-out hardware warm reset.

• **RSTFC Register**

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

“x”: unknown

Bit 7~4 Unimplemented, read as “0”

Bit 3 **RSTF**: Reset control register software reset flag  
 0: Not occurred  
 1: Occurred

This bit is set to 1 by the RSTC control register software reset and cleared by the application program. This bit can only be cleared to zero by application program.

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

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

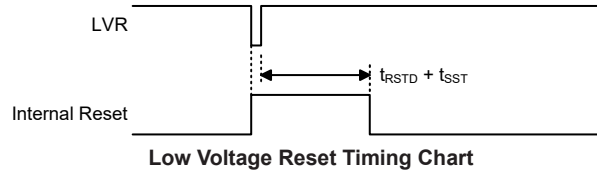
Bit 0 **WRF**: WDT control register software reset flag  
 Refer to the Watchdog Timer Control Register section.

**Low Voltage Reset – LVR**

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

The LVR function is always enabled except in the SLEEP and IDLE mode with a specific LVR voltage  $V_{\text{LVR}}$ . If the supply voltage of the device drops to within a range of  $0.9V \sim V_{\text{LVR}}$  such as might occur when changing the battery in battery powered applications, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set to 1. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between  $0.9V \sim V_{\text{LVR}}$  must exist for a time greater than that specified by  $t_{\text{LVR}}$  in the LVD/LVR Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not

perform a reset function. The actual  $t_{LVR}$  value can be selected by the TLVR1~TLVR0 bits in the TLVRC register. The actual  $V_{LVR}$  value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to any other values by environmental noise, the LVR will reset the device after a delay time,  $t_{SRESET}$ . When this happens, the LRF bit in the RSTFC register will be set to 1. After power on the register will have the value of 01010101B. Note that the LVR function will be automatically disabled when the device enters the SLEEP or IDLE mode.



• **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  
 01010101: 2.1V  
 00110011: 2.55V  
 10011001: 3.15V  
 10101010: 3.8V

Other values: Generates an MCU reset

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

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

• **TLVRC Register**

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **TLVR1~TLVR0**: Minimum low voltage width to reset time ( $t_{LVR}$ )  
 00:  $(7\sim8)\times t_{LIRC}$   
 01:  $(31\sim32)\times t_{LIRC}$   
 10:  $(63\sim64)\times t_{LIRC}$   
 11:  $(127\sim128)\times t_{LIRC}$

• **RSTFC Register**

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

“x”: unknown

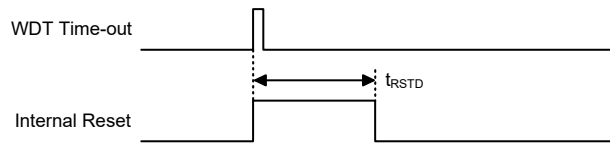
- Bit 7~4      Unimplemented, read as “0”
- Bit 3      **RSTF**: Reset control register software reset flag  
Refer to the Internal Reset Control section.
- Bit 2      **LVRF**: LVR function reset flag  
0: Not occurred  
1: Occurred  
This bit is set to 1 when a specific low voltage reset condition occurs. This bit can only be cleared to zero by application program.
- Bit 1      **LRF**: LVRC register software reset flag  
0: Not occurred  
1: Occurred  
This bit is set high if the LVRC register contains any non-defined LVR voltage register values. This in effect acts like a software reset function. This bit can only be cleared to zero by application program.
- Bit 0      **WRF**: WDT control register software reset flag  
Refer to the Watchdog Timer Control Register section.

**USB Reset**

The device contains a USB circuit, a reset signal can be generated by properly configuring the USB control register to reset the whole device. Refer to the “USB Interface” section for more associated details.

**Watchdog Time-out Reset during Normal Operation**

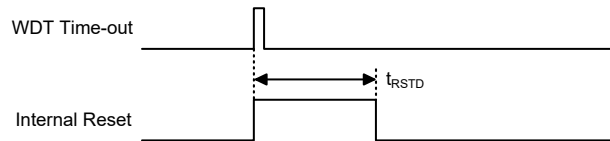
The Watchdog time-out Reset during normal operation in the FAST or SLOW Mode is the same as the LVR Reset except that the Watchdog time-out flag TO will be set to “1”.



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

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

The Watchdog time-out Reset during SLEEP or IDLE Mode is a little different from other kinds of reset. Most of the conditions remain unchanged except that the Program Counter and the Stack Pointer will be cleared to “0” and the TO flag will be set to “1”. Refer to the System Start Up Time Characteristics for  $t_{SST}$  details.



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

### Reset Initial Conditions

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

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

“u”: unchanged

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

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

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

Register	Power-On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
IAR0	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	-xxx xxxx	-uuu uuuu	-uuu uuuu	-uuu uuuu
STATUS	xx00 xxxx	uuuu uuuu	uu1u uuuu	uu11 uuuu
PBP	---- -000	---- --000	---- -000	---- -uuu
IAR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- 0x00	---- u1uu	---- uuuu	---- uuuu
SCC	010- 0000	010- 0000	010- 0000	uuu- uuuu
HIRCC	000- --01	uuu- --01	000- --01	uuu- --uu
HXTC	---- -000	---- -000	---- -000	---- -uuu
LXTC	---- -000	---- -000	---- -000	---- -uuu
PA	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	uuuu uuuu

Register	Power-On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PAPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAWU	0000 0000	0000 0000	0000 0000	uuuu uuuu
LVRC	0101 0101	0101 0101	0101 0101	uuuu uuuu
LVDC	0000 0000	0000 0000	0000 0000	uuuu uuuu
TLVRC	---- --01	---- --01	---- --01	---- --uu
MF10	0000 0000	0000 0000	0000 0000	uuuu uuuu
MF11	0000 0000	0000 0000	0000 0000	uuuu uuuu
MF12	-000 -000	-000 -000	-000 -000	-uuu -uuu
MF13	0000 0000	0000 0000	0000 0000	uuuu uuuu
WDTC	0101 0011	0101 0011	0101 0011	uuuu uuuu
INTEG0	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC0	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC3	-000 -000	-000 -000	-000 -000	-uuu -uuu
PB	---- 1111	---- 1111	---- 1111	---- uuuu
PBC	---- 1111	---- 1111	---- 1111	---- uuuu
PBPU	---- 0000	---- 0000	---- 0000	---- uuuu
PC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PSCR	---- --00	---- --00	---- --00	---- --uu
TB0C	0--- -000	0--- -000	0--- -000	u--- -uuu
TB1C	0--- -000	0--- -000	0--- -000	u--- -uuu
RSTC	0101 0101	0101 0101	0101 0101	uuuu uuuu
INTEG1	---- 0000	---- 0000	---- 0000	---- uuuu
PWRC	0--- -000	0--- -000	0--- -000	u--- -uuu
PGAC0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PGAC1	0000 000-	0000 000-	0000 000-	uuuu uu--
PGACS	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADRL	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRM	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADRH	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
ADCR0	-000 0000	-000 0000	-000 0000	-uuu uuuu
EEAL	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEAH	---- 0000	---- 0000	---- 0000	---- uuuu
EED	0000 0000	0000 0000	0000 0000	uuuu uuuu
ADCR1	0-00 0100	0-00 0100	0-00 0100	u-uu uuuu
ADCS	---0 0000	---0 0000	---0 0000	---u uuuu
CHOP	1010 0011	1010 0011	1010 0011	uuuu uuuu
IREFC	0--0 --00	0--0 --00	0--0 --00	u--u --uu
PVREF	0000 0000	0000 0000	0000 0000	uuuu uuuu
OPA1C	0000 0000	0000 0000	0000 0000	uuuu uuuu
AFEDA1C	---- --00	---- --00	---- --00	---- --uu
AFEDA1L	0000 ----	0000 ----	0000 ----	uuuu ----
AFEDA1H	0000 0000	0000 0000	0000 0000	uuuu uuuu
AFEDA2C	---- --00	---- --00	---- --00	---- --uu



Register	Power-On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
AFEDA2L	0000 ----	0000 ----	0000 ----	uuuu ----
AFEDA2H	0000 0000	0000 0000	0000 0000	uuuu uuuu
ATMC0	0000 0--0	0000 0--0	0000 0--0	uuuu u--u
ATMC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
ATMDL	0000 0000	0000 0000	0000 0000	uuuu uuuu
ATMDH	---- --00	---- --00	---- --00	---- --uu
ATMAL	0000 0000	0000 0000	0000 0000	uuuu uuuu
ATMAH	---- --00	---- --00	---- --00	---- --uu
ATMBL	0000 0000	0000 0000	0000 0000	uuuu uuuu
ATMBH	---- --00	---- --00	---- --00	---- --uu
ATMRP	0000 0000	0000 0000	0000 0000	uuuu uuuu
MDUWR0	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR1	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR2	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR3	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR4	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUWR5	xxxx xxxx	0000 0000	0000 0000	uuuu uuuu
MDUCTRL	00-- ----	00-- ----	00-- ----	uu-- ----
SYSC	0001 --0x	0001 --0x	0001 --0x	uuuu --ux
USB_STAT	11xx 0001	uuxx uuuu	uuxx uuuu	uuxx uuuu
UINT	---- 0000	---- uuuu	---- uuuu	---- uuuu
USC	1000 xxxx	uuuu xuux	uuuu xuux	uuuu xuux
UESR	---- xxxx	---- uuuu	---- uuuu	---- uuuu
UCC	0-0x 0-xx	u-uu u-uu	u-uu u-uu	u-uu u-uu
AWR	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
STL	---- xxxx	---- uuuu	---- uuuu	---- uuuu
SIES	xxxx xxxx	uxxx xuuu	uxxx xuuu	uxxx xuuu
MISC	xxx0 -xxx	xxuu -uux	xxuu -uux	xxuu -uux
SETIO	---- 1110	---- uuuu	---- uuuu	---- uuuu
FIFO3	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
FIFO2	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
FIFO1	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
FIFO0	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu
FRNUM0	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx
FRNUM1	---- -xxx	---- -xxx	---- -xxx	---- -uuu
PLLC	0-00 0001	0-uu uu0u	0-uu uu0u	u-uu uuuu
LCDC0	0000 -000	0000 -000	0000 -000	uuuu -uuu
LCDCP	---- 0-00	---- 0-00	---- 0-00	---- u-uu
LCDC2	000- -000	000- -000	000- -000	uuu- -uuu
ORMC	0000 0000	0000 0000	0000 0000	0000 0000
U0SR	0000 1011	0000 1011	0000 1011	uuuu uuuu
U0CR1	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
U0CR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
U0CR3	---- ---0	---- ---0	---- ---0	---- ---u
BRDH0	0000 0000	0000 0000	0000 0000	uuuu uuuu
BRDL0	0000 0000	0000 0000	0000 0000	uuuu uuuu
UFCR0	--00 0000	--00 0000	--00 0000	--uu uuuu

Register	Power-On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
TXR_RXR0	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RxCNT0	---- -000	---- -000	---- -000	---- -uuu
U1SR	0000 1011	0000 1011	0000 1011	uuuu uuuu
U1CR1	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
U1CR2	0000 0000	0000 0000	0000 0000	uuuu uuuu
U1CR3	---- ---0	---- ---0	---- ---0	---- ---u
BRDH1	0000 0000	0000 0000	0000 0000	uuuu uuuu
BRDL1	0000 0000	0000 0000	0000 0000	uuuu uuuu
UFCR1	--00 0000	--00 0000	--00 0000	--uu uuuu
TXR_RXR1	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
RxCNT1	---- -000	---- -000	---- -000	---- -uuu
SIMC0	111- 0000	111- 0000	111- 0000	uuu- uuuu
SIMC1	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMD	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMA/SIMC2	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMTOC	0000 0000	0000 0000	0000 0000	uuuu uuuu
SGC	0--- ---0	0--- ---0	0--- ---0	u--- ---u
SGN	0000 0000	0000 0000	0000 0000	uuuu uuuu
SGDN	--00 0000	--00 0000	--00 0000	--uu uuuu
OPA3C	000- 0000	000- 0000	000- 0000	uuu- uuuu
OPA2C	0000 0000	0000 0000	0000 0000	uuuu uuuu
ASWA10	0000 000-	0000 000-	0000 000-	uuuu uuu-
ASWA11	0000 000-	0000 000-	0000 000-	uuuu uuu-
ASWA12	0000 000-	0000 000-	0000 000-	uuuu uuu-
ASWA13	0000 000-	0000 000-	0000 000-	uuuu uuu-
ASWA14	0000 000-	0000 000-	0000 000-	uuuu uuu-
ASWA15	0000 0000	0000 0000	0000 0000	uuuu uuuu
ASWA16	0000 0000	0000 0000	0000 0000	uuuu uuuu
ASWA17	0000 0000	0000 0000	0000 0000	uuuu uuuu
ASWA18	--00 0000	--00 0000	--00 0000	--uu uuuu
ASWA19	--00 0000	--00 0000	--00 0000	--uu uuuu
ASWDAC	0--- 0000	0--- 0000	0--- 0000	u--- uuuu
EEC	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMC0	0000 0---	0000 0---	0000 0---	uuuu u---
STMC1	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMDL	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMDH	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMAL	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMAH	0000 0000	0000 0000	0000 0000	uuuu uuuu
STMRP	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0C0	0000 0---	0000 0---	0000 0---	uuuu u---
PTM0C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0DH	---- --00	---- --00	---- --00	---- --uu
PTM0AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM0AH	---- --00	---- --00	---- --00	---- --uu
PTM0RPL	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	Power-On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PTM0RPH	---- --00	---- --00	---- --00	---- --uu
PTM1C0	0000 0---	0000 0---	0000 0---	uuuu u---
PTM1C1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1DH	---- --00	---- --00	---- --00	---- --uu
PTM1AL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1AH	---- --00	---- --00	---- --00	---- --uu
PTM1RPL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTM1RPH	---- --00	---- --00	---- --00	---- --uu
STKPTR	0--- 0000	0--- 0000	0--- 0000	u--- 0000
IECC	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCRL	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCRH	0000 0000	0000 0000	0000 0000	uuuu uuuu
CRCCR	---- ---0	---- ---0	---- ---0	---- ---u
CRCIN	0000 0000	0000 0000	0000 0000	uuuu uuuu
CRCDL	0000 0000	0000 0000	0000 0000	uuuu uuuu
CRCDH	0000 0000	0000 0000	0000 0000	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PE	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PF	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFC	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFPU	0000 0000	0000 0000	0000 0000	uuuu uuuu
PG	---1 1111	---1 1111	---1 1111	---u uuuu
PGC	---1 1111	---1 1111	---1 1111	---u uuuu
PGPU	---0 0000	---0 0000	---0 0000	---u uuuu
FARL	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARH	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD0L	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD0H	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1L	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1H	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2L	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2H	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3L	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3H	0000 0000	0000 0000	0000 0000	uuuu uuuu
FCR	0000 0000	0000 0000	0000 0000	uuuu uuuu
FRCR	00-0 --00	00-u --00	00-u --00	uu-u --uu
IFS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS1	---- ---0	---- ---0	---- ---0	---- ---u
PAS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS0	0000 ----	0000 ----	0000 ----	uuuu ----

Register	Power-On Reset	LVR Reset (Normal Operation)	WDT Time-out (Normal Operation)	WDT Time-out (IDLE/SLEEP)
PCS1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS1	--00 0000	--00 0000	--00 0000	--uu uuuu
PES0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PES1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PFS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PFS1	0000 0000	0000 0000	0000 0000	uuuu uuuu
PGS0	0000 0000	0000 0000	0000 0000	uuuu uuuu
PGS1	---- --00	---- --00	---- --00	---- --uu
PMPS	---- --00	---- --00	---- --00	---- --uu

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

## Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	—	—	PB3	PB2	PB1	PB0
PBC	—	—	—	—	PBC3	PBC2	PBC1	PBC0
PBPU	—	—	—	—	PBPU3	PBPU2	PBPU1	PBPU0
PC	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
PCC	PCC7	PCC6	PCC5	PCC4	PCC3	PCC2	PCC1	PCC0
PCPU	PCPU7	PCPU6	PCPU5	PCPU4	PCPU3	PCPU2	PCPU1	PCPU0
PD	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0
PDC	PDC7	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
PDPU	PDPU7	PDPU6	PDPU5	PDPU4	PDPU3	PDPU2	PDPU1	PDPU0
PE	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
PEC	PEC7	PEC6	PEC5	PEC4	PEC3	PEC2	PEC1	PEC0
PEPU	PEPU7	PEPU6	PEPU5	PEPU4	PEPU3	PEPU2	PEPU1	PEPU0
PF	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
PFC	PFC7	PFC6	PFC5	PFC4	PFC3	PFC2	PFC1	PFC0

Register Name	Bit							
	7	6	5	4	3	2	1	0
PFPU	PFPU7	PFPU6	PFPU5	PFPU4	PFPU3	PFPU2	PFPU1	PFPU0
PG	—	—	—	PG4	PG3	PG2	PG1	PG0
PGC	—	—	—	PGC4	PGC3	PGC2	PGC1	PGC0
PGPU	—	—	—	PGPU4	PGPU3	PGPU2	PGPU1	PGPU0

“—”: Unimplemented, read as “0”

**I/O Logic Function Registers List**

**Pull-high Resistors**

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as an input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using the relevant pull-high control registers and are implemented using weak PMOS transistors.

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

• **PxPU Register**

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

**PxPUn:** I/O Port x Pin pull-high function control

0: Disable

1: Enable

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

**Port A Wake-up**

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

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

• **PAWU Register**

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

Bit 7~0 **PAWU7~PAWU0:** PA7~PA0 pin Wake-up function control

0: Disable

1: Enable

## I/O Port Control Registers

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

### • 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” is the Port name which can be A, B, C, D, E, F or G. However, the actual available bits for each I/O Port may be different.

## I/O Port Power Source Control

This device supports different I/O port power source selections for PC7~PC4. With the exception of OCDS, the multi-power function is only effective when the pin is set to have a digital input or output function.

The port power can come from the power pin VDD, VDDIO or V33O, which is determined using the PMPS1~PMPS0 bits 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/V33O pin should be equal to or less than the device supply power voltage  $V_{DD}$  when the VDDIO/V33O pin is selected as the port power supply pin.

### • PMPS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PMPS1	PMPS0
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **PMPS1~PMPS0:** PC7~PC4 pin power supply selection

0x:  $V_{DD}$

10:  $V_{DDIO}$

11:  $V_{V33O}$

If the PC3 pin is switched to the VDDIO function, and the PMPS1~PMPS0 bits are set to “10”, the VDDIO pin input voltage can be used for PC7~PC4 pin power.

## Pin-shared Functions

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

### Pin-shared Function Selection Registers

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
IFS0	SCSBPS1	SCSBPS0	SDISDAPS1	SDISDAPS0	SCKSCLPS1	SCKSCLPS0	STCKPS	PTP1IPS
IFS1	—	—	—	—	—	—	—	RX1TX1PS
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PCS0	PCS07	PCS06	PCS05	PCS04	—	—	—	—
PCS1	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
PDS1	—	—	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
PES0	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
PES1	PES17	PES16	PES15	PES14	PES13	PES12	PES11	PES10
PFS0	PFS07	PFS06	PFS05	PFS04	PFS03	PFS02	PFS01	PFS00
PFS1	PFS17	PFS16	PFS15	PFS14	PFS13	PFS12	PFS11	PFS10
PGS0	PGS07	PGS06	PGS05	PGS04	PGS03	PGS02	PGS01	PGS00
PGS1	—	—	—	—	—	—	PGS11	PGS10

**Pin-shared Function Selection Register List**

• **IFS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SCSBPS1	SCSBPS0	SDISDAPS1	SDISDAPS0	SCKSCLPS1	SCKSCLPS0	STCKPS	PTP1IPS
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     **SCSBPS1~SCSBPS0**:  $\overline{SCS}$  input source pin selection  
             00: PB2  
             01: PA5  
             10: PC4  
             11: PB2
- Bit 5~4     **SDISDAPS1~SDISDAPS0**: SDI/SDA input source pin selection  
             00: PD5  
             01: PA7  
             10: PC6  
             11: PD5
- Bit 3~2     **SCKSCLPS1~SCKSCLPS0**: SCK/SCL input source pin selection  
             00: PB3  
             01: PA6  
             10: PC5  
             11: PB3
- Bit 1       **STCKPS**: STCK input source pin selection  
             0: PA1  
             1: PA6
- Bit 0       **PTP1IPS**: PTP1I input source pin selection  
             0: PA3  
             1: PC2

• **IFS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	RX1TX1PS
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

- Bit 7~1     Unimplemented, read as “0”
- Bit 0       **RX1TX1PS**: RX1/TX1 input source pin selection  
             0: PC6  
             1: PA5

• **PAS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7~6     **PAS07~PAS06**: PA3 pin-shared function selection  
             00: PA3/PTP1I  
             01: PTP1  
             10: PA3/PTP1I  
             11: PA3/PTP1I
- Bit 5~4     **PAS05~PAS04**: PA2 pin-shared function selection  
             00: PA2/PTCK0  
             01: PA2/PTCK0  
             10: PA2/PTCK0  
             11: XT2



- Bit 3~2    **PAS03~PAS02**: PA1 pin-shared function selection  
 00: PA1/INT0/STCK  
 01: PA1/INT0/STCK  
 10: PA1/INT0/STCK  
 11: OSC1
- Bit 1~0    **PAS01~PAS00**: PA0 pin-shared function selection  
 00: PA0/PTP0I  
 01: PA0/PTP0I  
 10: PA0/PTP0I  
 11: XT1

• **PAS1 Register**

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

- Bit 7~6    **PAS17~PAS16**: PA7 pin-shared function selection  
 00: PA7/ATCK  
 01: SDI/SDA  
 10: PA7/ATCK  
 11: SEG30
- Bit 5~4    **PAS15~PAS14**: PA6 pin-shared function selection  
 00: PA6/STCK  
 01: SCK/SCL  
 10: PA6/STCK  
 11: SEG31
- Bit 3~2    **PAS13~PAS12**: PA5 pin-shared function selection  
 00: PA5/STPI  
 01: SCS  
 10: RX1/TX1  
 11: SEG32
- Bit 1~0    **PAS11~PAS10**: PA4 pin-shared function selection  
 00: PA4  
 01: SDO  
 10: TX1  
 11: PA4

• **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/INT3  
 01: SCK/SCL  
 10: PB3/INT3  
 11: SEG26
- Bit 5~4    **PBS05~PBS04**: PB2 pin-shared function selection  
 00: PB2/INT2  
 01: STP  
 10: SCS  
 11: SEG27

- Bit 3~2    **PBS03~PBS02**: PB1 pin-shared function selection  
 00: PB1/INT1  
 01: STPB  
 10: PB1/INT1  
 11: SEG28
- Bit 1~0    **PBS01~PBS00**: PB0 pin-shared function selection  
 00: PB0  
 01: PB0  
 10: PB0  
 11: SEG29

• **PCS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCS07	PCS06	PCS05	PCS04	—	—	—	—
R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	—	—	—	—

- Bit 7~6    **PCS07~PCS06**: PC3 pin-shared function selection  
 00: PC3/PTCK1  
 01: PTP0  
 10: PC3/PTCK1  
 11: VDDIO
- Bit 5~4    **PCS05~PCS04**: PC2 pin-shared function selection  
 00: PC2/INT5/PTP1I  
 01: PTP1  
 10: PC2/INT5/PTP1I  
 11: LVDIN
- Bit 3~0    Unimplemented, read as “0”

• **PCS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7~6    **PCS17~PCS16**: PC7 pin-shared function selection  
 00: PC7  
 01: SDO  
 10: TX1  
 11: PC7
- Bit 5~4    **PCS15~PCS14**: PC6 pin-shared function selection  
 00: PC6  
 01: SDI/SDA  
 10: RX1/TX1  
 11: PC6
- Bit 3~2    **PCS13~PCS12**: PC5 pin-shared function selection  
 00: PC5  
 01: SCK/SCL  
 10: PC5  
 11: PC5
- Bit 1~0    **PCS11~PCS10**: PC4 pin-shared function selection  
 00: PC4  
 01: SCS  
 10: PC4  
 11: PC4

• **PDS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
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     **PDS07~PDS06:** PD3 pin-shared function selection  
 00: PD3  
 01: PD3  
 10: COM7  
 11: SEG3
- Bit 5~4     **PDS05~PDS04:** PD2 pin-shared function selection  
 00: PD2  
 01: PD2  
 10: COM6  
 11: SEG2
- Bit 3~2     **PDS03~PDS02:** PD1 pin-shared function selection  
 00: PD1  
 01: PD1  
 10: COM5  
 11: SEG1
- Bit 1~0     **PDS01~PDS00:** PD0 pin-shared function selection  
 00: PD0  
 01: PD0  
 10: COM4  
 11: SEG0

• **PDS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	–	–	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
R/W	–	–	R/W	R/W	R/W	R/W	R/W	R/W
POR	–	–	0	0	0	0	0	0

- Bit 7~6     Unimplemented, read as “0”
- Bit 5~4     **PDS15~PDS14:** PD6 pin-shared function selection  
 00: PD6/INT4  
 01: PTP0B  
 10: PD6/INT4  
 11: OSC2
- Bit 3~2     **PDS13~PDS12:** PD5 pin-shared function selection  
 00: PD5  
 01: SDI/SDA  
 10: RX0/TX0  
 11: SEG25
- Bit 1~0     **PDS11~PDS10:** PD4 pin-shared function selection  
 00: PD4  
 01: SDO  
 10: TX0  
 11: SEG24

• **PES0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PES07	PES06	PES05	PES04	PES03	PES02	PES01	PES00
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     **PES07~PES06:** PE3 pin-shared function selection  
 00: PE3  
 01: PE3  
 10: PE3  
 11: SEG19
- Bit 5~4     **PES05~PES04:** PE2 pin-shared function selection  
 00: PE2  
 01: PE2  
 10: PE2  
 11: SEG18
- Bit 3~2     **PES03~PES02:** PE1 pin-shared function selection  
 00: PE1  
 01: PE1  
 10: PE1  
 11: SEG17
- Bit 1~0     **PES01~PES00:** PE0 pin-shared function selection  
 00: PE0  
 01: PE0  
 10: PE0  
 11: SEG16

• **PES1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PES17	PES16	PES15	PES14	PES13	PES12	PES11	PES10
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     **PES17~PES16:** PE7 pin-shared function selection  
 00: PE7  
 01: PE7  
 10: PE7  
 11: SEG23
- Bit 5~4     **PES15~PES14:** PE6 pin-shared function selection  
 00: PE6  
 01: PE6  
 10: PE6  
 11: SEG22
- Bit 3~2     **PES13~PES12:** PE5 pin-shared function selection  
 00: PE5  
 01: PE5  
 10: PE5  
 11: SEG21
- Bit 1~0     **PES11~PES10:** PE4 pin-shared function selection  
 00: PE4  
 01: PE4  
 10: PE4  
 11: SEG20

• **PFS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PFS07	PFS06	PFS05	PFS04	PFS03	PFS02	PFS01	PFS00
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     **PFS07~PFS06:** PF3 pin-shared function selection  
 00: PF3  
 01: PF3  
 10: PF3  
 11: SEG11
- Bit 5~4     **PFS05~PFS04:** PF2 pin-shared function selection  
 00: PF2  
 01: PF2  
 10: PF2  
 11: SEG10
- Bit 3~2     **PFS03~PFS02:** PF1 pin-shared function selection  
 00: PF1  
 01: PF1  
 10: PF1  
 11: SEG9
- Bit 1~0     **PFS01~PFS00:** PF0 pin-shared function selection  
 00: PF0  
 01: PF0  
 10: PF0  
 11: SEG8

• **PFS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PFS17	PFS16	PFS15	PFS14	PFS13	PFS12	PFS11	PFS10
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     **PFS17~PFS16:** PF7 pin-shared function selection  
 00: PF7  
 01: PF7  
 10: PF7  
 11: SEG15
- Bit 5~4     **PFS15~PFS14:** PF6 pin-shared function selection  
 00: PF6  
 01: PF6  
 10: PF6  
 11: SEG14
- Bit 3~2     **PFS13~PFS12:** PF5 pin-shared function selection  
 00: PF5  
 01: PF5  
 10: PF5  
 11: SEG13
- Bit 1~0     **PFS11~PFS10:** PF4 pin-shared function selection  
 00: PF4  
 01: PF4  
 10: PF4  
 11: SEG12

• **PGS0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PGS07	PGS06	PGS05	PGS04	PGS03	PGS02	PGS01	PGS00
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     **PGS07~PGS06:** PG3 pin-shared function selection  
 00: PG3  
 01: ATP/ATP\_PWM1  
 10: PG3  
 11: PG3
- Bit 5~4     **PGS05~PGS04:** PG2 pin-shared function selection  
 00: PG2  
 01: PG2  
 10: V2  
 11: SEG35
- Bit 3~2     **PGS03~PGS02:** PG1 pin-shared function selection  
 00: PG1  
 01: PG1  
 10: C2  
 11: SEG34
- Bit 1~0     **PGS01~PGS00:** PG0 pin-shared function selection  
 00: PG0  
 01: PG0  
 10: C1  
 11: SEG33

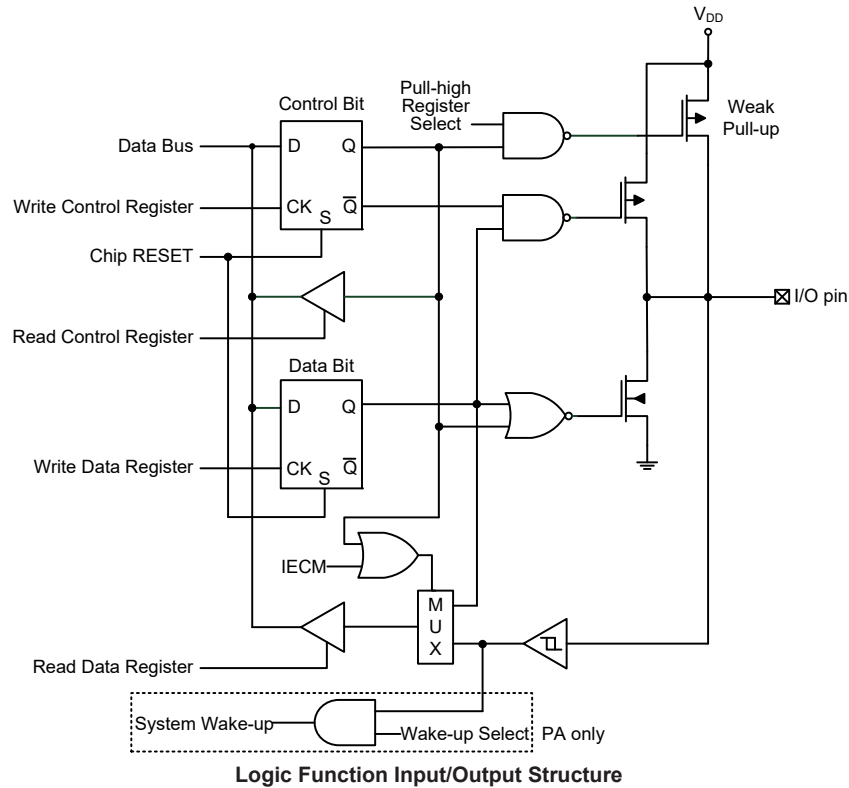
• **PGS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	PGS11	PGS10
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

- Bit 7~2     Unimplemented, read as “0”
- Bit 1~0     **PGS11~PGS10:** PG4 pin-shared function selection  
 00: PG4  
 01: ATPB/ATP\_PWM2  
 10: PG4  
 11: PG4

## I/O Pin Structures

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



## READ PORT function

The READ PORT function is used to manage the reading of the output data from the data latch or I/O pin, which is specially designed for the IEC 60730 self-diagnostic test on the I/O function and A/D paths. There is a register, IECC, which is used to control the READ PORT function. If the READ PORT function is disabled, the pin function will operate as the selected pin-shared function. When a specific data pattern, "11001010", is written into the IECC register, the internal signal named IECM will be set high to enable the READ PORT function. If the READ PORT function is enabled, the value on the corresponding pins will be passed to the accumulator ACC when the read port instruction "mov acc, Px" is executed where the "x" stands for the corresponding I/O port name. Note that the READ PORT mode can only control the input path and will not affect the pin-shared function assignment and the current MCU operation. However, when the IECC register content is set to any other values rather than "11001010", the IECM internal signal will be cleared to 0 to disable the READ PORT function, and the reading path will be set from the latch. If the READ PORT function is disabled, the pin function will operate as the selected pin-shared function.

• **IECC Register**

Bit	7	6	5	4	3	2	1	0
Name	IECS7	IECS6	IECS5	IECS4	IECS3	IECS2	IECS1	IECS0
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     **IECS7~IECS0**: READ PORT function enable control bit 7~ bit 0  
 11001010: IECM=1 – READ PORT function is enabled  
 Others: IECM=0 – READ PORT function is disabled

READ PORT Function Port Control Register Bit – PxC.n	Disabled		Enabled	
	1	0	1	0
I/O Function	Pin value	Data latch value	Pin value	
Digital Input Function				
Digital Output Function (except SIM and UARTn)	0			
SIM: SCK/SCL, SDI/SDA UARTn: RXn/TXn	Pin value			
Analog Function	0			

Note: The value in the above table is the content of the ACC register after “mov a, Px” instruction is executed where “x” means the relevant port name.

**Programming Considerations**

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

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



## Timer Modules – TM

One of the most fundamental functions in any microcontroller device is the ability to control and measure time. To implement time related functions the device includes 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, Input Capture, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has either two or three interrupts. The addition of input and output pins for each TM ensures that users are provided with timing units with a wide and flexible range of features.

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

### Introduction

The device contains several TMs and each individual TM can be categorised as a certain type, namely Standard Type TM, Periodic Type TM or Audio Type TM. Although similar in nature, the different TM types vary in their feature complexity. The common features to all of the Standard, Periodic and Audio 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 three types of TMs are summarised in the accompanying table.

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

### TM Clock Source

The clock source which drives the main counter in each TM can originate from various sources. The selection of the required clock source is implemented using the  $xTnCK2 \sim xTnCK0$  bits in the  $xTMn$  control registers, where “x” stands for S or P or A type TM and “n” stands for the specific TM serial number. For STM and ATM there is no serial number “n” in the relevant pin, register and control bit names since there are only an STM and an ATM in the device. The clock source can be a ratio of the system clock,  $f_{SYS}$ , or the internal high clock,  $f_{IH}$ , 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 Standard or Periodic type TMs has two internal interrupts, one for each of the internal comparator A or comparator P, which generate a TM interrupt when a compare match condition occurs. As the Audio Type TM has three internal comparators and comparator A or comparator B or comparator P compare match functions, it consequently has three internal interrupts. When a TM interrupt is generated, it can be used to clear the counter and also to change the state of the TM output pins.

### TM External Pins

Each of the TMs, irrespective of what type, has one TM input pin, with the label xTCKn while the STM and PTM has another input pin with the label xTPnI. 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 xTCKn input pin can be chosen to have either a rising or falling active edge. The xTCKn pins are also used as the external trigger input pin in single pulse output mode for the STM and PTMn.

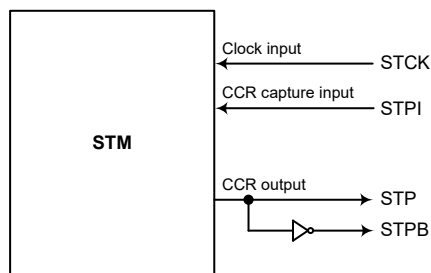
For STM and PTMn, another input pin, xTPnI, is the capture input whose active edge can be a rising edge, a falling edge or both rising and falling edges and the active edge transition type is selected using the xTnIO1~xTnIO0 bits in the xTMnC1 register. There is another capture input, PTCKn, for PTMn capture input mode, which can be used as the external trigger input source except the PTPnI pin.

The TMs each have two output pins with the label xTPn and xTPnB. When the TM is in the Compare Match Output Mode, this pin can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The xTPnB pin outputs the inverted signal of the xTPn. The external xTPn output pin is also the pin where the TM generates the PWM output waveform.

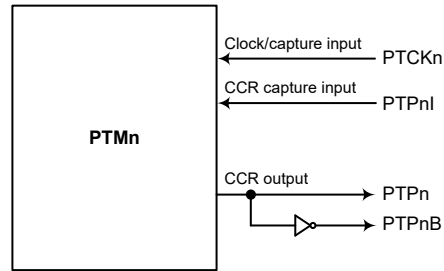
As the TM input and output pins are pin-shared with other functions, the TM input and output functions must first be selected using relevant pin-shared function selection. The details of the pin-shared function selection are described in the pin-shared function section.

STM		PTM		ATM	
Input	Output	Input	Output	Input	Output
STCK, STPI	STP, STPB	PTCK0, PTP0I PTCK1, PTP1I	PTP0, PTP0B PTP1, PTP1B	ATCK	ATP/ATP_PWM1, ATPB/ATP_PWM2

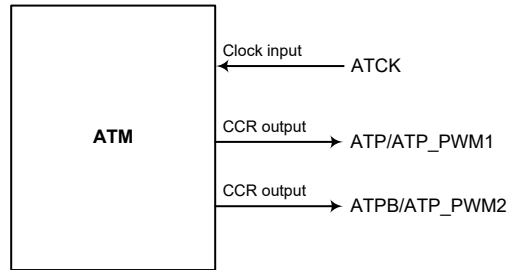
**TM External Pins**



**STM Function Pin Block Diagram**



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

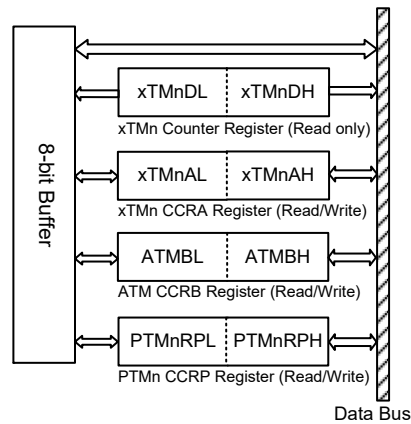


**ATM Function Pin Block Diagram**

### Programming Considerations

The TM Counter Registers and the Capture/Compare CCRA, CCRB 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, CCRB 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, CCRB and CCRP low byte registers, named xTMnAL, ATMBL and PTMnRPL, using the following access procedures. Accessing the CCRA, CCRB or CCRP low byte registers without following these access procedures will result in unpredictable values.

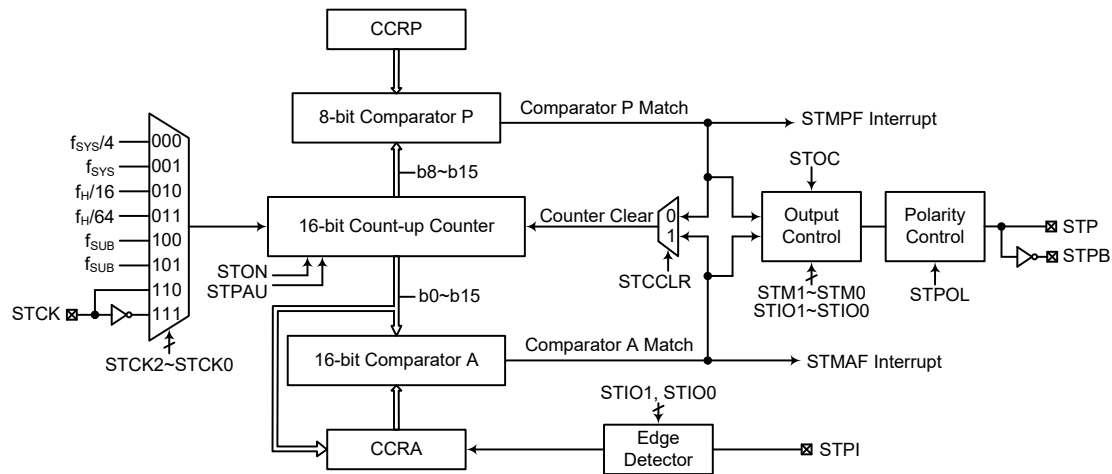


The following steps show the read and write procedures:

- Writing Data to CCRA or CCRB or CCRP
  - ♦ Step 1. Write data to Low Byte xTMnAL, ATMBL or PTMnRPL
    - Note that here data is only written to the 8-bit buffer.
  - ♦ Step 2. Write data to High Byte xTMnAH, ATMBH or PTMnRPH
    - 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, CCRB or CCRP
  - ♦ Step 1. Read data from the High Byte xTMnDH, xTMnAH, ATMBH or PTMnRPH
    - 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 xTMnDL, xTMnAL, ATMBL or PTMnRPL
    - This step reads data from the 8-bit buffer.

## Standard Type TM – STM

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



Note: 1. The STM external pins are pin-shared with other functions, therefore before using the STM function, the pin-shared function registers must have been set properly to enable the STM pin function. The STCK and STPI pins, if used, must also be set as an input by setting the corresponding bits in the port control register.

2. The STPB is the inverted signal of the STP.

**16-bit Standard Type TM Block Diagram**

## Standard Type TM Operation

The Standard Type TM core is a 16-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 comparator is 8-bit wide whose value is compared with the highest 8-bit in the counter while the CCRA is 16-bit and therefore compares all counter bits.

The only way of changing the value of the 16-bit counter using the application program, is to clear the counter by changing the STON 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, an STM interrupt signal will also usually be generated. The Standard Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control two output pins. All operating setup conditions are selected using relevant internal registers.

### Standard Type TM Register Description

Overall operation of the Standard TM is controlled using a series of registers. A read only register pair exists to store the internal counter 16-bit value, while a read/write register pair exists to store the internal 16-bit CCRA value. The STMRP register is used to store the 8-bit CCRP bits. 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
STMC0	STPAU	STCK2	STCK1	STCK0	STON	—	—	—
STMC1	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
STMDL	D7	D6	D5	D4	D3	D2	D1	D0
STMDH	D15	D14	D13	D12	D11	D10	D9	D8
STMAL	D7	D6	D5	D4	D3	D2	D1	D0
STMAH	D15	D14	D13	D12	D11	D10	D9	D8
STMRP	D7	D6	D5	D4	D3	D2	D1	D0

16-bit Standard TM Register List

#### • STMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	STPAU	STCK2	STCK1	STCK0	STON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **STPAU**: STM 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 STM 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 **STCK2~STCK0**: STM counter clock selection

000:  $f_{SYS}/4$   
 001:  $f_{SYS}$   
 010:  $f_H/16$   
 011:  $f_H/64$   
 100:  $f_{SUB}$   
 101:  $f_{SUB}$   
 110: STCK rising edge clock  
 111: STCK falling edge clock

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

Bit 3      **STON**: STM counter on/off control  
             0: Off  
             1: On

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

Bit 2~0      Unimplemented, read as “0”

• **STMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	STM1	STM0	STIO1	STIO0	STOC	STPOL	STDPX	STCCLR
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      **STM1~STM0**: STM operating mode selection  
             00: Compare Match Output Mode  
             01: Capture Input Mode  
             10: PWM Output Mode or Single Pulse Output Mode  
             11: Timer/Counter Mode

These bits setup the required operating mode for the STM. To ensure reliable operation the STM should be switched off before any changes are made to the STM1 and STM0 bits. In the Timer/Counter Mode, the STM output pin state is undefined.

Bit 5~4      **STIO1~STIO0**: STM external pin function selection

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

PWM Output Mode/Single Pulse Output Mode  
             00: PWM output inactive state  
             01: PWM output active state  
             10: PWM output  
             11: Single Pulse Output

Capture Input Mode  
             00: Input capture at rising edge of STPI  
             01: Input capture at falling edge of STPI  
             10: Input capture at rising/falling edge of STPI  
             11: Input capture disabled

Timer/Counter Mode  
             Unused

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

In the Compare Match Output Mode, the STIO1 and STIO0 bits determine how the STM output pin changes state when a compare match occurs from the Comparator A. The STM 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 STM output pin should be setup using the STOC bit in the STMC1 register. Note that the output

level requested by the STIO1 and STIO0 bits must be different from the initial value setup using the STOC bit otherwise no change will occur on the STM output pin when a compare match occurs. After the STM output pin changes state, it can be reset to its initial level by changing the level of the STON bit from low to high.

In the PWM Output Mode, the STIO1 and STIO0 bits determine how the STM 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 change the values of the STIO1 and STIO0 bits only after the STM has been switched off. Unpredictable PWM outputs will occur if the STIO1 and STIO0 bits are changed when the STM is running.

- Bit 3     **STOC:** STM STP 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 STM output pin. Its operation depends upon whether STM 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 STM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the STM 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 STM output pin when the STON bit changes from low to high.

- Bit 2     **STPOL:** STM STP output polarity control  
    0: Non-inverted  
    1: Inverted

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

- Bit 1     **STDPX:** STM PWM duty/period 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     **STCCLR:** STM 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 Standard TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the STCCLR 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 STCCLR bit is not used in the PWM Output, Single Pulse Output or Capture Input Mode.

• **STMDL 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:** STM Counter Low Byte Register bit 7 ~ bit 0  
STM 16-bit Counter bit 7 ~ bit 0

• **STMDH Register**

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

Bit 7~0     **D15~D8:** STM Counter High Byte Register bit 7 ~ bit 0  
STM 16-bit Counter bit 15 ~ bit 8

• **STMAL 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:** STM CCRA Low Byte Register bit 7 ~ bit 0  
STM 16-bit CCRA bit 7 ~ bit 0

• **STMAH 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:** STM CCRA High Byte Register bit 7 ~ bit 0  
STM 16-bit CCRA bit 15 ~ bit 8

• **STMRP 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:** STM CCRP 8-bit Register, Compared with the STM Counter bit 15~bit 8  
Comparator P Match period=  
0: 65536 STM clocks  
1~255: (1~255)×256 STM clocks

These eight bits are used to setup the value on the internal CCRP 8-bit register, which are then compared with the internal counter's highest eight bits. The result of this comparison can be selected to clear the internal counter if the STCCLR bit is cleared to zero. Setting the STCCLR 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 eight counter bits, the compare values exist in 256 clock cycle multiples. Clearing all eight bits to zero is in effect allowing the counter to overflow at its maximum value.



## Standard Type TM Operation Modes

The Standard Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the STM1 and STM0 bits in the STMC1 register.

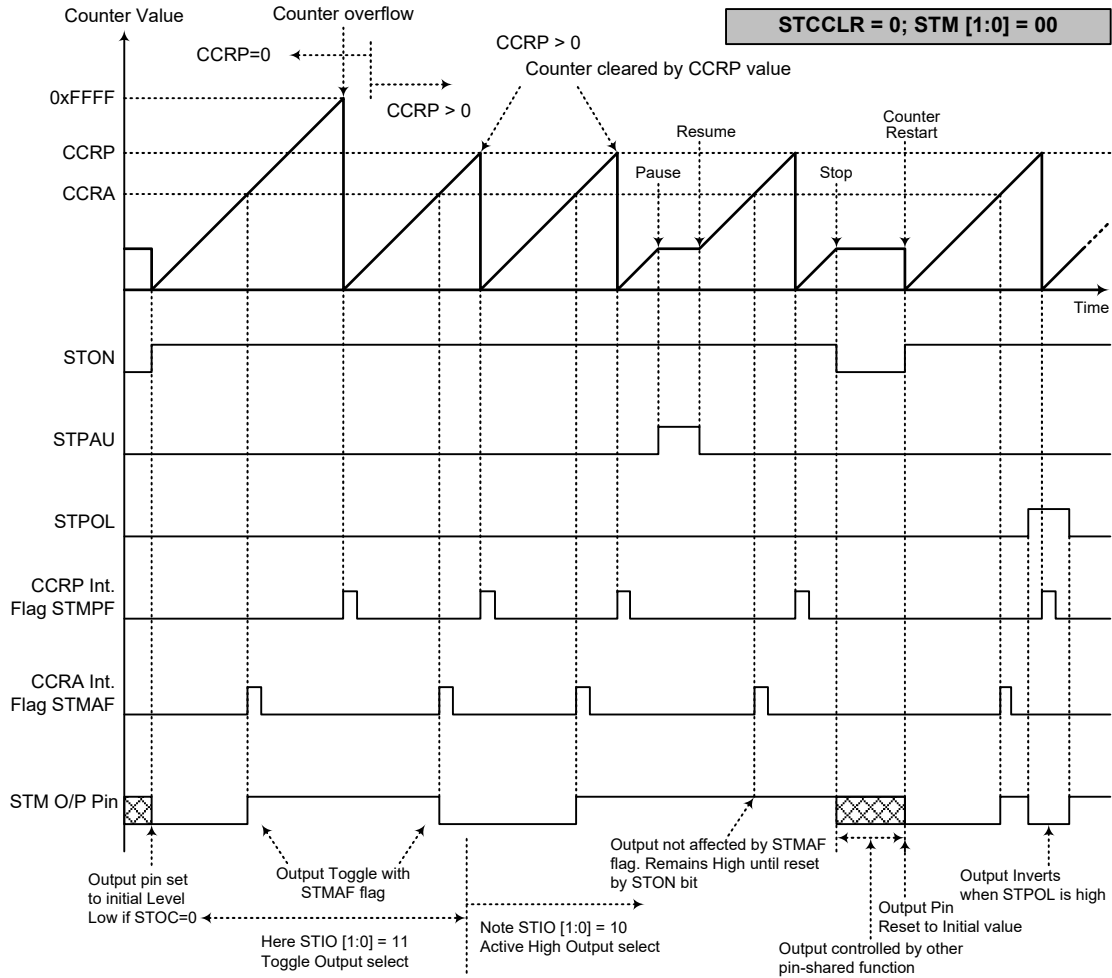
### Compare Match Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 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 STCCLR 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 STMAF and STMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the STCCLR bit in the STMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the STMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when STCCLR is high no STMPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA cannot be cleared to “0”.

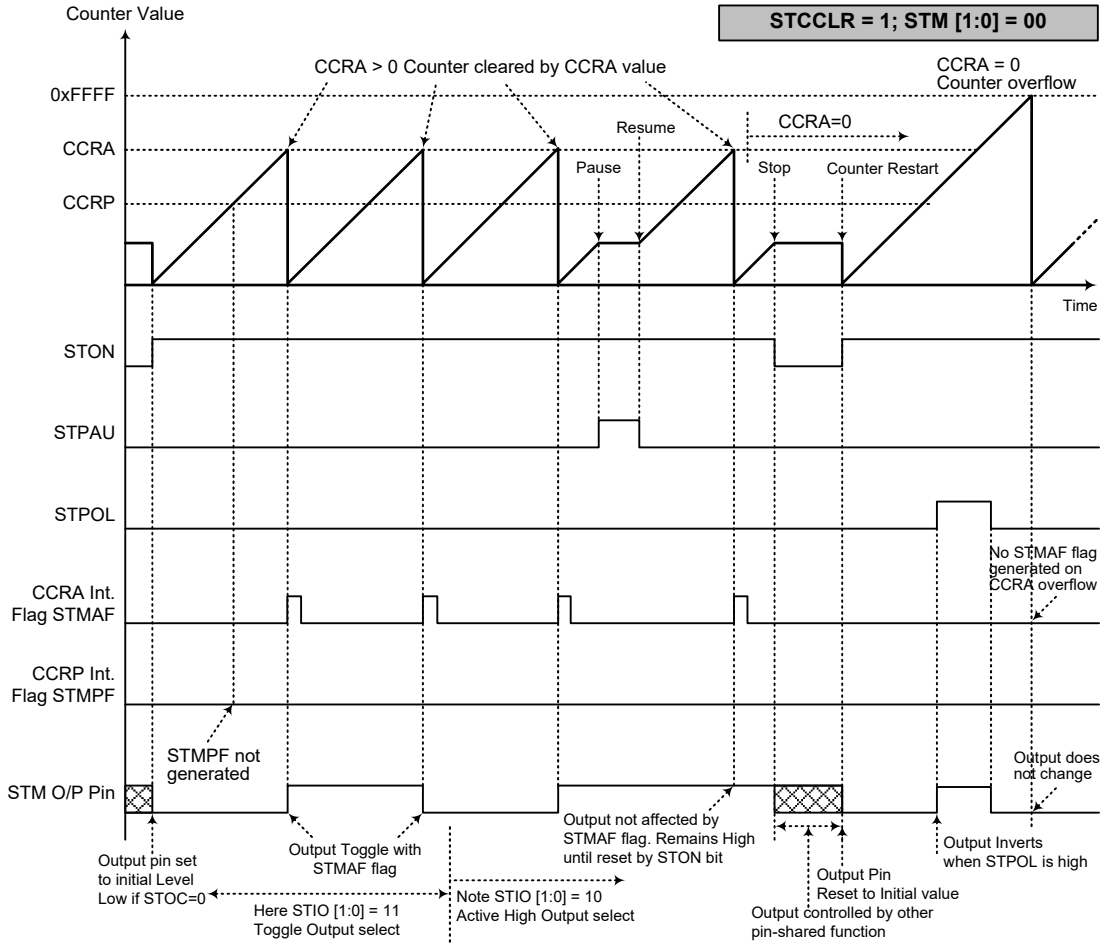
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 16-bit, FFFF Hex, value, however here the STMAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the STM output pin, will change state. The STM output pin condition however only changes state when an STMAF interrupt request flag is generated after a compare match occurs from Comparator A. The STMPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the STM output pin. The way in which the STM output pin changes state are determined by the condition of the STIO1 and STIO0 bits in the STMC1 register. The STM output pin can be selected using the STIO1 and STIO0 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 STM output pin, which is setup after the STON bit changes from low to high, is setup using the STOC bit. Note that if the STIO1 and STIO0 bits are zero then no pin change will take place.



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

- Note: 1. With STCCLR=0 a Comparator P match will clear the counter  
 2. The STM output pin is controlled only by the STMAF flag  
 3. The output pin is reset to initial state by an STON bit rising edge



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

- Note: 1. With STCCLR=1 a Comparator A match will clear the counter  
 2. The STM output pin is controlled only by the STMAF flag  
 3. The output pin is reset to its initial state by an STON bit rising edge  
 4. The STMPF flag is not generated when STCCLR=1

### Timer/Counter Mode

To select this mode, bits STM1 and STM0 in the STMC1 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 STM 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 STM 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 STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 10 respectively. The PWM function within the STM 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 STM output pin, a square waveform 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 STCCLR 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 register is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the STDPX bit in the STMC1 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 STOC bit in the STMC1 register is used to select the required polarity of the PWM waveform while the two STIO1 and STIO0 bits are used to enable the PWM output or to force the STM output pin to a fixed high or low level. The STPOL bit is used to reverse the polarity of the PWM output waveform.

• **16-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=0**

CCRP	1~255	0
Period	CCRP×256	65536
Duty	CCRA	

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

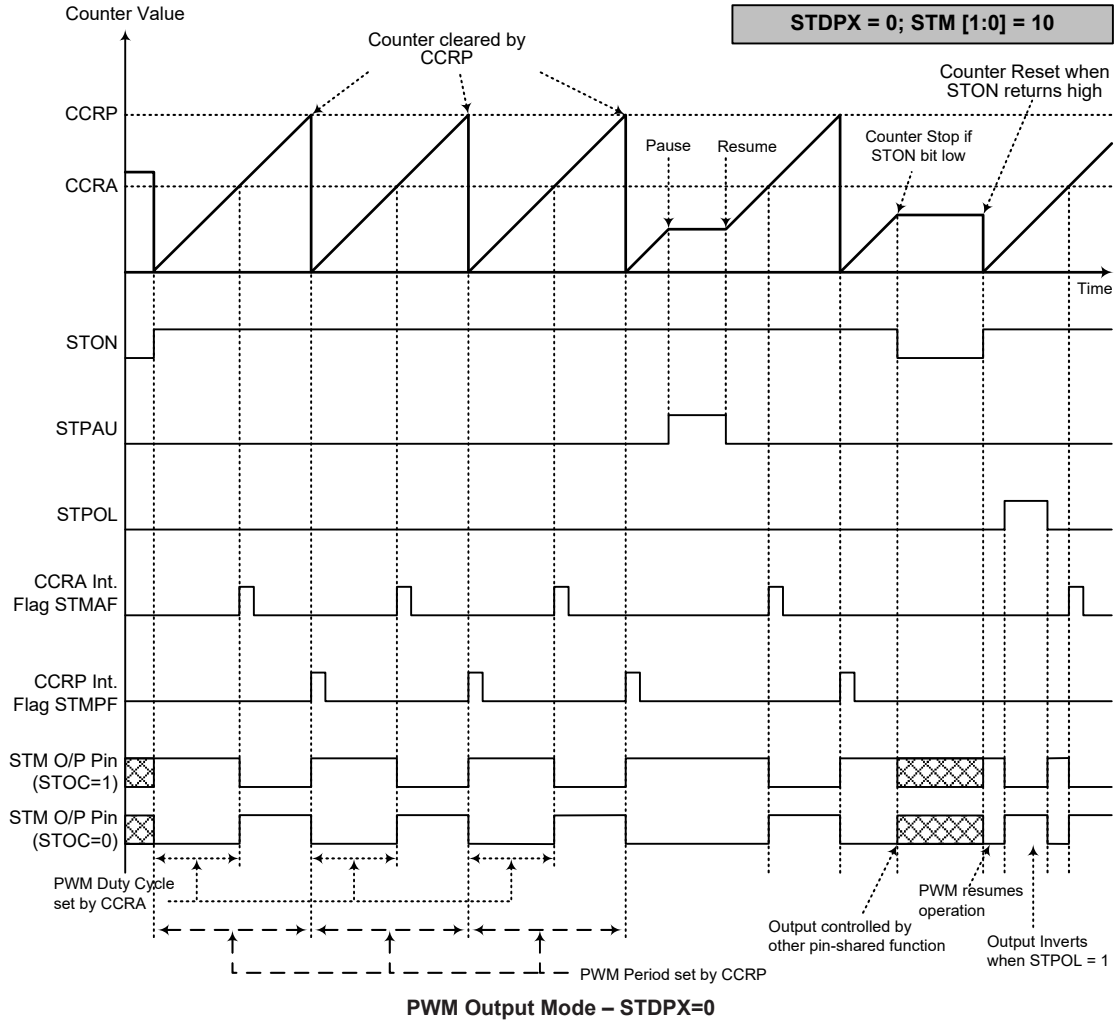
The STM PWM output frequency= $(f_{SYS}/4)/(2\times 256)=f_{SYS}/2048=4\text{kHz}$ , duty= $128/(2\times 256)=25\%$ .

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

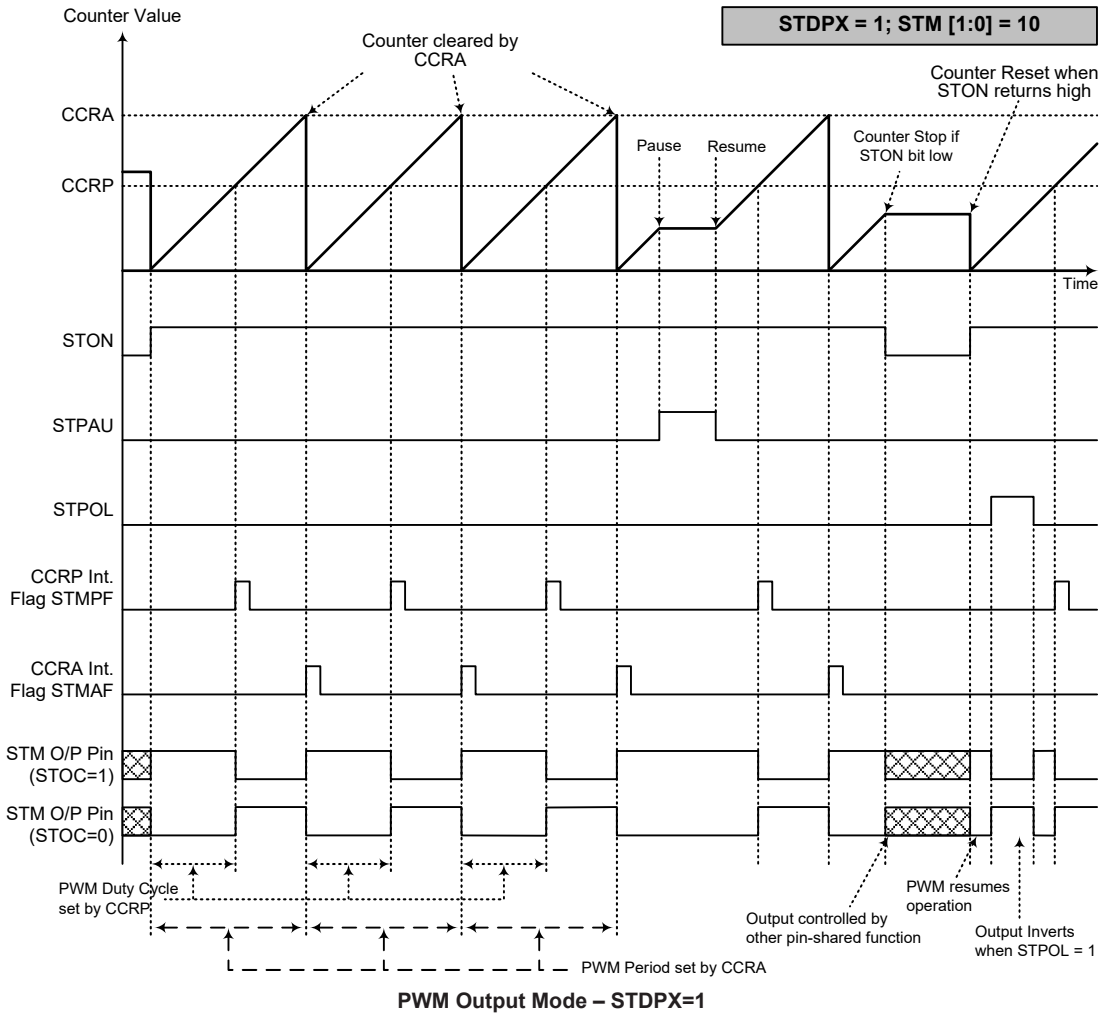
• **16-bit STM, PWM Output Mode, Edge-aligned Mode, STDPX=1**

CCRP	1~255	0
Period	CCRA	
Duty	CCRP×256	65536

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



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



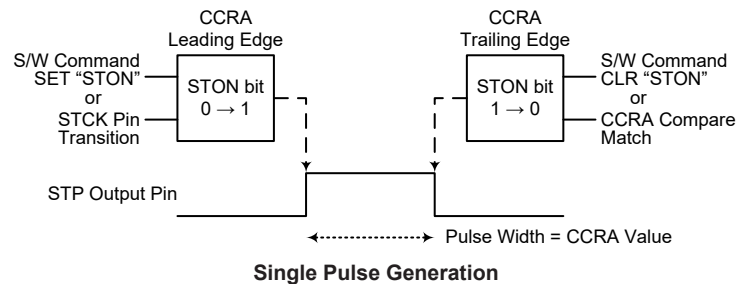
- Note: 1. Here STDPX=1 – Counter cleared by CCRA  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues even when STIO[1:0]=00 or 01  
 4. The STCCLR bit has no influence on PWM operation

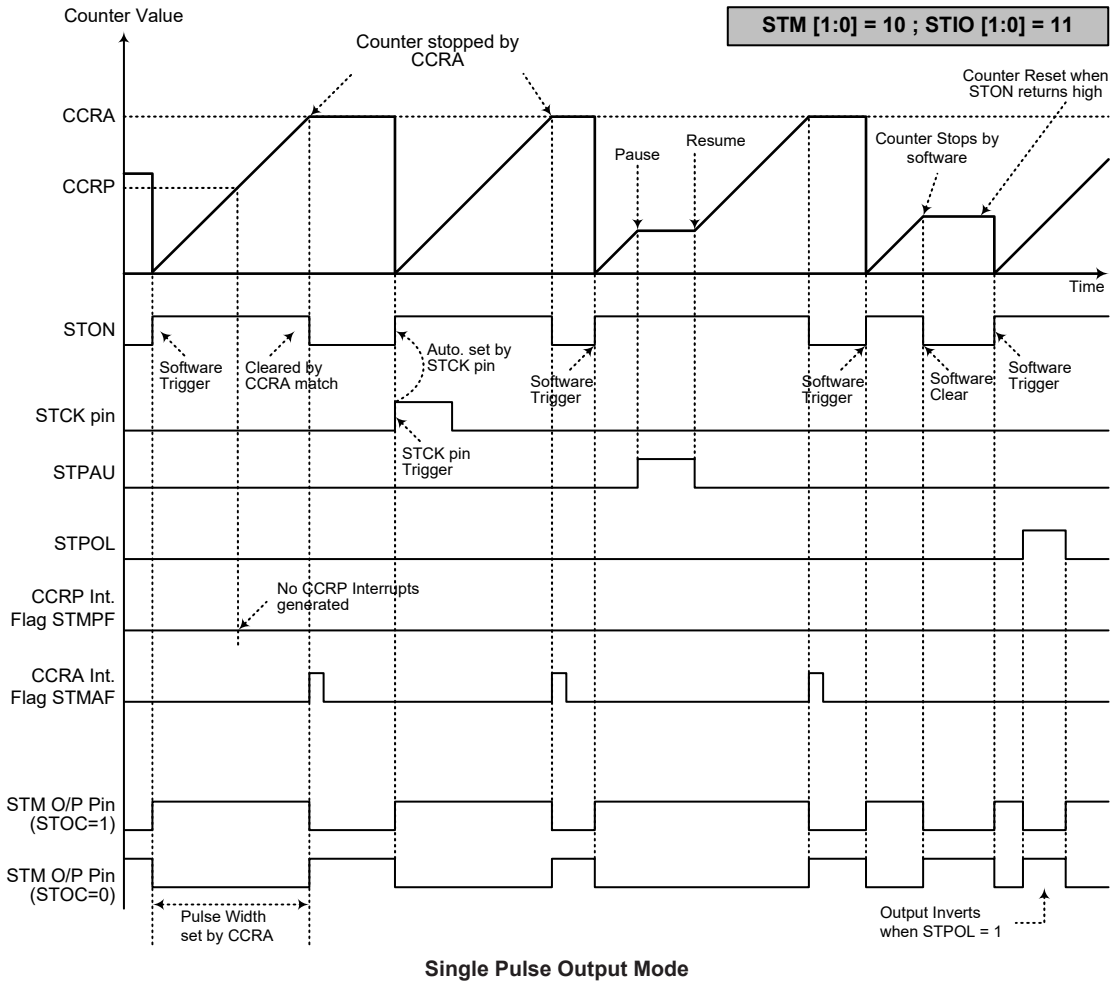
### Single Pulse Output Mode

To select this mode, bits STM1 and STM0 in the STMC1 register should be set to 10 respectively and also the STIO1 and STIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the STM output pin.

The trigger for the pulse output leading edge is a low to high transition of the STON bit, which can be implemented using the application program. However in the Single Pulse Output Mode, the STON bit can also be made to automatically change from low to high using the external STCK pin, which will in turn initiate the Single Pulse output. When the STON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The STON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the STON 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 STON 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 an STM interrupt. The counter can only be reset back to zero when the STON bit changes from low to high when the counter restarts. In the Single Pulse Output Mode CCRP is not used. The STCCLR and STDPX bits are not used in this mode.





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

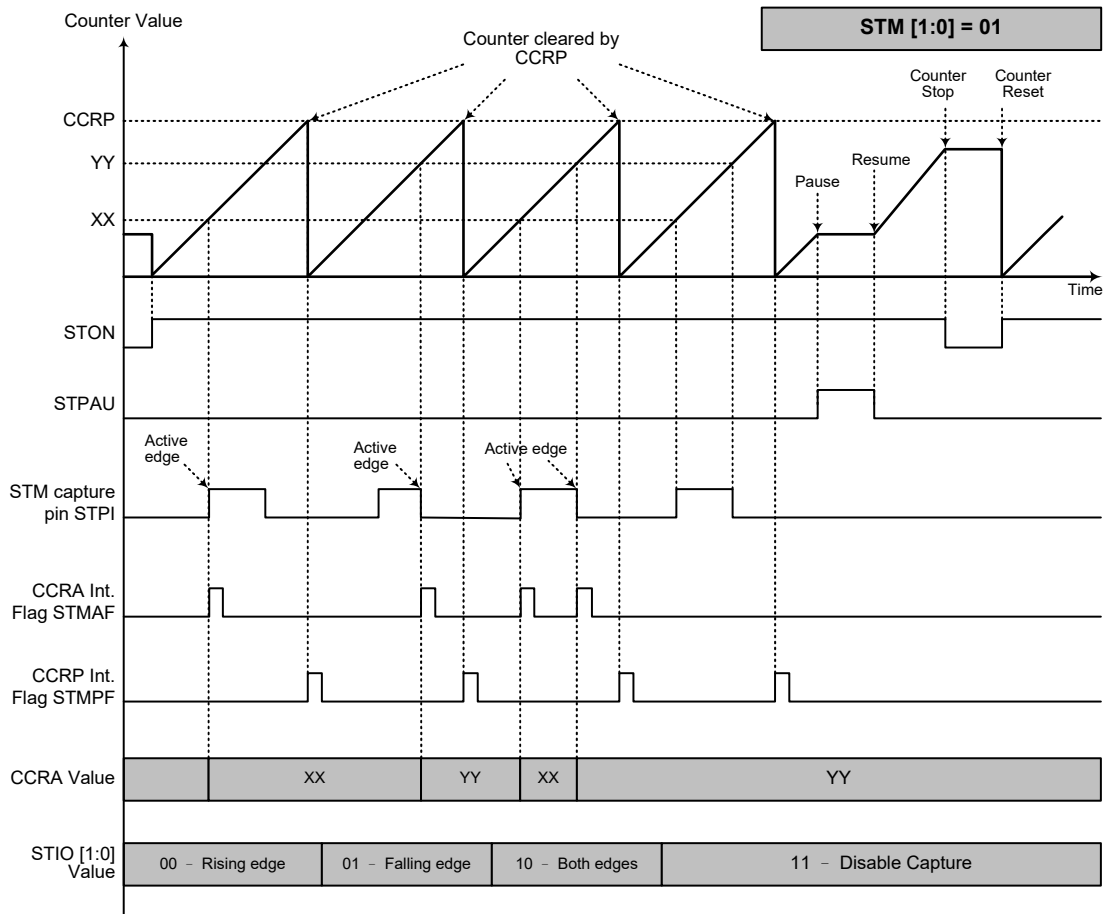


### **Capture Input Mode**

To select this mode bits STM1 and STM0 in the STMC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the STPI pin, whose active edge can be a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the STIO1 and STIO0 bits in the STMC1 register. The counter is started when the STON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the STPI pin the present value in the counter will be latched into the CCRA registers and an STM interrupt generated. Irrespective of what events occur on the STPI pin the counter will continue to free run until the STON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, an STM interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The STIO1 and STIO0 bits can select the active trigger edge on the STPI pin to be a rising edge, falling edge or both edge types. If the STIO1 and STIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the STPI pin, however it must be noted that the counter will continue to run. The STCCLR and STDPX bits are not used in this mode.

There are some considerations that should be noted. If the captured pulse width is less than 2 timer clock periods, it may be ignored by hardware. After the counter value is latched to the CCRA registers by an active capture edge, the STMAF flag will be set high after 0.5 timer clock periods. The delay time from the active capture edge received to the action of latching counter value to CCRA registers is less than 1.5 timer clock periods.

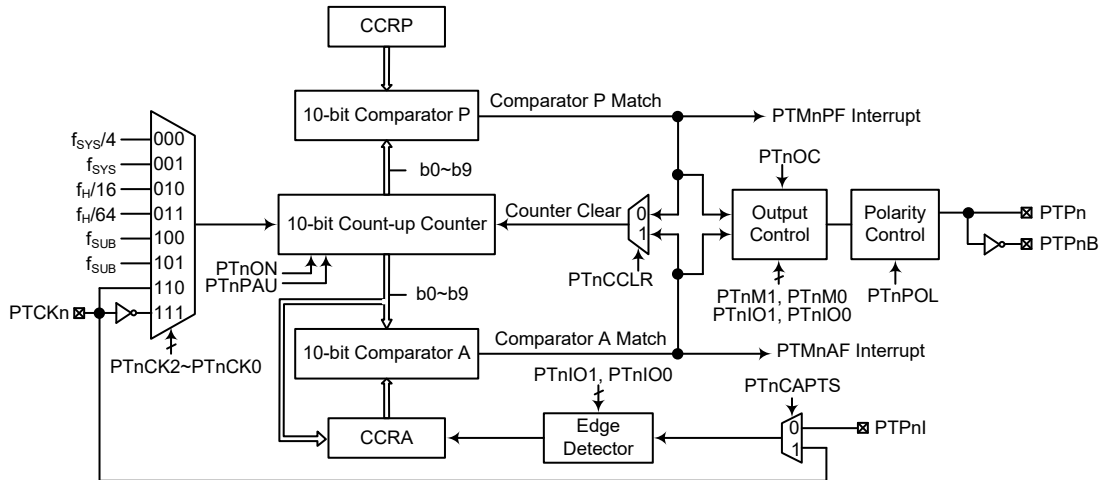


**Capture Input Mode**

- Note: 1. STM[1:0]=01 and active edge set by the STIO[1:0] bits  
 2. An STM Capture input pin active edge transfers the counter value to CCRA  
 3. STCCLR bit not used  
 4. No output function – STOC and STPOL bits are not used  
 5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero  
 6. The capture input mode cannot be used if the selected STM counter clock is not available

## Periodic Type TM – PTM

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



Note: 1. The PTMn external pins are pin-shared with other functions, therefore before using the PTMn function the pin-shared function registers must have been set properly to enable the PTMn pin function. The PTCKn and PTPnI pins, if used, must also be set as an input by setting the corresponding bits in the port control register.

2. The PTPnB is the inverted signal of the PTPn.

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

### 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 PTnON 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 PTMn 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 two output pins. All operating setup conditions are selected using relevant internal registers.

### Periodic Type TM Register Description

Overall operation of the Periodic 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
PTMnC0	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
PTMnC1	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCAPTS	PTnCCLR
PTMnDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnDH	—	—	—	—	—	—	D9	D8
PTMnAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnAH	—	—	—	—	—	—	D9	D8
PTMnRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMnRPH	—	—	—	—	—	—	D9	D8

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

• **PTMnC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTnPAU	PTnCK2	PTnCK1	PTnCK0	PTnON	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	—	—	—
POR	0	0	0	0	0	—	—	—

Bit 7 **PTnPAU**: PTMn 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 PTMn 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 **PTnCK2~PTnCK0**: PTMn counter clock selection  
 000:  $f_{SYS}/4$   
 001:  $f_{SYS}$   
 010:  $f_H/16$   
 011:  $f_H/64$   
 100:  $f_{SUB}$   
 101:  $f_{SUB}$   
 110: PTCKn rising edge clock  
 111: PTCKn falling edge clock

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

Bit 3 **PTnON**: PTMn counter on/off control  
 0: Off  
 1: On

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

Bit 2~0 Unimplemented, read as “0”

• **PTMnC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTnM1	PTnM0	PTnIO1	PTnIO0	PTnOC	PTnPOL	PTnCPTS	PTnCCLR
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 **PTnM1~PTnM0**: PTMn operating mode selection  
 00: Compare Match Output Mode  
 01: Capture Input Mode  
 10: PWM Output Mode or Single Pulse Output Mode  
 11: Timer/Counter Mode

These bits setup the required operating mode for the PTMn. To ensure reliable operation the PTMn should be switched off before any changes are made to the PTnM1 and PTnM0 bits. In the Timer/Counter Mode, the PTMn output pin state is undefined.

Bit 5~4 **PTnIO1~PTnIO0**: PTMn external pin function selection

Compare Match Output Mode

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

PWM Output Mode/Single Pulse Output Mode

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

Capture Input Mode

- 00: Input capture at rising edge of PTPnI or PTCKn
- 01: Input capture at falling edge of PTPnI or PTCKn
- 10: Input capture at rising/falling edge of PTPnI or PTCKn
- 11: Input capture disabled

Timer/Counter Mode

Unused

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

In the Compare Match Output Mode, the PTnIO1 and PTnIO0 bits determine how the PTMn output pin changes state when a compare match occurs from the Comparator A. The PTMn 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 PTMn output pin should be setup using the PTnOC bit in the PTMnC1 register. Note that the output level requested by the PTnIO1 and PTnIO0 bits must be different from the initial value setup using the PTnOC bit otherwise no change will occur on the PTMn output pin when a compare match occurs. After the PTMn output pin changes state, it can be reset to its initial level by changing the level of the PTnON bit from low to high.

In the PWM Output Mode, the PTnIO1 and PTnIO0 bits determine how the TM output pin changes state when a certain compare match condition occurs. The PTMn output function is modified by changing these two bits. It is necessary to only change the values of the PTnIO1 and PTnIO0 bits only after the PTMn has been switched off. Unpredictable PWM outputs will occur if the PTnIO1 and PTnIO0 bits are changed when the PTMn is running.

- Bit 3      **PTnOC**: PTMn PTPn 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 PTMn output pin. Its operation depends upon whether PTMn 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 PTMn is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the PTMn 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 PTMn output pin when the PTnON bit changes from low to high.

- Bit 2      **PTnPOL**: PTMn PTPn output polarity control  
     0: Non-inverted  
     1: Inverted

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

- Bit 1      **PTnCAPTS**: PTMn capture trigger source selection  
     0: From PTPnI pin  
     1: From PTCKn pin

- Bit 0      **PTnCCLR**: PTMn 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 TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the PTnCCLR 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 PTnCCLR bit is not used in the PWM Output, Single Pulse Output or Capture Input Mode.

• **PTMnDL 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**: PTMn Counter Low Byte Register bit 7 ~ bit 0  
 PTMn 10-bit Counter bit 7 ~ bit 0

• **PTMnDH 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**: PTMn Counter High Byte Register bit 1 ~ bit 0  
 PTMn 10-bit Counter bit 9 ~ bit 8

• **PTMnAL 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**: PTMn CCRA Low Byte Register bit 7 ~ bit 0  
 PTMn 10-bit CCRA bit 7 ~ bit 0

• **PTMnAH 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**: PTMn CCRA High Byte Register bit 1 ~ bit 0  
 PTMn 10-bit CCRA bit 9 ~ bit 8

• **PTMnRPL 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**: PTMn CCRP Low Byte Register bit 7 ~ bit 0  
 PTMn 10-bit CCRP bit 7 ~ bit 0

• **PTMnRPH 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**: PTMn CCRP High Byte Register bit 1 ~ bit 0  
 PTMn 10-bit CCRP bit 9 ~ bit 8

## Periodic Type TM Operation Modes

The Periodic Type TM can operate in one of five operating modes, Compare Match Output Mode, PWM Output Mode, Single Pulse Output Mode, Capture Input Mode or Timer/Counter Mode. The operating mode is selected using the PTnM1 and PTnM0 bits in the PTMnC1 register.

### Compare Match Output Mode

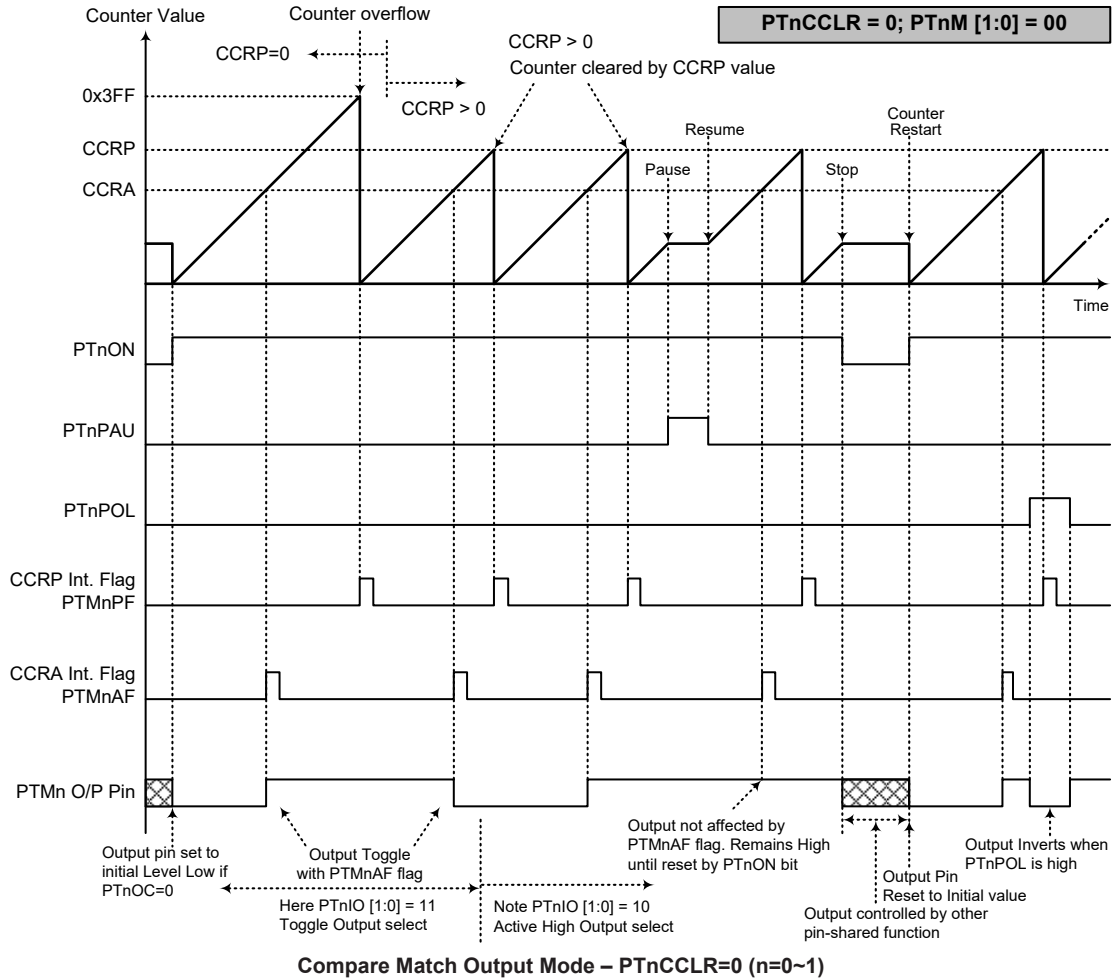
To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the PTnCCLR 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 PTMnAF and PTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the PTnCCLR bit in the PTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTnCCLR is high no PTMnPF interrupt request flag will be generated. In the Compare Match Output Mode, the CCRA cannot be cleared to “0”.

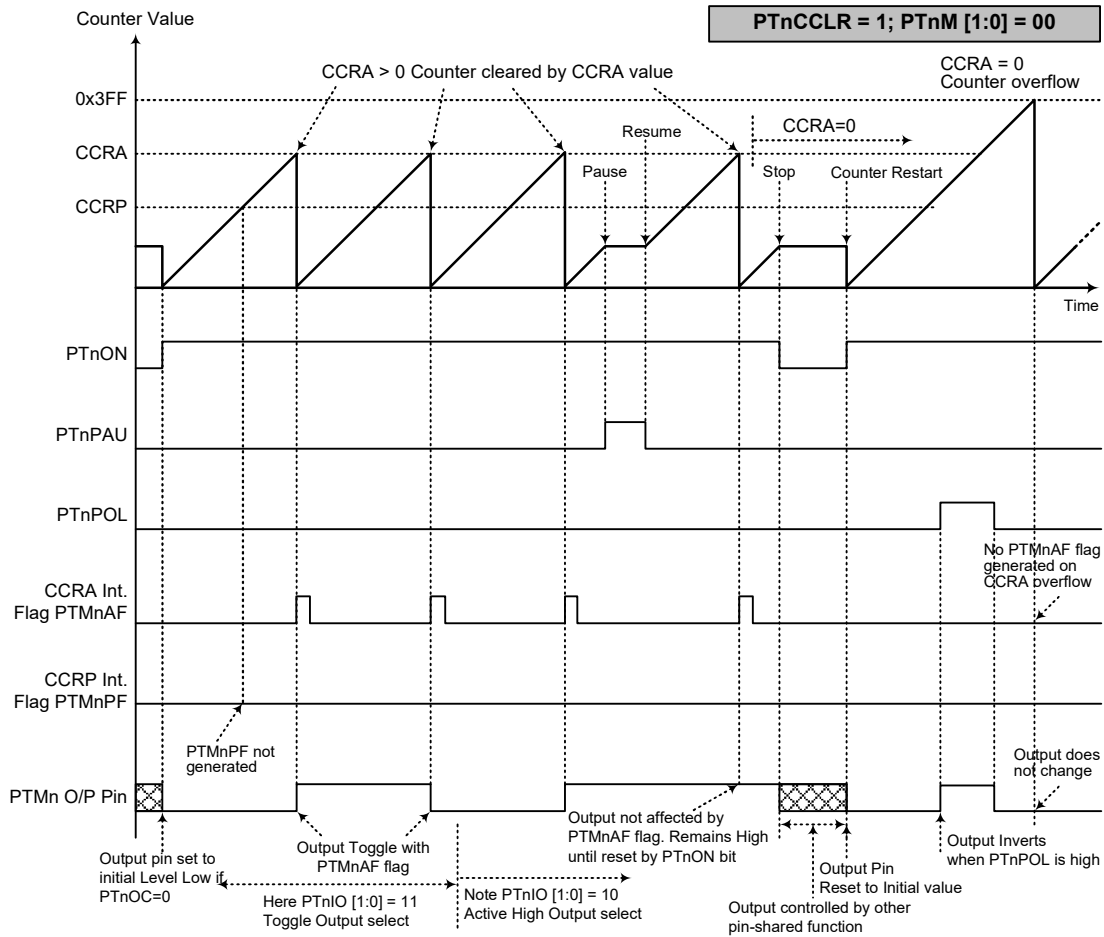
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the PTMnAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the PTMn output pin will change state. The PTMn output pin condition however only changes state when a PTMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The PTMnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the PTMn output pin. The way in which the PTMn output pin changes state are determined by the condition of the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The PTMn output pin can be selected using the PTnIO1 and PTnIO0 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 PTMn output pin, which is setup after the PTnON bit changes from low to high, is setup using the PTnOC bit. Note that if the PTnIO1 and PTnIO0 bits are zero then no pin change will take place.





- Note: 1. With PTnCCR=0, a Comparator P match will clear the counter  
 2. The PTMn output pin is controlled only by the PTMnAF flag  
 3. The output pin is reset to its initial state by a PTnON bit rising edge



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

- Note: 1. With PTnCCR=1, a Comparator A match will clear the counter  
 2. The PTMn output pin is controlled only by the PTMnAF flag  
 3. The output pin is reset to its initial state by a PTnON bit rising edge  
 4. A PTMnPF flag is not generated when PTnCCR=1

**Timer/Counter Mode**

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 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 PTMn 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 PTMn 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 PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 10 respectively. The PWM function within the PTMn 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 PTMn 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 PTnCCLR bit has no effect as the PWM period. Both of the CCRP and CCRA registers are used to generate the PWM waveform, the CCRP register is used to clear the internal counter and thus control the PWM waveform frequency, while the CCRA register 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 PTnOC bit in the PTMnC1 register is used to select the required polarity of the PWM waveform while the two PTnIO1 and PTnIO0 bits are used to enable the PWM output or to force the PTMn output pin to a fixed high or low level. The PTnPOL bit is used to reverse the polarity of the PWM output waveform.

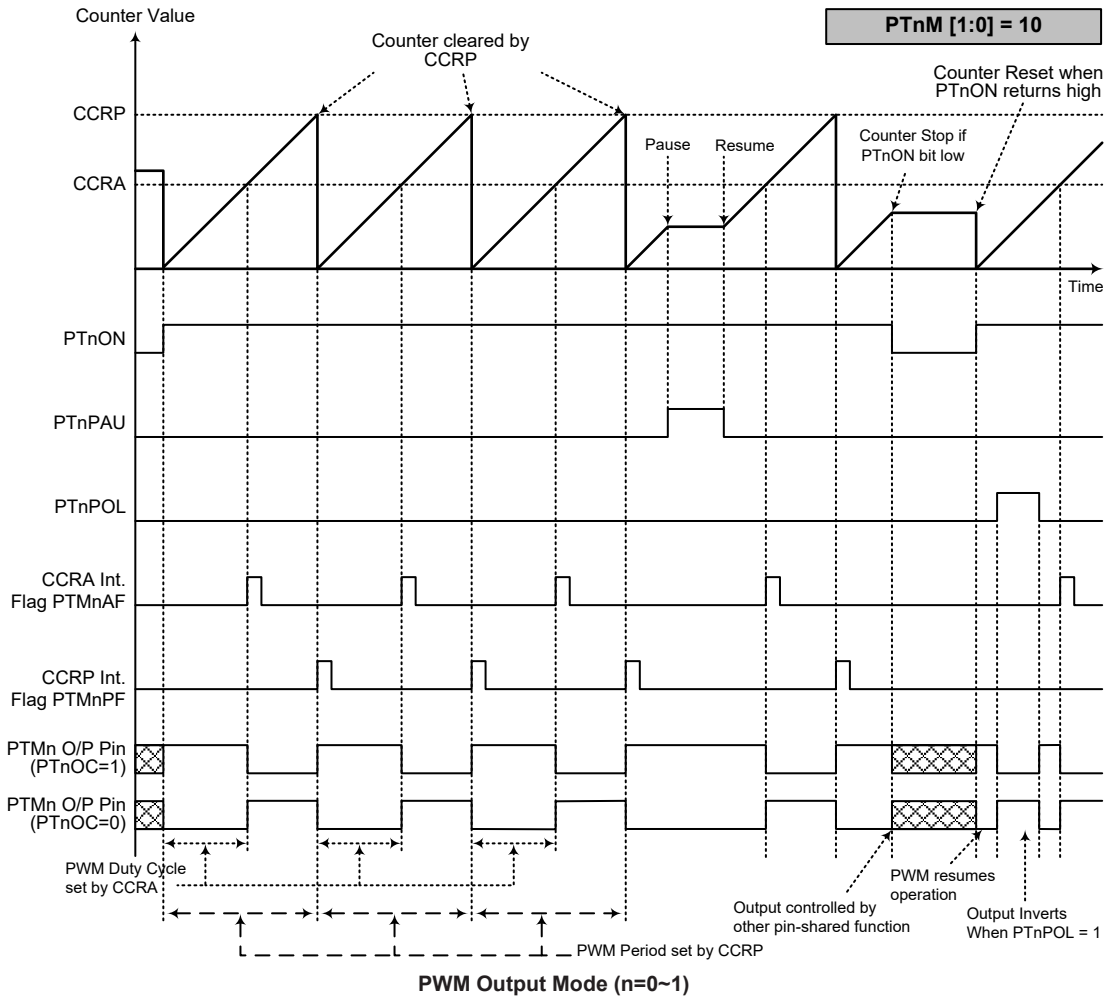
• **10-bit PTMn, PWM Output Mode, Edge-aligned Mode**

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

If  $f_{SYS}=8\text{MHz}$ , PTMn clock source select  $f_{SYS}/4$ , CCRP=512 and CCRA=128,

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



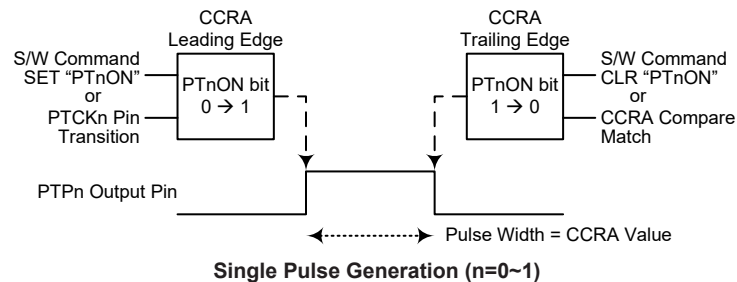
- Note: 1. The counter is cleared by CCRP  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues running even when PTnIO[1:0]=00 or 01  
 4. The PTnCCLR bit has no influence on PWM operation

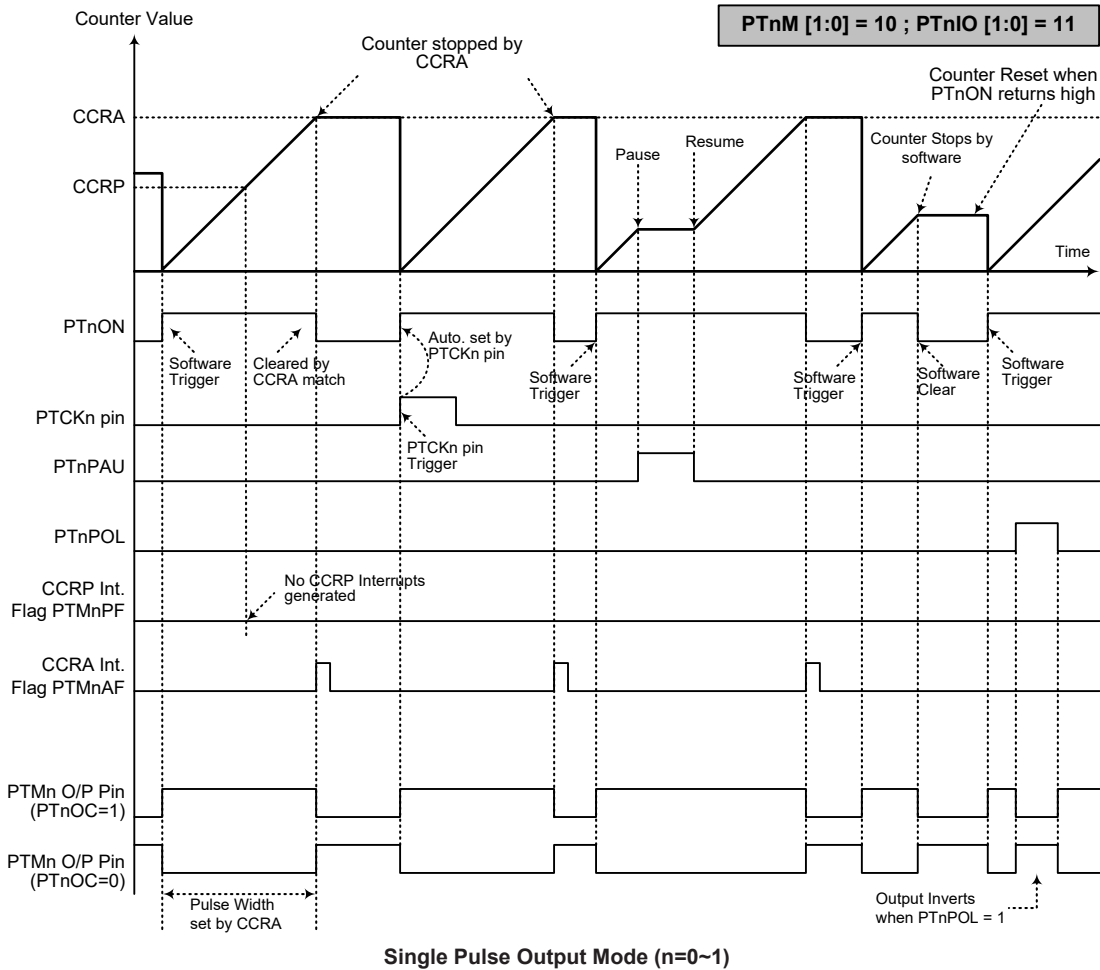
### Single Pulse Output Mode

To select this mode, bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 10 respectively and also the PTnIO1 and PTnIO0 bits should be set to 11 respectively. The Single Pulse Output Mode, as the name suggests, will generate a single shot pulse on the PTMn output pin.

The trigger for the pulse output leading edge is a low to high transition of the PTnON bit, which can be implemented using the application program. However in the Single Pulse Output Mode, the PTnON bit can also be made to automatically change from low to high using the external PTCKn pin, which will in turn initiate the Single Pulse output. When the PTnON bit transitions to a high level, the counter will start running and the pulse leading edge will be generated. The PTnON bit should remain high when the pulse is in its active state. The generated pulse trailing edge will be generated when the PTnON 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 PTnON 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 PTMn interrupt. The counter can only be reset back to zero when the PTnON bit changes from low to high when the counter restarts. In the Single Pulse Output Mode CCRP is not used. The PTnCCLR is not used in this mode.





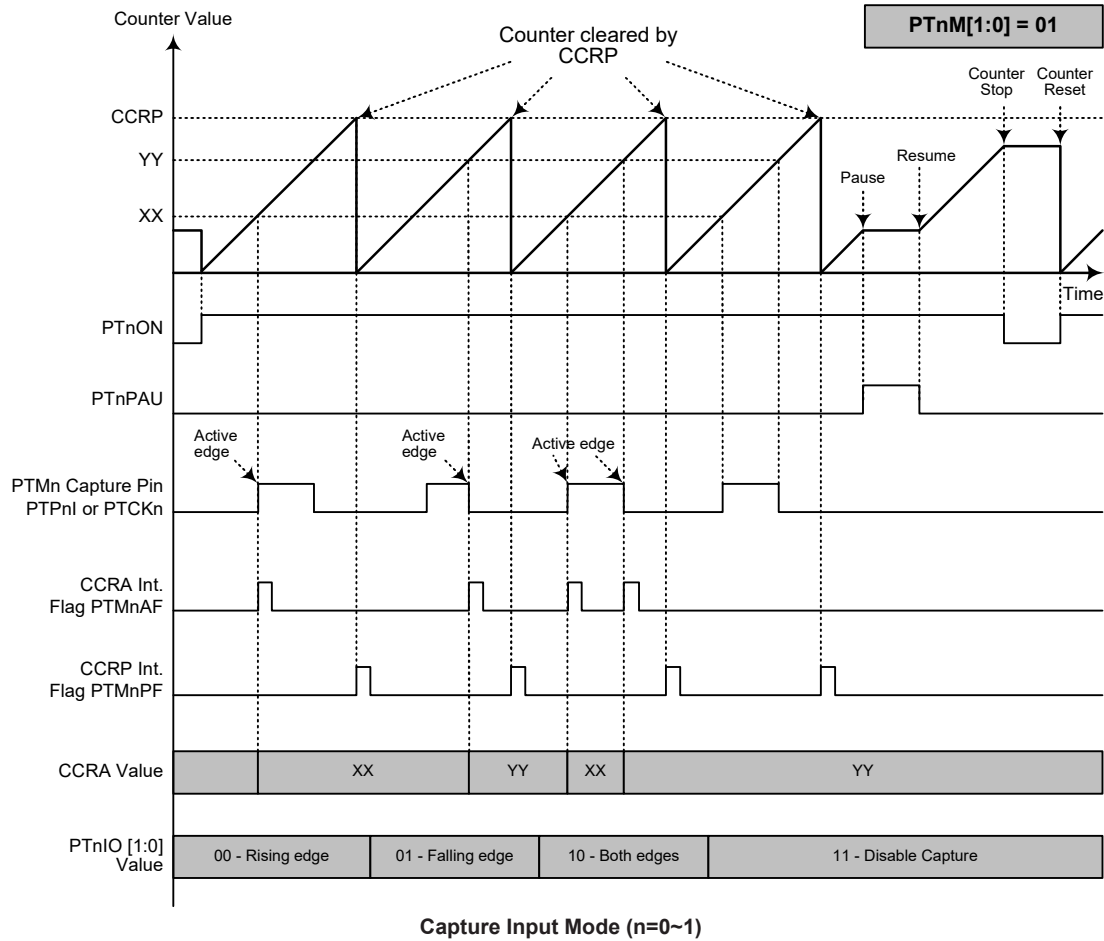
- Note:
1. Counter stopped by CCRA
  2. CCRP is not used
  3. The pulse triggered by the PTCKn pin or by setting the PTnON bit high
  4. A PTCKn pin active edge will automatically set the PTnON bit high
  5. In the Single Pulse Output Mode, PTnIO[1:0] must be set to "11" and cannot be changed

### **Capture Input Mode**

To select this mode bits PTnM1 and PTnM0 in the PTMnC1 register should be set to 01 respectively. This mode enables external signals to capture and store the present value of the internal counter and can therefore be used for applications such as pulse width measurements. The external signal is supplied on the PTPnI or PTCKn pin, selected by the PTnCAPTS bit in the PTMnC1 register. The input pin active edge can be either a rising edge, a falling edge or both rising and falling edges; the active edge transition type is selected using the PTnIO1 and PTnIO0 bits in the PTMnC1 register. The counter is started when the PTnON bit changes from low to high which is initiated using the application program.

When the required edge transition appears on the PTPnI or PTCKn pin the present value in the counter will be latched into the CCRA registers and a PTMn interrupt generated. Irrespective of what events occur on the PTPnI or PTCKn pin the counter will continue to free run until the PTnON bit changes from high to low. When a CCRP compare match occurs the counter will reset back to zero; in this way the CCRP value can be used to control the maximum counter value. When a CCRP compare match occurs from Comparator P, a PTMn interrupt will also be generated. Counting the number of overflow interrupt signals from the CCRP can be a useful method in measuring long pulse widths. The PTnIO1 and PTnIO0 bits can select the active trigger edge on the PTPnI or PTCKn pin to be a rising edge, falling edge or both edge types. If the PTnIO1 and PTnIO0 bits are both set high, then no capture operation will take place irrespective of what happens on the PTPnI or PTCKn pin, however it must be noted that the counter will continue to run. The PTnCCLR, PTnOC and PTnPOL bits are not used in this mode.

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

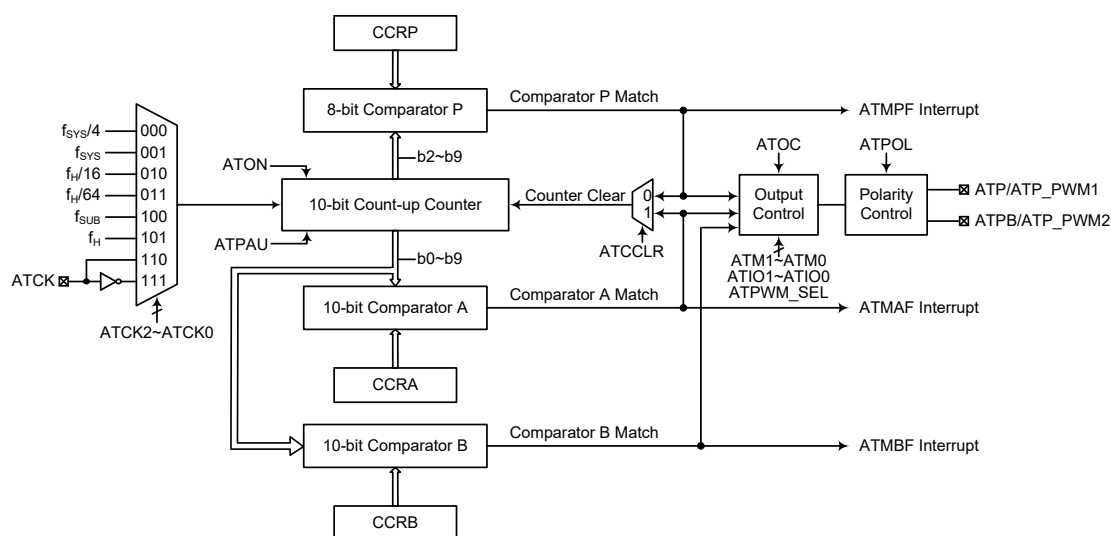


- Note: 1. PTnM[1:0]=01 and active edge set by the PTnIO[1:0] bits  
 2. A PTMn Capture input pin active edge transfers the counter value to CCRA  
 3. PTnCCLR bit not used  
 4. No output function – PTnOC and PTnPOL bits are not used  
 5. CCRP determines the counter value and the counter has a maximum count value when CCRP is equal to zero  
 6. The capture input mode cannot be used if the selected PTMn counter clock is not available



## Audio Type TM – ATM

The Audio Type TM contains three operating modes which are Compare Match Output, Timer/Event Counter and PWM Output modes. Here the PWM Output mode contains two sub-modes, namely Normal PWM Output mode and Audio PWM Output mode. The Audio type TM can also be controlled with an external input pin and can drive one or more external output pins. These output signals can be the same signal or the inverse signals. When the Audio PWM Output mode is enabled, the Audio PWM outputs will be generated on the ATP\_PWM1 and ATP\_PWM2 pins and can be used to drive speakers through an external PWM driver.



- Note: 1. The ATM external pins are pin-shared with other functions, therefore before using the ATM function, the pin-shared function registers have been set properly to enable the ATM pin function. The ATCK pin, if used, must also be set as an input by setting the corresponding bits in the port control register.
2. Only in the Audio PWM Output mode, the ATP\_PWM1 and ATP\_PWM2 pin functions are used. In other ATM modes including Compare match output mode and Normal PWM Output mode, the pins are used as ATP and ATPB functions. The ATPB is inverted signal of the ATP output.

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

### Audio Type TM Operation

The size of Audio 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 three internal comparators with the names, Comparator A, Comparator B and Comparator P. These comparators will compare the value in the counter with the CCRA, CCRB and CCRP registers. The CCRP comparator is 8-bit wide whose value is compared with the highest 8 bits in the counter while the CCRA and CCRB comparators are 10-bit wide and therefore 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 ATON 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, an ATM interrupt signal will also usually be generated. The Audio Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin and can also control one or more output pins. All operating setup conditions are selected using relevant internal registers.

## Audio Type TM Register Description

Overall operation of the Audio 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 CCRB values. The ATMRP register is used to store the 8-bit 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
ATMC0	ATPAU	ATCK2	ATCK1	ATCK0	ATON	—	—	ATPWM_SEL
ATMC1	ATM1	ATM0	ATIO1	ATIO0	ATOC	ATPOL	ATDPX	ATCCLR
ATMDL	D7	D6	D5	D4	D3	D2	D1	D0
ATMDH	—	—	—	—	—	—	D9	D8
ATMAL	D7	D6	D5	D4	D3	D2	D1	D0
ATMAH	—	—	—	—	—	—	D9	D8
ATMBL	D7	D6	D5	D4	D3	D2	D1	D0
ATMBH	—	—	—	—	—	—	D9	D8
ATMRP	D7	D6	D5	D4	D3	D2	D1	D0

**10-bit Audio Type TM Register List**

### • ATMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	ATPAU	ATCK2	ATCK1	ATCK0	ATON	—	—	ATPWM_SEL
R/W	R/W	R/W	R/W	R/W	R/W	—	—	R/W
POR	0	0	0	0	0	—	—	0

Bit 7 **ATPAU**: ATM 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 ATM 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 **ATCK2~ATCK0**: ATM counter clock selection  
000:  $f_{SYS}/4$   
001:  $f_{SYS}$   
010:  $f_H/16$   
011:  $f_H/64$   
100:  $f_{SUB}$   
101:  $f_H$   
110: ATCK rising edge clock  
111: ATCK falling edge clock

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

Bit 3 **ATON**: ATM counter on/off control  
0: Off  
1: On

This bit controls the overall on/off function of the ATM. Setting the bit high enables the counter to run while clearing the bit disables the ATM. Clearing this bit to zero

will stop the counter from counting and turn off the ATM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value. If the ATM is in the Compare Match Output Mode or the PWM Output Mode then the ATM output pin will be reset to its initial condition as specified by the ATOC bit, when the ATON bit changes from low to high.

- Bit 2~1 Unimplemented, read as “0”
- Bit 0 **ATPWM\_SEL**: ATM PWM output mode selection
  - 0: Normal PWM Output Mode
  - 1: Audio PWM Output Mode

This bit controls the PWM Output Mode selection. When the PWM output mode has been selected using the relevant bits in the ATMC1 register, then setting this bit high will switch the ATM operation from Normal PWM Output mode to the Audio PWM Output mode.

• **ATMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	ATM1	ATM0	ATIO1	ATIO0	ATOC	ATPOL	ATDPX	ATCCLR
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 **ATM1~ATM0**: ATM operating mode selection
  - 00: Compare Match Output Mode
  - 01: Undefined
  - 10: PWM Output Mode
  - 11: Timer/Counter Mode

These bits setup the required operating mode for the ATM. To ensure reliable operation the ATM should be switched off before any changes are made to the ATM1 and ATM0 bits. In the Timer/Counter Mode, the ATM output pin state is undefined.

- Bit 5~4 **ATIO1~ATIO0**: ATM external pin function selection
  - 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 ATM external pin changes state when a certain condition is reached. The function that these bits select depends upon the mode in which the ATM is running.

In the Compare Match Output Mode, the ATIO1 and ATIO0 bits determine how the ATM output pin changes state when a compare match occurs from Comparator A. The ATM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the ATM output pin should be setup using the ATOC bit in the ATMC1 register. Note that the output level requested by the ATIO1 and ATIO0 bits must be different from the initial value setup using the ATOC bit otherwise no change will occur on the ATM output pin when a compare match occurs. After the ATM output pin changes state, it can be reset to its initial level by changing the level of the ATON bit from low to high.

In the PWM Output Mode, the ATIO1 and ATIO0 bits determine how the ATM 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 ATIO1 and ATIO0 bits only after the ATM has been switched off. Unpredictable PWM outputs will occur if the ATIO1 and ATIO0 bits are changed when the ATM is running.

- Bit 3     **ATOC**: ATM output pin control  
 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 ATM output pins. Its operation depends upon whether the ATM is being used in the Compare Match Output Mode or in the PWM Output Mode. It has no effect if the ATM is in the Timer/Counter Mode. In the Compare Match Output Mode it determines the logic level of the ATP 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     **ATPOL**: ATM output polarity control  
     0: Non-invert  
     1: Invert

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

- Bit 1     **ATDPX**: ATM PWM period/duty control  
     0: CCRP – period; CCRA – duty  
     1: CCRP – duty; CCRA – period

This bit determines which register of the CCRA and CCRP is used to control either frequency or duty cycle of the PWM waveform in the Normal PWM Output Mode. However if the Audio PWM Output Mode is selected by setting the ATPWM\_SEL bit high, the ATDPX bit will be cleared to zero by hardware which means that the period values of output waveforms on the ATP\_PWM1 and ATP\_PWM2 will be controlled by the CCRP while their duty cycle values are controlled by the CCRA and CCRB.

- Bit 0     **ATCCLR**: ATM 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 Audio TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the ATCCLR 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 ATCCLR bit is not used in the PWM Output Mode.

• **ATMDL 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**: ATM Counter Low Byte Register bit 7 ~ bit 0  
 ATM 10-bit Counter bit 7 ~ bit 0

• **ATMDH 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**: ATM Counter High Byte Register bit 1 ~ bit 0  
 ATM 10-bit Counter bit 9 ~ bit 8

• **ATMAL 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**: ATM CCRA Low Byte Register bit 7 ~ bit 0  
 ATM 10-bit CCRA bit 7 ~ bit 0

• **ATMAH 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**: ATM CCRA High Byte Register bit 1 ~ bit 0  
 ATM 10-bit CCRA bit 9 ~ bit 8

• **ATMBL 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**: ATM CCRB Low Byte Register bit 7 ~ bit 0  
 ATM 10-bit CCRB bit 7 ~ bit 0

• **ATMBH 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**: ATM CCRB High Byte Register bit 1 ~ bit 0  
 ATM 10-bit CCRB bit 9 ~ bit 8

• **ATMRP 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:** ATM CCRP 8-bit register, compared with the ATM Counter bit 9~bit 2

Comparator P Match Period

0: 1024 ATM clocks

1~255: (1~255)×4 ATM clocks

These eight bits are used to setup the value on the internal CCRP 8-bit register, which are then compared with the internal counter's highest eight bits. The result of this comparison can be selected to clear the internal counter if the ATCCLR bit is cleared to zero. Setting the ATCCLR 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 eight counter bits, the compare values exist in 4 clock cycle multiples. Clearing all eight bits to zero is in effect allowing the counter to overflow at its maximum value.

### Audio Type TM Operation Modes

The Audio Type TM can operate in one of three operating modes, Compare Match Output Mode, Timer/Counter Mode and PWM Output Mode. The operating mode is selected using the ATM1 and ATM0 bits in the ATMC1 register.

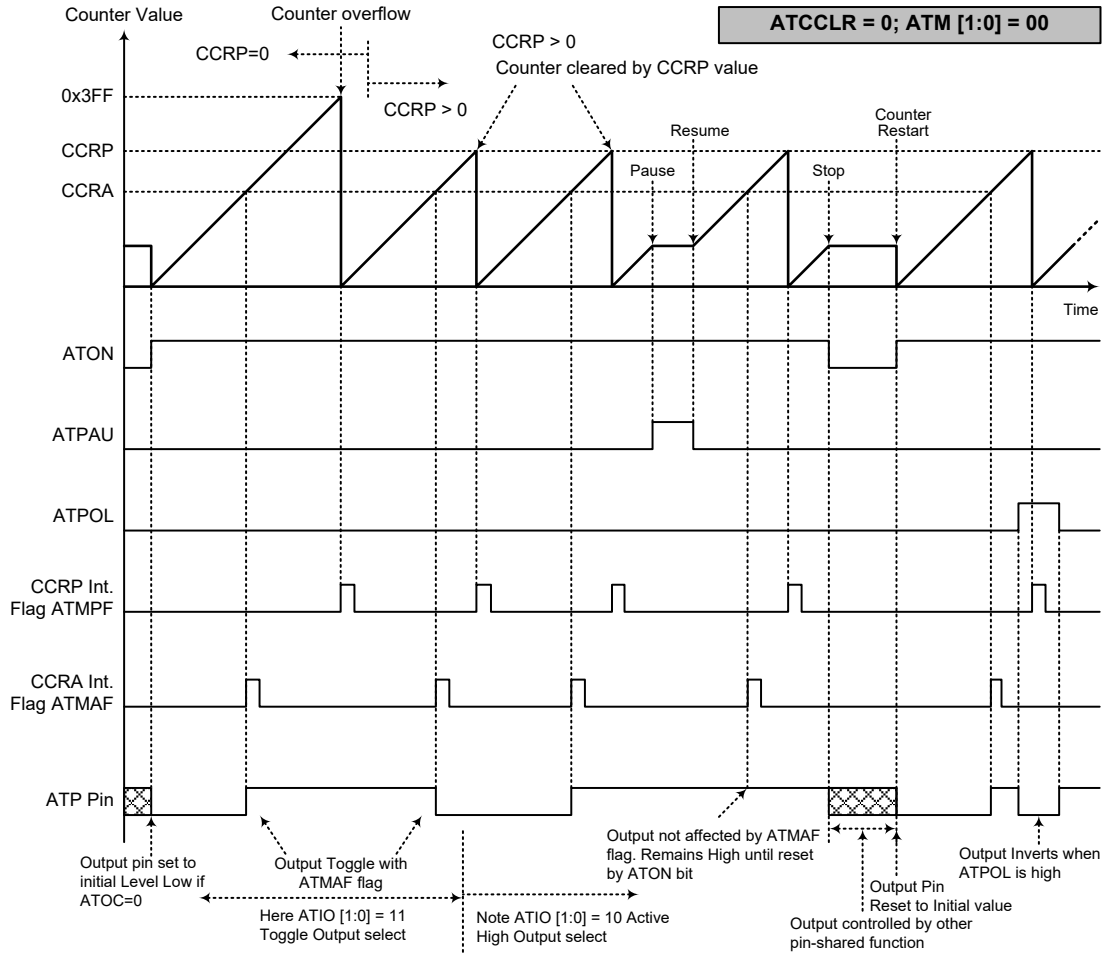
#### Compare Match Output Mode

To select this mode, bits ATM1 and ATM0 in the ATMC1 register, should be set to 00. 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 ATCCLR 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 ATMAF and ATMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

If the ATCCLR bit in the ATMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the ATMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when ATCCLR is high no ATMPF interrupt request flag will be generated.

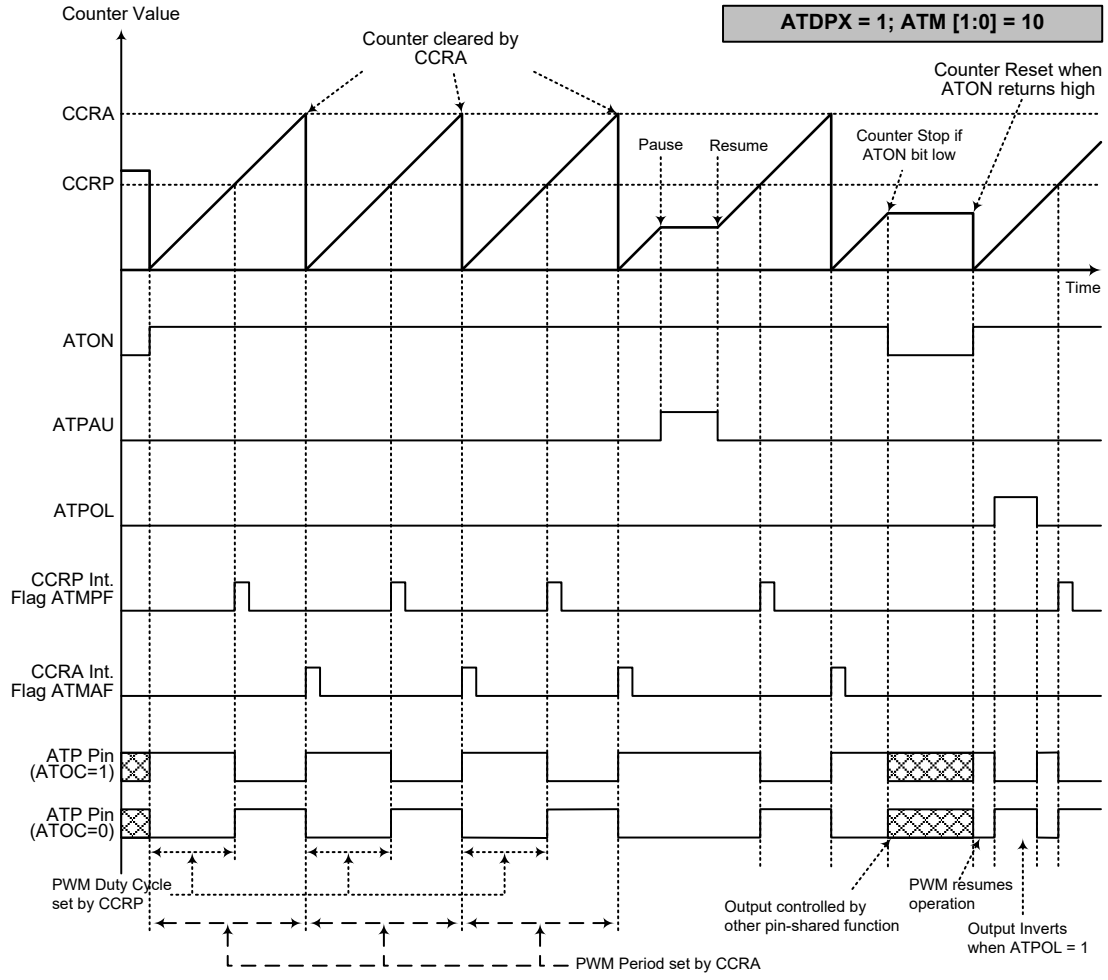
If the CCRA bits are all zero, the counter will overflow when it reaches its maximum 10-bit, 3FF Hex, value, however here the ATMAF interrupt request flag will not be generated.

As the name of the mode suggests, after a comparison is made, the ATM output pin will change state. The ATM output pin condition however only changes state when an ATMAF interrupt request flag is generated after a compare match occurs from Comparator A. The ATMPF interrupt request flag, generated from a compare match with Comparator P, will have no effect on the ATM output pin. The way in which the ATM output pin changes state are determined by the condition of the ATIO1 and ATIO0 bits in the ATMC1 register. The ATM output pin can be selected using the ATIO1 and ATIO0 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 ATM output pin, which is setup after the ATON bit changes from low to high, is setup using the ATOC bit. Note that if the ATIO1 and ATIO0 bits are zero then no pin change will take place.



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

- Note: 1. With ATCCLR=0, a Comparator P match will clear the counter  
 2. The ATM output pin is controlled only by the ATMAF flag  
 3. The output pin is reset to its initial state by an ATON bit rising edge



- Note:
1. With ATCCLR=1, a Comparator A match will clear the counter
  2. The ATM output pin is controlled only by the ATMAF flag
  3. The output pin is reset to its initial state by an ATON bit rising edge
  4. The ATMPF flag is not generated when ATCCLR=1



### Timer/Counter Mode

To select this mode, bits ATM1 and ATM0 in the ATMC1 register should be set to 11. 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 ATM 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 ATM 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 ATM1 and ATM0 in the ATMC1 register should first be set to 10. The PWM function within the ATM is useful for applications which require functions such as a Audio PWM output, motor control, heating control, illumination control, etc. By providing a signal of fixed frequency but of varying duty cycle on the ATM 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 ATCCLR bit has no effect on the PWM operation.

For this device, the PWM output can operate in two modes which are Normal PWM Output Mode and Audio PWM Output Mode and are selected using the ATPWM\_SEL bit in the ATMC0 register. The control bits and differences between the Normal PWM Output Mode and Audio PWM Output Mode are summarised in the accompanying table.

Item	Normal PWM Output Mode	Audio PWM Output Mode
ATPWM_SEL Bit	0	1
ATM CCRB Interrupt	×	√
Output Pins	ATP, ATPB	ATP_PWM1, ATP_PWM2
PWM Output Control	ATIO1~ATIO0 bits	
Output Active Level	ATOC bit	
Output Polarity Control	ATPOL bit	
Period/Duty Control	ATDPX bit	ATDPX bit is always 0. The Audio PWM period is controlled by CCRP.

**Normal/Audio PWM Output Mode Comparison Table**

### Normal PWM Output Mode

In the Normal PWM Output Mode, 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. Which register is used to control either frequency or duty cycle is determined using the ATDPX bit in the ATMC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

If the defined Duty value is equal to or greater than the Period value, then the PWM output duty is undefined.

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 ATOC bit in the ATMC1 register is used to select the required polarity of the PWM waveform while the two ATIO1 and ATIO0 bits are used to enable the PWM output or to force the ATM output pin to a fixed high or low level. The ATPOL bit is used to reverse the polarity of the PWM output waveform.

• **10-bit ATM, Normal PWM Output Mode, Edge-aligned Mode, ATDPX=0**

CCRP	1~255	0
Period	CCRP×4	1024
Duty	CCRA	

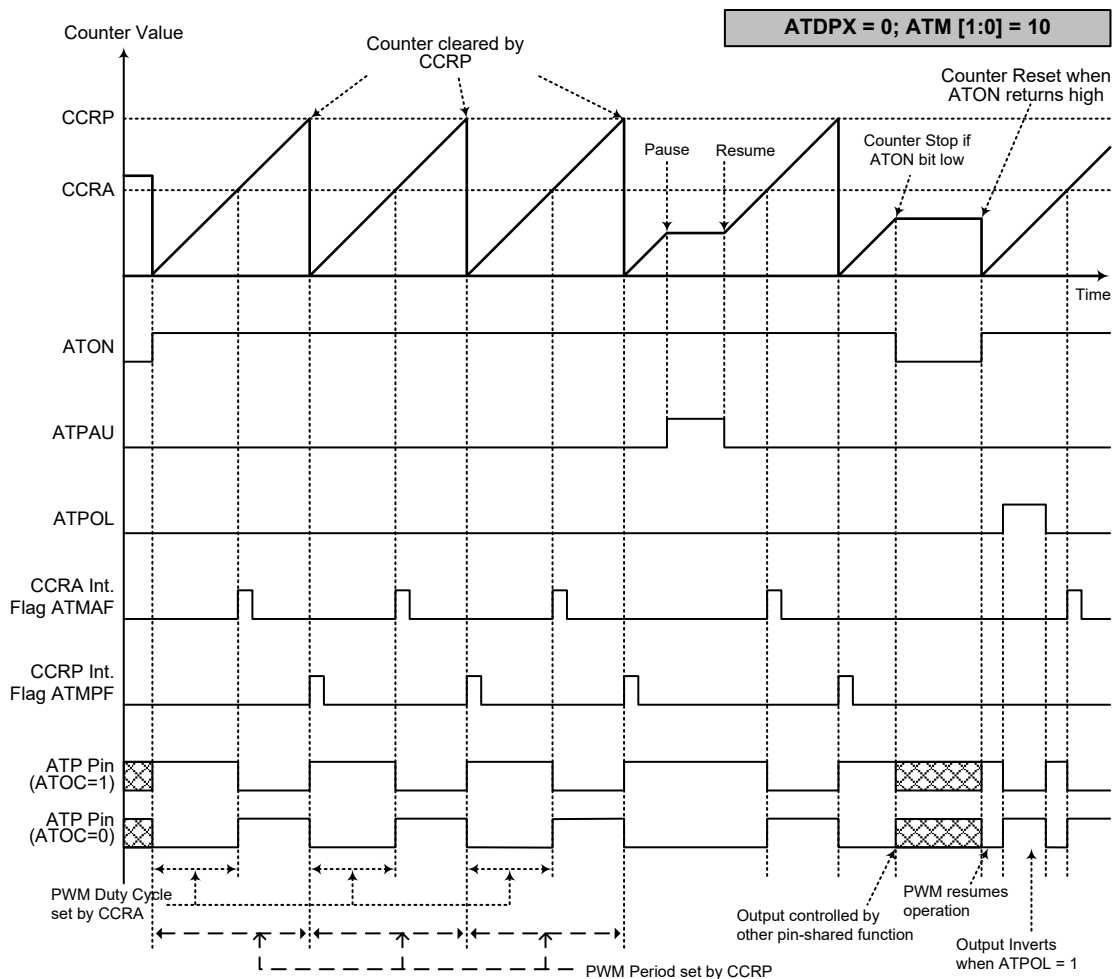
If  $f_{SYS}=8\text{MHz}$ , ATM clock source is  $f_{SYS}/4$ ,  $CCRP=64$  and  $CCRA=64$ ,

The ATM PWM output frequency= $(f_{SYS}/4)/(64\times 4)=f_{SYS}/1024=7.8125\text{kHz}$ ,  $duty=64/(64\times 4)=25\%$ ,

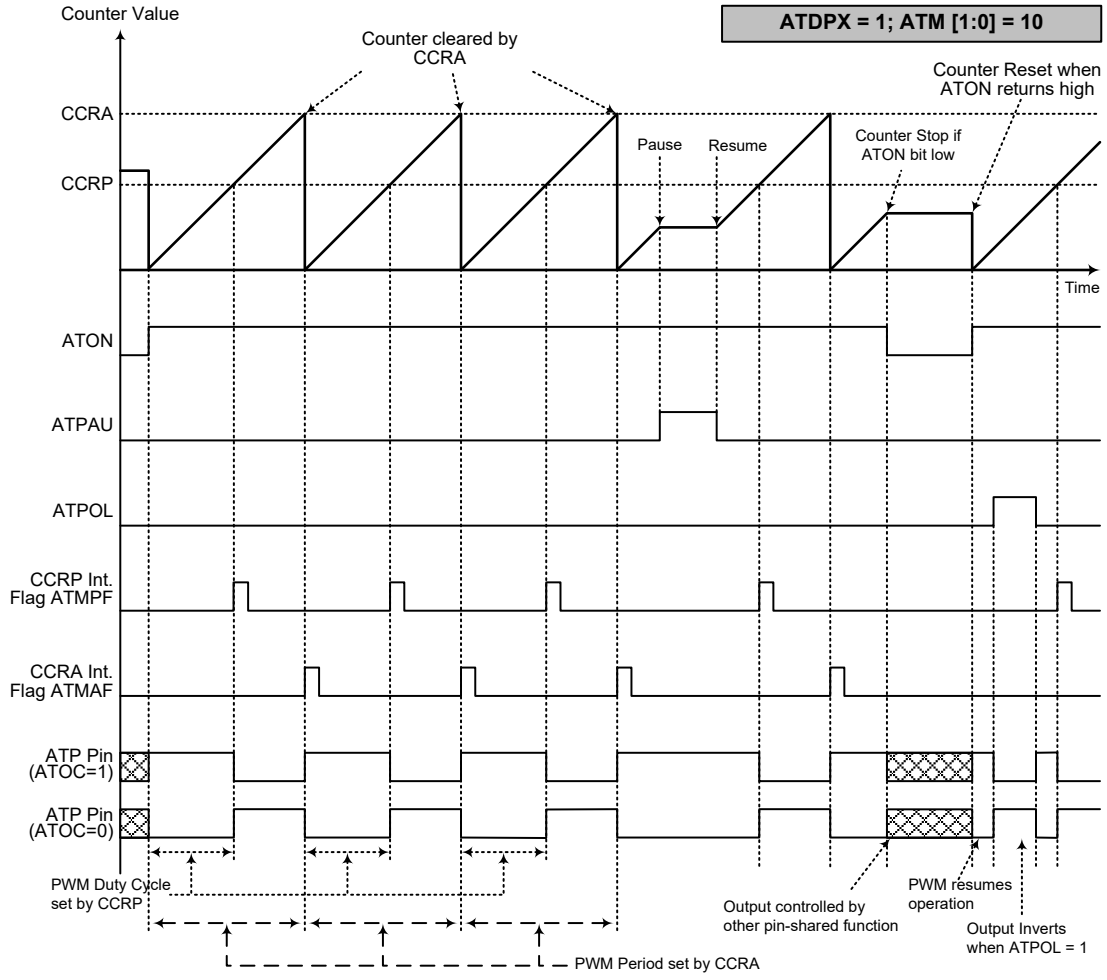
• **10-bit ATM, Normal PWM Output Mode, Edge-aligned Mode, ATDPX=1**

CCRP	1~255	0
Period	CCRA	
Duty	CCRP×4	1024

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



- Note: 1. Here  $ATM[1:0]=10$ ,  $ATPWM\_SEL=0$  and  $ATDPX=0$ , the counter is cleared by CCRP  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues running even when  $ATIO[1:0]=00$  or  $01$   
 4. The  $ATCCLR$  bit has no influence on PWM operation



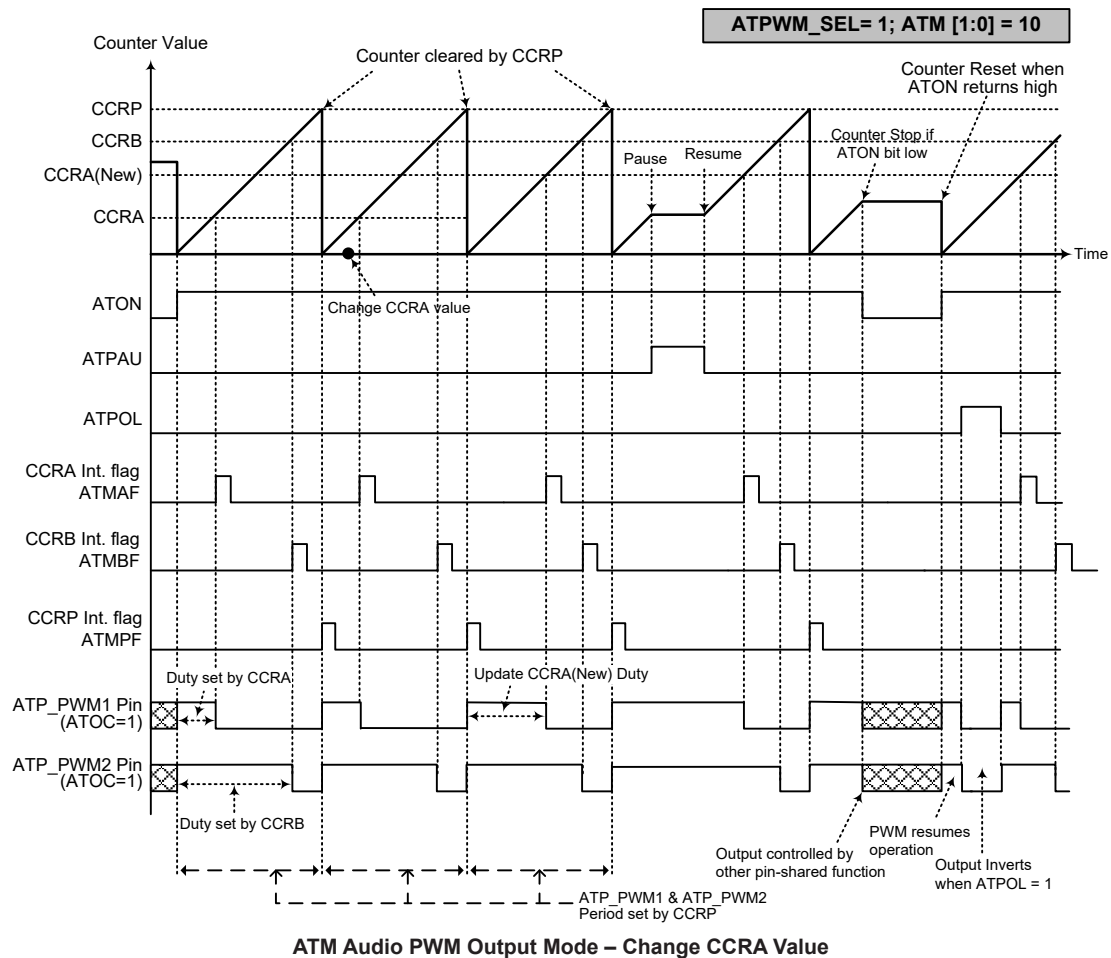
- Note: 1. Here ATM[1:0]=10, ATPWM\_SEL=0 and ATDPX=1, the counter is cleared by CCRA  
 2. A counter clear sets the PWM Period  
 3. The internal PWM function continues running even when ATIO[1:0]=00 or 01  
 4. The ATCCLR bit has no influence on PWM operation

### Audio PWM Output Mode

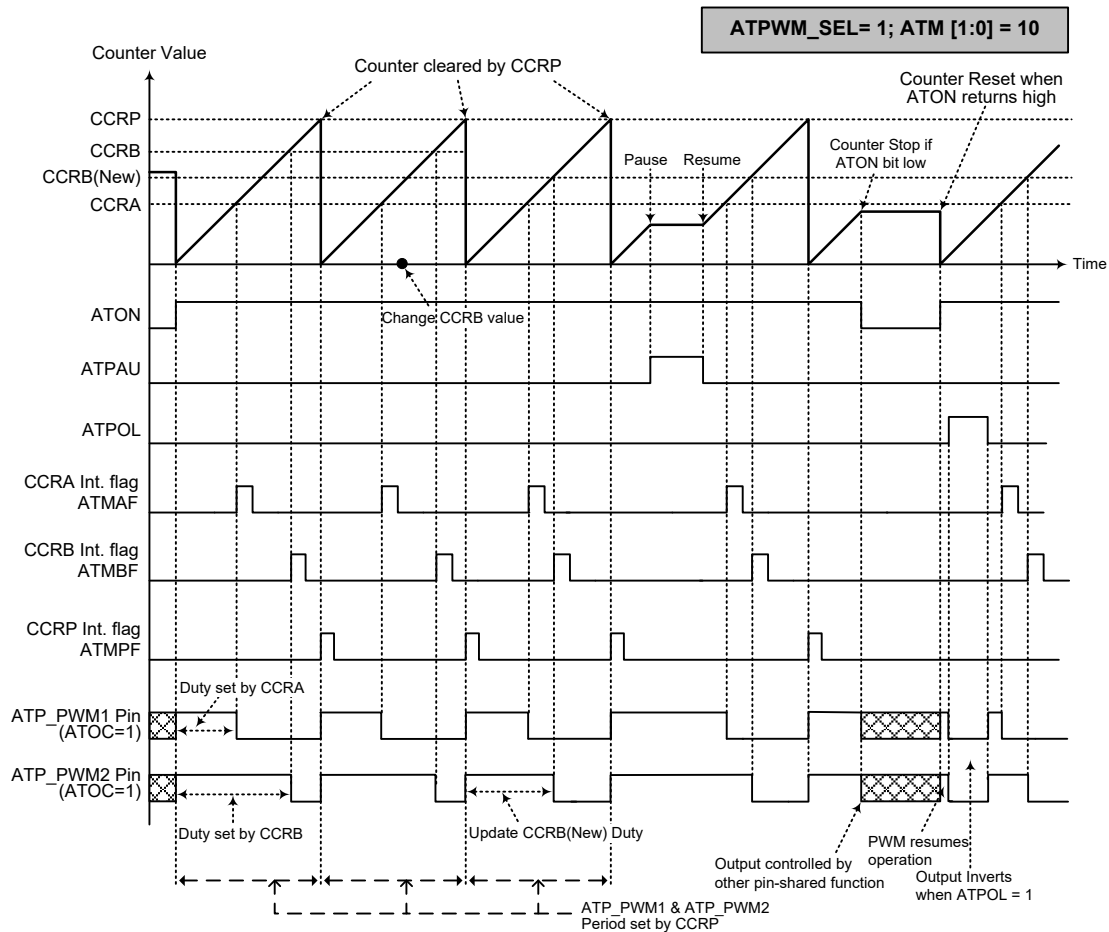
In the Audio PWM Output Mode, the CCRA, CCRB and CCRP registers are all used to generate the PWM waveforms on ATP\_PWM1 and ATP\_PWM2. The CCRP register is used to clear the internal counter and thus control the frequency of the two PWM outputs on the ATP\_PWM1 and ATP\_PWM2 pins, while the CCRA and CCRB registers are used to control the duty cycles of these two signals respectively.

An interrupt can be generated when a compare match occurs from either Comparator A, Comparator B or Comparator P. Except that the ATDPX bit is maintained at a zero value by hardware in the Audio PWM Output Mode, other control bits that control the ATP\_PWM1 and ATP\_PWM2 are the same as the Normal PWM Output mode.

Note that as in the Audio PWM Output Mode, the CCRP, CCRA and CCRB all have a shadow function, when writing a new data into the CCRA, CCRB or CCRP registers, the CCRA, CCRB or CCRP will not be updated immediately by hardware until the current CCRP counter overflows and an ATMPF interrupt flag is set.

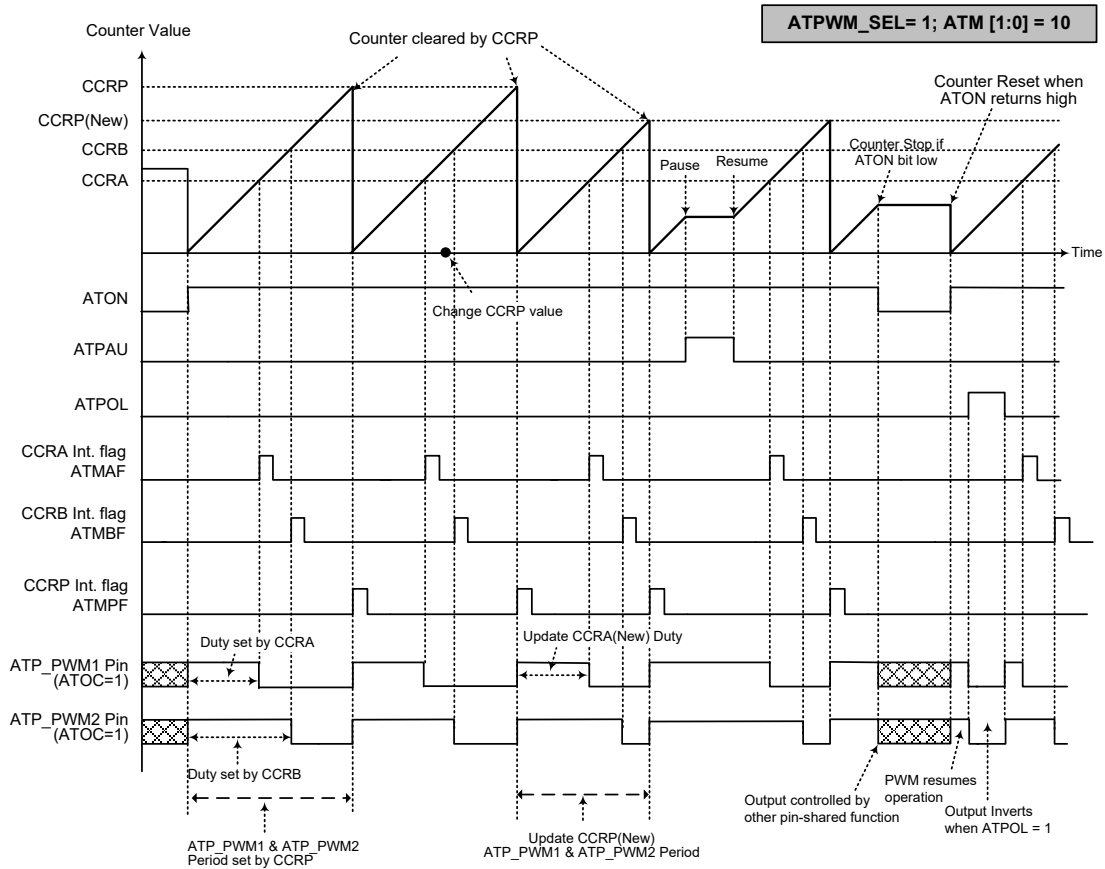


- Note: 1. ATM[1:0]=10, ATPWM\_SEL=1, the counter is cleared by CCRP  
 2. A counter clear sets the PWM Period  
 3. CCRA defines ATP\_PWM1 duty while CCRB defines ATP\_PWM2 duty  
 4. The internal PWM function continues running even when ATIO[1:0]=00 or 01  
 5. The ATCLLR bit has no influence on PWM operation, the ATDPX bit is kept at 0 by hardware



**ATM Audio PWM Output Mode – Change CCRB Value**

- Note: 1. ATM[1:0]=10, ATPWM\_SEL=1, the counter is cleared by CCRP  
 2. A counter clear sets the PWM Period  
 3. CCRA defines ATP\_PWM1 duty while CCRB defines ATP\_PWM2 duty  
 4. The internal PWM function continues running even when ATIO[1:0]=00 or 01  
 5. The ATCCLR bit has no influence on PWM operation, the ATDPX bit is kept at 0 by hardware

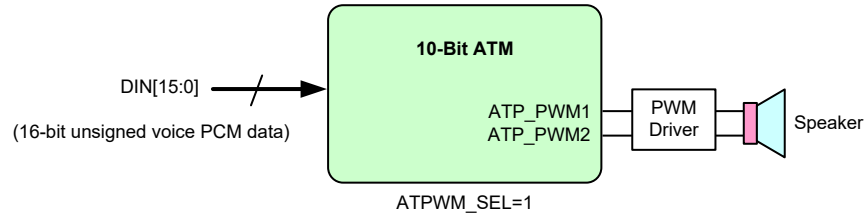


**ATM Audio PWM Output Mode – Change CCRP Value**

- Note: 1. ATM[1:0]=10, ATPWM\_SEL=1, the counter is cleared by CCRP  
 2. A counter clear sets the PWM Period  
 3. CCRA defines ATP\_PWM1 duty while CCRB defines ATP\_PWM2 duty  
 4. The internal PWM function continues running even when ATIO[1:0]=00 or 01  
 5. The ATCCLR bit has no influence on PWM operation, the ATPDX bit is kept at 0 by hardware

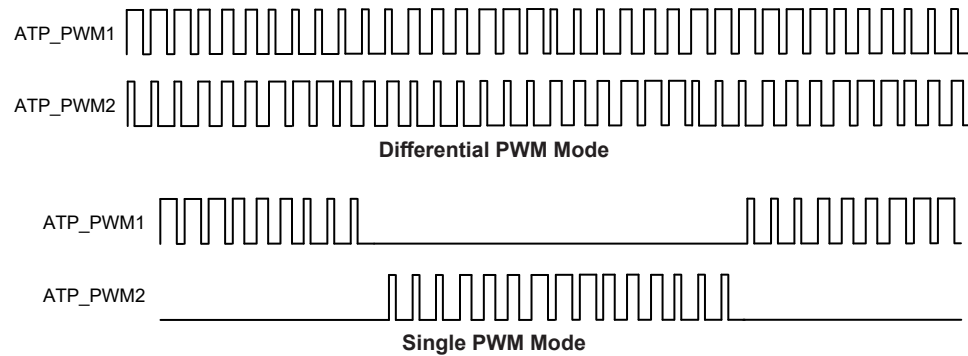
### Audio PWM Output Usage Description

When operating in the Audio PWM Output Mode, the ATM is used to generate audio PWM waveforms according to decompressed PCM voice data that can then drive speakers through an external PWM driver. The following example introduces how to process 16-bit unsigned PCM voice data and write correct data into the CCRP, CCRA and CCRB registers to generate the corresponding audio PWM outputs.



**Audio PWM Output Application Schematic Diagram**

The Audio PWM has two output modes, one is called the Differential Mode and the other is the Single Mode. Their output waveforms are illustrated in the following diagrams.



The following provides a description of how to use the ATM function in this device to generate the Differential and Single Audio PWM outputs. In this example, the PWM resolution is 8-bit and the system clock is derived from  $f_H$  which is 8MHz. To generate the following Audio PWM output signals, the CCRA, CCRB and CCRP register together with the relevant ATM control registers should be setup according to the steps below.

- Step 1 – Configure the ATMC0 Register  
 Select the ATM counter clock to be from either  $f_{SYS}$  or  $f_H$  by setting the ATCK2~ATCK0 bits in the ATMC0 register to “001” or “101”. Set the ATPWM\_SEL bit in the ATMC0 register to “1” to select the Audio PWM Output Mode. The ATMC0 register value will now be “00010--1b”.
- Step 2 – Configure the ATMC1 Register  
 Set the ATM1~ATM0 bits in the ATMC1 register to “10” and ATIO1~ATIO0 bits to “10” to ensure that the ATM is used with the PWM output. Then set the ATOC bit to “1” and ATPOL bit to “0” to select the PWM output to be active high and non-inverted. The ATMC1 register value will now be “10101000b”.
- Step 3 – Configure the ATMRP Register  
 Set the ATMRP register to “01000000b” to select the Comparator P match period as 256 ATM clocks. Therefore, the Audio PWM frequency= $(f_{SYS})/256=31.25kHz$ . Of course, if the ATCK2~ATCK0 bits value is “000”, the corresponding Audio PWM frequency is  $(f_{SYS}/4)/256=7.8125kHz$ .

- Step 4 – Pin-shared function selection  
Correctly set the PGS07~PGS06 bits in the PGS0 register or set the PGS11~PGS10 bits in the PGS1 register to select the pin as corresponding ATP\_PWM1 and ATP\_PWM2 function.
- Step 5 – Enable the ATM function  
After setting up the other ATM related functions, such as interrupts, etc., then set the ATON bit in the ATMC0 register to “1” to enable the ATM counter to run.
- Step 6 – Update CCRA and CCRB Values  
According to the input data, the CCRA and CCRB values should be updated. Refer to the following section for the calculation of the CCRA and CCRB values.  
As there are two types of Audio PWM output waveforms, Differential mode and Single mode, the calculation methods are different for these two types.

**Example 1 – Audio PWM Output Differential Mode, DIN[15:0]=4000H**

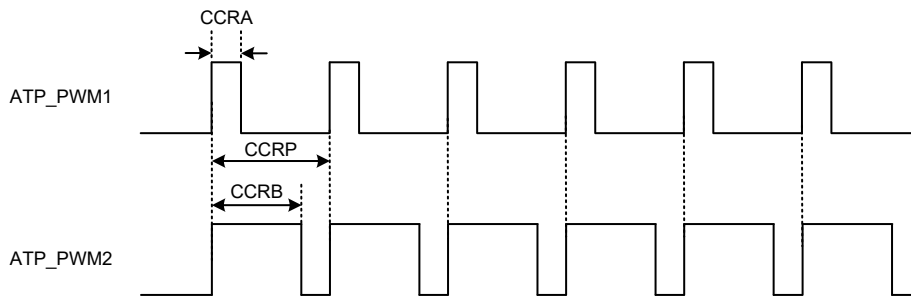
For example, DIN[15:0]=4000H (unsigned voice data), to generate the differential PWM outputs on ATP\_PWM1 and ATP\_PWM2 as shown in the timing diagram below.

DIN[15:0]=4000H, DINB[15:0]=~DIN[15:0]=BFFFH

The higher 8-bit of DIN: DIN[15:8]=40H=64, DINB[15:8]=BFH=191

For ATP\_PWM1: 8-bit DIN[15:8]=40H=64=CCRA, Duty=64/256≈25%

For ATP\_PWM2: 8-bit DINB[15:8]=BFH=191=CCRB, Duty=191/256≈75%



**Example 2 – Audio PWM Output Differential Mode, DIN[15:0]=C000H**

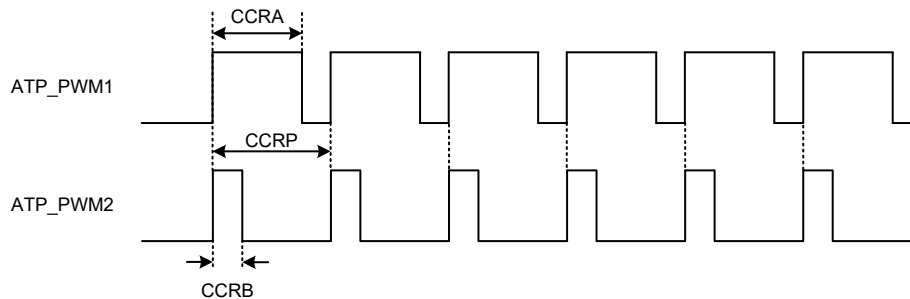
For example, DIN[15:0]=C000H (unsigned voice data), to generate the differential PWM outputs on ATP\_PWM1 and ATP\_PWM2 as shown in the timing diagram below.

DIN[15:0]=C000H, DINB[15:0]=~DIN[15:0]=3FFFH

The higher 8-bit of DIN: DIN[15:8]=C0H=192, DINB[15:8]=3FH=63

For ATP\_PWM1: 8-bit DIN[15:8]=C0H=192=CCRA, Duty=192/256≈75%

For ATP\_PWM2: 8-bit DINB[15:8]=3FH=63=CCRB, Duty=63/256≈25%





From the Example 1 ~ Example 2 above, the calculation method for the CCRA/CCRB values converted from the DIN[15:0] unsigned voice data for Audio PWM Differential mode can be obtained.

$$CCRA = \text{DIN}[15:8]$$

$$CCRB = \text{DINB}[15:8] = \sim\text{DIN}[15:8]$$

The ATP\_PWM1 and ATP\_PWM2 output signal duty and the corresponding CCRA and CCRB values in the PWM output differential mode are shown in the following table.

DIN[15:0] (Unsigned Voice Data)	ATP_PWM1 Output	ATP_PWM2 Output	CCRA/CCRB Settings
(00H, 00H)	Pulse Duty≈0%	Pulse Duty≈100%	CCRA=DIN[15:8]=00H CCRB=~DIN[15:8]=FFH
(20H, 00H)	Pulse Duty≈12.5%	Pulse Duty≈87.5%	CCRA=DIN[15:8]=20H CCRB=~DIN[15:8]=DFH
(40H, 00H)	Pulse Duty≈25%	Pulse Duty≈75%	CCRA=DIN[15:8]=40H CCRB=~DIN[15:8]=BFH
(60H, 00H)	Pulse Duty≈37.5%	Pulse Duty≈62.5%	CCRA=DIN[15:8]=60H CCRB=~DIN[15:8]=9FH
(80H, 00H)	Pulse Duty≈50%	Pulse Duty≈50%	CCRA=DIN[15:8]=80H CCRB=~DIN[15:8]=7FH
(A0H, 00H)	Pulse Duty≈62.5%	Pulse Duty≈37.5%	CCRA=DIN[15:8]=A0H CCRB=~DIN[15:8]=5FH
(C0H, 00H)	Pulse Duty≈75%	Pulse Duty≈25%	CCRA=DIN[15:8]=C0H CCRB=~DIN[15:8]=3FH
(E0H, 00H)	Pulse Duty≈87.5%	Pulse Duty≈12.5%	CCRA=DIN[15:8]=E0H CCRB=~DIN[15:8]=1FH
(FFH, FFH)	Pulse Duty≈100%	Pulse Duty≈0%	CCRA=DIN[15:8]=FFH CCRB=~DIN[15:8]=00H

**CCRA/CCRB Values – Audio PWM Output Differential Mode**

**Example 3 – Audio PWM Output Single Mode, DIN[15:0]=4000H**

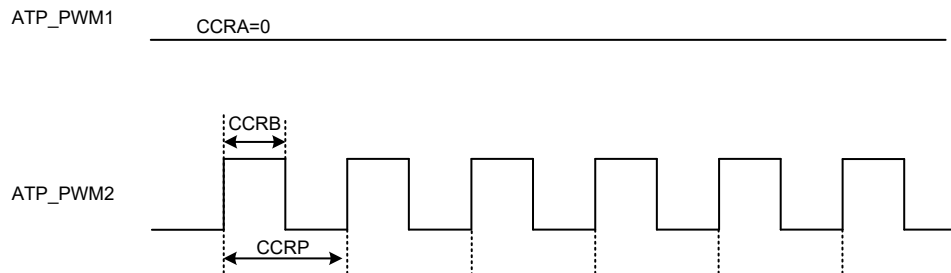
For example, DIN[15:0]=4000H (unsigned voice data), to generate the single PWM output on ATP\_PWM1 or ATP\_PWM2 as shown in the timing diagram below.

$$\text{DIN}[15:0]=4000\text{H}, \text{DINB}[15:0]=\sim\text{DIN}[15:0]=\text{BFFFH}$$

The Bit 14~Bit 7 of DINB:  $\text{DINB}[14:7]=7\text{FH}=127$

For ATP\_PWM1: As  $\text{DIN}[15]=0$ ,  $\text{CCRA}=0$ ,  $\text{Duty}=0/256\approx 0\%$

For ATP\_PWM2:  $\text{DINB}[14:7]=7\text{FH}=127=\text{CCRB}$ ,  $\text{Duty}=127/256\approx 50\%$



**Example 4 – Audio PWM Output Single Mode, DIN[15:0]=C000H**

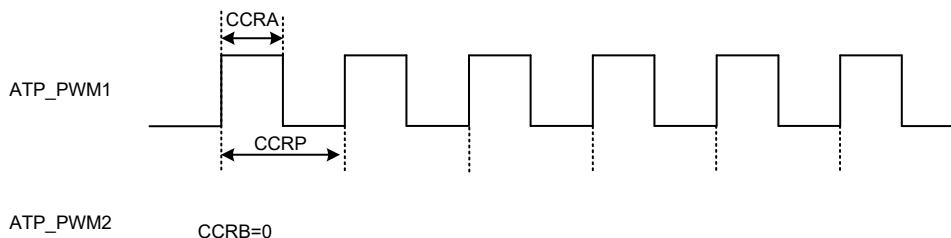
For example, DIN[15:0]=C000H (unsigned voice data), to generate the single PWM output on ATP\_PWM1 or ATP\_PWM2 as shown in the timing diagram below.

DIN[15:0]=C000H, DINB[15:0]=~DIN[15:0]=3FFFH

The Bit 14~Bit 7 of DIN: DIN[14:7]=80H=128

For the ATP\_PWM1: DIN[14:7]=80H=128=CCRA, Duty=128/256≈50%

For the ATP\_PWM2: As DIN[15]=1, CCRB=0, Duty=0/256≈0%



From the Example 3 ~ Example 4 above, the calculation method for the CCRA/CCRB values converted from the DIN[15:0] unsigned voice data for the Audio PWM single mode can be obtained.

- If DIN[15]=0, CCRA=0 (ATP\_PWM1 outputs 0), CCRB=~DIN[14:7]
- If DIN[15]=1, CCRA=DIN[14:7], CCRB=0 (ATP\_PWM2 outputs 0)

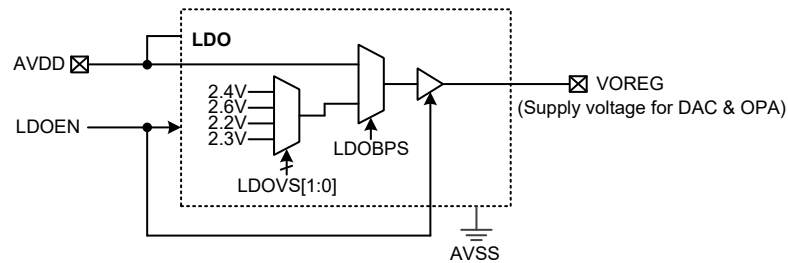
The ATP\_PWM1 and ATP\_PWM2 output signal duty and the corresponding CCRA and CCRB values in the PWM output single mode are shown in the following table.

DIN[15:0] (Unsigned Voice Data)	ATP_PWM1 Output	ATP_PWM2 Output	CCRA/CCRB Settings
(00H, 00H)	0	Pulse Duty≈100%	CCRA=00H CCRB=~DIN[14:7]=FFH
(20H, 00H)	0	Pulse Duty≈75%	CCRA=00H CCRB=~DIN[14:7]=BFH
(40H, 00H)	0	Pulse Duty≈50%	CCRA=00H CCRB=~DIN[14:7]=7FH
(60H, 00H)	0	Pulse Duty≈25%	CCRA=00H CCRB=~DIN[14:7]=3FH
(80H, 00H)	0	0	CCRA=DIN[14:7]=00H CCRB=00H
(A0H, 00H)	Pulse Duty≈25%	0	CCRA=DIN[14:7]=40H CCRB=00H
(C0H, 00H)	Pulse Duty≈50%	0	CCRA=DIN[14:7]=80H CCRB=00H
(E0H, 00H)	Pulse Duty≈75%	0	CCRA=DIN[14:7]=C0H CCRB=00H
(FFH, FFH)	Pulse Duty≈100%	0	CCRA=DIN[14:7]=FFH CCRB=00H

**CCRA/CCRB Values – Audio PWM Output Single Mode**

## Internal Power Supply

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



**Internal Power Supply Block Diagram**

### • PWRC Register

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

Bit 7 **LDOEN**: LDO function control  
 0: Disable  
 1: Enable

If the LDO is disabled, there will be no power consumption and the LDO output pin will remain at a low level using a weak internal pull-low resistor.

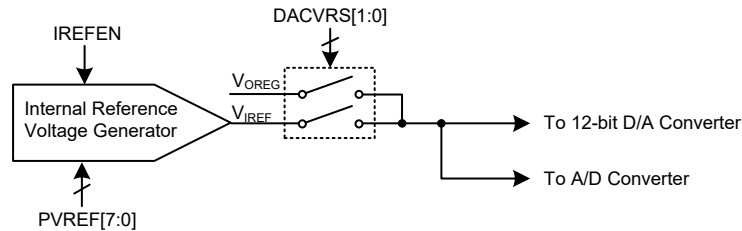
Bit 6~3 Unimplemented, read as “0”

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

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

## Internal Reference Voltage Generator

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



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

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

"x": Don't care

## Internal Reference Voltage Register Description

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

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

Internal Reference Voltage Register List

### • IREFC Register

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

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

Bit 6~5 Unimplemented, read as "0"

Bit 4 **IREFEN**: Internal Reference Voltage Generator control  
0: Disable  
1: Enable

This bit controls the internal reference voltage generator on/off. When this bit is set high, the internal reference voltage  $V_{IREF}$  can be generated and used as the reference voltage. When this bit is cleared to zero, the internal reference voltage generator will enter the power down mode and the output will be in a floating state. If the internal

referenc voltage is not used by any circuits, the IREFEN bit should be cleared to zero to reduce power consumption.

- Bit 3~2 Unimplemented, read as “0”
- Bit 1~0 **DACVRS1~DACVRS0**: 12-bit D/A Converter reference voltage  $V_{DACVREF}$  selection
  - 00/01:  $V_{IREF}$
  - 10:  $V_{OREG}$
  - 11: Floating

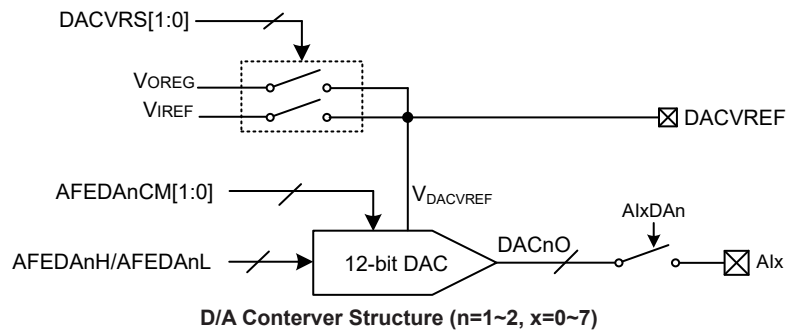
• **PVREF Register**

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

- Bit 7~0 **D7~D0**: Internal Reference Voltage Generator fine tune control  
 Setting the register can fine tune the internal reference voltage with a range of -60mV~+60mV (based on PVREF=80H). When the PVREF register value is increased by one, the internal reference voltage will decrease around 500μV, vice verse.

**D/A Converter – DAC**

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



**D/A Converter Register Description**

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
IREFC	OP1DO	—	—	IREFEN	—	—	DACVRS1	DACVRS0
AFEDAnC	—	—	—	—	—	—	AFEDAnCM1	AFEDAnCM0
AFEDAnL	D3	D2	D1	D0	—	—	—	—
AFEDAnH	D11	D10	D9	D8	D7	D6	D5	D4
ASWA1x (x=0~4)	A1xDA1	A1xDA2	A1XSINO	A1xADIP	A1xADIN	A1xOP2N	A1xOP1N	—

Register Name	Bit							
	7	6	5	4	3	2	1	0
ASWAlx (x=5~7)	AlxDA1	AlxDA2	AlxADIP	AlxADIN	AlxOP2P	AlxOP2O	AlxOP1P	AlxOP1O

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

• **IREFC Register**

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

- Bit 7      **OP1DO**: OPA1 digital output; positive logic  
Refer to the Operational Amplifier section.
- Bit 6~5    Unimplemented, read as “0”
- Bit 4      **IREFEN**: Internal Reference Voltage Generator control  
Refer to the Internal Reference Voltage Generator section.
- Bit 3~2    Unimplemented, read as “0”
- Bit 1~0    **DACVRS1~DACVRS0**: 12-bit D/A Converter reference voltage  $V_{DACVREF}$  selection  
00/01:  $V_{IREF}$   
10:  $V_{OREG}$   
11: Floating

• **AFEDAnC Register (n=1~2)**

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

- Bit 7~2    Unimplemented, read as “0”
- Bit 1~0    **AFEDAnCM1~AFEDAnCM0**: 12-bit D/A Converter mode control  
00: DAC Disable, output in a high impedance state  
01/11: DAC Enable  
10: DAC Disable, output in a ground state

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

Register	AFEDAnH								AFEDAnL							
	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
Name	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	—	—	—	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—	—	—	—
POR	0	0	0	0	0	0	0	0	0	0	0	0	—	—	—	—

“—”: Unimplemented, read as “0”

- D11~D0**: 12-bit D/A Converter output control bits  
The bit 7 ~ bit 0 in the AFEDAnH register combine with the bit 7 ~ bit 4 in the AFEDAnL register to form a 12-bit DAC value of 0~4095.  
DAC Output Voltage ( $V_{DACnO}$ )= $V_{DACVREF} \times (\text{DAC value}/4096)$

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

• **ASWAIx Register (x=0~4)**

Bit	7	6	5	4	3	2	1	0
Name	AIxDA1	AIxDA2	AIxSINO	AIxADIP	AIxADIN	AIxOP2N	AIxOP1N	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	0	0	0	0	0	0	0	—

Bit 7      **AIxDA1**: DAC10 to AIx switch control bit

0: Off  
 1: On

Bit 6      **AIxDA2**: DAC20 to AIx switch control bit

0: Off  
 1: On

Bit 5~1    Refer to other sections

Bit 0      Unimplemented, read as “0”

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

• **ASWAIx Register (x=5~7)**

Bit	7	6	5	4	3	2	1	0
Name	AIxDA1	AIxDA2	AIxADIP	AIxADIN	AIxOP2P	AIxOP2O	AIxOP1P	AIxOP1O
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7      **AIxDA1**: DAC10 to AIx switch control bit

0: Off  
 1: On

Bit 6      **AIxDA2**: DAC20 to AIx switch control bit

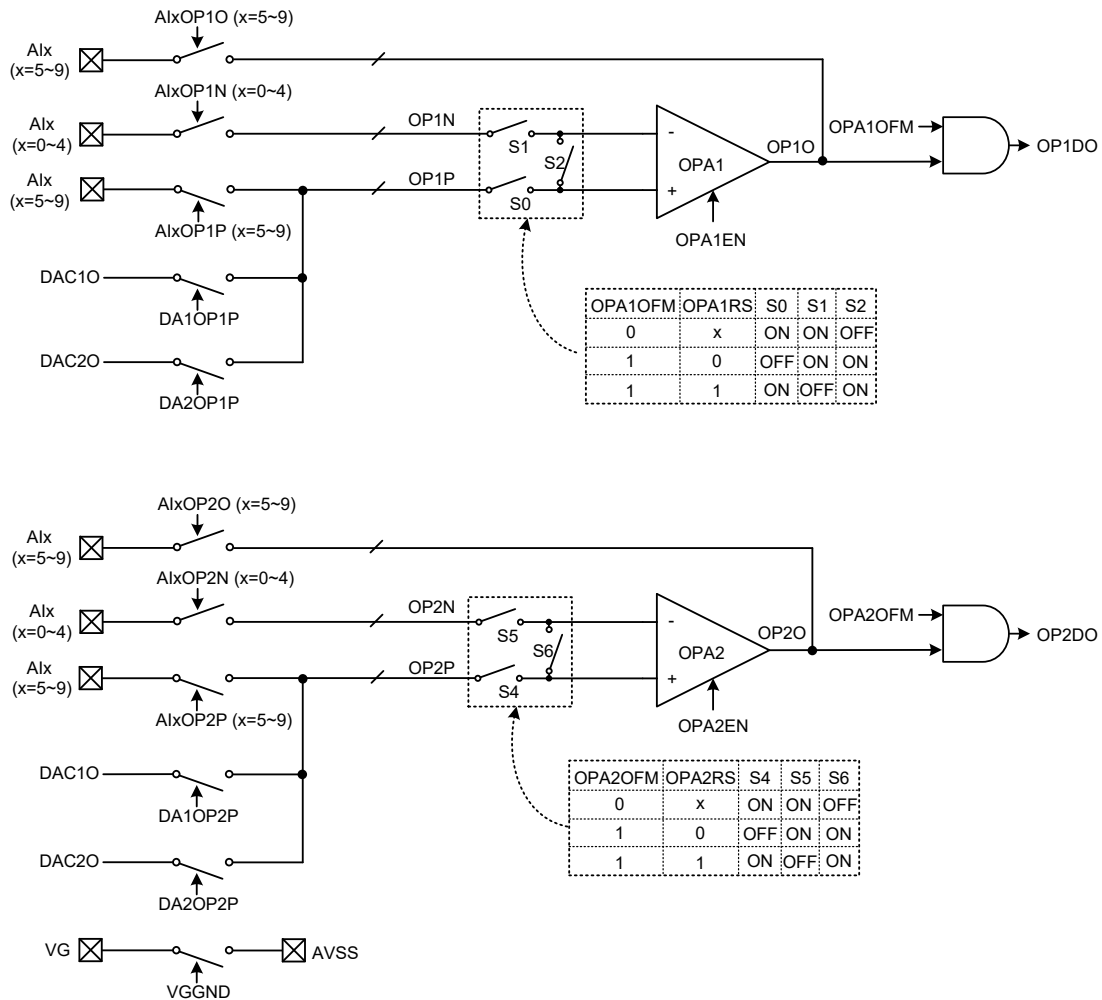
0: Off  
 1: On

Bit 5~0    Refer to other sections

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

## Operational Amplifier – OPA

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



**Operational Amplifier Structure**

### OPA Register Description

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
IREFC	OP1DO	—	—	IREFEN	—	—	DACVRS1	DACVRS0
OPA1C	OPA1OFM	OPA1RS	OPA1EN	OPA1OF4	OPA1OF3	OPA1OF2	OPA1OF1	OPA1OF0



Register Name	Bit							
	7	6	5	4	3	2	1	0
OPA2C	OPA2OFM	OPA2RS	OPA2EN	OPA2OF4	OPA2OF3	OPA2OF2	OPA2OF1	OPA2OF0
OPA3C	OPA3EN	IQ_FWR	OP2DO	—	OP3G3	OP3G2	OP3G1	OP3G0
ASWA1x (x=0~4)	AlxDA1	AlxDA2	AlxSINO	AlxADIP	AlxADIN	AlxOP2N	AlxOP1N	—
ASWA1x (x=5~7)	AlxDA1	AlxDA2	AlxADIP	AlxADIN	AlxOP2P	AlxOP2O	AlxOP1P	AlxOP1O
ASWA1x (x=8~9)	—	—	AlxADIP	AlxADIN	AlxOP2P	AlxOP2O	AlxOP1P	AlxOP1O
ASWDAC	VGGND	—	—	—	DA1OP1P	DA1OP2P	DA2OP1P	DA2OP2P

**OPA Register List**

• **IREFC Register**

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

- Bit 7      **OP1DO**: OPA1 digital output; positive logic  
The OP1DO bit is cleared to zero when the OPA1 function is disabled.  
When the OPA1OFM bit is set high, the OP1DO value is defined as OPA1 output status. Refer to the offset calibration procedure.
- Bit 6~5    Unimplemented, read as “0”
- Bit 4      **IREFEN**: Internal Reference Voltage Generator control  
Refer to the Internal Reference Voltage Generator section.
- Bit 3~2    Unimplemented, read as “0”
- Bit 1~0    **DACVRS1~DACVRS0**: 12-bit D/A Converter reference voltage  $V_{DACVREF}$  selection  
Refer to the D/A Converter section.

• **OPA1C Register**

Bit	7	6	5	4	3	2	1	0
Name	OPA1OFM	OPA1RS	OPA1EN	OPA1OF4	OPA1OF3	OPA1OF2	OPA1OF1	OPA1OF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7      **OPA1OFM**: OPA1 normal operation or input offset voltage calibration mode selection  
0: Normal operation  
1: Offset calibration mode  
When the input offset calibration mode is selected, the reference voltage input should be derived from the positive input.
- Bit 6      **OPA1RS**: OPA1 input offset voltage calibration reference selection  
0: Input reference voltage comes from OP1N  
1: Input reference voltage comes from DACnO  
Note: The OPA1 input offset voltage calibration can be executed only when the OPA1RS bit is set high.
- Bit 5      **OPA1EN**: OPA1 enable/disable control  
0: Disable  
1: Enable
- Bit 4~0    **OPA1OF4~OPA1OF0**: OPA1 input offset voltage calibration control

• **OPA2C Register**

Bit	7	6	5	4	3	2	1	0
Name	OPA2OFM	OPA2RS	OPA2EN	OPA2OF4	OPA2OF3	OPA2OF2	OPA2OF1	OPA2OF0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7      **OPA2OFM**: OPA2 normal operation or input offset voltage calibration mode selection  
0: Normal operation  
1: Offset calibration mode  
When the input offset calibration mode is selected, the reference voltage input should be derived from the positive input.
- Bit 6      **OPA2RS**: OPA2 input offset voltage calibration reference selection  
0: Input reference voltage comes from OP2N  
1: Input reference voltage comes from DACnO  
Note: The OPA2 input offset voltage calibration can be executed only when the OPA2RS bit is set high.
- Bit 5      **OPA2EN**: OPA2 enable/disable control  
0: Disable  
1: Enable
- Bit 4~0    **OPA2OF4~OPA2OF0**: OPA2 input offset voltage calibration control

• **OPA3C Register**

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

- Bit 7      **OPA3EN**: OPA3 enable/disable control  
Refer to the Bio-impedance Analysis Function section.
- Bit 6      **IQ\_FWR**: IQ/FWR mode selection  
Refer to the Bio-impedance Analysis Function section.
- Bit 5      **OP2DO**: OPA2 digital output; positive logic  
The OP2DO bit is cleared to zero when the OPA2 function is disabled.  
When the OPA2OFM bit is set high, the OP2DO value is defined as OPA2 output status. Refer to the offset calibration procedure.
- Bit 4      Unimplemented, read as “0”
- Bit 3~0    **OP3G3~OP3G0**: OPA3 gain control bit  
Refer to the Bio-impedance Analysis Function section.

• **ASWAlx Register (x=0~4)**

Bit	7	6	5	4	3	2	1	0
Name	AIxDA1	AIxDA2	AIxSINO	AIxADIP	AIxADIN	AIxOP2N	AIxOP1N	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	0	0	0	0	0	0	0	—

- Bit 7~3    Refer to other sections
- Bit 2      **AIxOP2N**: OP2N to AIx switch control bit  
0: Off  
1: On
- Bit 1      **AIxOP1N**: OP1N to AIx switch control bit  
0: Off  
1: On

Bit 0 Unimplemented, read as “0”

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

• **ASWAIx Register (x=5~7)**

Bit	7	6	5	4	3	2	1	0
Name	AIxDA1	AIxDA2	AIxADIP	AIxADIN	AIxOP2P	AIxOP2O	AIxOP1P	AIxOP1O
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~4 Refer to other sections

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

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

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

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

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

• **ASWAIx Register (x=8~9)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	AIxADIP	AIxADIN	AIxOP2P	AIxOP2O	AIxOP1P	AIxOP1O
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5~4 Refer to other sections

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

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

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

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

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

• **ASWDAC Register**

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

- Bit 7      **VGGND**: VG to ground switch control bit  
0: Off  
1: On
- Bit 6~4    Unimplemented, read as “0”
- Bit 3      **DA1OP1P**: DAC1O to OP1P switch control bit  
0: Off  
1: On
- Bit 2      **DA1OP2P**: DAC1O to OP2P switch control bit  
0: Off  
1: On
- Bit 1      **DA2OP1P**: DAC2O to OP1P switch control bit  
0: Off  
1: On
- Bit 0      **DA2OP2P**: DAC2O to OP2P switch control bit  
0: Off  
1: On

Note: Only one of the DA1OP1P and DA2OP1P bits can be on each time.  
Only one of the DA1OP2P and DA2OP2P bits can be on each time.

**Input Offset Calibration**

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

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

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

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

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

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

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

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

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

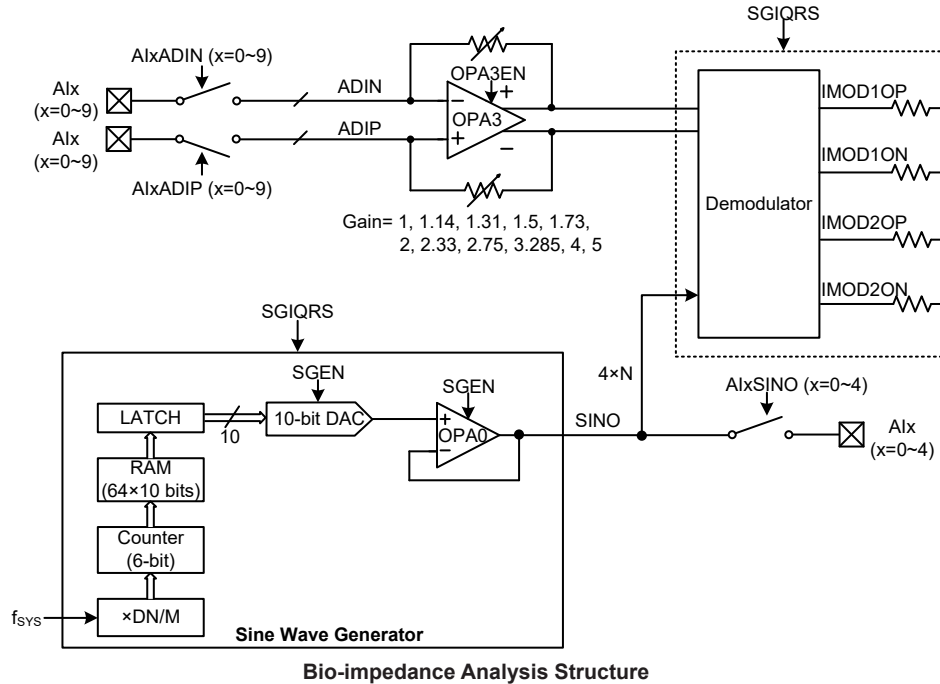
If the OPnDO bit state changes, record the OPAnOF[4:0] bits value as  $V_{OS2}$  and then go to Step 6.

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

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

## Bio-impedance Analysis Function

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



### Sine Wave Generator

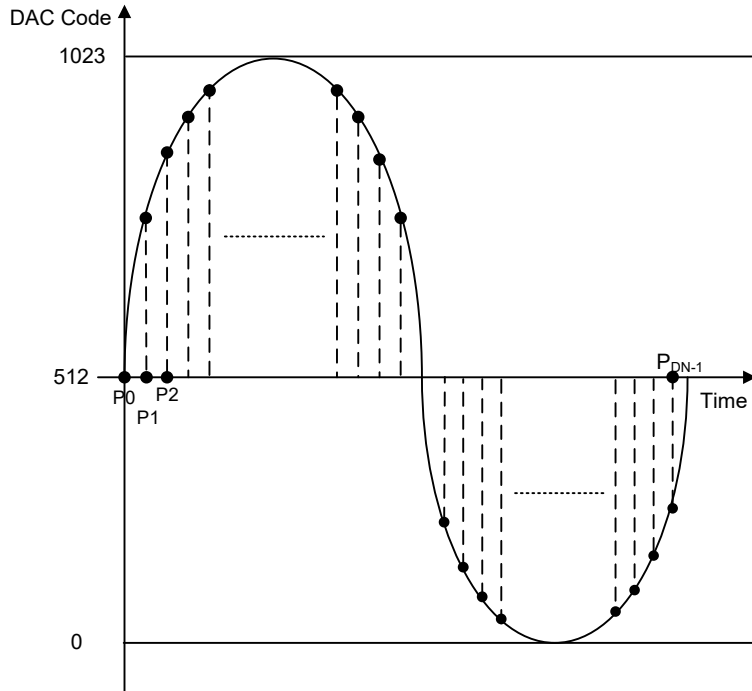
The sine generator consists of a frequency divider, a 6-bit counter, RAM, a 10-bit D/A converter and the operational amplifier 0. It offers a sine wave ranging from 5kHz to 500kHz and a 64×10 bits RAM for sine wave pattern which is set by software. The frequency divider will multiply by DN/M to generate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

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

Refer to following table for details.

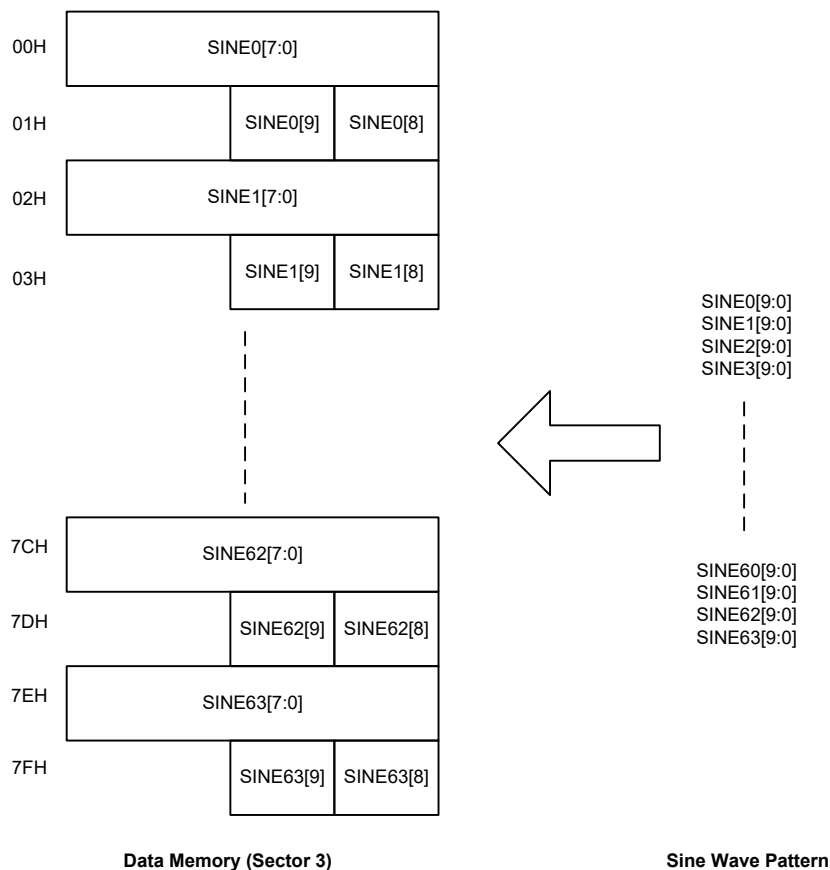
System Frequency	4MHz			8MHz			12MHz		
Sine Wave Frequency (kHz)	500	50	5	500	50	5	500	50	5
M	8	80	800	16	160	1600	24	240	2400
N	1	2	20	1	4	25	1	5	50
DN	8	40	40	16	40	64	24	48	48

Note: The sine wave generator circuit consists of a 10-bit D/A converter and a smoothing filter whose frequency at -3dB is equal to 489kHz. When the output frequency is 500kHz, the output wave amplitude will decrease and therefore it is impossible to achieve a full range of  $V_{OREG} \sim 0$ .



Users only need to prepare a whole sine wave pattern  $P_0 \sim P_{DN-1}$  and store them in RAM Sector 3 (00H~7FH). The bits 7~0 of the sine wave patterns are stored in even addresses and the bits 9~8 of the sine wave patterns are stored in odd addresses. Once the sine wave generator is enabled, the CPU cannot write/read any data to/from this RAM and the sine wave generator will read RAM data to the 10-bit D/A converter.

The controller reads the whole sine wave patterns from RAM and generates a sine waveform to the AIX pin. Refer to following figure.



### Bio-impedance Analysis Register Description

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

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

**Bio-impedance Analysis Register List**

• **SGC Register**

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

Bit 7      **SGEN**: Sine Wave Generator control  
 0: Disable  
 1: Enable  
 When this bit is equal to “0”, the OPA0 and 10-bit D/A converter will be in power down mode.

Bit 6~1    Unimplemented, read as “0”

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

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

• **SGN Register**

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

Bit 7~0    **D7~D0**: The multiplier of system clock frequency (N)  
 The multiplier of system clock frequency (N)=D[7:0]+1.

• **SGDN Register**

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

Bit 7~6    Unimplemented, read as “0”

Bit 5~0    **D5~D0**: Data number of sample  
 A whole sine wave cycle numerical value is stored in Data memory Sector 3. DN is equal to D[5:0]+1. When in the IQ mode, the DN value must be a multiple of 8 for normal operations.

• **OPA3C Register**

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

Bit 7      **OPA3EN**: OPA3 enable/disable control  
 0: Disable  
 1: Enable  
 When this bit is equal to “0”, OPA3 will be in power down mode.

Bit 6      **IQ\_FWR**: IQ/FWR mode selection  
 0: IQ mode  
 1: FWR mode

Bit 5      **OP2DO**: OPA2 digital output; positive logic  
 Refer to the Operational Amplifier section.



- Bit 4 Unimplemented, read as “0”
- Bit 3~0 **OP3G3~OP3G0**: OPA3 gain control bit
  - 0001: 1.14
  - 0010: 1.31
  - 0011: 1.5
  - 0100: 1.73
  - 0101: 2
  - 0110: 2.33
  - 0111: 2.75
  - 1000: 3.285
  - 1001: 4
  - 1010: 5
  - Others: 1

• **ASWA<sub>x</sub> Register (x=0~4)**

Bit	7	6	5	4	3	2	1	0
Name	AIxDA1	AIxDA2	AIxSINO	AIxADIP	AIxADIN	AIxOP2N	AIxOP1N	—
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	—
POR	0	0	0	0	0	0	0	—

- Bit 7~6 Refer to other sections
- Bit 5 **AIxSINO**: SINO to AIx switch control bit
  - 0: Off
  - 1: On
- Bit 4 **AIxADIP**: ADIP to AIx switch control bit
  - 0: Off
  - 1: On
- Bit 3 **AIxADIN**: ADIN to AIx switch control bit
  - 0: Off
  - 1: On
- Bit 2~1 Refer to other sections
- Bit 0 Unimplemented, read as “0”

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

• **ASWA<sub>x</sub> Register (x=5~7)**

Bit	7	6	5	4	3	2	1	0
Name	AIxDA1	AIxDA2	AIxADIP	AIxADIN	AIxOP2P	AIxOP2O	AIxOP1P	AIxOP1O
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7~6 Refer to other sections
- Bit 5 **AIxADIP**: ADIP to AIx switch control bit
  - 0: Off
  - 1: On
- Bit 4 **AIxADIN**: ADIN to AIx switch control bit
  - 0: Off
  - 1: On
- Bit 3~0 Refer to other sections

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

• **ASWAIx Register (x=8~9)**

Bit	7	6	5	4	3	2	1	0
Name	—	—	AIxADIP	AIxADIN	AIxOP2P	AIxOP2O	AIxOP1P	AIxOP1O
R/W	—	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	—	0	0	0	0	0	0

Bit 7~6 Unimplemented, read as “0”

Bit 5 **AIxADIP**: ADIP to AIx switch control bit  
0: Off  
1: On

Bit 4 **AIxADIN**: ADIN to AIx switch control bit  
0: Off  
1: On

Bit 3~0 Refer to other sections

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

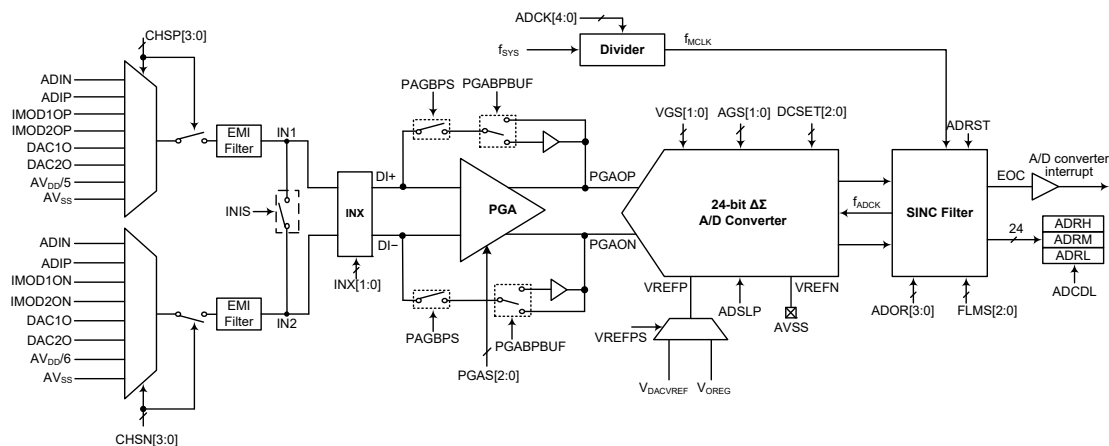
## Analog to Digital Converter – ADC

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

### A/D Converter Overview

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

In addition, PGA gain control, A/D converter gain control and A/D converter reference voltage gain control determine the amplification gain for A/D converter input signal. The designer can select the best gain combination for the desired amplification applied to the input signal. The following block diagram illustrates the A/D converter basic operational function. The A/D converter external input channel can be arranged as sixteen single-ended A/D converter input channels or eight differential input channels. The input signal can be amplified by PGA before entering the 24-bit Delta Sigma A/D converter. The A/D converter module will output one bit converted data to SINC filter which can transform the converted one-bit data to 24 bits and store them into the specific data registers. With high accuracy and performance, the device is very suitable for differential output sensor applications such as weight measurement scales and other related products.



Note: The PGA and A/D converter are power supplied by the AV<sub>DD</sub> and AV<sub>SS</sub>. The SINC Filter is power supplied by the V<sub>DD</sub> and V<sub>SS</sub>.

### A/D Converter Structure

### A/D Converter Register Description

Overall operation of the A/D converter is controlled using several registers. Three read only registers exist to store the A/D converter data 24-bit value. The remaining registers are control registers which setup the gain selections and control function of the A/D converter.

Register Name	Bit							
	7	6	5	4	3	2	1	0
PGAC0	PGABPS	VGS1	VGS0	AGS1	AGS0	PGAS2	PGAS1	PGAS0
PGAC1	PGABPBUF	INIS	INX1	INX0	DCSET2	DCSET1	DCSET0	—
PGACS	CHSN3	CHSN2	CHSN1	CHSN0	CHSP3	CHSP2	CHSP1	CHSP0
ADRL	D7	D6	D5	D4	D3	D2	D1	D0
ADRM	D15	D14	D13	D12	D11	D10	D9	D8
ADRH	D23	D22	D21	D20	D19	D18	D17	D16
ADCR0	—	FLMS2	FLMS1	FLMS0	ADOR3	ADOR2	ADOR1	ADOR0
ADCR1	VREFPS	—	ADCDL	EOC	ADRST	ADOFF	ADSLP	EDGE_SEL
ADCS	—	—	—	ADCK4	ADCK3	ADCK2	ADCK1	ADCK0
CHOP	D7	D6	D5	D4	D3	D2	R_FILSEL	R_CKCHOP

**A/D Converter Register List**

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

There are three registers related to the programmable gain control, PGAC0, PGAC1 and PGACS. The PGAC0 register is used to select the PGA gain, A/D Converter gain, A/D Converter reference voltage gain and PGA bypass control. The PGAC1 register is used to define the input connection, PGA enable/disable control and PGA bypass gain with/without buffer control. In addition, the PGACS register is used to select the PGA inputs. Therefore, the input channels have to be determined by the CHSP3~CHSP0 and CHSN3~CHSN0 bits to determine which analog channel signals, OPA inputs, DAC outputs or internal power supply are actually connected to the internal differential A/D converter.

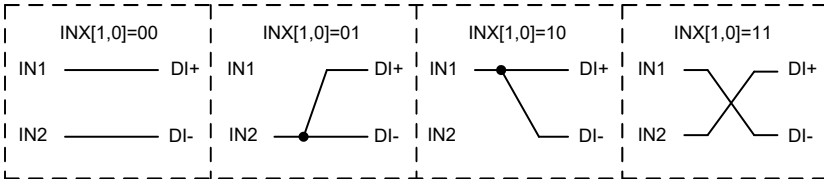
• **PGAC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	PGABPS	VGS1	VGS0	AGS1	AGS0	PGAS2	PGAS1	PGAS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7      **PGABPS**: PGA DI+/DI- differential channel input bypass selection  
 0: Not bypass  
 1: Bypass
- Bit 6~5    **VGS1~VGS0**: VREFP/VREFN differential reference voltage gain selection  
 00: VREFGN=1  
 01: VREFGN=1/2  
 10: VREFGN=1/4  
 11: Reserved
- Bit 4~3    **AGS1~AGS0**: A/D converter PGAOP/PAGON differential input signal gain selection  
 00: ADGN=1  
 01: ADGN=2  
 10: ADGN=4  
 11: Reserved
- Bit 2~0    **PGAS2~PGAS0**: PGA DI+/DI- differential channel input gain selection  
 000: PGAGN=1  
 001: PGAGN=2  
 010: PGAGN=4  
 011: PGAGN=8  
 100: PGAGN=16  
 101: PGAGN=32  
 110: PGAGN=64  
 111: PGAGN=128

• **PGAC1 Register**

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

- Bit 7      **PGABPBUF**: DI+/DI- differential channel input bypass gain with buffer  
 0: Without buffer  
 1: With buffer  
 This bit is used to set the signal bypass gain with buffer or without buffer when the PGABPS bit is set high to enable the bypass circuit.
- Bit 6      **INIS**: Selected inputs, IN1/IN2, internal connection control  
 0: Not connected  
 1: Connected
- Bit 5~4    **INX1~INX0**: Selected inputs, IN1/IN2, and the PGA differential input ends, DI+/DI- connection control bit
- 

INX[1,0]=00      INX[1,0]=01      INX[1,0]=10      INX[1,0]=11
- Bit 3~1    **DCSET2~DCSET0**: Differential input signal PGAOP/PAGON offset selection  
 000: DCSET=+0V  
 001: DCSET=+0.25×ΔVR\_I

- 010: DCSET=+0.5×ΔVR\_I
- 011: DCSET=+0.75×ΔVR\_I
- 100: DCSET=+0V
- 101: DCSET=-0.25×ΔVR\_I
- 110: DCSET=-0.5×ΔVR\_I
- 111: DCSET=-0.75×ΔVR\_I

The voltage, ΔVR\_I, is the differential reference voltage after amplification, which is amplified by specific gain selection based on the selected inputs.

Bit 0 Unimplemented, read as “0”

• **PGACS Register**

Bit	7	6	5	4	3	2	1	0
Name	CHSN3	CHSN2	CHSN1	CHSN0	CHSP3	CHSP2	CHSP1	CHSP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~4 **CHSN3~CHSN0**: Negative input end IN2 selection

- 0000: ADIN
  - 0001: ADIP
  - 0010~0011: Reserved
  - 0100: IMOD1ON
  - 0101: IMOD2ON
  - 0110: DAC1O
  - 0111: DAC2O
  - 1000: AV<sub>DD</sub>/6
  - 1001: AV<sub>SS</sub>
  - 1010~1111: Reserved
- These bits are used to select the negative input, IN2.

Bit 3~0 **CHSP3~CHSP0**: Positive input end IN1 selection

- 0000: ADIN
  - 0001: ADIP
  - 0010~0011: Reserved
  - 0100: IMOD1OP
  - 0101: IMOD2OP
  - 0110: DAC1O
  - 0111: DAC2O
  - 1000: AV<sub>DD</sub>/5
  - 1001: AV<sub>SS</sub>
  - 1010~1111: Reserved
- These bits are used to select the positive input, IN1.

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

The 24-bit Delta Sigma A/D converter requires three data registers to store the converted value. These are a high byte register, known as ADRH, a middle byte register, known as ADRM, and a low byte register, known as ADRL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value, D23~D0.

• **ADRL Register**

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

“x”: unknown

Bit 7~0 **D7~D0**: A/D conversion data register bit 7 ~ bit 0

• **ADRM Register**

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

“x”: unknown

Bit 7~0     **D15~D8**: A/D conversion data register bit 15 ~ bit 8

• **ADRH Register**

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

“x”: unknown

Bit 7~0     **D23~D16**: A/D conversion data register bit 23 ~ bit 16

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

To control the function and operation of the A/D converter, several control registers known as ADCR0, ADCR1 and ADCS are provided. These 8-bit registers define functions such as the A/D converter clock source, A/D conversion oversampling data rate as well as controlling the power-up function and monitoring the A/D converter end of conversion status.

• **ADCR0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	FLMS2	FLMS1	FLMS0	ADOR3	ADOR2	ADOR1	ADOR0
R/W	—	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	—	0	0	0	0	0	0	0

Bit 7     Unimplemented, read as “0”

Bit 6~4     **FLMS2~FLMS0**: A/D converter clock f<sub>ADCK</sub> divider and sampled data doubling function control

000: CHOP=2, f<sub>ADCK</sub>=f<sub>MCLK</sub>/30

010: CHOP=2, f<sub>ADCK</sub>=f<sub>MCLK</sub>/12

100: CHOP=1, f<sub>ADCK</sub>=f<sub>MCLK</sub>/30

110: CHOP=1, f<sub>ADCK</sub>=f<sub>MCLK</sub>/12

Others: Reserved

When the CHOP is equal to 2, it means that the sampled data amount will be doubled for the normal conversion mode. However, it can be regarded as a low latency conversion mode if the CHOP is equal to 1, which means that the sampled data doubling function is disabled.

Bit 3~0     **ADOR3~ADOR0**: A/D conversion oversampling rate selection

0000: Oversampling rate OSR=32768

0001: Oversampling rate OSR=16384

0010: Oversampling rate OSR=8192

0011: Oversampling rate OSR=4096

0100: Oversampling rate OSR=2048

0101: Oversampling rate OSR=1024

0110: Oversampling rate OSR=512

0111: Oversampling rate OSR=256

1000: Oversampling rate OSR=128

Others: Reserved

• **ADCR1 Register**

Bit	7	6	5	4	3	2	1	0
Name	VREFPS	—	ADCDL	EOC	ADRST	ADOFF	ADSLP	EDGE_SEL
R/W	R/W	—	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	—	0	0	0	1	0	0

Bit 7 **VREFPS**: A/D converter VREFP reference voltage selection  
 0:  $V_{DACVREF}$   
 1:  $V_{OREG}$

Bit 6 Unimplemented, read as “0”

Bit 5 **ADCDL**: A/D converted data latch function control  
 0: Disable data latch function  
 1: Enable data latch function

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

Bit 4 **EOC**: End of A/D conversion flag  
 0: A/D conversion in progress  
 1: A/D conversion ended

This bit must be cleared by software.

Bit 3 **ADRST**: A/D converter software reset control  
 0: Disable  
 1: Enable

This bit is used to reset the A/D converter internal digital SINC filter. This bit is set low for A/D normal operations. However, if set high, the internal digital SINC filter will be reset and the current A/D converted data will be aborted. A new A/D data conversion process will not be initiated until this bit is set low again.

Bit 2 **ADOFF**: A/D converter module power on/off control  
 0: Power on  
 1: Power off

This bit controls the power of the A/D converter module. This bit should be cleared to zero to enable the A/D converter. If the bit is set high then the A/D converter will be switched off reducing the device power consumption. As the A/D converter will consume a limited amount of power, even when not executing a conversion, this may be an important consideration in power sensitive battery powered applications.

It is recommended to set the ADOFF bit high before the device enters the IDLE/SLEEP mode for saving power. Setting the ADOFF bit high will power down the A/D converter module regardless of the ADSLP and ADRST bit settings.

Bit 1 **ADSLP**: A/D converter sleep mode enable control  
 0: Normal mode  
 1: Sleep mode

This bit is used to determine whether the A/D converter enters the sleep mode or not when the A/D converter is powered on by setting the ADOFF bit low. When the A/D converter is powered on and the ADSLP bit is low, the A/D converter will operate normally. However, the A/D converter will enter the sleep mode if the ADSLP bit is set high as the A/D converter has been powered on. The whole A/D converter circuit

will be switched off except the PGA and internal Bandgap circuit to reduce the power consumption.

- Bit 0     **EDGE\_SEL**: A/D converter latch clock selection  
           0: Normal  
           1: Reversed

• **ADCS Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	ADCK4	ADCK3	ADCK2	ADCK1	ADCK0
R/W	—	—	—	R/W	R/W	R/W	R/W	R/W
POR	—	—	—	0	0	0	0	0

Bit 7~5     Unimplemented, read as “0”

Bit 4~0     **ADCK4~ADCK0**: A/D converter clock source  $f_{MCLK}$  setup  
           00000~11110:  $f_{MCLK}=f_{SYS}/2/(ADCK[4:0]+1)$   
           11111:  $f_{MCLK}=f_{SYS}$

**SINC Filter Register – CHOP**

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

• **CHOP Register**

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

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

Bit 1     **R\_FILSEL**: Output digital filter selection  
           0: SINC2 filter (R\_CKCHOP should be logic low), Data latency=3 output data clocks  
           1: SINC3 filter (R\_CKCHOP should be logic high), Data latency=4 output data clocks

Bit 0     **R\_CKCHOP**: System chopper function selection  
           0: Enable (R\_FILSEL should be logic low)  
           1: Disable (R\_FILSEL should be logic high)

**A/D Converter Rate Definition**

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

Data Rate for SINC2

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

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

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

Data Rate for SINC3

$$=f_{ADCK}/OSR$$

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

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

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

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

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



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

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

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

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

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

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

### A/D Converter Operation

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

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

Note: “x” means unknown

#### A/D Operating Mode Summary

To enable the A/D Converter, the first step is to disable the A/D converter power down and sleep mode by clearing the ADOFF and ADPSLP bits to make sure the A/D converter is powered on. The ADRST bit in the ADCR1 register is used to start and reset the A/D converter after power on. When the microcontroller changes the ADRST bit from low to high and then low again, an analog to digital conversion in the SINC filter will be initiated. After this setup is completed, the A/D Converter is ready for operation. These three bits are used to control the overall start operation of the internal analog to digital converter.

The EOC bit in the ADCR1 register is used to indicate when the analog to digital conversion process is complete. This bit will be automatically set high by the microcontroller after a conversion cycle has ended. In addition, the corresponding A/D converter interrupt request flag will be set in the

interrupt control register, and if the A/D converter and global interrupts are enabled, an appropriate internal interrupt signal will be generated. This A/D converter internal interrupt signal will direct the program flow to the associated A/D converter internal interrupt address for processing. If the A/D converter internal interrupt is disabled, the microcontroller can poll the EOC bit in the ADCR1 register to check whether it has been set to “1” as an alternative method of detecting the end of an A/D conversion cycle. The A/D converted data will be updated continuously by the new converted data. If the A/D converted data latch function is enabled, the latest converted data will be latched and the following newly converted data will be discarded until this data latch function is disabled.

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

The differential reference voltage supply to the A/D Converter can be supplied from VREFP and VREFN. The desired VREFP selection is made using the VREFPS bit in the ADCR1 register.

### Summary of A/D Conversion Steps

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

- Step 1  
Select the PGA, A/D converter gains and reference voltage gain by the PGAC0 register.
- Step 2  
Select the input connection and bypass control by the PGAC1 register.
- Step 3  
Select the required A/D conversion clock source by correctly programming bits ADCK4~ADCK0 in the ADCS register.
- Step 4  
Select output data rate by configuring the ADOR3~ADOR0 and FLMS2~FLMS0 bits in the ADCR0 register.
- Step 5  
Select which channel is to be connected to the internal PGA by correctly programming the CHSP3~CHSP0 and CHSN3~CHSN0 bits which are contained in the PGACS register.
- Step 6  
Release the power down mode and sleep mode by clearing the ADOFF and ADSLP bits in ADCR1 register.
- Step 7  
Reset the A/D by setting the ADRST in the ADCR1 register to high, and then release the reset status by clearing this bit to zero.
- Step 8  
If the A/D converter interrupt is to be used, the related interrupt control registers must be correctly configured to ensure the A/D converter interrupt function is active. The master interrupt control bit, EMI, and the A/D converter interrupt bit, ADE, must both be set high in advance.
- Step 9  
To check when the analog to digital conversion process is completed, the EOC bit in the ADCR1 register can be polled. The conversion process is completed when this bit goes high. When this occurs the A/D converter data registers ADRL, ADRM and ADRH can be read to obtain the conversion value. As an alternative method, if the A/D converter interrupt is enabled and the stack is not full, the program can wait for an A/D converter interrupt to occur.

Note: When checking for the end of the conversion process, if the method of polling the EOC bit in the ADCR1 register is used, the interrupt enable step above can be omitted.

### Programming Considerations

During microcontroller operations where the A/D converter is not being used, the A/D converter internal circuitry can be switched off to reduce power consumption, by setting the ADOFF bit high. When this happens, the internal A/D converter circuits will not consume power irrespective of what analog voltage is applied to their input lines.

### A/D Converter Transfer Function

The device contains a 24-bit Delta Sigma A/D converter and its full-scale converted digitized value is from 8388607 to -8388608 in decimal value. The converted data format is formed using a two's complement binary value. The MSB of the converted data is the signed bit. Since the full-scale analog input value is equal to  $\Delta VR_I$ , which is the amplified value of the differential reference input voltage  $\Delta VR_{\pm}$  selected by the VREFPS bit in ADCR1 register, this gives a single bit analog input value of  $\Delta VR_I$  divided by 8388608.

$$1 \text{ LSB} = \Delta VR_I / 8388608$$

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

$$\Delta SI_I = (PGAGN \times ADGN \times \Delta DI_{\pm}) + DCSET$$

$$\Delta VR_I = VREFGN \times \Delta VR_{\pm}$$

$$ADC\_Conversion\_Data = (\Delta SI_I / \Delta VR_I) \times K$$

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

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

2.  $\Delta SI_I$ : Differential Input Signal after amplification and offset adjustment.
3. PGAGN: Programmable Gain Amplifier gain
4. ADGN: A/D Converter gain
5. VREFGN: Reference voltage gain
6.  $\Delta DI_{\pm}$ : Differential input signal derived from external channels or internal signals
7. DCSET: Offset voltage
8.  $\Delta VR_{\pm}$ : Differential Reference voltage
9.  $\Delta VR_I$ : Differential Reference input voltage after amplification

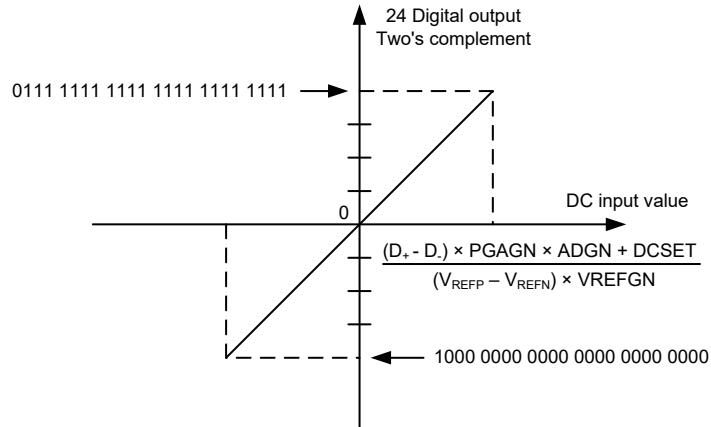
Due to the digital system design of the Delta Sigma A/D Converter, the maximum A/D converted value is 8388607 and the minimum value is -8388608. Therefore, there is a middle value of 0. The ADC\_Conversion\_Data equation illustrates this range of converted data variation.

A/D Conversion Data (2's complement, Hexadecimal)	Decimal Value
0x7FFFFFFF	8388607
0x800000	-8388608

**A/D Conversion Data Range**

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

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



### A/D Converted Data

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

### A/D Converted Data to Voltage

The converted data can be recovered using the following equations:

If MSB=0 – Positive Converted data

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

If the MSB=1 – Negative Converted data

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

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

## Serial Interface Module – SIM

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

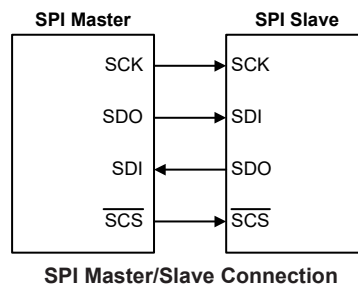
### SPI Interface

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

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

### SPI Interface Operation

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

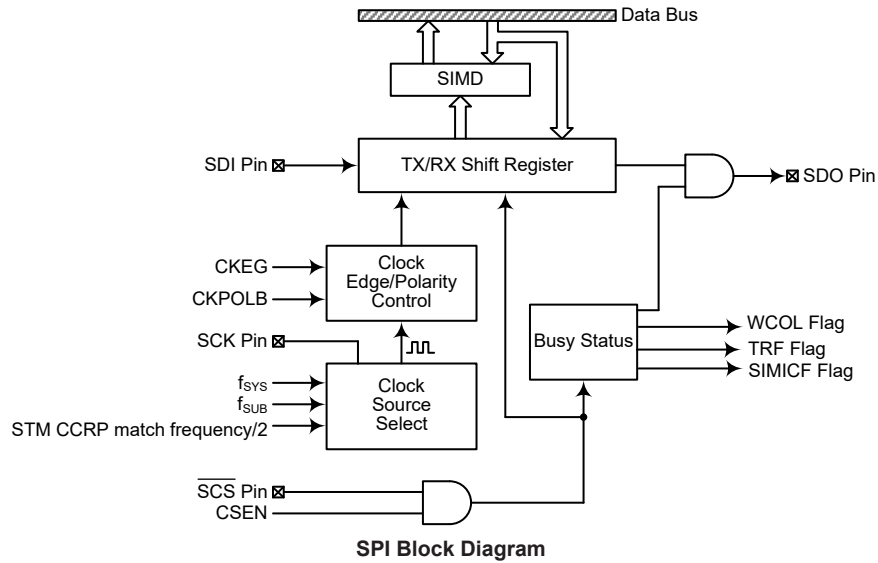


The SPI function in this device offers the following features:

- Full duplex synchronous data transfer
- Both Master and Slave modes

- LSB first or MSB first data transmission modes
- Transmission complete flag
- Rising or falling active clock edge

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



### SPI Registers

There are three internal registers which control the overall operation of the SPI interface. These are the SIMD data register and two registers SIMC0 and SIMC2. The SIMC1 register is only used by the I<sup>2</sup>C interface.

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

**SPI Register List**

### SPI Data Register

The SIMD register is used to store the data being transmitted and received. The same register is used by both the SPI and 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**: SIM data register bit 7 ~ bit 0

**SPI Control Registers**

There are also two control registers for the SPI interface, SIMC0 and SIMC2. Note that the SIMC2 register also has the name SIMA which is used by the I<sup>2</sup>C function. The SIMC1 register is not used by the SPI function, only by the I<sup>2</sup>C function. Register SIMC0 is used to control the enable/disable function and to set the data transmission clock frequency. Register SIMC2 is used for other control functions such as LSB/MSB selection, write collision flag, etc.

• **SIMC0 Register**

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

Bit 7~5     **SIM2~SIM0**: SIM Operating Mode Control  
 000: SPI master mode; SPI clock is  $f_{SYS}/4$   
 001: SPI master mode; SPI clock is  $f_{SYS}/16$   
 010: SPI master mode; SPI clock is  $f_{SYS}/64$   
 011: SPI master mode; SPI clock is  $f_{SUB}$   
 100: SPI master mode; SPI clock is STM CCRP match frequency/2  
 101: SPI slave mode  
 110: I<sup>2</sup>C slave mode  
 111: Non SIM function

These bits setup the overall operating mode of the SIM 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 STM and  $f_{SUB}$ . If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4     Unimplemented, read as “0”

Bit 3~2     **SIMDEB1~SIMDEB0**: I<sup>2</sup>C Debounce Time Selection  
 The SIMDEB1~SIMDEB0 bits are only used in the I<sup>2</sup>C mode and the detailed definition is described in the I<sup>2</sup>C section.

Bit 1     **SIMEN**: SIM Enable Control  
 0: Disable  
 1: Enable

The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and  $\overline{SCS}$ , or SDA and SCL lines will lose their SPI or I<sup>2</sup>C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I<sup>2</sup>C interface via the 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**: SIM SPI slave mode Incomplete Transfer Flag  
           0: SIM SPI slave mode incomplete condition not occurred  
           1: SIM SPI slave mode incomplete condition occurred

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

• **SIMC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6     **D7~D6**: Undefined bits  
 These bits can be read or written by the application program.

Bit 5     **CKPOLB**: SPI clock line base condition selection  
           0: The SCK line will be high when the clock is inactive  
           1: The SCK line will be low when the clock is inactive

The CKPOLB bit determines the base condition of the clock line, if the bit is high, then the SCK line will be low when the clock is inactive. When the CKPOLB bit is low, then the SCK line will be high when the clock is inactive.

Bit 4     **CKEG**: SPI SCK clock active edge type selection  
 CKPOLB=0  
           0: SCK is high base level and data capture at SCK rising edge  
           1: SCK is high base level and data capture at SCK falling edge  
 CKPOLB=1  
           0: SCK is low base level and data capture at SCK falling edge  
           1: SCK is low base level and data capture at SCK rising edge

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

Bit 3     **MLS**: SPI data shift order  
           0: LSB first  
           1: MSB first

This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

Bit 2     **CSEN**: SPI  $\overline{SCS}$  pin control  
           0: Disable  
           1: Enable

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

Bit 1     **WCOL**: SPI write collision flag  
           0: No collision  
           1: Collision



The WCOL flag is used to detect whether a data collision has occurred or not. If this bit is high, it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. This bit can be cleared by the application program.

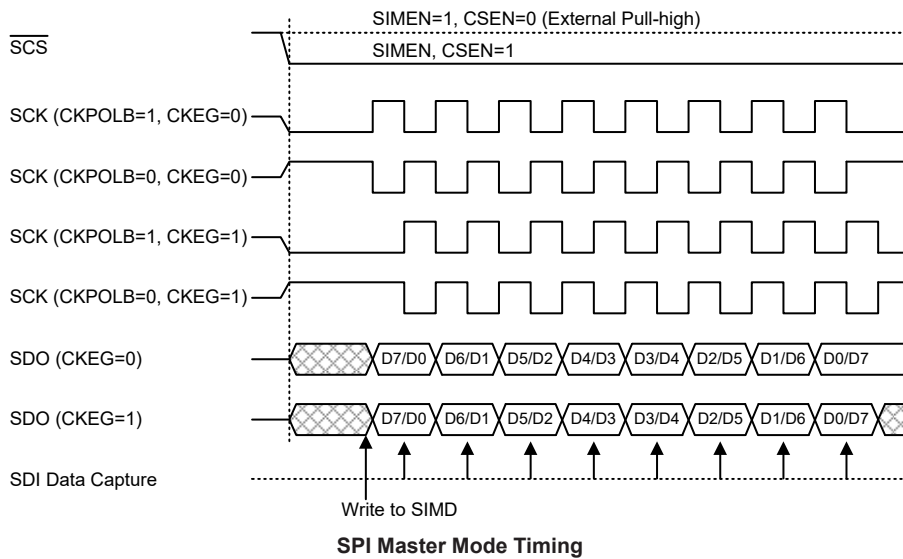
Bit 0 **TRF**: SPI Transmit/Receive complete flag  
 0: SPI data is being transferred  
 1: SPI data transfer is completed

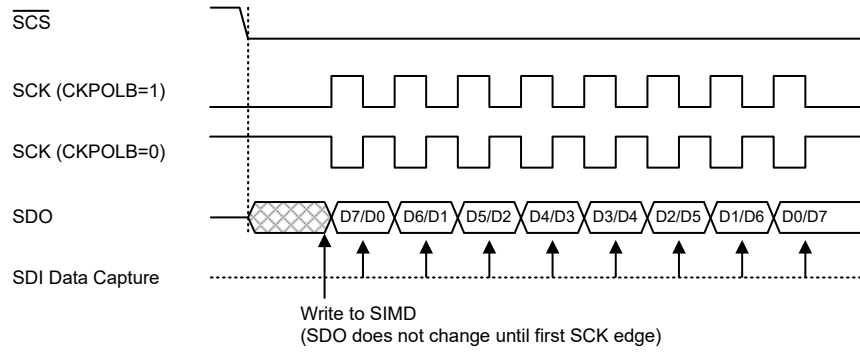
The TRF bit is the Transmit/Receive Complete flag and is set to 1 automatically when an SPI data transfer is completed, but must be cleared to 0 by the application program. It can be used to generate an interrupt.

**SPI Communication**

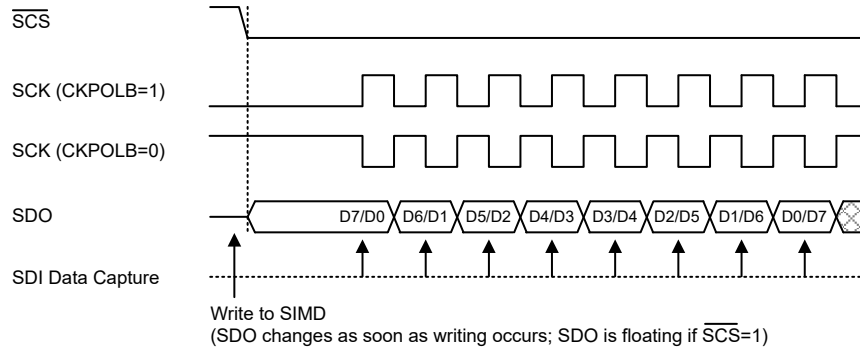
After the SPI interface is enabled by setting the SIMEN bit high, then in the Master Mode, when data is written to the SIMD register, transmission/reception will begin simultaneously. When the data transfer is complete, the TRF flag will be set automatically, but must be cleared using the application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

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



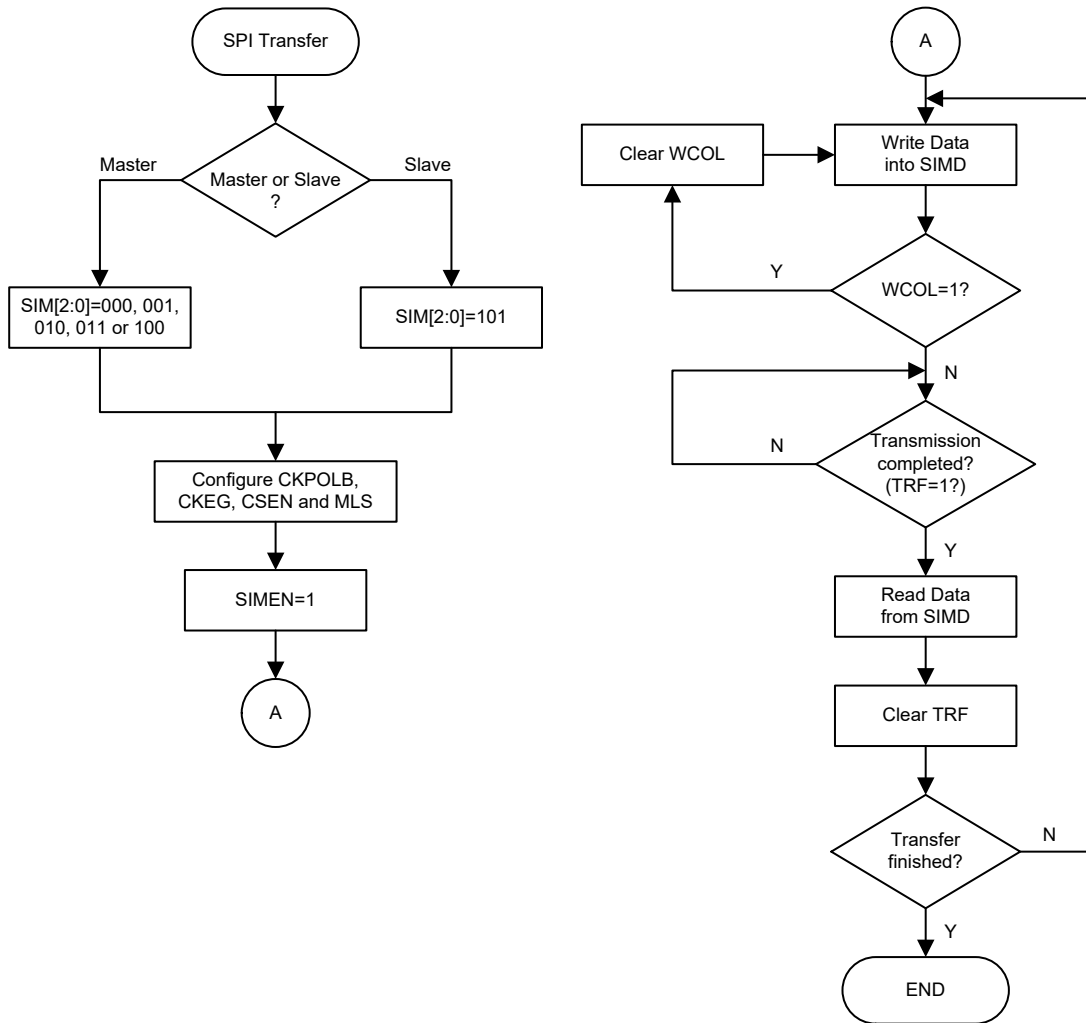


**SPI Slave Mode Timing – CKEG=0**



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

**SPI Slave Mode Timing – CKEG=1**



**SPI Transfer Control Flow Chart**

**SPI Bus Enable/Disable**

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

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

**SPI Operation Steps**

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

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

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

#### Master Mode

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

#### Slave Mode

- Step 1  
Select the SPI Slave mode using the SIM2~SIM0 bits in the SIMC0 control register
- Step 2  
Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master devices.
- Step 3  
Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.
- Step 4  
For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCK and  $\overline{SCS}$  signal. After this, go to step 5.

For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.

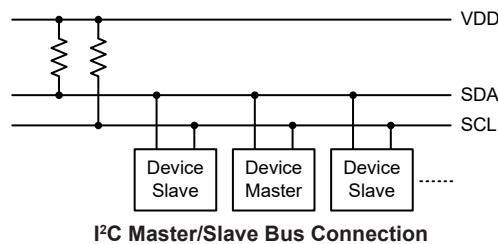
- Step 5  
Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6  
Check the TRF bit or wait for an SIM SPI serial bus interrupt.
- Step 7  
Read data from the SIMD register.
- Step 8  
Clear TRF.
- Step 9  
Go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

### Error Detection

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

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

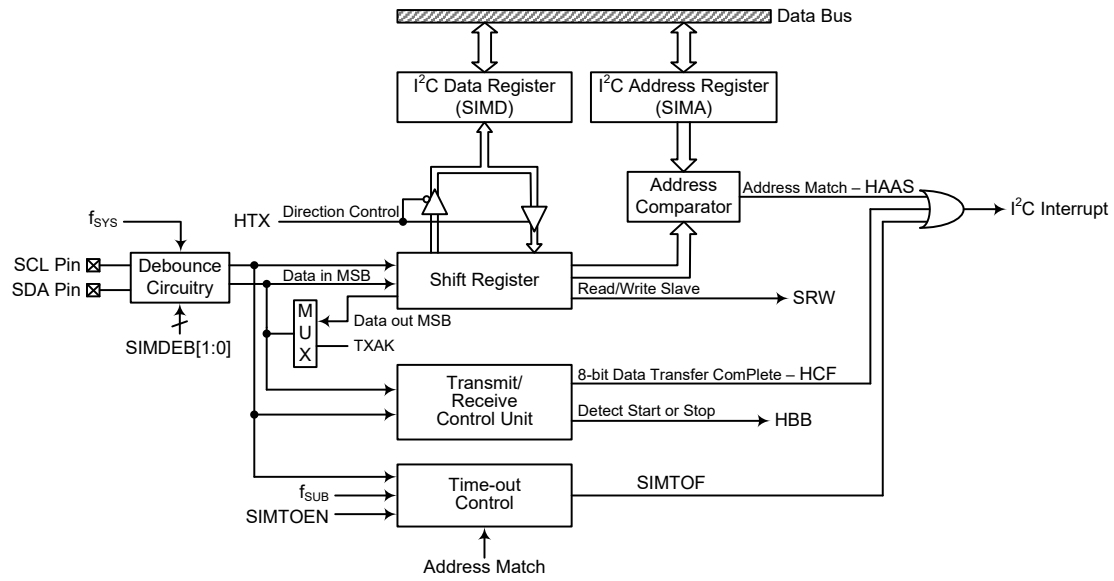
The I<sup>2</sup>C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple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



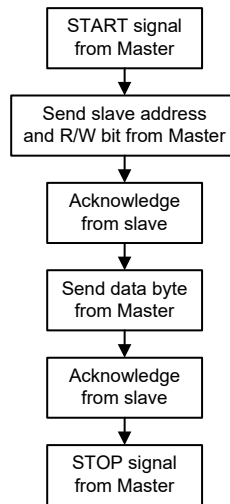
### I<sup>2</sup>C Interface Operation

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



**I<sup>2</sup>C Interface Block Diagram**



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

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

I <sup>2</sup> C Debounce Time Selection	I <sup>2</sup> C Standard Mode (100kHz)	I <sup>2</sup> C Fast Mode (400kHz)
No Debounce	$f_{SYS} > 2\text{MHz}$	$f_{SYS} > 4\text{MHz}$
2 system clock debounce	$f_{SYS} > 4\text{MHz}$	$f_{SYS} > 8\text{MHz}$
4 system clock debounce	$f_{SYS} > 4\text{MHz}$	$f_{SYS} > 8\text{MHz}$

**I<sup>2</sup>C Minimum  $f_{SYS}$  Frequency Requirement**

### 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.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SIMC0	SIM2	SIM1	SIM0	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
SIMC1	HCF	HAAS	HBB	HTX	TXAK	SRW	IAMWU	RXAK
SIMA	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
SIMD	D7	D6	D5	D4	D3	D2	D1	D0
SIMTOC	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0

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**: SIM 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. Bits 7~1 of the SIMA register define the device slave address. Bit 0 is not implemented.

When a master device, which is connected to the 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. Note that the SIMA register is the same register address as SIMC2 which is used by the SPI interface.

#### • SIMA Register

Bit	7	6	5	4	3	2	1	0
Name	SIMA6	SIMA5	SIMA4	SIMA3	SIMA2	SIMA1	SIMA0	D0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~1      **SIMA6~SIMA0**: I<sup>2</sup>C slave address  
 SIMA6~SIMA0 is the I<sup>2</sup>C slave address bit 6~bit 0.

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

### 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 register SIMC0 is used to control the enable/disable function and to select the I<sup>2</sup>C slave mode and debounce time. 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	—	SIMDEB1	SIMDEB0	SIMEN	SIMICF
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	1	1	1	—	0	0	0	0

Bit 7~5 **SIM2~SIM0**: SIM Operating Mode Control  
 000: SPI master mode; SPI clock is  $f_{SYS}/4$   
 001: SPI master mode; SPI clock is  $f_{SYS}/16$   
 010: SPI master mode; SPI clock is  $f_{SYS}/64$   
 011: SPI master mode; SPI clock is  $f_{SUB}$   
 100: SPI master mode; SPI clock is STM CCRP match frequency/2  
 101: SPI slave mode  
 110: I<sup>2</sup>C slave mode  
 111: Non SIM function

These bits setup the overall operating mode of the SIM 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 STM and  $f_{SUB}$ . If the SPI Slave Mode is selected then the clock will be supplied by an external Master device.

Bit 4 Unimplemented, read as “0”

Bit 3~2 **SIMDEB1~SIMDEB0**: I<sup>2</sup>C Debounce Time Selection  
 00: No debounce  
 01: 2 system clock debounce  
 1x: 4 system clock debounce

These bits are used to select the I<sup>2</sup>C debounce time when the SIM is configured as the I<sup>2</sup>C interface function by setting the SIM2~SIM0 bits to “110”.

Bit 1 **SIMEN**: SIM Enable Control  
 0: Disable  
 1: Enable

The bit is the overall on/off control for the SIM interface. When the SIMEN bit is cleared to zero to disable the SIM interface, the SDI, SDO, SCK and  $\overline{SCS}$ , or SDA and SCL lines will lose their SPI or I<sup>2</sup>C function and the SIM operating current will be reduced to a minimum value. When the bit is high the SIM interface is enabled. The SIM configuration option must have first enabled the SIM interface for this bit to be effective. If the SIM is configured to operate as an SPI interface via the SIM2~SIM0 bits, the contents of the SPI control registers will remain at the previous settings when the SIMEN bit changes from low to high and should therefore be first initialised by the application program. If the SIM is configured to operate as an I<sup>2</sup>C interface via the 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**: SIM SPI Incomplete Flag  
 The SIMICF bit is only used in the SPI mode and the detailed definition is described in the SPI section.



• **SIMC1 Register**

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

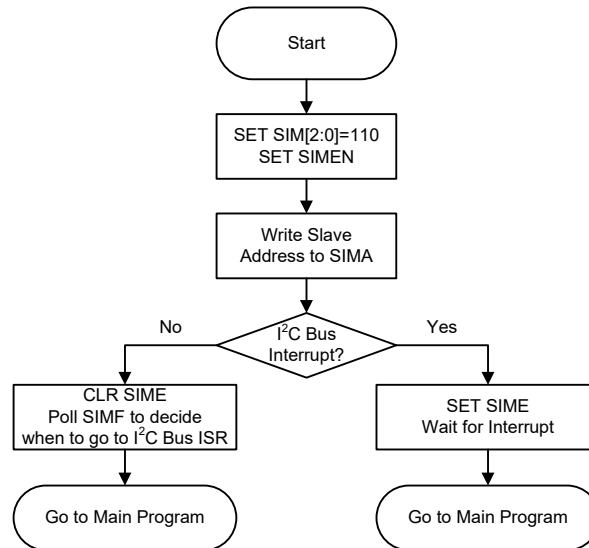
- Bit 7 HCF:** I<sup>2</sup>C Bus data transfer completion flag  
 0: Data is being transferred  
 1: Completion of an 8-bit data transfer  
 The HCF flag is the data transfer completion flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated. Below is an example of the flow of a two-byte I<sup>2</sup>C data transfer. First, I<sup>2</sup>C slave device receives a start signal from I<sup>2</sup>C master and then HCF bit is automatically cleared to zero. Second, I<sup>2</sup>C slave device finishes receiving the 1st data byte and then HCF bit is automatically set high. Third, user read the 1st data byte from SIMD register by the application program and then HCF bit is automatically cleared to zero. Fourth, I<sup>2</sup>C slave device finishes receiving the 2nd data byte and then HCF bit is automatically set to one and so on. Finally, I<sup>2</sup>C slave device receives a stop signal from I<sup>2</sup>C master and then HCF bit is automatically set high.
- Bit 6 HAAS:** I<sup>2</sup>C Bus data transfer completion flag  
 0: Not address match  
 1: Address match  
 The HAAS flag is the address match flag. This flag is used to determine if the slave device address is the same as the master transmit address. If the addresses match then this bit will be high, if there is no match then the flag will be low.
- Bit 5 HBB:** 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 cleared 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 transmitter/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 sends acknowledge flag  
 1: Slave does not send acknowledge flag  
 The TXAK flag is the transmit acknowledge flag. After the slave device has received 8 bits of data, this flag will be transmitted to the bus on the 9<sup>th</sup> clock from the slave device. The slave device must always set the 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	<b>IAMWU:</b> I <sup>2</sup> C Address Match Wake-Up control 0: Disable 1: Enable – must be cleared by the application program after wake-up This bit should be set to 1 to enable the 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 by the application program after wake-up to ensure correction device operation.
Bit 0	<b>RXAK:</b> I <sup>2</sup> C bus receive acknowledge flag 0: Slave receives acknowledge flag 1: Slave does not receive acknowledge flag The RXAK flag is the receiver acknowledge flag. When the RXAK flag is “0”, it means that a acknowledge signal has been received at the 9 <sup>th</sup> clock, after 8 bits of data have been transmitted. When the slave device is in the transmit mode, the slave device checks the RXAK flag to determine if the master receiver wishes to receive the next byte. The slave transmitter will therefore continue sending out data until the RXAK flag is “1”. When this occurs, the slave transmitter will release the SDA line to allow the master to send a STOP signal to release the 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 SIM interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and SIMTOF bits to determine whether the interrupt source originates from either an address match or the completion of an 8-bit data transfer or the 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 8<sup>th</sup> 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 SIM2~SIM0 bits to “110” and SIMEN bit to “1” in the SIMC0 register 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 SIME interrupt enable bit of the interrupt control register to enable the SIM 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 SIM I<sup>2</sup>C bus interrupt signal will be generated. The next bit following the address, which is the 8<sup>th</sup> 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 9<sup>th</sup> bit. The slave device will also set the status flag HAAS when the addresses match.

As an SIM I<sup>2</sup>C bus interrupt can come from three sources, when the program enters the interrupt subroutine, the HAAS and SIMTOF bits should be examined to see whether the interrupt source has come from either a matching slave address, the completion of a data byte transfer or the 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.

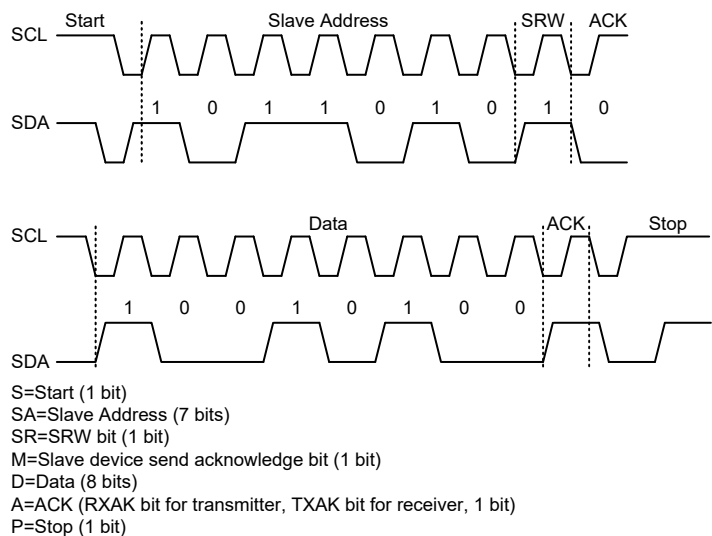
**I<sup>2</sup>C Bus Slave Address Acknowledge Signal**

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 cleared to “0”.

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

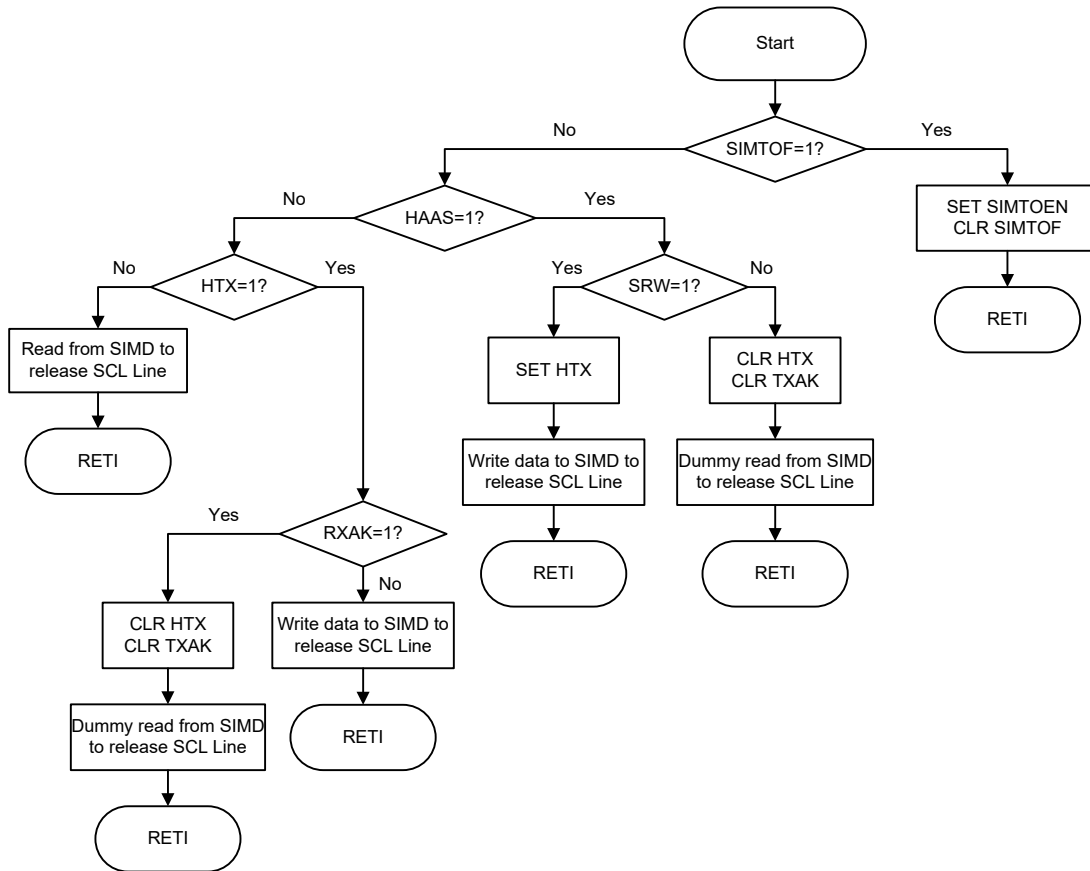
The transmitted data is 8-bit wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8 bits of data, the receiver must transmit an acknowledge signal, level “0”, before it can 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 9<sup>th</sup> 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.



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.

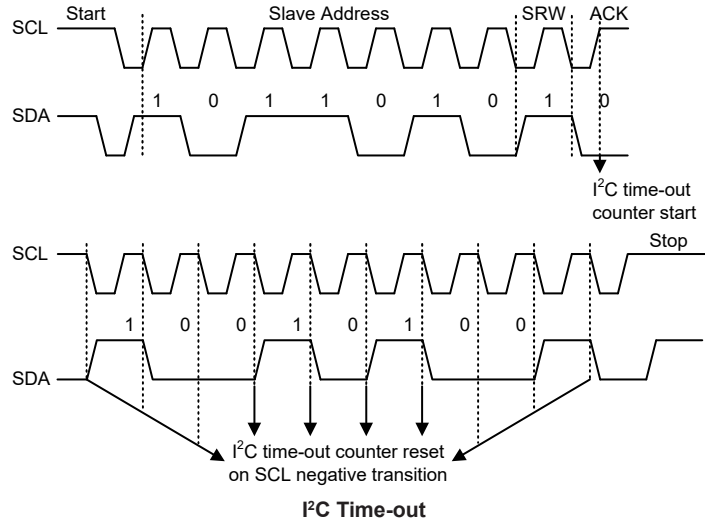
**I<sup>2</sup>C Communication Timing Diagram**



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

**I<sup>2</sup>C Time-out Control**

In order to reduce the I<sup>2</sup>C lockup problem due to reception of erroneous clock sources, a time-out function is provided. If the clock source connected to the I<sup>2</sup>C bus is not received for a while, then the I<sup>2</sup>C circuitry and registers will be reset after a certain time-out period. The time-out counter starts to count on an I<sup>2</sup>C bus “START” & “address match” condition, and is cleared by an SCL falling edge. Before the next SCL falling edge arrives, if the time elapsed is greater than the time-out period specified by the SIMTOC register, then a time-out condition will occur. The time-out function will stop when an I<sup>2</sup>C “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 SIM 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 period selections which can be selected using the SIMTOS5~SIMTOS0 bits in the SIMTOC register. The time-out duration is calculated by the formula:  $((1\sim64)\times(32/f_{SUB}))$ . This gives a time-out period which ranges from about 1ms to 64ms.

• **SIMTOC Register**

Bit	7	6	5	4	3	2	1	0
Name	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7     **SIMTOEN**: SIM I<sup>2</sup>C Time-out control  
 0: Disable  
 1: Enable

Bit 6     **SIMTOF**: SIM I<sup>2</sup>C Time-out flag  
 0: No time-out occurred  
 1: Time-out occurred

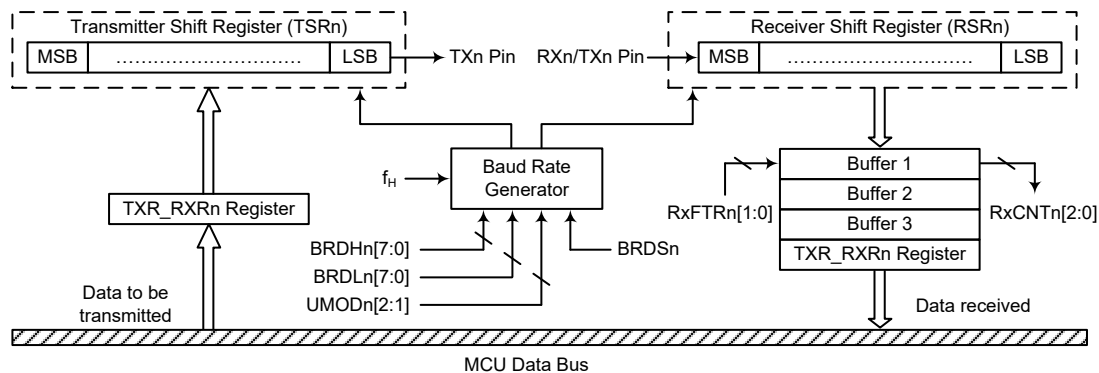
Bit 5~0   **SIMTOS5~SIMTOS0**: SIM I<sup>2</sup>C Time-out period selection  
 I<sup>2</sup>C Time-out clock source is  $f_{SUB}/32$ .  
 I<sup>2</sup>C Time-out period is equal to  $(SIMTOS[5:0]+1)\times(32/f_{SUB})$ .

## UART Interface

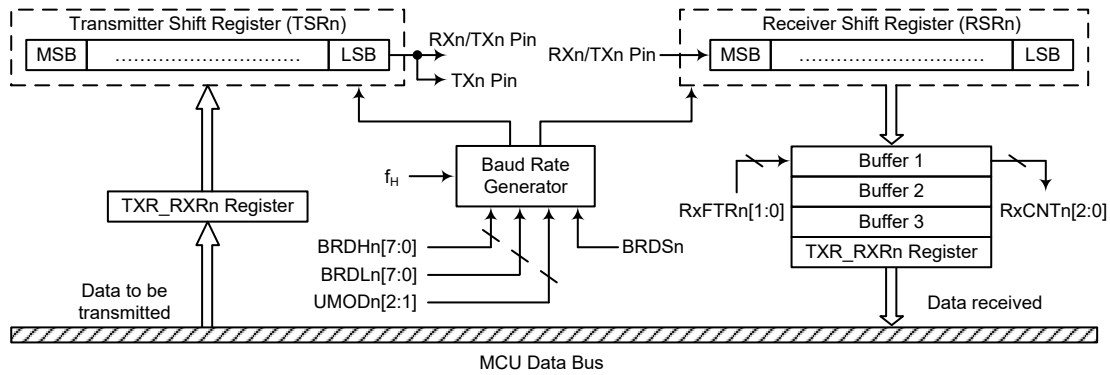
The device contains two integrated full-duplex or half-duplex asynchronous serial communications UART interfaces that enable communication with external devices that contain a serial interface. Each UART function has many features and can transmit and receive data serially by transferring a frame of 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. Each UART function possesses its own internal interrupt which can be used to indicate when a reception occurs or when a transmission terminates.

The integrated UART functions contain the following features:

- Full-duplex or half-duplex (single wire mode), asynchronous communication
- 8 or 9 bits character length
- Even, odd, mark, space or no parity options
- One or two stop bits configurable for receiver
- Two stop bits for transmitter
- Baud rate generator with 16-bit prescaler
- Parity, framing, noise and overrun error detection
- Support for interrupt on address detect (last character bit=1)
- Separately enabled transmitter and receiver
- 4-byte Deep FIFO Receive Data Buffer
- 1-byte Deep FIFO Transmit Data Buffer
- RXn/TXn pin wake-up function
- Transmit and receive interrupts
- Interrupts can be triggered by the following conditions:
  - ♦ Transmitter Empty
  - ♦ Transmitter Idle
  - ♦ Receiver reaching FIFO trigger level
  - ♦ Receiver Overrun
  - ♦ Address Mode Detect



**UARTn Data Transfer Block Diagram – SWMn=0 (n=0~1)**



UARTn Data Transfer Block Diagram – SWMn=1 (n=0~1)

### UART External Pins

To communicate with an external serial interface, the internal UARTn has two external pins known as TXn and RXn/TXn, which are pin-shared with I/O or other pin functions. The TXn and RXn/TXn pin function should first be selected by the corresponding pin-shared function selection register before the UARTn function is used. Along with the UARTENn bit, the TXENn and RXENn 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 TXn or RXn/TXn pin function is disabled by clearing the UARTENn, TXENn or RXENn bit, the TXn or RXn/TXn pin will be set to a floating state. At this time whether the internal pull-high resistor is connected to the TXn or RXn/TXn pin or not is determined by the corresponding I/O pull-high function control bit.

### UART Single Wire Mode

The UARTn function also supports the Single Wire Mode communication which is selected using the SWMn bit in the UnCR3 register. When the SWMn bit is set high, the UARTn function will be in the single wire mode. In the single wire mode, a single RXn/TXn pin can be used to transmit and receive data depending upon the corresponding control bits. When the RXENn bit is set high, the RXn/TXn pin is used as a receiver pin. When the RXENn bit is cleared to zero and the TXENn bit is set high, the RXn/TXn pin will act as a transmitter pin.

It is recommended not to set both the RXENn and TXENn bits high in the single wire mode. If both the RXENn and TXEN bits are set high, the RXENn bit will have the priority and the UARTn 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 TXn pin mentioned in this chapter should be replaced by the RXn/TXn pin to understand the whole UART single wire mode function.

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

### UART Data Transfer Scheme

The UARTn Data Transfer Block Diagram shows the overall data transfer structure arrangement for the UARTn. The actual data to be transmitted from the MCU is first transferred to the TXR\_RXRn register by the application program. The data will then be transferred to the Transmit Shift Register



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

It should be noted that the actual register for data transmission and reception only exists as a single shared register, TXR\_RXR<sub>n</sub>, in the Data Memory.

## UART Status and Control Registers

There are nine control registers associated with the UART function. The SWM<sub>n</sub> bit in the UnCR3 register is used to enable/disable the UART Single Wire Mode. The UnSR, UnCR1, UnCR2, UFCR<sub>n</sub> and RxCNT<sub>n</sub> registers control the overall function of the UART<sub>n</sub>, while the BRDH<sub>n</sub> and BRDL<sub>n</sub> registers control the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR\_RXR<sub>n</sub> data register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
UnSR	PERR <sub>n</sub>	NF <sub>n</sub>	FERR <sub>n</sub>	OERR <sub>n</sub>	RIDLE <sub>n</sub>	RXIF <sub>n</sub>	TIDLE <sub>n</sub>	TXIF <sub>n</sub>
UnCR1	UARTEN <sub>n</sub>	BN0 <sub>n</sub>	PREN <sub>n</sub>	PRTn1	PRTn0	TXBRK <sub>n</sub>	RX8 <sub>n</sub>	TX8 <sub>n</sub>
UnCR2	TXEN <sub>n</sub>	RXEN <sub>n</sub>	STOPS <sub>n</sub>	ADDEN <sub>n</sub>	WAKEN	RIE <sub>n</sub>	TIIE <sub>n</sub>	TEIE <sub>n</sub>
UnCR3	—	—	—	—	—	—	—	SWM <sub>n</sub>
TXR_RXR <sub>n</sub>	D7	D6	D5	D4	D3	D2	D1	D0
BRDH <sub>n</sub>	D7	D6	D5	D4	D3	D2	D1	D0
BRDL <sub>n</sub>	D7	D6	D5	D4	D3	D2	D1	D0
UFCR <sub>n</sub>	—	—	UMODn2	UMODn1	UMODn0	BRDS <sub>n</sub>	RxFTRn1	RxFTRn0
RxCNT <sub>n</sub>	—	—	—	—	—	D2	D1	D0

UART<sub>n</sub> Register List (n=0~1)

### • UnSR Register

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

Bit	7	6	5	4	3	2	1	0
Name	PERR <sub>n</sub>	NF <sub>n</sub>	FERR <sub>n</sub>	OERR <sub>n</sub>	RIDLE <sub>n</sub>	RXIF <sub>n</sub>	TIDLE <sub>n</sub>	TXIF <sub>n</sub>
R/W	R	R	R	R	R	R	R	R
POR	0	0	0	0	1	0	1	1

Bit 7 **PERR<sub>n</sub>**: Parity error flag  
 0: No parity error is detected  
 1: Parity error is detected

The PERR<sub>n</sub> flag is the parity error flag. When this read only flag is “0”, it indicates a parity error has not been detected. When the flag is “1”, it indicates that the parity of the received word is incorrect. This error flag is applicable only if the parity is enabled and the parity type (odd, even, mark or space) is selected. The flag can also be cleared by a software sequence which involves a read to the status register UnSR followed by an access to the TXR\_RXR<sub>n</sub> data register.

- Bit 6      **NFn**: Noise flag  
             0: No noise is detected  
             1: Noise is detected  
 The NFn 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 UARTn has detected noise on the receiver input. The NFn flag is set during the same cycle as the RXIFn flag but will not be set in the case of an overrun. The NFn flag can be cleared by a software sequence which will involve a read to the status register UnSR followed by an access to the TXR\_RXRn data register.
- Bit 5      **FERRn**: Framing error flag  
             0: No framing error is detected  
             1: Framing error is detected  
 The FERRn 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 by a software sequence which will involve a read to the status register UnSR followed by an access to the TXR\_RXRn data register.
- Bit 4      **OERRn**: Overrun error flag  
             0: No overrun error is detected  
             1: Overrun error is detected  
 The OERRn flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is “0”, it indicates that there is no overrun error. When the flag is “1”, it indicates that an overrun error occurs which will inhibit further transfers to the TXR\_RXRn receive data register. The flag is cleared by a software sequence, which is a read to the status register UnSR followed by an access to the TXR\_RXRn data register.
- Bit 3      **RIDLEn**: Receiver status  
             0: Data reception is in progress (Data being received)  
             1: No data reception is in progress (Receiver is idle)  
 The RIDLEn 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 RIDLEn bit is “1” indicating that the UARTn receiver is idle and the RXn/TXn pin stays in logic high condition.
- Bit 2      **RXIFn**: Receive TXR\_RXRn data register status  
             0: TXR\_RXRn data register is empty  
             1: TXR\_RXRn data register has available data and reach Receiver FIFO trigger level  
 The RXIFn flag is the receive data register status flag. When this read only flag is “0”, it indicates that the TXR\_RXRn read data register is empty. When the flag is “1”, it indicates that the TXR\_RXRn read data register contains new data and reaches the Receiver FIFO trigger level. When the contents of the shift register are transferred to the TXR\_RXRn register and reach Receiver FIFO trigger level, an interrupt is generated if RIEn=1 in the UnCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NFn, FERRn, and/or PERRn are set within the same clock cycle. The RXIFn flag is cleared when the UnSR register is read with RXIFn set, followed by a read from the TXR\_RXRn register, and if the TXR\_RXRn register has no data available.
- Bit 1      **TIDLEn**: Transmission idle  
             0: Data transmission is in progress (Data being transmitted)  
             1: No data transmission is in progress (Transmitter is idle)  
 The TIDLEn 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 TXIFn flag is “1” and when there is no transmit data or break character being transmitted. When TIDLEn is equal to “1”, the TXn pin becomes idle with the

pin state in logic high condition. The TIDL<sub>EN</sub> flag is cleared by reading the UnSR register with TIDL<sub>EN</sub> set and then writing to the TXR\_RXR<sub>n</sub> register. The flag is not generated when a data character or a break is queued and ready to be sent.

- Bit 0 **TXIF<sub>n</sub>**: Transmit TXR\_RXR<sub>n</sub> data register status  
 0: Character is not transferred to the transmit shift register  
 1: Character has transferred to the transmit shift register (TXR\_RXR<sub>n</sub> data register is empty)

The TXIF<sub>n</sub> flag is the transmit data register empty flag. When this read only flag is “0”, it indicates that the character is not transferred to the transmitter shift register. When the flag is “1”, it indicates that the transmitter shift register has received a character from the TXR\_RXR<sub>n</sub> data register. The TXIF<sub>n</sub> flag is cleared by reading the UART<sub>n</sub> status register (UnSR) with TXIF<sub>n</sub> set and then writing to the TXR\_RXR<sub>n</sub> data register. Note that when the TXEN<sub>n</sub> bit is set, the TXIF<sub>n</sub> flag will also be set since the transmit data register is not yet full.

• **UnCR1 Register**

The UnCR1 register together with the UnCR2 and UnCR3 registers are the three UART<sub>n</sub> control registers that are used to set the various options for the UART<sub>n</sub> 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	UARTEN <sub>n</sub>	BNO <sub>n</sub>	PREN <sub>n</sub>	PRTn1	PRTn0	TXBRK <sub>n</sub>	RX8 <sub>n</sub>	TX8 <sub>n</sub>
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

“x”: unknown

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

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

When the UART<sub>n</sub> 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<sub>n</sub> is disabled, all error and status flags will be reset. Also the TXEN<sub>n</sub>, RXEN<sub>n</sub>, TXBRK<sub>n</sub>, RXIF<sub>n</sub>, OERR<sub>n</sub>, FERR<sub>n</sub>, PERR<sub>n</sub> and NFN bits as well as the RxCNT<sub>n</sub> register will be cleared to zero, while the TIDL<sub>EN</sub>, TXIF<sub>n</sub> and RIDL<sub>EN</sub> bits will be set high. Other control bits in UnCR1, UnCR2, UnCR3, UFCR<sub>n</sub>, BRDH<sub>n</sub> and BRDL<sub>n</sub> registers will remain unaffected. If the UART<sub>n</sub> is active and the UARTEN<sub>n</sub> 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<sub>n</sub> is re-enabled, it will restart in the same configuration.

- Bit 6 **BNO<sub>n</sub>**: Number of data transfer bits selection  
 0: 8-bit data transfer  
 1: 9-bit data transfer

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

Note that the 9th bit of data if BNO<sub>n</sub>=1, or the 8th bit of data if BNO<sub>n</sub>=0, which is used as the parity bit, does not transfer to RX8<sub>n</sub> or TXR\_RXR<sub>n</sub>.7 respectively when the parity function is enabled.

- Bit 5      **PREn**: Parity function enable control  
             0: Parity function is disabled  
             1: Parity function is enabled  
 This is the parity enable bit. When this bit is equal to “1”, the parity function will be enabled. If the bit is equal to “0”, then the parity function will be disabled. Replace the most significant bit position with a parity bit.
- Bit 4~3    **PRTn1~PRTn0**: Parity type selection bits  
             00: Even parity for parity generator  
             01: Odd parity for parity generator  
             10: Mark parity for parity generator  
             11: Space parity for parity generator  
 These bits are the parity type selection bits. When these bits are equal to 00b, even parity type will be selected. If these bits are equal to 01b, then odd parity type will be selected. If these bits are equal to 10b, then a 1 (Mark) in the parity bit location will be selected. If these bits are equal to 11b, then a 0 (Space) in the parity bit location will be selected.
- Bit 2      **TXBRKn**: Transmit break character  
             0: No break character is transmitted  
             1: Break characters transmit  
 The TXBRKn bit is the Transmit Break Character bit. When this bit is “0”, there are no break characters and the TXn 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 TXBRKn bit is reset.
- Bit 1      **RX8n**: 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 RX8n. The BNO n bit is used to determine whether data transfers are in 8-bit or 9-bit format.
- Bit 0      **TX8n**: 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 TX8n. The BNO n bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• **UnCR2 Register**

The UnCR2 register is the second of the two UARTn control registers and serves several purposes. One of its main functions is to control the basic enable/disable operation of the UARTn Transmitter and Receiver as well as enabling the various UARTn interrupt sources. The register also serves to control the receiver STOP bit number selection, receiver wake-up enable and the address detect enable. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	TXENn	RXENn	STOPSn	ADDENn	WAKEn	RIEn	TIIE n	TEIE n
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      **TXENn**: UARTn Transmitter enabled control  
             0: UARTn transmitter is disabled  
             1: UARTn transmitter is enabled  
 The bit named TXENn 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 TXn pin will be set in a floating state.  
 If the TXENn bit is equal to “1” and the UARTE n bit are also equal to “1”, the transmitter will be enabled and the TXn pin will be controlled by the UARTn. Clearing the TXENn bit during a transmission will cause the data transmission to be aborted

- and will reset the transmitter. If this situation occurs, the TXn pin will be set in a floating state.
- Bit 6     **RXENn**: UARTn Receiver enabled control  
          0: UARTn receiver is disabled  
          1: UARTn receiver is enabled  
  
The bit named RXENn 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 RXn/TXn pin will be set in a floating state. If the RXENn bit is equal to “1” and the UARTENn bit is also equal to “1”, the receiver will be enabled and the RXn/TXn pin will be controlled by the UARTn. Clearing the RXENn bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RXn/TXn pin will be set in a floating state.
- Bit 5     **STOPSn**: Number of Stop bits selection for receiver  
          0: One stop bit format is used  
          1: Two stop bits format is used  
  
This bit determines if one or two stop bits are to be used for receiver. When this bit is equal to “1”, two stop bits are used. If this bit is equal to “0”, then only one stop bit is used. Two stop bits are used for transmitter.
- Bit 4     **ADDENn**: Address detect function enable control  
          0: Address detect function is disabled  
          1: Address detect function is enabled  
  
The bit named ADDENn 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 TXR\_RXRn.7 if BNO<sub>n</sub>=0 or the 9th bit, which corresponds to RX8<sub>n</sub> if BNO<sub>n</sub>=1, has a value of “1”, then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNO<sub>n</sub>. 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     **WAKEN**: RXn/TXn pin wake-up UARTn function enable control  
          0: RXn/TXn pin wake-up UARTn function is disabled  
          1: RXn/TXn pin wake-up UARTn function is enabled  
  
This bit is used to control the wake-up UARTn function when a falling edge on the RXn/TXn pin occurs. Note that this bit is only available when the UARTn clock (f<sub>HT</sub>) is switched off. There will be no RXn/TXn pin wake-up UARTn function if the UARTn clock (f<sub>HT</sub>) exists. If the WAKEN bit is set to 1 as the UARTn clock (f<sub>HT</sub>) is switched off, a UARTn wake-up request will be initiated when a falling edge on the RXn/TXn pin occurs. When this request happens and the corresponding interrupt is enabled, an RXn/TXn pin wake-up UARTn interrupt will be generated to inform the MCU to wake up the UARTn function by switching on the UARTn clock (f<sub>HT</sub>) via the application program. Otherwise, the UARTn function cannot resume even if there is a falling edge on the RXn/TXn pin when the WAKEN bit is cleared to 0.
- Bit 2     **RIEn**: 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 OERRn or receive data available flag RXIFn is set, the UARTn interrupt request flag will be set. If this bit is equal to “0”, the UARTn interrupt request flag will not be influenced by the condition of the OERRn or RXIFn flags.
- Bit 1     **TIEn**: 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 TIDLEn is set, due to a transmitter idle condition, the UARTn interrupt request flag will be set. If this bit is equal to “0”, the UARTn interrupt request flag will not be influenced by the condition of the TIDLEn flag.

Bit 0     **TEIE**n: 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 TXIFn is set, due to a transmitter empty condition, the UARTn interrupt request flag will be set. If this bit is equal to “0”, the UARTn interrupt request flag will not be influenced by the condition of the TXIFn flag.

• **UnCR3 Register**

The UnCR3 register is used to enable the UARTn Single Wire Mode communication. As the name suggests in the single wire mode the UARTn communication can be implemented in one single line, RXn/TXn, together with the control of the RXENn and TXENn bits in the UnCR2 register.

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	SWMn
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1     Unimplemented, read as “0”

Bit 0     **SWM**n: Single Wire Mode enable control  
           0: Disable, the RXn/TXn pin is used as UARTn receiver function only  
           1: Enable, the RXn/TXn pin can be used as UARTn receiver or transmitter function controlled by the RXENn and TXENn bits

Note that when the Single Wire Mode is enabled, if both the RXENn and TXENn bits are high, the RXn/TXn pin will just be used as UARTn receiver input.

• **TXR\_RXRn Register**

The TXR\_RXRn register is the data register which is used to store the data to be transmitted on the TXn pin or being received from the RXn/TXn pin.

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**: UARTn Transmit/Receive Data bit 7 ~ bit 0

• **BRDHn Register**

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

Bit 7~0     **D7~D0**: Baud rate divider high byte  
 The baud rate divider BRDn (BRDHn/BRDLn) defines the UARTn clock divider ratio.  
 $Baud\ Rate = f_{i}/(BRDn + UMODn/8)$   
 $BRDn = 16 \sim 65535$  or  $8 \sim 65535$  depending on BRDSn  
 Note: 1. BRDn value should not be set to less than 16 when BRDSn=0 or less than 8 when BRDSn=1, otherwise errors may occur.  
 2. The BRDLn must be written first and then BRDHn, otherwise errors may occur.  
 3. The BRDHn register should not be modified during data transmission process.

• **BRDLn Register**

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

- Bit 7~0 **D7~D0**: Baud rate divider low byte  
 The baud rate divider BRDn (BRDHn/BRDLn) defines the UARTn clock divider ratio.  
 $Baud\ Rate = f_{ih} / (BRDn + UMODn / 8)$   
 $BRDn = 16 \sim 65535$  or  $8 \sim 65535$  depending on BRDSn  
 Note: 1. BRDn value should not be set to less than 16 when BRDSn=0 or less than 8 when BRDSn=1, otherwise errors may occur.  
 2. The BRDLn must be written first and then BRDHn, otherwise errors may occur.  
 3. The BRDLn register should not be modified during data transmission process.

• **UFCRn Register**

The UFCRn register is the FIFO control register which is used for UARTn modulation control, BRDn range selection and trigger level selection for RXIFn and interrupt.

Bit	7	6	5	4	3	2	1	0
Name	—	—	UMODn2	UMODn1	UMODn0	BRDSn	RxFTRn1	RxFTRn0
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 **UMODn2~UMODn0**: UARTn Modulation Control bits  
 The modulation control bits are used to correct the baud rate of the received or transmitted UARTn signal. These bits determine if the extra UARTn clock cycle should be added in a UARTn bit time. The UMODn2~UMODn0 will be added to internal accumulator for every UARTn bit time. Until a carry to bit 3, the corresponding UARTn bit time increases a UARTn clock cycle.
- Bit 2 **BRDSn**: BRDn range selection  
 0: BRDn range is from 16 to 65535  
 1: BRDn range is from 8 to 65535  
 The BRDSn is used to control the sampling point in a UARTn bit time. If the BRDSn is cleared to zero, the sampling point will be  $BRDn/2$ ,  $BRDn/2 + 1 \times f_{ih}$ , and  $BRDn/2 + 2 \times f_{ih}$  in a UARTn bit time. If the BRDSn is set high, the sampling point will be  $BRDn/2 - 1 \times f_{ih}$ ,  $BRDn/2$ , and  $BRDn/2 + 2 \times f_{ih}$  in a UARTn bit time.  
 Note that the BRDSn bit should not be modified during data transmission process.
- Bit 1~0 **RxFTRn1~RxFTRn0**: Receiver FIFO trigger level (bytes)  
 00: 4 bytes in Receiver FIFO  
 01: 1 or more bytes in Receiver FIFO  
 10: 2 or more bytes in Receiver FIFO  
 11: 3 or more bytes in Receiver FIFO  
 For the receiver these bits define the number of received data bytes in the Receiver FIFO that will trigger the RXIFn bit being set high, an interrupt will also be generated if the RIEn bit is enabled. To prevent OERRn from being set high, the receiver FIFO trigger level can be set to 2 bytes, avoiding an overrun state that cannot be processed by the program in time when more than 4 data bytes are received. After the reset the Receiver FIFO is empty.

• **RxCNTn Register**

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

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

Bit 7~3 Unimplemented, read as “0”

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

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

**Baud Rate Generator**

To setup the speed of the serial data communication, the UARTn function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 16-bit timer, the period of which is determined by two factors. The first of these is the value placed in BRDHn/BRDLn register and the second is the UARTn modulation control bits UMODn2~UMODn0. To prevent accumulated error of the receiver baud rate frequency, it is recommended to use two stop bits for resynchronization after each byte is received. If a baud rate BR is required with UARTn clock  $f_H$ .

$$f_H/BR = \text{Integer Part} + \text{Fractional Part}$$

The integer part is loaded into BRDn (BRDHn/BRDLn). The fractional part is multiplied by 8 and rounded, then loaded into the UMODn bit field below:

$$BRDn = \text{TRUNC}(f_H/BR)$$

$$UMODn = \text{ROUND}[\text{MOD}(f_H/BR) \times 8]$$

Therefore, the actual baud rate is calculated as follows:

$$\text{Baud rate} = f_H / [BRDn + (UMODn/8)]$$

**Calculating the Baud Rate and Error Values**

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

$$\text{From the above formula, the } BRDn = \text{TRUNC}(f_H/BR) = \text{TRUNC}(17.36111) = 17$$

$$\text{The } UMODn = \text{ROUND}[\text{MOD}(f_H/BR) \times 8] = \text{ROUND}(0.36111 \times 8) = \text{ROUND}(2.88888) = 3$$

$$\text{The actual Baud Rate} = f_H / [BRDn + (UMODn/8)] = 230215.83$$

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

**Modulation Control Example**

To get the best-fitting bit sequence for UARTn modulation control bits UMODn2~UMODn0, the following algorithm can be used: Firstly, the fractional part of the theoretical division factor is multiplied by 8. Then the product will be rounded and UMODn2~UMODn0 bits will be filled with the rounded value. The UMODn2~UMODn0 bits will be added to internal accumulator for every UARTn bit time. Until a carry to bit 3, the corresponding UARTn bit time increases a UARTn clock cycle. The following is an example using the fraction 0.36111 previously calculated:  $UMODn[2:0] = \text{ROUND}(0.36111 \times 8) = 011b$ .



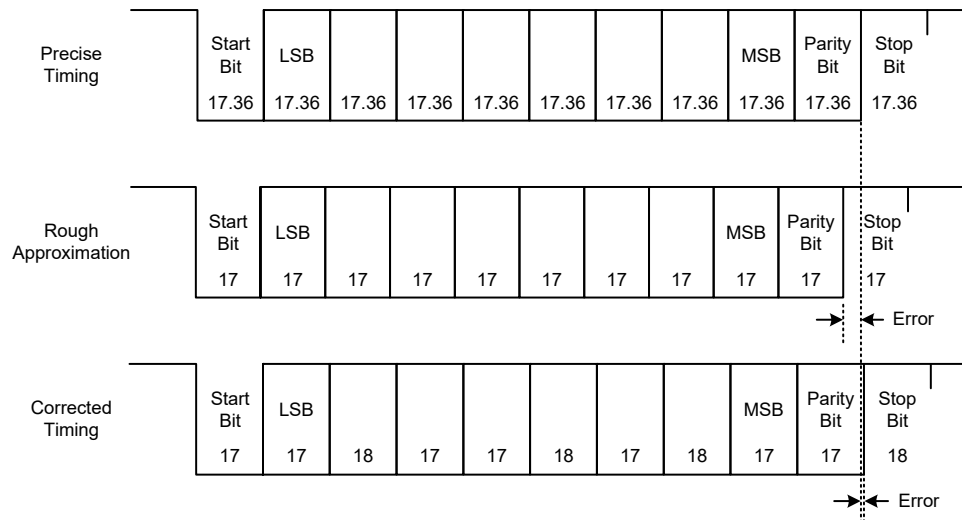
Fraction Addition	Carry to Bit 3	UARTn Bit Time Sequence	Extra UARTn Clock Cycle
0000b+0011b=0011b	No	Start bit	No
0011b+0011b=0110b	No	D0	No
0110b+0011b=1001b	Yes	D1	Yes
1001b+0011b=1100b	No	D2	No
1100b+0011b=1111b	No	D3	No
1111b+0011b=0010b	Yes	D4	Yes
0010b+0011b=0101b	No	D5	No
0101b+0011b=1000b	Yes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

**Baud Rate Correction Example**

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

The following figure shows three different frames:

- The upper frame is the correct one, with a bit-length of 17.36  $f_H$  cycles ( $4000000/230400=17.36$ ).
- The middle frame uses a rough estimate, with 17  $f_H$  cycles for the bit length.
- The lower frame shows a corrected frame using the best fit for the UARTn modulation control bits UMODn2~UMODn0.



**UART Setup and Control**

For data transfer, the UARTn 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 UARTn hardware, and can be setup to be even, odd, mark, space or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits along with the parity are setup by programming the BNO<sub>n</sub>, PRTn1~PRTn0 and PREN<sub>n</sub> bits. The transmitter always uses two stop bits while the receiver uses one or two stop bits which is determined by the STOPS<sub>n</sub> bit. The baud rate used to transmit and receive data is setup using the internal 16-bit baud

rate generator, while the data is transmitted and received LSB first. Although the UARTn 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 UARTn function is controlled using the UARTENn bit in the UnCR1 register. If the UARTENn, TXENn and RXENn bits are set, then these two UARTn pins will act as normal TXn output pin and RXn/TXn input pin respectively. If no data is being transmitted on the TXn pin, then it will default to a logic high value.

Clearing the UARTENn bit will disable the TXn and RXn/TXn pins and allow these two pins to be used as normal I/O or other pin-shared functional pins by configuring the corresponding pin-shared control bits. When the UARTn function is disabled the buffer will be reset to an empty condition, at the same time discarding any remaining residual data. Disabling the UARTn will also reset the error and status flags with bits TXENn, RXENn, TXBRKn, RXIFn, OERRn, FERRn, PERRn and NFn as well as register RxCNTn being cleared while bits TIDLEn, TXIFn and RIDLEn will be set. The remaining control bits in the UnCR1, UnCR2, UnCR3, UFCRn, BRDHn and BRDLn registers will remain unaffected. If the UARTENn bit in the UnCR1 register is cleared while the UARTn is active, then all pending transmissions and receptions will be immediately suspended and the UARTn will be reset to a condition as defined above. If the UARTn is then subsequently re-enabled, it will restart again in the same configuration.

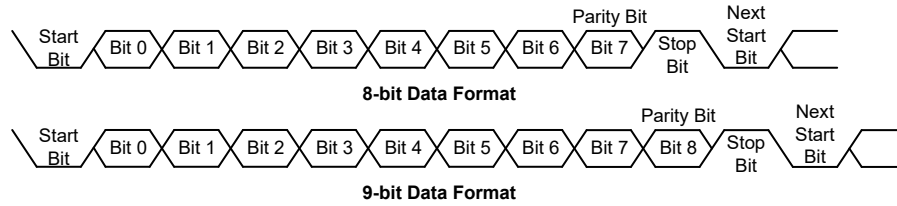
### Data, Parity and Stop Bit Selection

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 UnCR1 and UnCR2 registers. The BNO n bit controls the number of data bits which can be set to either 8 or 9, the PRTn1~PRTn0 bits control the choice of odd, even, mark or space parity, the PRENn bit controls the parity on/off function and the STOPSn bit decides whether one or two stop bits are to be used for the receiver, while the transmitter always uses two stop bits. The following table shows various formats for data transmission. The address bit, which is the MSB of the data byte, identifies the frame as an address character or data if the address detect function is enabled. The number of stop bits, which can be either one or two, is independent of the data length and is only configurable for the receiver. The transmitter uses two stop bits.

Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
<b>Example of 8-bit Data Formats</b>				
1	8	0	0	1 or 2
1	7	0	1	1 or 2
1	7	1	0	1 or 2
<b>Example of 9-bit Data Formats</b>				
1	9	0	0	1 or 2
1	8	0	1	1 or 2
1	8	1	0	1 or 2

**Transmitter Receiver Data Format**

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



## UART Transmitter

Data word lengths of either 8 or 9 bits can be selected by programming the BNO<sub>n</sub> bit in the UnCR1 register. When BNO<sub>n</sub> bit is set, the word length will be set to 9 bits. In this case the 9th bit, which is the MSB, needs to be stored in the TX8<sub>n</sub> bit in the UnCR1 register. At the transmitter core lies the Transmitter Shift Register, more commonly known as the TSR<sub>n</sub>, whose data is obtained from the transmit data register, which is known as the TXR\_RXR<sub>n</sub> register. The data to be transmitted is loaded into this TXR\_RXR<sub>n</sub> register by the application program. The TSR<sub>n</sub> 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<sub>n</sub> can then be loaded with new data from the TXR\_RXR<sub>n</sub> register, if it is available. It should be noted that the TSR<sub>n</sub> register, unlike many other registers, is not directly mapped into the Data Memory area and as such is not available to the application program for direct read/write operations. An actual transmission of data will normally be enabled when the TXEN<sub>n</sub> bit is set, but the data will not be transmitted until the TXR\_RXR<sub>n</sub> register has been loaded with data and the baud rate generator has defined a shift clock source. However, the transmission can also be initiated by first loading data into the TXR\_RXR<sub>n</sub> register, after which the TXEN<sub>n</sub> bit can be set. When a transmission of data begins, the TSR<sub>n</sub> is normally empty, in which case a transfer to the TXR\_RXR<sub>n</sub> register will result in an immediate transfer to the TSR<sub>n</sub>. If during a transmission the TXEN<sub>n</sub> bit is cleared, the transmission will immediately cease and the transmitter will be reset. The TX<sub>n</sub> 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<sub>n</sub> is transmitting data, the data is shifted on the TX<sub>n</sub> pin from the shift register, with the least significant bit LSB first. In the transmit mode, the TXR\_RXR<sub>n</sub> register forms a buffer between the internal bus and the transmitter shift register. It should be noted that if 9-bit data format has been selected, then the MSB will be taken from the TX8<sub>n</sub> bit in the UnCR1 register. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of the BNO<sub>n</sub>, PRTn1~PRTn0 and PREN<sub>n</sub> bits to define the required word length and parity type. Two stop bits are used for the transmitter.
- Setup the BRDH<sub>n</sub>, BRDL<sub>n</sub> registers and UMODn2~UMODn0 bits to select the desired baud rate.
- Set the TXEN<sub>n</sub> bit to ensure that the TX<sub>n</sub> pin is used as a UART<sub>n</sub> transmitter pin.
- Access the UnSR register and write the data that is to be transmitted into the TXR\_RXR<sub>n</sub> register. Note that this step will clear the TXIF<sub>n</sub> bit.

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

It should be noted that when TXIF<sub>n</sub>=0, data will be inhibited from being written to the TXR\_RXR<sub>n</sub> register. Clearing the TXIF<sub>n</sub> flag is always achieved using the following software sequence:

1. A UnSR register access
2. A TXR\_RXR<sub>n</sub> register write execution

The read-only TXIFn flag is set by the UARTn hardware and if set indicates that the TXR\_RXRn register is empty and that other data can now be written into the TXR\_RXRn register without overwriting the previous data. If the TEIE n bit is set then the TXIFn flag will generate an interrupt.

During a data transmission, a write instruction to the TXR\_RXRn register will place the data into the TXR\_RXRn register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the TXR\_RXRn register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIFn bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLE n bit will be set. To clear the TIDLE n bit the following software sequence is used:

1. A UnSR register access
2. A TXR\_RXRn register write execution

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

### Transmitting Break

If the TXBRKn bit is set and the state keeps for a time greater than  $(BRDn+1) \times t_{th}$  while TIDLE n=1, then break characters will be sent on the next transmission. Break character transmission consists of a start bit, followed by  $13 \times N$  '0' bits and stop bits, where  $N=1, 2, \text{etc.}$  If a break character is to be transmitted then the TXBRKn 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 TXBRKn 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 TXBRKn bit, the transmitter will finish transmitting the last break character and subsequently send out two stop bits. The automatic logic highs at the end of the last break character will ensure that the start bit of the next frame is recognized.

### UART Receiver

The UARTn is capable of receiving word lengths of either 8 or 9 bits. If the BNO n bit is set, the word length will be set to 9 bits with the MSB being stored in the RX8n bit of the UnCR1 register. At the receiver core lies the Receive Serial Shift Register, commonly known as the RSRn. The data which is received on the RXn/TXn external input pin is sent to the data recovery block. The data recovery block operating speed is 16 times that of the baud rate, while the main receive serial shifter operates at the baud rate. After the RXn/TXn pin is sampled for the stop bit, the received data in RSRn is transferred to the receive data register, if the register is empty. The data which is received on the external RXn/TXn input pin is sampled three times by a majority detect circuit to determine the logic level that has been placed onto the RXn/TXn pin. It should be noted that the RSRn 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 UARTn receiver is receiving data, the data is serially shifted in on the external RXn/TXn input pin, LSB first. In the read mode, the TXR\_RXRn register forms a buffer between the internal bus and the receiver shift register. The TXR\_RXRn register is a four-byte deep FIFO data buffer, where four bytes can be held in the FIFO while a fifth byte can continue to be received. Note that the application program must ensure that the data is read from TXR\_RXRn before the fifth byte has been completely shifted in, otherwise this fifth byte will be discarded and an overrun error OERRn will be subsequently indicated. For continuous multi-byte data transmission, it is strongly recommended that the receiver uses two stop bits to avoid a receiving error caused by the accumulated error of the receiver baud rate frequency.

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

- Make the correct selection of BNO<sub>n</sub>, PRT<sub>n1</sub>~PRT<sub>n0</sub>, PREN<sub>n</sub> and STOPS<sub>n</sub> bits to define the word length and parity type and number of stop bits.
- Setup the BRDH<sub>n</sub> and BRDL<sub>n</sub> registers and the UMOD<sub>n2</sub>~UMOD<sub>n0</sub> bits to select the desired baud rate.
- Set the RXEN<sub>n</sub> bit to ensure that the RX<sub>n</sub>/TX<sub>n</sub> pin is used as a UART<sub>n</sub> receiver pin.

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

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

- The RXIF<sub>n</sub> bit in the UnSR register will be set when the TXR\_RXR<sub>n</sub> register has data available, the number of the available data bytes can be checked by polling the RxCNT<sub>n</sub> register content.
- When the contents of the shift register have been transferred to the TXR\_RXR<sub>n</sub> register and reach Receiver FIFO trigger level if the RIEN bit is set, then an interrupt will be generated.
- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

The RXIF<sub>n</sub> bit can be cleared using the following software sequence:

1. A UnSR register access
2. A TXR\_RXR<sub>n</sub> register read execution

### Receiving Break

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

- The framing error flag, FERR<sub>n</sub>, will be set.
- The receive data register, TXR\_RXR<sub>n</sub>, will be cleared.
- The OERR<sub>n</sub>, NF<sub>n</sub>, PERR<sub>n</sub>, RIDLE<sub>n</sub> or RXIF<sub>n</sub> 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 UnSR register, otherwise known as the RIDLE<sub>n</sub> flag, will have a zero value. In between the reception of a stop bit and the detection of the next start bit, the RIDLE<sub>n</sub> flag will have a high value, which indicates the receiver is in an idle condition.

### Receiver Interrupt

The read only receive interrupt flag RXIF<sub>n</sub> in the UnSR register is set by an edge generated by the receiver. An interrupt is generated if RIEN=1, when a word is transferred from the Receive Shift

Register, RSRn, to the Receive Data Register, TXR\_RXRn. An overrun error can also generate an interrupt if RIEn=1.

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

## Managing Receiver Errors

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

### Overrun Error – OERRn

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

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

- The OERRn flag in the UnSR register will be set.
- The TXR\_RXRn contents will not be lost.
- The shift register will be overwritten.
- An interrupt will be generated if the RIEn bit is set.

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

The OERRn flag can be cleared by an access to the USR register followed by a read to the TXR\_RXRn register.

### Noise Error – NFn

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, NFn, in the UnSR register will be set on the rising edge of the RXIFn bit.
- Data will be transferred from the Shift register to the TXR\_RXRn register.
- No interrupt will be generated. However this bit rises at the same time as the RXIFn bit which itself generates an interrupt.

Note that the NFn flag is reset by a UnSR register read operation followed by a TXR\_RXRn register read operation.

### Framing Error – FERRn

The read only framing error flag, FERRn, in the UnSR 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 FERRn flag will be set. The FERRn flag and the received data will be recorded in the UnSR and TXR\_RXRn registers respectively, and the flag is cleared in any reset.

### Parity Error – PERRn

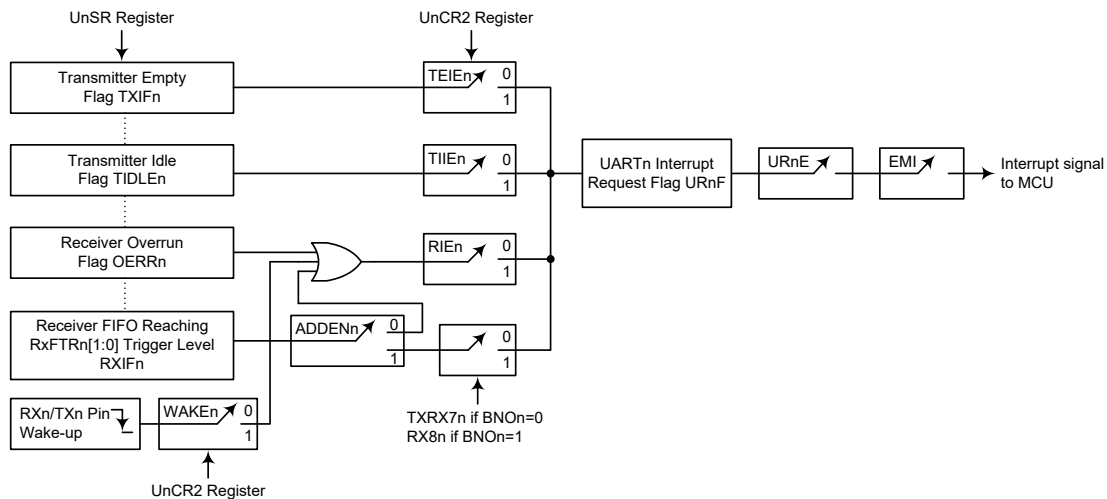
The read only parity error flag, PERRn, in the UnSR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PRENn=1, and if the parity type, odd, even, mark or space is selected. The read only PERRn flag and the received data will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

### UART Interrupt Structure

Several individual UARTn conditions can generate a UARTn interrupt. When these conditions exist, a low pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RXn/TXn pin wake-up. When any of these conditions are created, if the global interrupt enable bit and its corresponding interrupt control bit are enabled and the stack is not full, the program will jump to its corresponding interrupt vector where it can be serviced before returning to the main program. Four of these conditions have the corresponding UnSR register flags which will generate a UARTn interrupt if its associated interrupt enable control bit in the UnCR2 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 UARTn interrupt sources.

The address detect condition, which is also a UARTn interrupt source, does not have an associated flag, but will generate a UARTn interrupt when an address detect condition occurs if its function is enabled by setting the ADDENn bit in the UnCR2 register. An RXn/TXn pin wake-up, which is also a UARTn interrupt source, does not have an associated flag, but will generate a UARTn interrupt if the UARTn clock ( $f_{H1}$ ) source is switched off and the WAKEn and RIEn bits in the UnCR2 register are set when a falling edge on the RXn/TXn pin occurs.

Note that the UnSR 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 UARTn, the details of which are given in the UARTn register section. The overall UARTn interrupt can be disabled or enabled by the related interrupt enable control bits in the interrupt control registers of the microcontroller to decide whether the interrupt requested by the UARTn module is masked out or allowed.



**UARTn Interrupt Structure (n=0~1)**

### Address Detect Mode

Setting the Address Detect Mode bit, ADDEN<sub>n</sub>, in the UnCR2 register, enables this special mode. If this bit is enabled then an additional qualifier will be placed on the generation of a Receiver Data Available interrupt, which is requested by the RXIF<sub>n</sub> flag. If the ADDEN<sub>n</sub> 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 URnE and EMI interrupt enable bits must also be enabled for correct interrupt generation. This highest address bit is the 9th bit if BNO<sub>n</sub>=1 or the 8th bit if BNO<sub>n</sub>=0. If this bit is high, then the received word will be defined as an address rather than data. A Data Available interrupt will be generated every time the last bit of the received word is set. If the ADDEN<sub>n</sub> bit is not enabled, then a Receiver Data Available interrupt will be generated each time the RXIF<sub>n</sub> flag is set, irrespective of the data last bit status. The address detect mode and parity enable are mutually exclusive functions. Therefore if the address detect mode is enabled, then to ensure correct operation, the parity function should be disabled by resetting the parity enable bit PREN<sub>n</sub> to zero.

ADDEN <sub>n</sub>	9th Bit if BNO <sub>n</sub> =1, 8th Bit if BNO <sub>n</sub> =0	UART <sub>n</sub> Interrupt Generated
0	0	√
	1	√
1	0	×
	1	√

**ADDEN<sub>n</sub> Bit Function (n=0~1)**

### UART Power Down and Wake-up

When the UART<sub>n</sub> clock, f<sub>h</sub>, is switched off, the UART<sub>n</sub> will cease to function, all clock sources to the module are shutdown. If the UART<sub>n</sub> clock, f<sub>h</sub>, is off while a transmission is still in progress, then the transmission will be paused until the UART<sub>n</sub> 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 UnSR, UnCR1, UnCR2, UnCR3, UFCR<sub>n</sub>, RxCNT<sub>n</sub> and TXR\_RXR<sub>n</sub> registers, as well as the BRDH<sub>n</sub>/BRDL<sub>n</sub> register will not be affected. It is recommended to make sure first that the UART<sub>n</sub> data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

The UART<sub>n</sub> function contains a receiver RX<sub>n</sub>/TX<sub>n</sub> pin wake-up function, which is enabled or disabled by the WAKEN bit in the UnCR2 register. If this bit, along with the UART<sub>n</sub> enable bit, UARTEN<sub>n</sub>, the receiver enable bit, RXEN<sub>n</sub> and the receiver interrupt bit, RInE, are all set when the UART<sub>n</sub> clock (f<sub>h</sub>) is off, then a falling edge on the RX<sub>n</sub>/TX<sub>n</sub> pin will trigger an RX<sub>n</sub>/TX<sub>n</sub> pin wake-up UART<sub>n</sub> interrupt. Note that as it takes certain system clock cycles after a wake-up, before normal microcontroller operation resumes, any data received during this time on the RX<sub>n</sub>/TX<sub>n</sub> pin will be ignored.

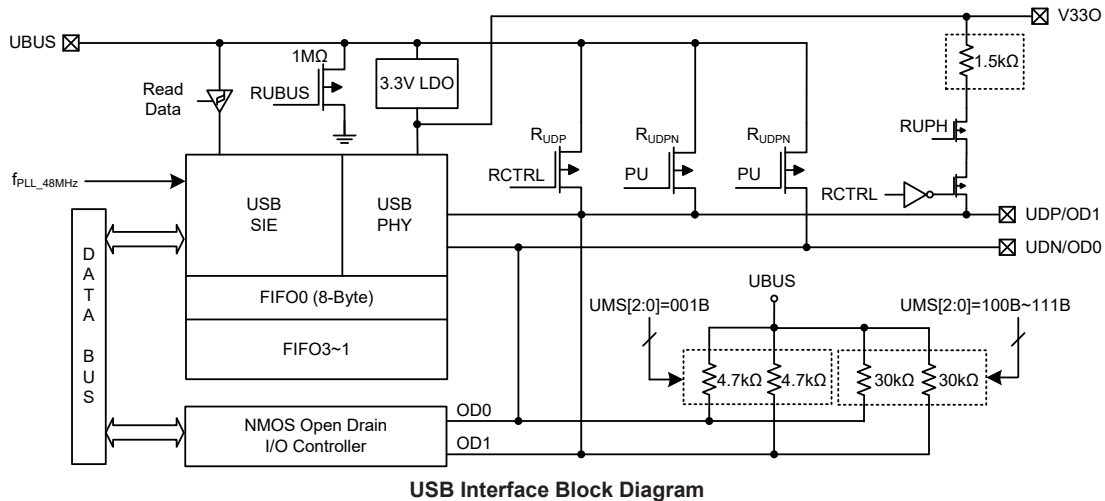
For a UART<sub>n</sub> wake-up interrupt to occur, in addition to the bits for the wake-up being set, the global interrupt enable bit, EMI, and the UART<sub>n</sub> interrupt enable bit, URnE, must be set. If the EMI and URnE bits are not set then only a wake-up event will occur and no interrupt will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



## USB Interface

The USB interface is a 4-wire serial bus that allows communication between a host device and up to 127 max peripheral devices on the same bus. A token based protocol method is used by the host device for communication control. Other advantages of the USB bus include live plugging and unplugging and dynamic device configuration. As the complexity of USB data protocol does not permit comprehensive USB operation information to be provided in this datasheet, the reader should therefore consult other external information for a detailed USB understanding.

The device includes a USB interface function allowing for the convenient design of USB peripheral products.



USB Interface Block Diagram

## USB Interface Operation

To communicate with an external USB host, the internal USB module has the external pins known as UDP and UDN along with the 3.3V regulator output V33O. A Serial Interface Engine (SIE) decodes the incoming USB data stream and transfers it to the correct endpoint buffer memory known as the FIFO. The USB module has 4 endpoints, EP0~EP3. The endpoint 0 has 8-byte size, the endpoint 1 has 16-byte size, the endpoint 2 has 32-byte size and the endpoint 3 has 64-byte size. The endpoint 0 supports the Control transfer while the endpoint 1 ~ endpoint 3 support the Interrupt or Bulk transfer.

## USB Interface Registers

The USB function control is implemented using a series of registers. A series of status registers provide the user with the USB data transfer situation as well as any error conditions. The USB contains its own independent interrupt which can be used to indicate when the USB FIFOs are accessed by the host device or a change of the USB operating conditions including the USB suspend mode, resume event or USB reset occurs.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SYSC	BITWE	USBDIS	RUBUS	UBUSF	—	—	D1	ESDF
USB_STAT	OD1O	OD0O	OD1I	OD0I	SE1	SE0	PU	RUPH
UINT	—	—	—	—	EP3EN	EP2EN	EP1EN	EP0EN
USC	URD	UMS2	UMS1	UMS0	RESUME	URST	RMWK	SUSP
UESR	—	—	—	—	EP3F	EP2F	EP1F	EP0F

Register Name	Bit							
	7	6	5	4	3	2	1	0
UCC	RCTRL	—	JSUSP	SUSP2	USBCKEN	—	EPS1	EPS0
AWR	AD6	AD5	AD4	AD3	AD2	AD1	AD0	WKEN
STL	—	—	—	—	STL3	STL2	STL1	STL0
SIES	NMI	UERR2	UERR1	UERR0	IN	OUT	UFERR	ASET
MISC	LEN0	READY	SETCMD	V33OS	—	CLEAR	TX	REQUEST
SETIO	—	—	—	—	SETIO3	SETIO2	SETIO1	DATATG
FRNUM0	D7	D6	D5	D4	D3	D2	D1	D0
FRNUM1	—	—	—	—	—	D10	D9	D8
FIFO0	D7	D6	D5	D4	D3	D2	D1	D0
FIFO1	D7	D6	D5	D4	D3	D2	D1	D0
FIFO2	D7	D6	D5	D4	D3	D2	D1	D0
FIFO3	D7	D6	D5	D4	D3	D2	D1	D0

**USB Interface Register List**

• **SYSC Register**

Bit	7	6	5	4	3	2	1	0
Name	BITWE	USBDIS	RUBUS	UBUSF	—	—	D1	ESDF
R/W	R/W	R/W	R/W	R	—	—	R/W	R/W
POR	0	0	0	1	—	—	0	x

“x”: unknown

- Bit 7     **BITWE**: Software write permission enable control for certain bits  
           0: Enable – Software can write the URST, UFERR, LEN0 and SETCMD bits  
           1: Disable – Software cannot write the URST, UFERR, LEN0 and SETCMD bits
- Bit 6     **USBDIS**: USB SIE function control  
           0: Enable  
           1: Disable  
           This bit is used to control the USB SIE function. When this bit is set to 1, the USB SIE function will be disabled.
- Bit 5     **RUBUS**: UBUS pin pull low function control  
           0: Enable  
           1: Disable
- Bit 4     **UBUSF**: UBUS pin input status  
           0: Low level  
           1: High level
- Bit 3~2   Unimplemented, read as “0”
- Bit 1     **D1**: Reserved bit, cannot be used and must be fixed at 0
- Bit 0     **ESDF**: ESD issue flag  
           This bit will be set to 1 when there is an ESD issue. It is set by SIE and cleared by software.

• **USB\_STAT Register**

Bit	7	6	5	4	3	2	1	0
Name	OD1O	OD0O	OD1I	OD0I	SE1	SE0	PU	RUPH
R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W
POR	1	1	x	x	0	0	0	1

“x”: unknown

- Bit 7      **OD1O**: Output data on OD1 pin, open drain NMOS output
- Bit 6      **OD0O**: Output data on OD0 pin, open drain NMOS output
- Bit 5      **OD1I**: OD1 pin input status
- Bit 4      **OD0I**: OD0 pin input status
- Bit 3      **SE1**: USB bus SE1 noise indication  
 This bit is used to indicate that the SIE has detected an SE1 noise on the USB bus.  
 This bit is set by SIE and cleared by software.
- Bit 2      **SE0**: USB bus SE0 noise indication  
 This bit is used to indicate that the SIE has detected an SE0 noise on the USB bus.  
 This bit is set by SIE and cleared by software.
- Bit 1      **PU**: UDP/UDN pins pull-high function control  
 0: Disable – no internal pull-high resistor  
 1: Enable – internal 600kΩ pull-high resistor on UDP/UDN pins
- Bit 0      **RUPH**: 1.5kΩ resistor connection control  
 0: No 1.5kΩ resistor is connected between UDP and V33O  
 1: A 1.5kΩ resistor is connected between UDP and V33O

• **UINT Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	EP3EN	EP2EN	EP1EN	EP0EN
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

- Bit 7~4      Unimplemented, read as “0”
- Bit 3      **EP3EN**: USB endpoint 3 interrupt enable control  
 0: Disable  
 1: Enable
- Bit 2      **EP2EN**: USB endpoint 2 interrupt enable control  
 0: Disable  
 1: Enable
- Bit 1      **EP1EN**: USB endpoint 1 interrupt enable control  
 0: Disable  
 1: Enable
- Bit 0      **EP0EN**: USB endpoint 0 interrupt enable control  
 0: Disable  
 1: Enable

• **USC Register**

Bit	7	6	5	4	3	2	1	0
Name	URD	UMS2	UMS1	UMS0	RESUME	URST	RMWK	SUSP
R/W	R/W	R/W	R/W	R/W	R	R/W	R/W	R
POR	1	0	0	0	x	x	x	x

“x”: unknown

- Bit 7**      **URD:** USB reset signal reset function control  
0: USB reset signal cannot reset MCU  
1: USB reset signal will reset MCU
- Bit 6~4**    **UMS2~UMS0:** USB and OD mode selection  
000: No mode available – The V33O output will be floating. The relevant external pins will be in an input floating state.  
001: Open drain output mode – The V33O output will be pulled high to VDD. The relevant external pins will become OD0/OD1 pins with a pull-high resistor respectively connected to VDD.  
01x: USB mode – The V33O function will be enabled. The relevant external pins will be used as the UDP and UDN pins.  
100~111: Undefined, the OD0/OD1 will be in a floating state.
- Bit 3**      **RESUME:** USB resume indication  
0: Resume signal is not asserted or USB device has left the suspend mode  
1: Resume signal is asserted and USB device is going to leave the suspend mode  
This bit is read only. When the resume event occurs, this bit will be set high by SIE and then an interrupt will also be generated to wake up the MCU. In order to detect the suspend state, the MCU should set the USBCKEN bit to 1 and clear the SUSP2 bit to 0. When the USB device leaves the suspend mode, the SUSP bit will be cleared to 0 and then the RESUME bit will also be cleared to 0. The resume signal which causes the MCU to wake up should be noted and taken into consideration when the MCU is detecting the suspend mode.
- Bit 2**      **URST:** USB reset indication  
0: No USB reset event occurs  
1: USB reset event occurs  
This bit is only set by the USB SIE and only cleared by the MCU when the BITWE bit is cleared to zero. When the URST bit is set high, it indicates that a USB reset event has occurred and a USB interrupt will be generated.
- Bit 1**      **RMWK:** USB remote wake-up command  
0: No USB remote wake-up command initiated  
1: Initiate USB remote wake-up command  
The RMWK bit is set to 1 by the MCU to force the USB host leaving the suspend mode. Setting the RMWK bit to 1 will initiate a remote wake-up command. The RMWK bit should be kept high for at least 1 USB clock to make sure that the remote wake-up command is accepted by the SIE.
- Bit 0**      **SUSP:** USB suspend indication  
0: USB leaves the suspend mode  
1: USB enters the suspend mode  
This bit is read only and set to 1 by the SIE to indicate that the USB has entered the suspend mode. The corresponding interrupt will also be generated when the SUSP bit changes from low to high.

• **UESR Register**

The UESR register is the USB endpoint interrupt status register and is used to indicate which endpoint is accessed and to select the USB bus. The endpoint request flags, EPnF, are used to indicate which endpoints are accessed. If an endpoint is accessed, the related endpoint request flag will be set to “1” and the USB interrupt will occur if the USB interrupt is enabled and the stack is not full. When the active endpoint request flag is serviced, the endpoint request flag has to be cleared to “0” by software.

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	EP3F	EP2F	EP1F	EP0F
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	x	x	x	x

“x”: unknown

- Bit 7~4 Unimplemented, read as “0”
- Bit 3 **EP3F**: Endpoint 3 access interrupt request flag  
 0: Not accessed  
 1: Accessed
- Bit 2 **EP2F**: Endpoint 2 access interrupt request flag  
 0: Not accessed  
 1: Accessed
- Bit 1 **EP1F**: Endpoint 1 access interrupt request flag  
 0: Not accessed  
 1: Accessed
- Bit 0 **EP0F**: Endpoint 0 access interrupt request flag  
 0: Not accessed  
 1: Accessed

• **UCC Register**

Bit	7	6	5	4	3	2	1	0
Name	RCTRL	—	JSUSP	SUSP2	USBCKEN	—	EPS1	EPS0
R/W	R/W	—	R	R/W	R/W	—	R/W	R/W
POR	0	—	0	x	0	—	x	x

“x”: unknown

- Bit 7 **RCTRL**: RUDP resistor between UDP and UBUS connection control  
 0: Disable – No R<sub>UDP</sub> resistor is connected between UDP and UBUS lines  
 1: Enable – R<sub>UDP</sub> resistor is connected between UDP and UBUS lines
- Bit 6 Unimplemented, read as “0”
- Bit 5 **JSUSP**: USB J-STATE Suspend mode Indication  
 0: USB interface is not in the J-STATE suspend mode  
 1: USB interface is in the J-STATE suspend mode  
 This bit indicates whether the USB interface is in the J-STATE suspend mode or not.
- Bit 4 **SUSP2**: USB suspend mode current reduction control  
 0: Current reduction is disabled in suspend mode  
 1: Current reduction is enabled in suspend mode  
 The current can be reduced to meet the USB standard specification if this bit is set to 1 when entering the suspend mode.
- Bit 3 **USBCKEN**: USB clock enable control  
 0: Disable  
 1: Enable
- Bit 2 Unimplemented, read as “0”

- Bit 1~0    **EPS1~EPS0**: Endpoint FIFO access selection  
                   00: Endpoint 0 FIFO is selected  
                   01: Endpoint 1 FIFO is selected  
                   10: Endpoint 2 FIFO is selected  
                   11: Endpoint 3 FIFO is selected

• **AWR Register**

The AWR register contains the current address and a remote wake up function control bit. The initial value of AWR is “00H”. Whether the address value extracted from the USB host command is immediately loaded into this register or not is determined by the ASET bit in the SIES register.

Bit	7	6	5	4	3	2	1	0
Name	AD6	AD5	AD4	AD3	AD2	AD1	AD0	WKEN
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 7~1    **AD6~AD0**: USB device address bit 6 ~ bit 0  
 Bit 0      **WKEN**: USB remote wake-up enable control  
                   0: Disable  
                   1: Enable

• **STL Register**

The STL register shows whether the corresponding endpoint has worked properly or not. As soon as an endpoint improper operation occurs, the related bit in the STL register has to be set high by application program. The STL register content will be cleared by a USB reset signal and a setup token event.

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	STL3	STL2	STL1	STL0
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	x	x	x	x

“x”: unknown

- Bit 7~4    Unimplemented, read as “0”  
 Bit 3      **STL3**: USB endpoint 3 FIFO stall indication  
                   0: Endpoint 3 FIFO is not stalled  
                   1: Endpoint 3 FIFO is stalled  
 Bit 2      **STL2**: USB endpoint 2 FIFO stall indication  
                   0: Endpoint 2 FIFO is not stalled  
                   1: Endpoint 2 FIFO is stalled  
 Bit 1      **STL1**: USB endpoint 1 FIFO stall indication  
                   0: Endpoint 1 FIFO is not stalled  
                   1: Endpoint 1 FIFO is stalled  
 Bit 0      **STL0**: USB endpoint 0 FIFO stall indication  
                   0: Endpoint 0 FIFO is not stalled  
                   1: Endpoint 0 FIFO is stalled

• **SIES Register**

The SIES register is used to indicate the present signal state which the SIE receives and also controls whether the SIE changes the device address automatically or not.

Bit	7	6	5	4	3	2	1	0
Name	NMI	UERR2	UERR1	UERR0	IN	OUT	UFERR	ASET
R/W	R/W	R/W	R/W	R/W	R	R	R/W	R/W
POR	x	x	x	x	x	x	x	x

“x”: unknown

- Bit 7 NMI:** NAK token interrupt mask control  
 0: NAK token interrupt is not masked  
 1: NAK token interrupt is masked  
 If this bit is set to 1, the interrupt will not be generated when the device sends an NAK token to the USB host. Otherwise, the endpoint n NAK token interrupt will be generated if the corresponding endpoint interrupt control is enabled when this bit is cleared to 0 and the device sends an NAK token to the USB host.
- Bit 6~4 UERR2~UERR0:** USB SIE error status  
 0xx: No error  
 100: USB PID error  
 101: Bit stuffing error  
 110: CRC error  
 111: Host no response to SIE  
 These bits indicate which kind of error is detected by the USB SIE. These bits are set by the USB SIE and cleared by the application program.
- Bit 3 IN:** IN token indication  
 0: The received token packet is not IN token  
 1: The received token packet is IN token  
 The IN bit is used to indicate whether the current token packet received from the USB host is IN token or not.
- Bit 2 OUT:** OUT token indication  
 0: The received token packet is not OUT token  
 1: The received token packet is OUT token  
 The OUT bit is used to indicate whether the token received from the USB host is OUT token or not except the OUT zero length token. This bit should be cleared to 0 by application program after an OUT data has been read. Note that this bit will also be cleared when the next valid SETUP token is received.
- Bit 1 UFERR:** FIFO access error indication  
 0: No error occurs  
 1: Error occurs  
 This bit is used to indicate whether the USB bus errors, such as CRC error, PID error or bit stuffing error, etc., has occurred or not when the FIFO is accessed. This bit is only set by SIE and only cleared by application program when the BITWE bit is cleared to zero.
- Bit 0 ASET:** Device address update method control  
 0: Device address is immediately updated when an address is written into the AWR register  
 1: Device address is updated after the device IN token data have completely been read by the USB host  
 This bit is used to configure the SIE to automatically change the device address by the value stored in the AWR register. When this bit is set to “1” by software, the SIE will update the device address by the value stored in the AWR register after the USB host has successfully read the data from the device by an IN operation. Otherwise, when this bit is cleared to “0”, the SIE will update the device address immediately after an address is written to the AWR register. Therefore, in order to operate properly, the software has to clear this bit after a next valid SETUP token is received.

• **MISC Register**

Bit	7	6	5	4	3	2	1	0
Name	LEN0	READY	SETCMD	V33OS	—	CLEAR	TX	REQUEST
R/W	R/W	R	R/W	R/W	—	R/W	R/W	R/W
POR	x	x	x	0	—	x	x	x

“x”: unknown

- Bit 7**     **LEN0**: Zero-length packet indication  
           0: The received packet is not zero-length packet  
           1: The received packet is zero-length packet  
 This bit is used to show whether the USB host sends a zero-length packet or not. It is only set by hardware and only cleared by software when the BITWE bit is cleared to zero. It will also be cleared by hardware after the MCU receives a valid USB SETUP token.
- Bit 6**     **READY**: Endpoint FIFO ready indication  
           0: The desired endpoint FIFO is not ready  
           1: The desired endpoint FIFO is ready
- Bit 5**     **SETCMD**: SETUP command indication  
           0: The data in the FIFO is not SETUP token  
           1: The data in the FIFO is SETUP token  
 This bit is only set by hardware and only cleared by application program when the BITWE bit is cleared to zero.
- Bit 4**     **V33OS**: Voltage select  
           0: Internal V330 voltage  
           1: External 3.3V LDO
- Bit 3**     Unimplemented, read as “0”
- Bit 2**     **CLEAR**: FIFO clear function enable control  
           0: Disable – No operation  
           1: Enable – Clear the requested endpoint FIFO  
 This bit is used to clear the requested FIFO even if the corresponding FIFO is not ready. The CLEAR bit should be set to 1 to generate a positive pulse with a pulse width to clear the requested FIFO and then clear this bit to zero. After clearing the FIFO, the USB interface output pipe endpoint can receive new data from the host and input pipe endpoint can transfer new data to the Host.
- Bit 1**     **TX**: Data transfer direction indication  
           0: MCU read data from the USB FIFO  
           1: MCU write data to the USB FIFO  
 This bit defines the data transfer direction between the MCU and USB endpoint FIFO. When the TX bit is set to 1, it means that the MCU wants to write data to the USB endpoint FIFO. After the MCU write operation has completed, this bit has to be cleared to 0 before terminating the FIFO request to indicate the end of the data transfer. For an MCU read operation this bit has to be cleared to 0 to indicate that the MCU wants to read data from the USB endpoint FIFO. Then this bit has to be set to 1 before terminating the FIFO request to indicate the end of the data transfer after an MCU read operation completion.
- Bit 0**     **REQUEST**: FIFO request control  
           0: No request or request completion  
           1: Request desired FIFO  
 This bit is used to request an operation of the desired endpoint FIFO. After selecting the desired endpoint FIFO, the FIFO can be requested by setting this bit high. Then this bit should be cleared to zero after the operation completion.



• **SETIO Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	SETIO3	SETIO2	SETIO1	DATATG
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	1	1	1	0

- Bit 7~4 Unimplemented, read as “0”
- Bit 3 **SETIO3**: Endpoint 3 FIFO control  
 0: Output pipe  
 1: Input pipe
- Bit 2 **SETIO2**: Endpoint 2 FIFO control  
 0: Output pipe  
 1: Input pipe
- Bit 1 **SETIO1**: Endpoint 1 FIFO control  
 0: Output pipe  
 1: Input pipe
- Bit 0 **DATATG**: Data token toggle bit  
 0: DATA0 will be sent first  
 1: DATA1 will be sent first

This bit is used to select the Data token toggle bit. When this bit is cleared to 0, a DATA0 will be sent first in the following IN or OUT data pipe for the requested endpoint FIFO. Otherwise, a DATA1 will be sent first followed by the successive IN or OUT data transfer.

- Note: 1. If the related SETIO<sub>n</sub> bit is set to “1”, the corresponding endpoint FIFO is configured as input pipe for IN token operation. Otherwise, the corresponding endpoint FIFO is configured as output pipe for OUT token operation. The purpose of this function is to avoid the USB host from abnormally sending only an IN token or OUT token and disable the related endpoint.
2. As the USB specification defined, when the USB host sends a “Set Configuration” SETUP token, the Data pipe should send the DATA0 (Data toggle) first. Therefore, when the USB device receives a “Set Configuration” SETUP token, user needs to clear the DATATG bit to zero to ensure that the next data will send a DATA0 first.

Data toggle bit setup procedure:

- (1) Select the desired endpoint by configuring the UCC register.
  - (2) Configure the data token toggle bit by configuring the DATATG bit.
  - (3) Configure the input or output pipe direction by configuring the TX bit in the MISC register.
  - (4) Set the REQUEST bit.
  - (5) Toggle the TX bit.
  - (6) Clear the REQUEST bit.
3. This register will be reset only by power-on reset.

• **FRNUM0 Register**

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

“x”: unknown

- Bit 7~0 **D7~D0**: Frame number D7~D0 for each USB SOF packet

• **FRNUM1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	D10	D9	D8
R/W	—	—	—	—	—	R	R	R
POR	—	—	—	—	—	x	x	x

“x”: unknown

Bit 7~3      Unimplemented, read as “0”

Bit 2~0      **D10~D8**: Frame number D10~D8 for each USB SOF packet

• **FIFO<sub>n</sub> Registers (n=0~3)**

The FIFO<sub>n</sub> Register is used for data transactions storages between the USB device and the USB host. The MCU reads data from or writes data to the FIFO<sub>n</sub> via the specific combination of the corresponding control and selection bits.

Name	Type	POR	Descriptions
FIFO0	R/W	xxxx xxxx	Endpoint 0 Data Pipe
FIFO1	R/W	xxxx xxxx	Endpoint 1 Data Pipe
FIFO2	R/W	xxxx xxxx	Endpoint 2 Data Pipe
FIFO3	R/W	xxxx xxxx	Endpoint 3 Data Pipe

“x”: unknown

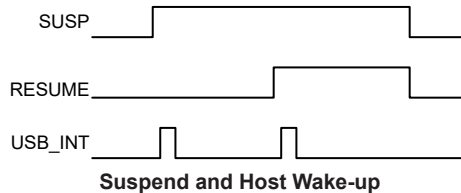
**USB Suspend Mode and Wake-Up**

**USB Suspend Mode**

If there is no signal on the USB bus for over 3ms, the USB device will enter the suspend mode. The Suspend flag, SUSP, in the USC register will then be set high and a USB interrupt will be generated to indicate that the device should jump to the suspend state to meet the requirements of the USB suspend current specification. In order to meet the requirements of the suspend current, the software should disable the USB clock by clearing the USBCKEN bit to “0”. The suspend mode current can be further decreased by setting the SUSP2 bit in the UCC register.

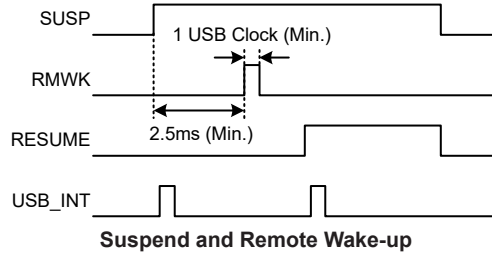
**USB Host Wake-up**

When the resume signal is asserted by the USB host, the device will be woken up by the USB interrupt and the RESUME bit in the USC register will be set. To ensure correct device operation, the application program should set the USBCKEN bit high and the USB host will start to communicate with the USB device. Then the SUSP2 bit will be cleared low together with the RESUME bit when the USB device actually leaves the suspend mode. Therefore, when the device detects the suspend bit, SUSP2, the resume bit, RESUME, should be monitored and taken into consideration.



### USB Remote Wake-up

As the USB device has a remote wake-up function, the USB device can wake up the USB host by sending a remote wake-up pulse which is generated by setting the RMWK bit high. Once the USB host receives a remote wake-up signal from the USB device, the host will send a resume signal to device.



### USB Interrupts

Several USB conditions can generate a USB interrupt. When one of these conditions exists, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are the USB suspended, USB resumed, USB reset and USB endpoint FIFO access events. When the USB interrupt caused by any of these conditions occurs, if the corresponding interrupt control is enabled and the stack is not full, the program will jump to the corresponding interrupt vector where it can be serviced before returning to the main program.

For the USB Endpoint FIFO access event, there are the corresponding indication flags to indicate which endpoint FIFO is accessed. As the Endpoint FIFO access flag is set, it will generate a USB interrupt if the associated Endpoint FIFO pipe and interrupt control are both enabled. The Endpoint FIFO access flags should be cleared by the application program. As the USB suspended or USB resume condition occurs, the corresponding indication flag, known as SUSP and RESUME bits, will be set and a USB interrupt will directly generate without enabling the associated interrupt control bit. The SUSP and RESUME bits are read only and set or cleared by the USB SIE. For a USB interrupt occurred to be serviced, in addition to the bits for the corresponding interrupt enable control in USB module being set, the global interrupt enable control and the related interrupt enable control bits in the host MCU must also be set. If these bits are not set, then no interrupt will be serviced.

In addition, each USB start of frame packet will generate an interrupt if the global interrupt enable control and the related interrupt enable control bit SOFE are both set. Refer to the Interrupts section for more detailed information.

## LCD Driver

For large volume applications, which incorporate an LCD in their design, the use of a custom display rather than a more expensive character based display reduces costs significantly. However, the corresponding COM and SEG signals required, which vary in both amplitude and time, to drive such a custom display require many special considerations for proper LCD operation to occur. This device contains an LCD Driver function, which with their internal LCD signal generating circuitry and various options will automatically generate these time and amplitude varying signals to provide a means of direct driving and easy interfacing to a range of custom LCDs.

This device includes a wide range of options to enable LCD displays of various types to be driven. The table shows the range of options available across the device range.

Driver No.	Duty	Bias Level	Bias Type	Waveform Type
36×4	1/4	1/3	R or C	A or B
34×6	1/6	1/3	R or C	A or B
32×8	1/8	1/3	R	A or B
32×8	1/8	1/4	R	A or B

**LCD Driver Output Selection**

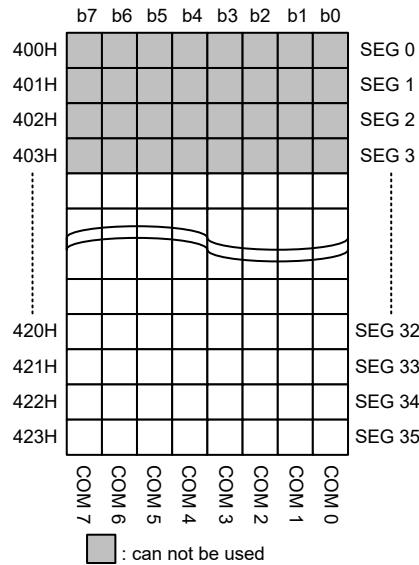
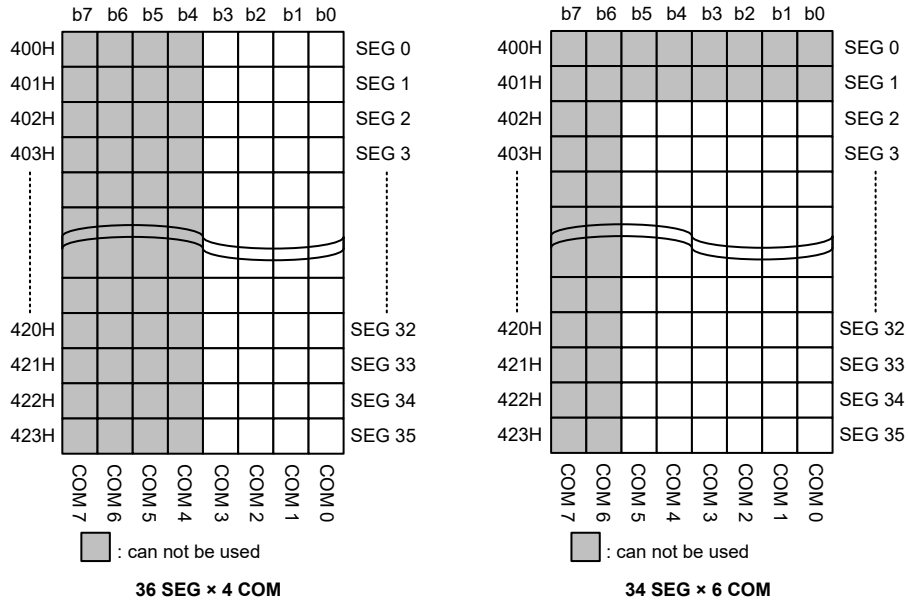
## LCD Display Memory

An area of Data Memory is especially reserved for use for the LCD display data. This data area is known as the LCD Memory. Any data written here will be automatically read by the internal display driver circuits, which will in turn automatically generate the necessary LCD driving signals. Therefore any data written into this Memory will be immediately reflected into the actual display connected to the microcontroller.

This device provides an area of embedded data memory for the LCD display. The area is located at 00H to 23H in Sector 4 of the Data Memory. The LCD display memory can be read and written to by indirect addressing mode using MP1L/MP1H and MP2L/MP2H, or by direct addressing mode using the corresponding extended instructions. If using the indirect addressing to access the Display Memory therefore requires first that Sector 4 is selected by writing a value of 04H to MP1H or MP2H. After this, the memory can then be accessed by using indirect addressing through the use of MP1L or MP2L. With Sector 4 selected, then using MP1L/MP2L to read or write to the memory area, from 00H to 23H, will result in operations to the LCD memory.

When data is written into the display data area, it is automatically read by the LCD driver which then generates the corresponding LCD driving signals. To turn the display on or off, a “1” or a “0” is written to the corresponding bit of the display memory, respectively. The figure illustrates the mapping between the display memory and LCD pattern for the device.

The unimplemented LCD RAM bits cannot be used as general purpose RAM for application. For example, if the LCD duty is selected as 1/4 duty (4-COM), the COM b4~b7 will be read as 0 only.



**LCD Memory Map**

### LCD Clock Source

The LCD clock source is the internal clock signal,  $f_{SUB}$ , divided by 8, using an internal divider circuit. The  $f_{SUB}$  internal clock is supplied by either the LIRC or LXT oscillator, the choice of which is determined by the FSS bit in the SCC register. For proper LCD operation, this arrangement is provided to generate an ideal LCD clock source frequency of 4kHz.

$f_{SUB}$ Clock Source	LCD Clock Frequency
LIRC	4kHz
LXT	4kHz

**LCD Clock Source**

## LCD Registers

Control Registers in the Data Memory, are used to control the various setup features of the LCD Driver. There are three control registers for the LCD function, LCDC0, LCDCP and LCDC2.

The TYPE bit in the LCDC0 register is used to select whether Type A or Type B LCD control signals are used. The RCT bit in the LCDC0 register is used to select whether R type or C type LCD drive bias. Bits LCDP1 and LCDP0 in the LCDC0 register are used to select the power source to supply the C type LCD panel with the correct bias voltages. Bits LCDIS1 and LCDIS0 in the LCDC0 register are used to select the internal bias current to supply the R type LCD panel with the correct bias voltages. A choice to best match the LCD panel used in the application can be selected also to minimise bias current. The LCDEN bit in the LCDC0 register, which provides the overall LCD enable/disable function, will only be effective when this device is in the FAST, SLOW or IDLE Mode. If this device is in the SLEEP Mode then the display will always be disabled.

The LCDPR bit in the LCDCP register is used to select the PLCD pin or the internal charge pump regulator to supply the power for the R type LCD COMs and SEGs pins. Bits CPVS1 and CPVS0 in the same register are used to select an appropriate charge pump output voltage level for the R type LCD.

The LCDC2 register is used for the C type LCD pump clock divider as well as LCD duty and bias selection.

Register Name	Bit							
	7	6	5	4	3	2	1	0
LCDC0	TYPE	RCT	LCDP1	LCDP0	—	LCDIS1	LCDIS0	LCDEN
LCDCP	—	—	—	—	LCDPR	—	CPVS1	CPVS0
LCDC2	LCDPCK2	LCDPCK1	LCDPCK0	—	—	DTYC1	DTYC0	BIAS

**LCD Register List**

### • LCDC0 Register

Bit	7	6	5	4	3	2	1	0
Name	TYPE	RCT	LCDP1	LCDP0	—	LCDIS1	LCDIS0	LCDEN
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 **TYPE**: LCD waveform type selection

0: Type A  
1: Type B

Bit 6 **RCT**: R or C type selection

0: R type  
1: C type

When this bit is clear to 0, the LCDP[1:0] bits should be fixed at “00” and the power source is from the PLCD pin. The C1/C2/V2 pins are pin-shared with I/O or other functions, when the RCT=1, selected I/O or other pin-shared functions will interference with the C1/C2/V2 functions.

Bit 5~4 **LCDP1~LCDP0**: C type LCD power source selection

00: From external PLCD/V1/V2 pin input  
01: From internal  $V_C=V_{REFIN}$   
10: From internal  $V_B=3V$   
11: From internal  $V_A=V_{DD}$

The  $V_{REFIN}$  is an internal reference voltage with an approximate level of 1.0V.

Bit 3 Unimplemented, read as “0”

Bit 2~1 **LCDIS1~LCDIS0**: LCD bias current selection for R type LCD ( $V_A=V_{PLCD}=V_{DD}$ , 1/3 bias)  
 00: 25 $\mu$ A  
 01: 50 $\mu$ A  
 10: 100 $\mu$ A  
 11: 200 $\mu$ A

When using the C type LCD, these bits should be fixed at 00.

Bit 0 **LCDEN**: LCD enable control  
 0: Disable  
 1: Enable

In the FAST, SLOW or IDLE mode, the LCD on/off function can be controlled by this bit. In the SLEEP mode, the LCD is always off.

• **LCDCP Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	LC DPR	—	CPVS1	CPVS0
R/W	—	—	—	—	R/W	—	R/W	R/W
POR	—	—	—	—	0	—	0	0

Bit 7~4 Unimplemented, read as “0”

Bit 3 **LC DPR**: R type LCD power selection  
 0: PLCD pin  
 1: R type internal charge pump

This bit is only available for R type LCD applications. When the LC DPR bit is cleared to 0, the R type LCD power will be derived from the PLCD pin and the internal charge pump circuit will be disabled. This internal charge pump will also be disabled when the C type LCD driver is selected by setting the RCT bit to 1 or the LCD driver is disabled by clearing the LCDEN bit to 0.

Bit 2 Unimplemented, read as “0”

Bit 1~0 **CPVS1~CPVS0**: Charge pump output voltage selection for R type  
 00: 3.3V  
 01: 3.0V  
 10: 2.7V  
 11: 4.5V

• **LCDC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	LC DPCK2	LC DPCK1	LC DPCK0	—	—	DTYC1	DTYC0	BIAS
R/W	R/W	R/W	R/W	—	—	R/W	R/W	R/W
POR	0	0	0	—	—	0	0	0

Bit 7~5 **LC DPCK2~LC DPCK0**: C type LCD charge pump clock divider selection  
 000: 250Hz ( $f_{SUB}/128$ )  
 001: 500Hz ( $f_{SUB}/64$ )  
 010: 1kHz ( $f_{SUB}/32$ )  
 011: 2kHz ( $f_{SUB}/16$ )  
 100: 4kHz ( $f_{SUB}/8$ )  
 101: 8kHz ( $f_{SUB}/4$ )  
 110: 16kHz ( $f_{SUB}/2$ )  
 111: 16kHz ( $f_{SUB}/2$ )

These frequency options are figured out based on the LCD clock source of 32kHz.

Bit 4~3 Unimplemented, read as “0”

- Bit 2~1    **DTYC1~DTYC0**: LCD duty selection  
00: 1/4 duty – COM0~COM3 used  
01: 1/6 duty – COM0~COM5 used  
10: 1/8 duty – COM0~COM7 used, for R type only  
11: Unimplemented
- The unused COM pins can be configured as normal I/O or other pin-shared functions using the corresponding pin-shared selection register.
- Bit 0    **BIAS**: LCD bias selection  
0: 1/3 bias  
1: 1/4 bias, for R type only

### LCD Voltage Source and Biasing

The time and amplitude varying signals generated by the LCD Driver function require the generation of several voltage levels for their operation. The device can have either R type or C type biasing selected via the RCT bit. Selecting the C type biasing will enable C type internal charge pump circuitry.

#### R Type Biasing

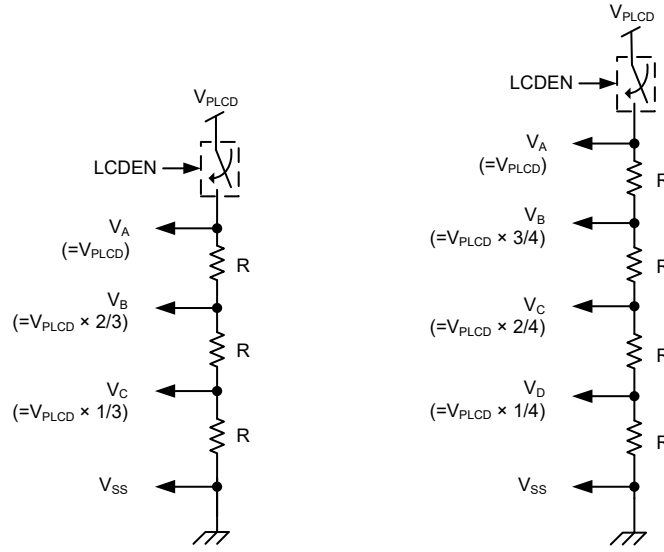
For R type biasing an external LCD voltage source must be supplied on pin PLCD or internal charge pump regulator, selected by the LCDPR bit in the LCDCP register, to generate the internal biasing voltages. The source on the PLCD pin could be the microcontroller power supply  $V_{DD}$  or some other voltage sources. There are four kinds of the internal charge pump voltage output, managed by the CPVS[1:0] bits in the LCDCP register.

For the R type 1/3 bias scheme, four voltage levels  $V_{SS}$ ,  $V_A$ ,  $V_B$  and  $V_C$  are utilised. The voltage  $V_A$  is equal to  $V_{PLCD}$ . The voltage  $V_B$  is equal to  $V_{PLCD} \times 2/3$  while the voltage  $V_C$  is equal to  $V_{PLCD} \times 1/3$ .

For the R type 1/4 bias scheme, five voltage levels  $V_{SS}$ ,  $V_A$ ,  $V_B$ ,  $V_C$  and  $V_D$  are utilised. The voltage  $V_A$  is equal to  $V_{PLCD}$ . The voltage  $V_B$  is equal to  $V_{PLCD} \times 3/4$ , the voltage  $V_C$  is equal to  $V_{PLCD} \times 2/4$  while the voltage  $V_D$  is equal to  $V_{PLCD} \times 1/4$ .

Different values of internal bias current can be selected using the LCDIS1~LCDIS0 bits in the LCDC0 register. The VMAX pin should be connected to the PLCD or VDD pin which provides the maximum voltage.





**R Type 1/3 Bias**

**R Type 1/4 Bias**

- Note: 1. The DC path will be switched off when the LCD is disabled.  
 2. When LCDPR=1, the PLCD pin should externally connect a 4.7μF capacitor; when LCDPR=0, the PLCD pin does not require an external capacitor.

**R Type Bias Voltage Generation**

**C Type Biasing**

For C type biasing the LCD voltage source can be supplied on the external pin PLCD, V1 or V2 or derived from the internal power source to generate the required biasing voltages. The C type bias voltage source is selected using the LCDP1 and LCDP0 bits in the LCDC0 register.

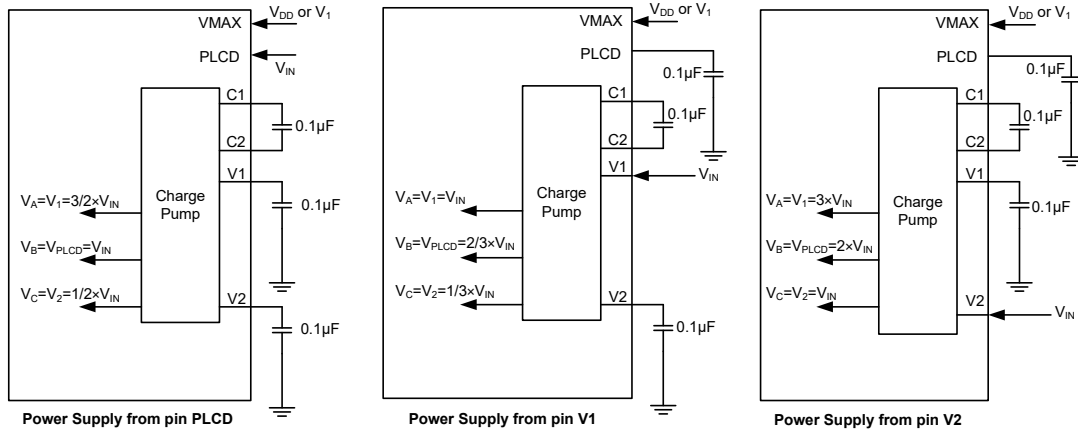
When the LCD voltage source is from the PLCD pin, the C type biasing scheme uses its internal charge pump circuit, which can generate voltages higher than what is supplied on PLCD. This feature is useful in applications where the microcontroller supply voltage is less than the supply voltage required by the LCD. The charge pump clock divider is selected using the LCDPCK2~LCDPCK0 bits in the LCDC2 register. An additional charge pump capacitor must also be connected between pins C1 and C2 to generate the necessary voltage levels.

For C type 1/3 bias scheme, whether the LCD power is selected from external pin or internal voltage, four voltage levels  $V_{SS}$ ,  $V_A$ ,  $V_B$  and  $V_C$  are utilised. These bias voltages have different levels depending upon different LCD power supply schemes.

LCD Power Supply		$V_A$ Voltage	$V_B$ Voltage	$V_C$ Voltage
External Power Supply	$V_{IN}$ from V1 pin	$V_{IN}$	$2/3 \times V_{IN}$	$1/3 \times V_{IN}$
	$V_{IN}$ from PLCD pin	$3/2 \times V_{IN}$	$V_{IN}$	$1/2 \times V_{IN}$
	$V_{IN}$ from V2 pin	$3 \times V_{IN}$	$2 \times V_{IN}$	$V_{IN}$
Internal Power Supply	$V_A$ ( $V_A = V_{DD}$ )	$V_{DD}$	$2/3 \times V_{DD}$	$1/3 \times V_{DD}$
	$V_B$ ( $V_B = 3V$ )	$3/2 \times V_B$	$V_B$	$1/2 \times V_B$
	$V_C$ ( $V_C = V_{REFIN}$ ) (Note)	$3 \times V_{REFIN}$	$2 \times V_{REFIN}$	$V_{REFIN}$

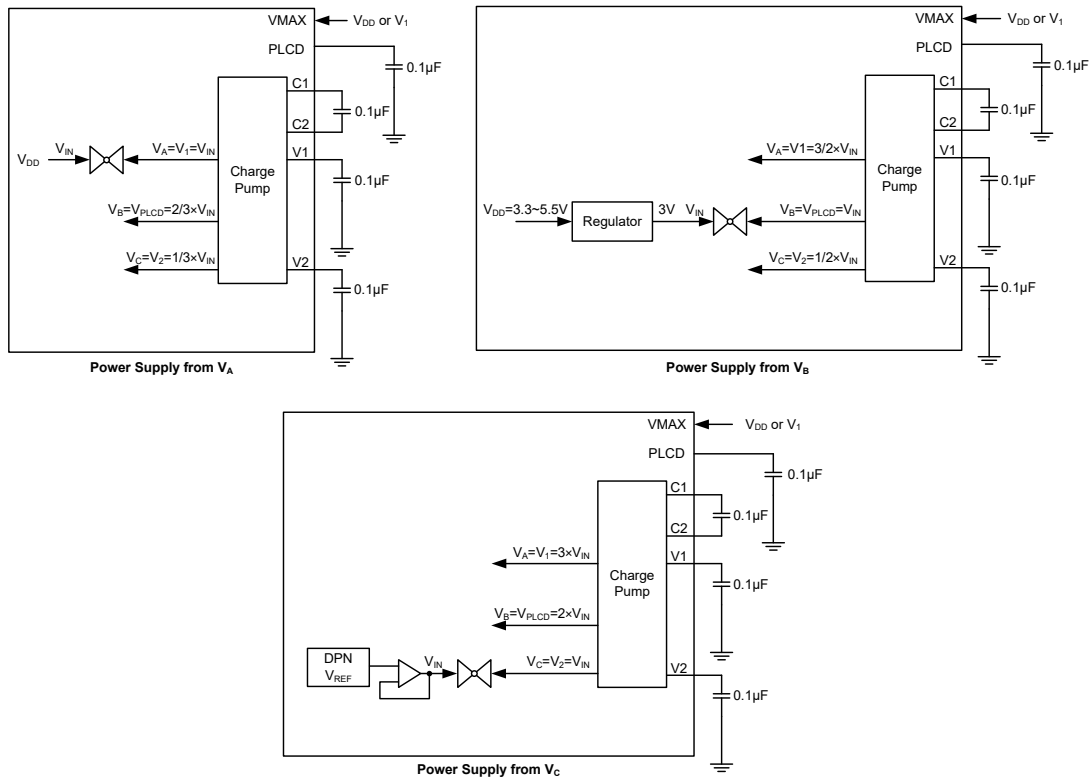
Note: The  $V_{REFIN}$  is an internal reference voltage with an approximate level of 1.0V.

**C Type Bias Power Supply Scheme**



Note: The pin VMAX must be connected to the maximum voltage to prevent from the pad leakage.

**C Type Bias External Power Supply Configuration – 1/3 Bias**



Note: The pin VMAX must be connected to the maximum voltage to prevent from the pad leakage.

**C Type Bias Internal Power Supply Configuration – 1/3 Bias**

The connection to the VMAX pin depends upon the voltage that is applied to the PLCD pin. It is extremely important to ensure that these charge pump generated internal voltages do not exceed the maximum  $V_{DD}$  voltage of 5.5V.

Condition	VMAX Connection
$V_{DD} > V_{PLCD} \times 1.5$	Connect VMAX to VDD
Otherwise	Connect VMAX to V1

**C Type Bias VMAX Pin Connection**

### LCD Reset Status

The LCD has an internal reset function. Clearing the LCDEN bit to zero will reset the LCD function. When the LCDEN bit is set to 1 to enable the LCD driver and then an MCU reset occurs, the LCD driver will be reset and the COM and SEG outputs will be in a floating state during the MCU reset duration. The reset operation will take a time of  $t_{RSTD}+t_{SST}$ . Refer to the System Start Up Time Characteristics for  $t_{RSTD}$  and  $t_{SST}$  details.

MCU Reset	LCDEN	LCD Reset	COM & SEG Voltage Level
No	1	No	Normal Operation
No	0	Yes	Low
Yes	x	Yes	Floating

Note: 1. The Watchdog time-out reset in the IDLE or SLEEP Mode is excluded from the MCU Reset conditions.

2. "x": Don't care.

### LCD Reset Status

### LCD Driver Output

The number of COM and SEG outputs supplied by the LCD driver, the biasing and waveform type selections are dependent upon how the LCD control bits are programmed. The bias type, whether C type or R type is also selected by the RCT bit.

The nature of Liquid Crystal Displays require that only AC voltages can be applied to their pixels as the application of DC voltages to LCD pixels may cause permanent damage. For this reason the relative contrast of an LCD display is controlled by the actual RMS voltage applied to each pixel, which is equal to the RMS value of the voltage on the COM pin minus the voltage applied to the SEG pin. This differential RMS voltage must be greater than the LCD saturation voltage for the pixel to be on and less than the threshold voltage for the pixel to be off.

The requirement to limit the DC voltage to zero and to control as many pixels as possible with a minimum number of connections requires that both a time and amplitude signal is generated and applied to the application LCD. These time and amplitude varying signals are automatically generated by the LCD driver circuits in the microcontroller. What is known as the duty determines the number of common lines used, which are also known as backplanes or COMs. For example, the duty is 1/4 and equates to a COM number of 4, therefore defines the number of time divisions within each LCD signal frame. Two types of signal generation are also provided, known as Type A and Type B, the required type is selected via the TYPE bit in the LCDC0 register. Type B offers lower frequency signals, however lower frequencies may introduce flickering and influence display clarity.

**R & C Type, 4-COM, 1/3 Bias**

**LCD Display Off Mode**

COM0 ~ COM3

All segment outputs

**Normal Operation Mode**

1 Frame

COM0

COM1

COM2

COM3

All segments are OFF

COM0 side segments are ON

COM1 side segments are ON

COM2 side segments are ON

COM3 side segments are ON

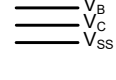
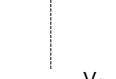
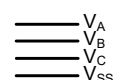
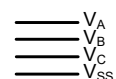
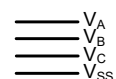
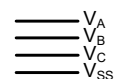
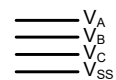
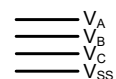
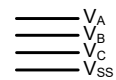
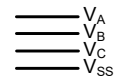
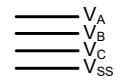
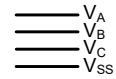
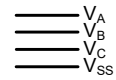
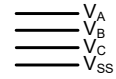
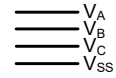
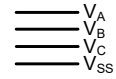
COM0,1 side segments are ON

COM0,2 side segments are ON

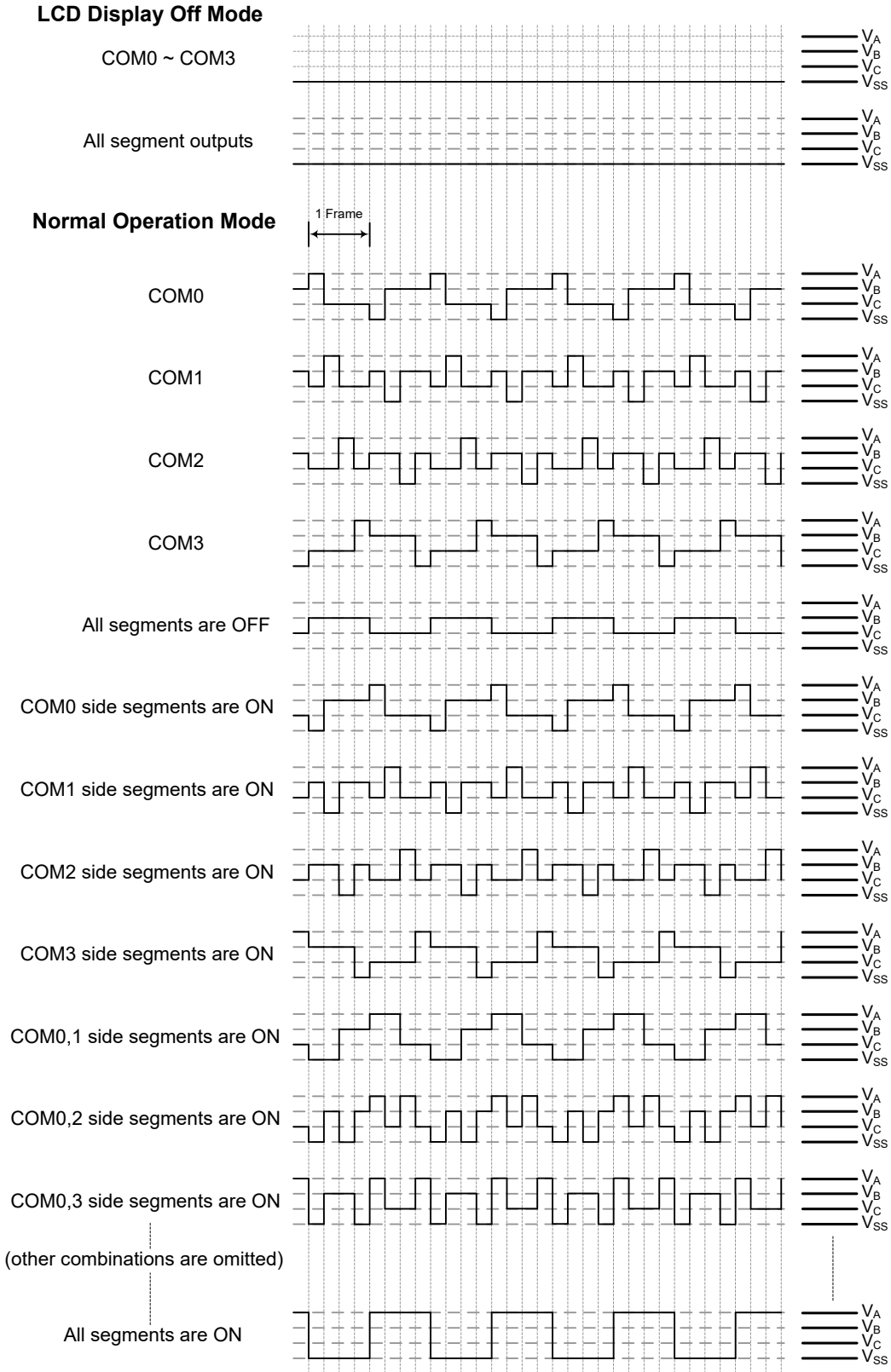
COM0,3 side segments are ON

(other combinations are omitted)

All segments are ON

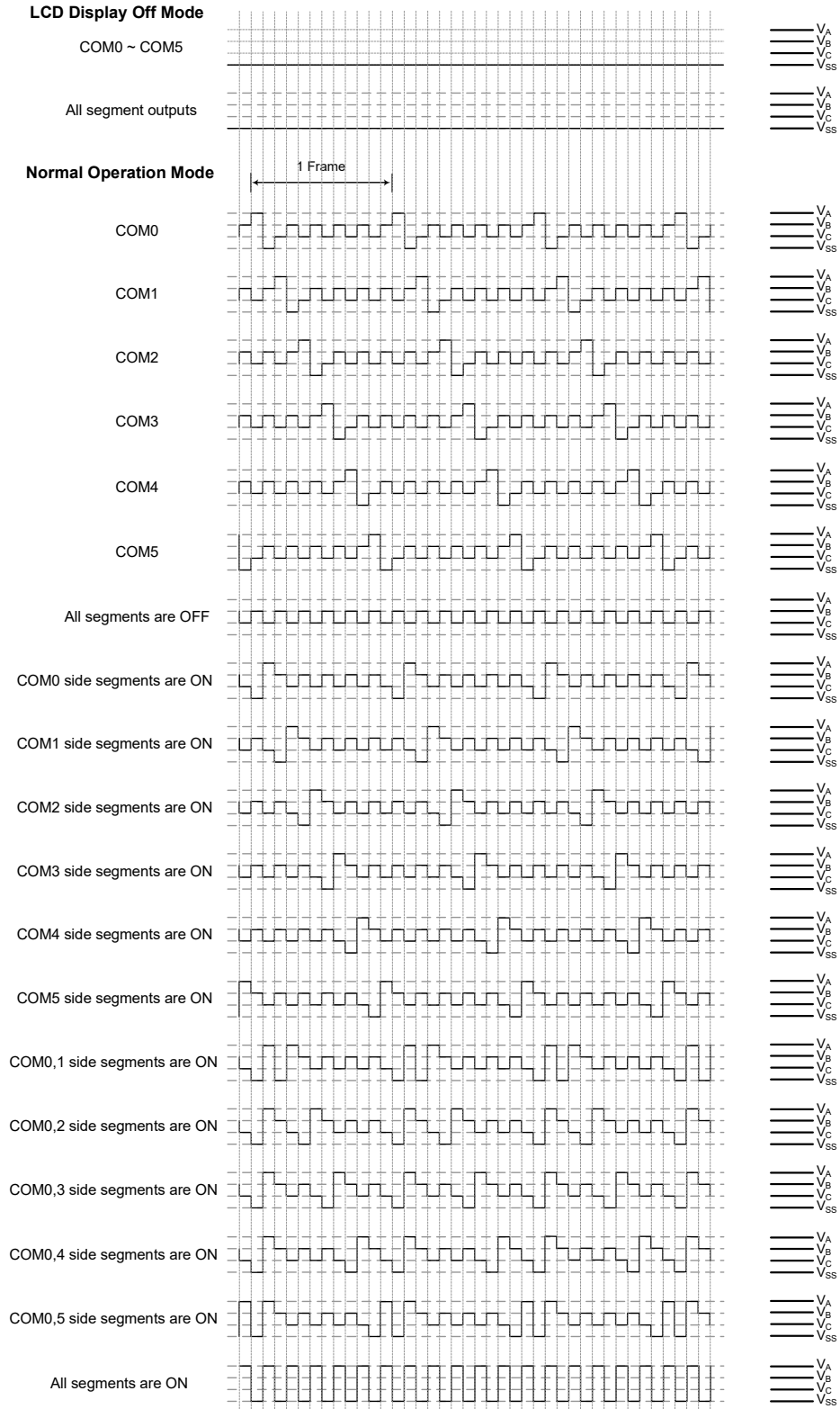


**LCD Driver Output – Type A, 1/4 Duty, 1/3 Bias**



LCD Driver Output – Type B, 1/4 Duty, 1/3 Bias

**R & C Type, 6-COM, 1/3 Bias**

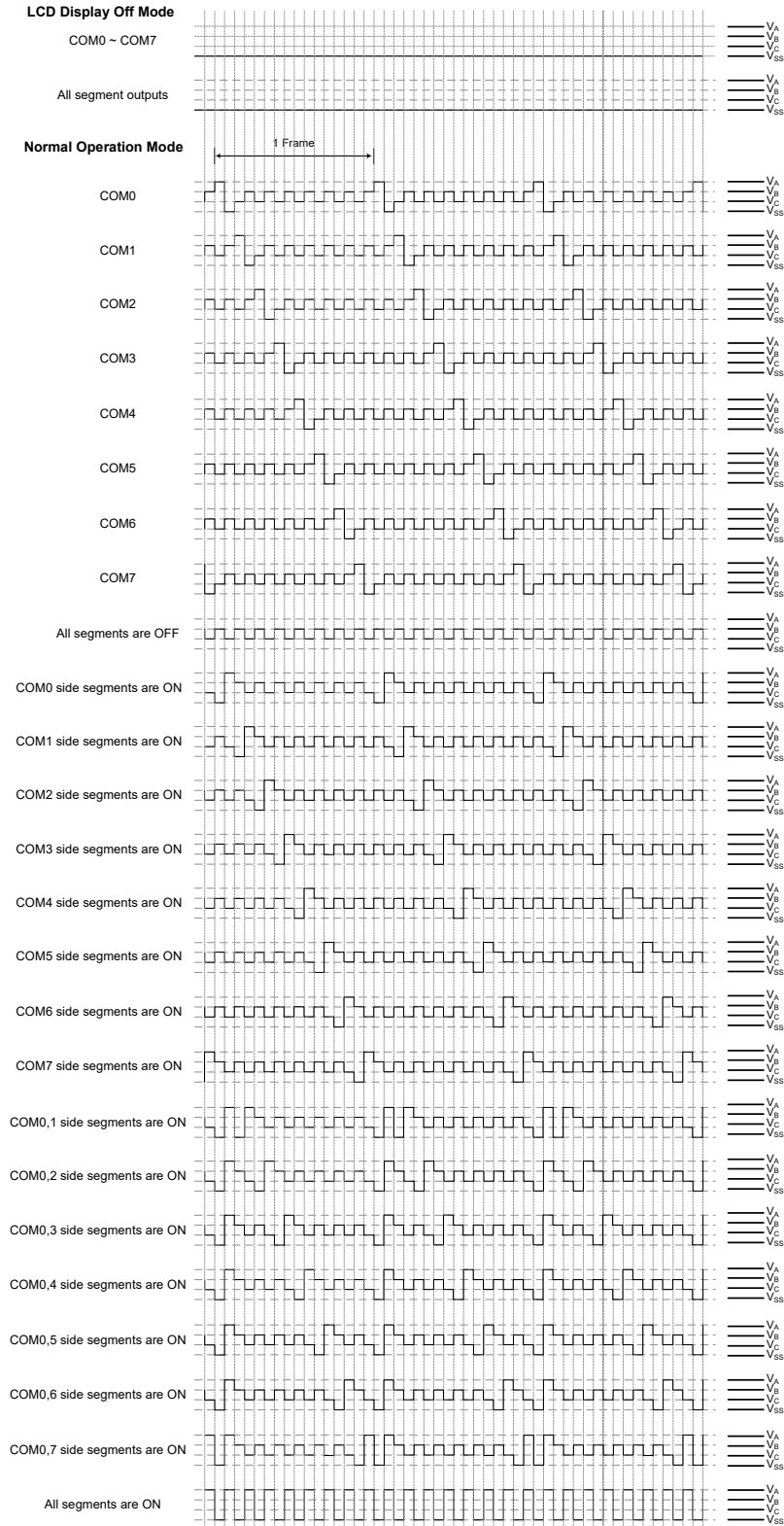


**LCD Driver Output – Type A, 1/6 Duty, 1/3 Bias**



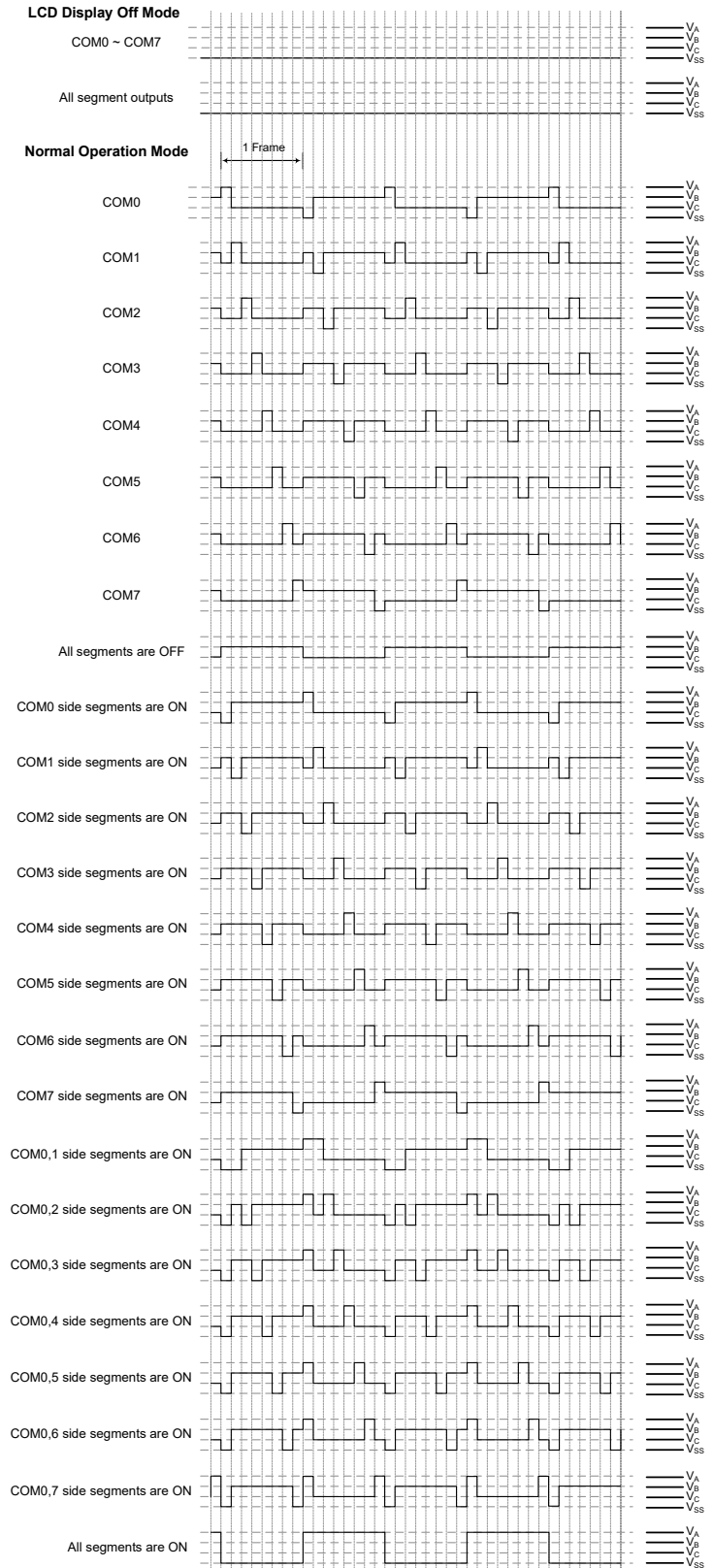
**LCD Driver Output – Type B, 1/6 Duty, 1/3 Bias**

**R Type, 8-COM, 1/3 Bias**



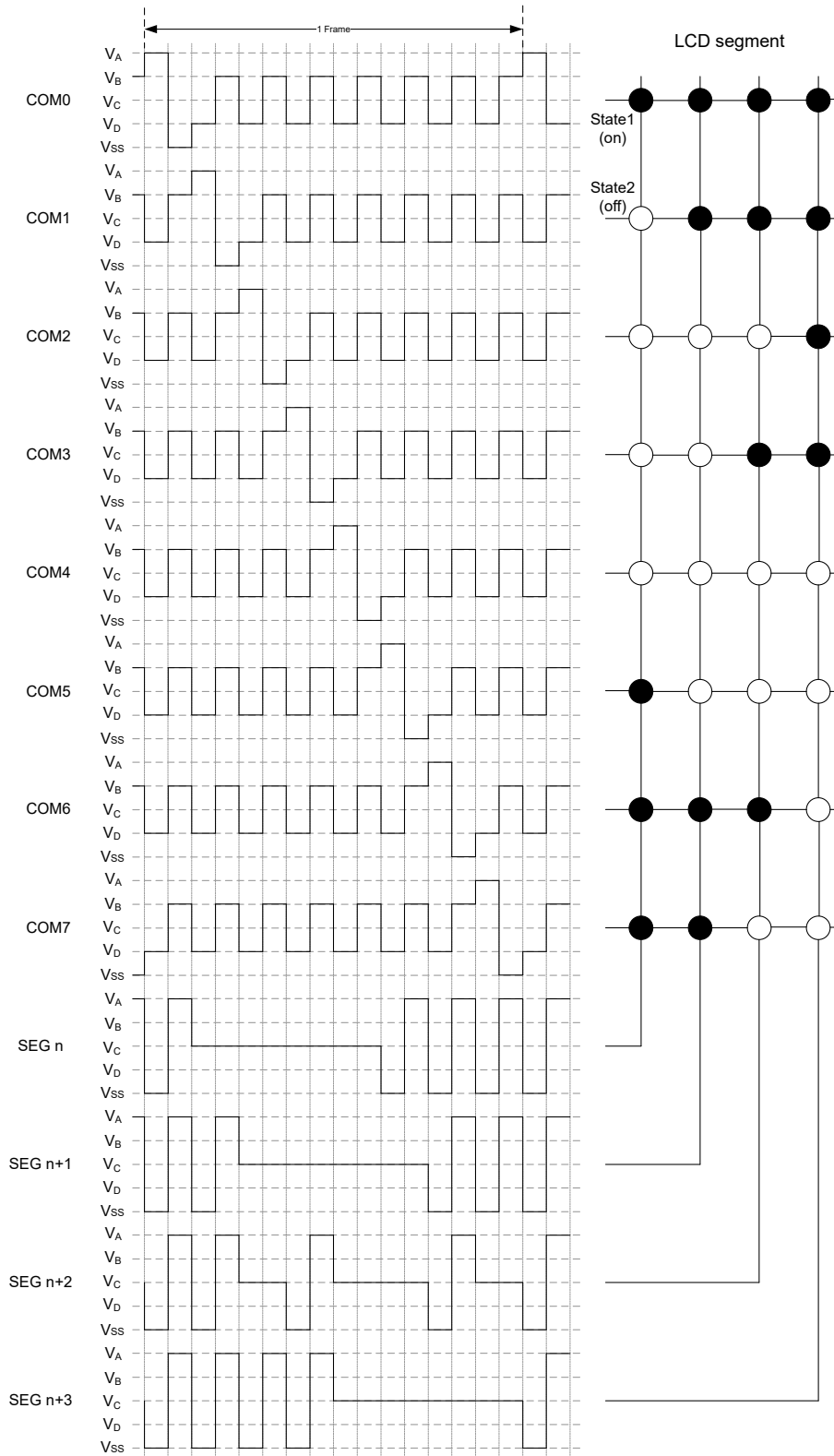
**LCD Driver Output – Type A, 1/8 Duty, 1/3 Bias**



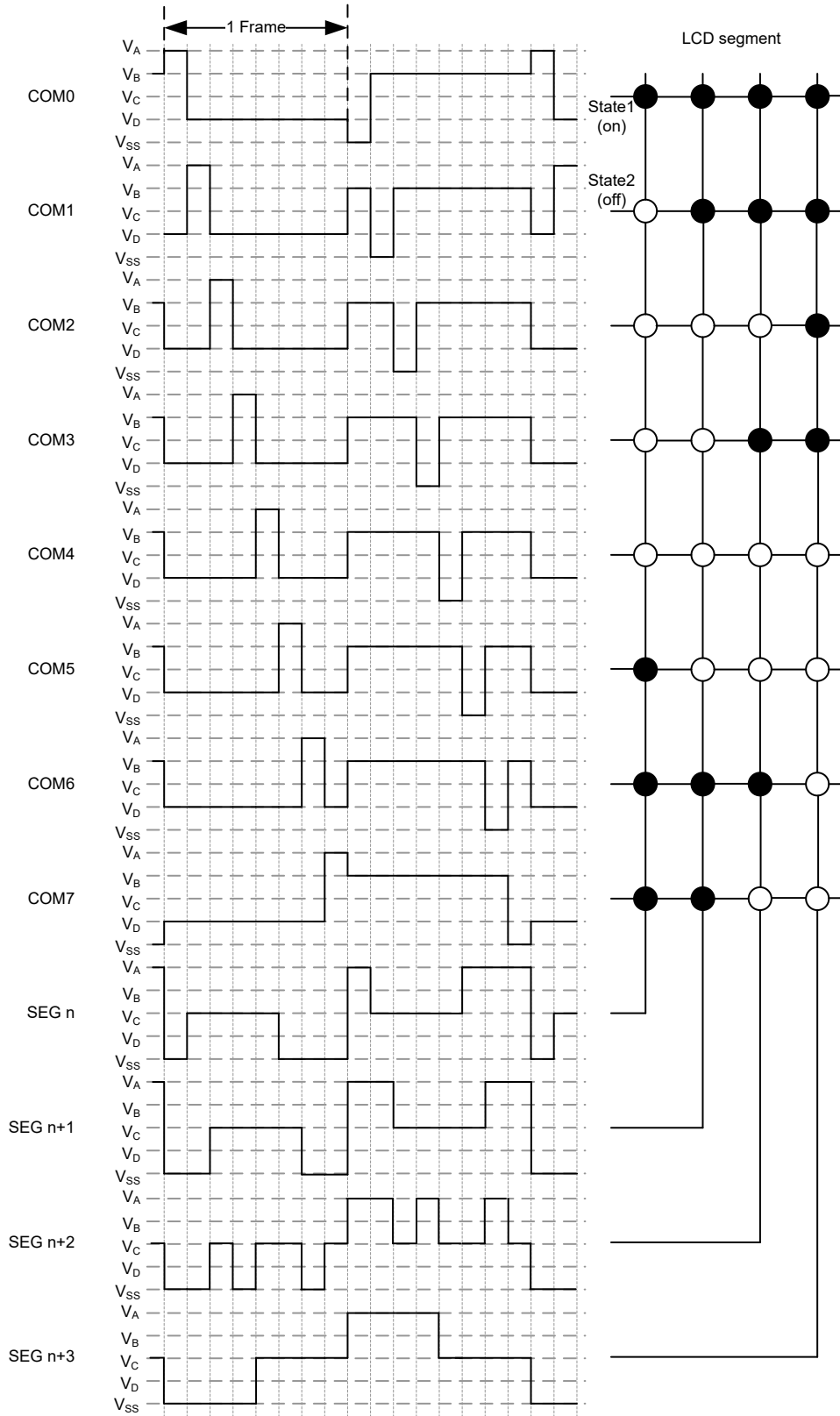


**LCD Driver Output – Type B, 1/8 Duty, 1/3 Bias**

R Type, 8-COM, 1/4 Bias



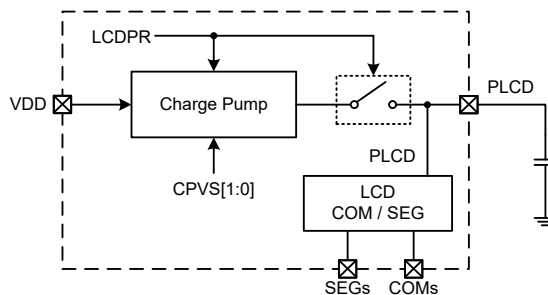
LCD Driver Output – Type A, 1/8 Duty, 1/4 Bias



**LCD Driver Output – Type B, 1/8 Duty, 1/4 Bias**

### LCD Charge Pump

For the R type LCD, the COMs and SEGs pins can be powered up by the external PLCD pin or internal charge pump circuit which is determined by the LCDPR bit in the LCDCP register. When the LCDPR bit is set low, the LCD driver power is supplied by the external PLCD pin. If the LCDPR bit is set high, the LCD driver power is supplied by the internal charge pump circuit. There are four charge pump output voltage levels which are selected by the CPVS1~CPVS0 bits in the LCDCP register. If the internal charge pump circuit is used, an external 4.7μF capacitor should be connected to the external PLCD pin for output voltage stability. In addition, when using the C type LCD, the LCDPR bit should be fixed at 0.



**R Type LCD Driver Charge Pump Circuit**

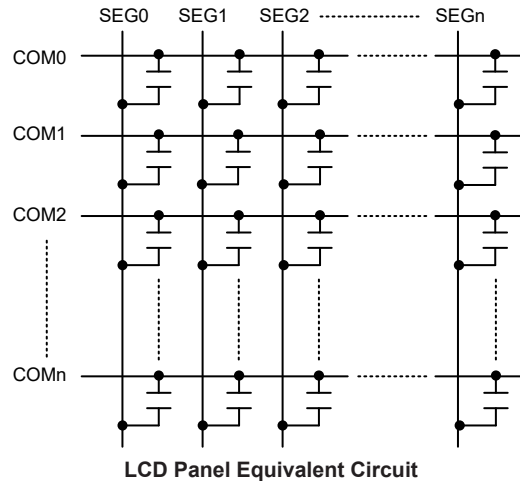
### Programming Considerations

Certain precautions must be taken when programming the LCD. One of these is to ensure that the LCD Memory is properly initialised after the microcontroller is powered on. Like the General Purpose Data Memory, the contents of the LCD Memory are in an unknown condition after power-on. As the contents of the LCD Memory will be mapped into the actual display, it is important to initialise this memory area into a known condition soon after applying power to obtain a proper display pattern.

Consideration must also be given to the capacitive load of the actual LCD used in the application. As the load presented to the microcontroller by LCD pixels can be generally modeled as mainly capacitive in nature, it is important that this is not excessive, a point that is particularly true in the case of the COM lines which may be connected to many LCD pixels. The accompanying diagram depicts the equivalent circuit of the LCD.

One additional consideration that must be taken into account is what happens when the microcontroller enters the IDLE or SLEEP Mode. The LCDEN control bit in the LCDC0 register permits the display to be powered off to reduce power consumption. If this bit is zero, the driving signals to the display will cease, producing a blank display pattern but reducing any power consumption associated with the LCD.

After Power-on, note that as the LCDEN bit is cleared to zero, the display function will be disabled.



## Low Voltage Detector – LVD

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

### LVD Register

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

#### • LVDC Register

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

Bit 7~6 **TLVD1~TLVD0**: Minimum low voltage width to interrupt time ( $t_{LVD}$ )

00:  $(1\sim2) \times t_{LIRC}$

01:  $(3\sim4) \times t_{LIRC}$

10:  $(7\sim8) \times t_{LIRC}$

11:  $(1\sim2) \times t_{LIRC}$

Bit 5 **LVDO**: LVD Output flag

0: No Low Voltage Detected

1: Low Voltage Detected

- Bit 4      **LV DEN**: Low Voltage Detector Enable control  
            0: Disable  
            1: Enable
- Bit 3      **VB GEN**: Bandgap Voltage Output Enable control  
            0: Disable  
            1: Enable

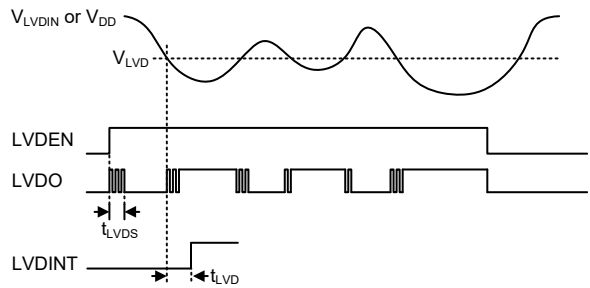
Note that the Bandgap circuit is enabled when the LVD or LVR function is enabled or when the VB GEN bit is set high.

- Bit 2~0    **VLVD2~VLVD0**: LVD Voltage selection  
            000:  $V_{LVDIN} \leq 1.23V$   
            001: 2.2V  
            010: 2.4V  
            011: 2.7V  
            100: 3.0V  
            101: 3.3V  
            110: 3.6V  
            111: 4.0V

When the VLVD2~VLVD0 bits are set to 000B, the LVD function will be implemented by comparing the LVDIN pin input voltage with the LVD reference voltage of 1.23V. When the VLVD2~VLVD0 bits are set to any other value except 000B, the LVD function will operate by comparing the  $V_{DD}$  voltage level with the LVD reference voltage with a specific voltage value which is generated by the internal LVD circuit.

**LVD Operation**

The Low Voltage Detector function operates by comparing the power supply voltage,  $V_{DD}$ , or the LVDIN pin input voltage, with a pre-specified voltage level stored in the LVDC register. This has a range of between 1.23V and 4.0V. When the power supply voltage,  $V_{DD}$ , or the LVDIN pin input voltage, falls below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. When the device is in the SLEEP mode, the low voltage detector will be disabled even if the LV DEN bit is high. After enabling the Low Voltage Detector, a time delay  $t_{LVDS}$  should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the  $V_{DD}$  or  $V_{LVDIN}$  voltage may rise and fall rather slowly, at the voltage nears that of  $V_{LVD}$ , there may be multiple bit LVDO transitions.

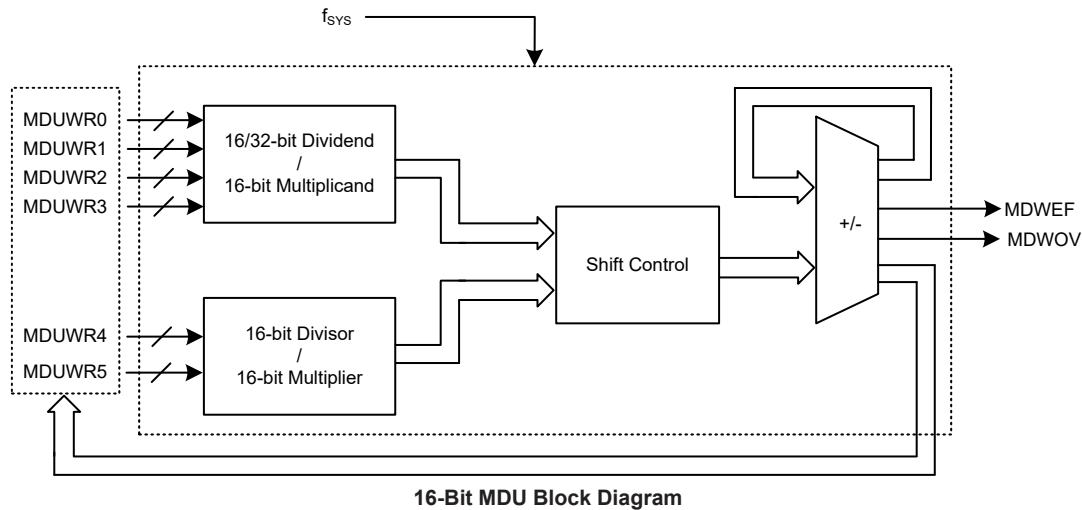


**LVD Operation**

The Low Voltage Detector also has its own interrupt which is contained within one of the Multi-function interrupts, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of  $t_{LVD}$  after the LVDO bit has been set high by a low voltage condition. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if  $V_{DD}$  or  $V_{LVDIN}$  falls below the preset LVD voltage. This will cause the device to wake up from the IDLE Mode, however if the Low Voltage Detector wake-up function is not required then the LVF flag should be first set high before the device enters the IDLE Mode.

## 16-bit Multiplication Division Unit – MDU

The device has a 16-bit Multiplication Division Unit, MDU, which integrates a 16-bit unsigned multiplier and a 32-bit/16-bit divider. The MDU, in replacing the software multiplication and division operations, can therefore save large amounts of computing time as well as the Program and Data Memory space. It also reduces the overall microcontroller loading and results in the overall system performance improvements.



### MDU Registers

The multiplication and division operations are implemented in a specific way, a specific write access sequence of a series of MDU data registers. The status register, MDUWCTRL, provides the indications for the MDU operation. The data register each is used to store the data regarded as the different operand corresponding to different MDU operations.

Register Name	Bit							
	7	6	5	4	3	2	1	0
MDUWR0	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR1	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR2	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR3	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR4	D7	D6	D5	D4	D3	D2	D1	D0
MDUWR5	D7	D6	D5	D4	D3	D2	D1	D0
MDUWCTRL	MDWEF	MDWOV	—	—	—	—	—	—

**MDU Register List**

#### • MDUWRn Register (n=0~5)

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**: 16-bit MDU data register n

• **MDUWCTRL Register**

Bit	7	6	5	4	3	2	1	0
Name	MDWEF	MDWOV	—	—	—	—	—	—
R/W	R	R	—	—	—	—	—	—
POR	0	0	—	—	—	—	—	—

- Bit 7      **MDWEF**: 16-bit MDU error flag  
             0: Normal  
             1: Abnormal  
             This bit will be set to 1 if the data register MDUWR<sub>n</sub> is written or read as the MDU operation is executing. This bit should be cleared to 0 by reading the MDUWCTRL register if it is equal to 1 and the MDU operation is completed.
- Bit 6      **MDWOV**: 16-bit MDU overflow flag  
             0: No overflow occurs  
             1: Multiplication product > FFFFH or Divisor=0  
             When an operation is completed, this bit will be updated by hardware to a new value corresponding to the current operation situation.
- Bit 5~0    Unimplemented, read as “0”

**MDU Operation**

For this MDU the multiplication or division operation is carried out in a specific way and is determined by the write access sequence of the six MDU data registers, MDUWR0~MDUWR5. The low byte data, regardless of the dividend, multiplicand, divisor or multiplier, must first be written into the corresponding MDU data register followed by the high byte data. All MDU operations will be executed after the MDUWR5 register is write-accessed together with the correct specific write access sequence of the MDUWR<sub>n</sub>. Note that it is not necessary to consecutively write data into the MDU data registers but must be in a correct write access sequence. Therefore, a non-write MDUWR<sub>n</sub> instruction or an interrupt, etc., can be inserted into the correct write access sequence without destroying the write operation. The relationship between the write access sequence and the MDU operation is shown in the following.

- 32-bit/16-bit division operation: Write data sequentially into the six MDU data registers from MDUWR0 to MDUWR5.
- 16-bit/16-bit division operation: Write data sequentially into the specific four MDU data registers in a sequence of MDUWR0, MDUWR1, MDUWR4 and MDUWR5 with no write access to MDUWR2 and MDUWR3.
- 16-bit×16-bit multiplication operation: Write data sequentially into the specific four MDU data register in a sequence of MDUWR0, MDUWR4, MDUWR1 and MDUWR5 with no write access to MDUWR2 and MDUWR3.

After the specific write access sequence is determined, the MDU will start to perform the corresponding operation. The calculation time necessary for these MDU operations are different. During the calculation time any read/write access to the six MDU data registers is forbidden. After the completion of each operation, it is necessary to check the operation status in the MDUWCTRL register to make sure that whether the operation is correct or not. Then the operation result can be read out from the corresponding MDU data registers in a specific read access sequence if the operation is correctly finished. The necessary calculation time for different MDU operations is listed in the following.

- 32-bit/16-bit division operation:  $17 \times t_{SYS}$ .
- 16-bit/16-bit division operation:  $9 \times t_{SYS}$ .
- 16-bit×16-bit multiplication operation:  $11 \times t_{SYS}$ .



The operation results will be stored in the corresponding MDU data registers and should be read out from the MDU data registers in a specific read access sequence after the operation is completed. Note that it is not necessary to consecutively read data out from the MDU data registers but must be in a correct read access sequence. Therefore, a non-read MDUWRn instruction or an interrupt, etc., can be inserted into the correct read access sequence without destroying the read operation. The relationship between the operation result read access sequence and the MDU operation is shown in the following.

- 32-bit/16-bit division operation: Read the quotient from MDUWR0 to MDUWR3 and remainder from MDUWR4 and MDUWR5 sequentially.
- 16-bit/16-bit division operation: Read the quotient from MDUWR0 and MDUWR1 and remainder from MDUWR4 and MDUWR5 sequentially.
- 16-bit×16-bit multiplication operation: Read the product sequentially from MDUWR0 to MDUWR3.

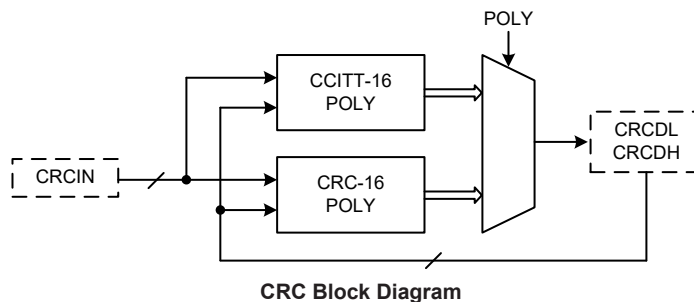
The overall important points for the MDU read/write access sequence and calculation time are summarized in the following table. Note that the device should not enter the IDLE or SLEEP mode until the MDU operation is totally completed, otherwise the MDU operation will fail.

Operations Items	32-bit / 16-bit Division	16-bit / 16-bit Division	16-bit × 16-bit Multiplication
<b>Write Sequence</b> First write ↓ ↓ ↓ ↓ Last write	Dividend Byte 0 written to MDUWR0 Dividend Byte 1 written to MDUWR1 Dividend Byte 2 written to MDUWR2 Dividend Byte 3 written to MDUWR3 Divisor Byte 0 written to MDUWR4 Divisor Byte 1 written to MDUWR5	Dividend Byte 0 written to MDUWR0 Dividend Byte 1 written to MDUWR1 Divisor Byte 0 written to MDUWR4 Divisor Byte 1 written to MDUWR5	Multiplicand Byte 0 written to MDUWR0 Multiplier Byte 0 written to MDUWR4 Multiplicand Byte 1 written to MDUWR1 Multiplier Byte 1 written to MDUWR5
<b>Calculation Time</b>	17×t <sub>sys</sub>	9×t <sub>sys</sub>	11×t <sub>sys</sub>
<b>Read Sequence</b> First read ↓ ↓ ↓ ↓ Last read	Quotient Byte 0 read from MDUWR0 Quotient Byte 1 read from MDUWR1 Quotient Byte 2 read from MDUWR2 Quotient Byte 3 read from MDUWR3 Remainder Byte 0 read from MDUWR4 Remainder Byte 1 read from MDUWR5	Quotient Byte 0 read from MDUWR0 Quotient Byte 1 read from MDUWR1 Remainder Byte 0 read from MDUWR4 Remainder Byte 1 read from MDUWR5	Product Byte 0 read from MDUWR0 Product Byte 1 read from MDUWR1 Product Byte 2 read from MDUWR2 Product Byte 3 read from MDUWR3

**MDU Operations Summary**

## Cyclic Redundancy Check – CRC

The Cyclic Redundancy Check, CRC, calculation unit is an error detection technique test algorithm and uses to verify data transmission or storage data correctness. A CRC calculation takes a data stream or a block of data as input and generates a 16-bit output remainder. Ordinarily, a data stream is suffixed by a CRC code and used as a checksum when being sent or stored. Therefore, the received or restored data stream is calculated by the same generator polynomial as described in the following section.



### CRC Registers

The CRC generator contains an 8-bit CRC data input register, CRCIN, and a CRC checksum register pair, CRCDH and CRCDL. The CRCIN register is used to input new data and the CRCDH and CRCDL registers are used to hold the previous CRC calculation result. A CRC control register, CRCCR, is used to select which CRC generating polynomial is used.

Register Name	Bit							
	7	6	5	4	3	2	1	0
CRCIN	D7	D6	D5	D4	D3	D2	D1	D0
CRCDL	D7	D6	D5	D4	D3	D2	D1	D0
CRCDH	D15	D14	D13	D12	D11	D10	D9	D8
CRCCR	—	—	—	—	—	—	—	POLY

**CRC Register List**

#### • CRCIN 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**: CRC input data register

#### • CRCDL 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**: 16-bit CRC checksum low byte data register

• **CRCDH 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:** 16-bit CRC checksum high byte data register

• **CRCCR Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	POLY
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1      Unimplemented, read as “0”

Bit 0      **POLY:** 16-bit CRC generating polynomial selection  
 0: CRC-CCITT:  $X^{16}+X^{12}+X^5+1$   
 1: CRC-16:  $X^{16}+X^{15}+X^2+1$

**CRC Operation**

The CRC generator provides the 16-bit CRC result calculation based on the CRC16 and CCITT CRC16 polynomials. In this CRC generator, there are only these two polynomials available for the numeric values calculation. It can not support the 16-bit CRC calculations based on any other polynomials.

The following two expressions can be used for the CRC generating polynomial which is determined using the POLY bit in the CRC control register, CRCCR. The CRC calculation result is called as the CRC checksum, CRCSUM, and stored in the CRC checksum register pair, CRCDH and CRCDL.

- CRC-CCITT:  $X^{16}+X^{12}+X^5+1$
- CRC-16:  $X^{16}+X^{15}+X^2+1$

**CRC Computation**

Each write operation to the CRCIN register creates a combination of the previous CRC value stored in the CRCDH and CRCDL registers and the new data input. The CRC unit calculates the CRC data register value is based on byte by byte. It will take one MCU instruction cycle to calculate the CRC checksum.

**CRC Calculation Procedures:**

1. Clear the checksum register pair, CRCDH and CRCDL.
2. Execute an “Exclusive OR” operation with the 8-bit input data byte and the 16-bit CRCSUM high byte. The result is called the temporary CRCSUM.
3. Shift the temporary CRCSUM value left by one bit and move a “0” into the LSB.
4. Check the shifted temporary CRCSUM value after procedure 3.

If the MSB is 0, then this shifted temporary CRCSUM will be considered as a new temporary CRCSUM.

Otherwise, execute an “Exclusive OR” operation with the shifted temporary CRCSUM in procedure 3 and a data “8005H”. Then the operation result will be regarded as the new temporary CRCSUM.

Note that the data to be perform an “Exclusive OR” operation is “8005H” for the CRC-16 polynomial while for the CRC-CCITT polynomial the data is “1021H”.

5. Repeat the procedure 3 ~ procedure 4 until all bits of the input data byte are completely calculated.
6. Repeat the procedure 2~ procedure 5 until all of the input data bytes are completely calculated. Then, the latest calculated result is the final CRC checksum, CRCSUM.

**CRC Calculation Examples:**

- Write 1 byte input data into the CRCIN register and the corresponding CRC checksum are individually calculated as the following table shown.

CRC Data Input CRC Polynomial	00H	01H	02H	03H	04H	05H	06H	07H
CRC-CCITT ( $X^{16}+X^{12}+X^5+1$ )	0000H	1021H	2042H	3063H	4084H	50A5H	60C6H	70E7H
CRC-16 ( $X^{16}+X^{15}+X^2+1$ )	0000H	8005H	800FH	000AH	801BH	001EH	0014H	8011H

Note: The initial value of the CRC checksum register pair, CRCDH and CRCDL, is zero before each CRC input data is written into the CRCIN register.

- Write 4 bytes input data into the CRCIN register sequentially and the CRC checksum are sequentially listed in the following table.

CRC Data Input CRC Polynomial	CRCIN=78H→56H→34H→12H
CRC-CCITT ( $X^{16}+X^{12}+X^5+1$ )	(CRCDH, CRCDL)=FF9FH→BBC3H→A367H→D0FAH
CRC-16 ( $X^{16}+X^{15}+X^2+1$ )	(CRCDH, CRCDL)=0110h→91F1h→F2DEh→5C43h

Note: The initial value of the CRC checksum register pair, CRCDH and CRCDL, is zero before the sequential CRC data input operation.

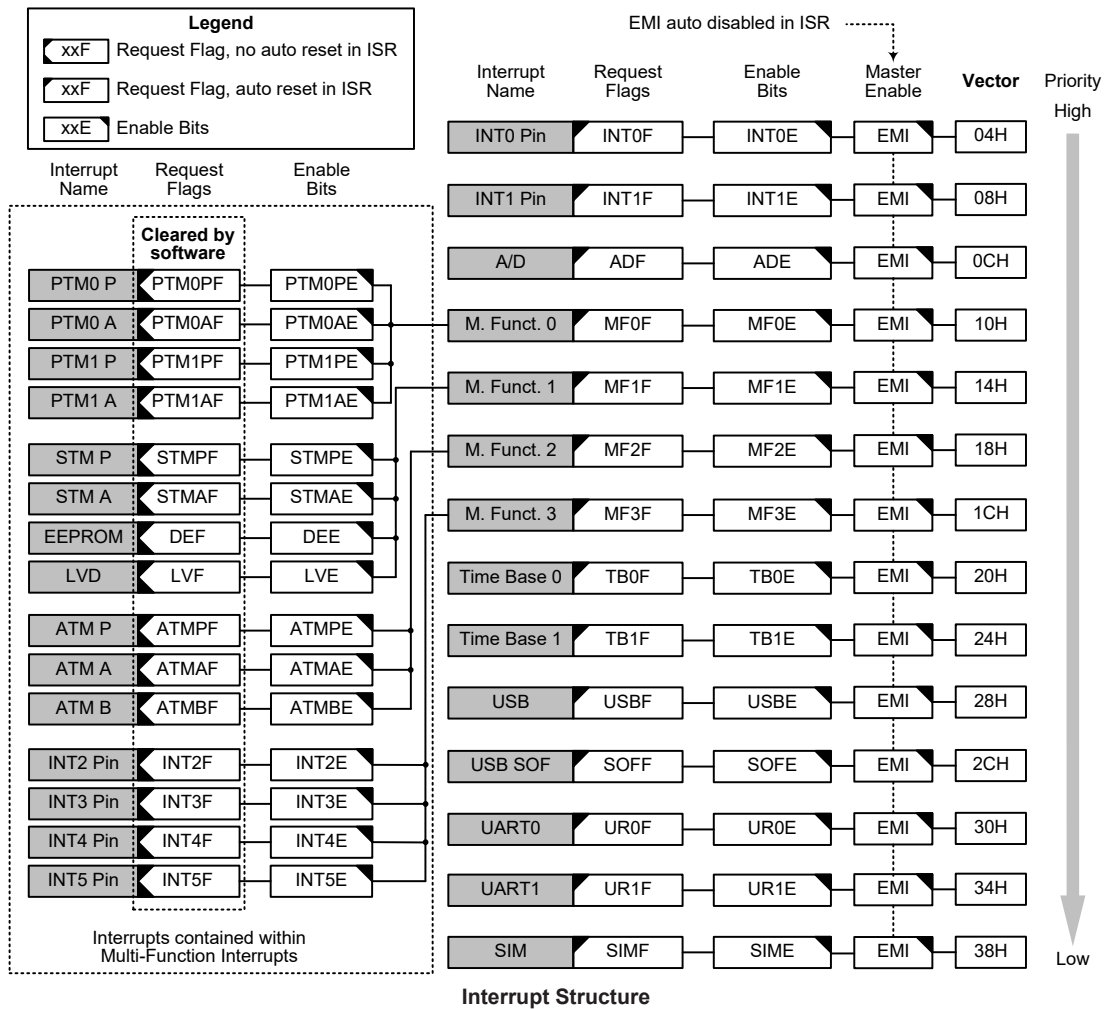
**Program Memory CRC Checksum Calculation Example:**

1. Clear the checksum register pair, CRCDH and CRCDL.
2. Select the CRC-CCITT or CRC-16 polynomial as the generating polynomial using the POLY bit in the CRCCR register.
3. Execute the table read instruction to read the program memory data value.
4. Write the table data low byte into the CRCIN register and execute the CRC calculation with the current CRCSUM value. Then a new CRCSUM result will be obtained and stored in the CRC checksum register pair, CRCDH and CRCDL.
5. Write the table data high byte into the CRCIN register and execute the CRC calculation with the current CRCSUM value. Then a new CRCSUM result will be obtained and stored in the CRC checksum register pair, CRCDH and CRCDL.
6. Repeat the procedure 3 ~ procedure 5 to read the next program memory data value and execute the CRC calculation until all program memory data are read followed by the sequential CRC calculation. Then the value in the CRC checksum register pair is the final CRC calculation result.

## Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupts functions. The external interrupts are generated by the action of the external INT0~INT5 pins, while the internal interrupts are generated by various internal functions such as the TMs, Time Base, EEPROM, SIM, UART, LVD and the A/D converter, etc.

The various interrupt enable bits, together with their associated request flags, are shown in the accompanying diagrams with their order of priority. Some interrupt sources have their own individual vector while others share the same multi-function interrupt vector.



## Interrupt Registers

Overall interrupt control, which basically means the setting of request flags when certain microcontroller conditions occur and the setting of interrupt enable bits by the application program, is controlled by a series of registers, located in the Special Purpose Data Memory, as shown in the accompanying table. The number of registers depends upon the device chosen but fall into three

categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MF10~MF13 registers which setup the Multi-function interrupts. Finally there are two registers, INTEG0 and INTEG1, to setup the external interrupt trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INTn Pins	INTnE	INTnF	n=0~5
A/D Converter	ADE	ADF	—
Multi-function	MFnE	MFnF	n=0~3
Time Base	TBnE	TBnF	n=0~1
USB	USBE	USBF	—
USB start of frame	SOFE	SOFF	—
UART Interface	URnE	URnF	n=0~1
SIM Interface	SIME	SIMF	—
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
PTM	PTMnPE	PTMnPF	n=0~1
	PTMnAE	PTMnAF	
STM	STMPE	STMPF	—
	STMAE	STMAF	
ATM	ATMPE	ATMPF	—
	ATMAE	ATMAF	
	ATMBE	ATMBF	

**Interrupt Register Bit Naming Conventions**

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG0	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
INTEG1	—	—	—	—	INT5S1	INT5S0	INT4S1	INT4S0
INTC0	—	ADF	INT1F	INT0F	ADE	INT1E	INT0E	EMI
INTC1	MF3F	MF2F	MF1F	MF0F	MF3E	MF2E	MF1E	MF0E
INTC2	SOFF	USBF	TB1F	TB0F	SOFE	USBE	TB1E	TB0E
INTC3	—	SIMF	UR1F	UR0F	—	SIME	UR1E	UR0E
MF10	PTM1AF	PTM1PF	PTM0AF	PTM0PF	PTM1AE	PTM1PE	PTM0AE	PTM0PE
MF11	LVF	DEF	STMAF	STMPF	LVE	DEE	STMAE	STMPE
MF12	—	ATMBF	ATMAF	ATMPF	—	ATMBE	ATMAE	ATMPE
MF13	INT5F	INT4F	INT3F	INT2F	INT5E	INT4E	INT3E	INT2E

**Interrupt Register List**

• **INTEG0 Register**

Bit	7	6	5	4	3	2	1	0
Name	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
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     **INT3S1~INT3S0**: Interrupt edge control for INT3 pin  
 00: Disable  
 01: Rising edge  
 10: Falling edge  
 11: Rising and falling edges
- Bit 5~4     **INT2S1~INT2S0**: Interrupt edge control for INT2 pin  
 00: Disable  
 01: Rising edge  
 10: Falling edge  
 11: Rising and falling edges
- 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

• **INTEG1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	INT5S1	INT5S0	INT4S1	INT4S0
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     **INT5S1~INT5S0**: Interrupt edge control for INT5 pin  
 00: Disable  
 01: Rising edge  
 10: Falling edge  
 11: Rising and falling edges
- Bit 1~0     **INT4S1~INT4S0**: Interrupt edge control for INT4 pin  
 00: Disable  
 01: Rising edge  
 10: Falling edge  
 11: Rising and falling edges

• **INTC0 Register**

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

- Bit 7     Unimplemented, read as “0”
- Bit 6     **ADF**: A/D Converter interrupt request flag  
 0: No request  
 1: Interrupt request

- Bit 5      **INT1F**: INT1 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **INT0F**: INT0 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3      **ADE**: A/D Converter interrupt control  
0: Disable  
1: Enable
- Bit 2      **INT1E**: INT1 interrupt control  
0: Disable  
1: Enable
- Bit 1      **INT0E**: INT0 interrupt control  
0: Disable  
1: Enable
- Bit 0      **EMI**: Global interrupt control  
0: Disable  
1: Enable

• **INTC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	MF3F	MF2F	MF1F	MF0F	MF3E	MF2E	MF1E	MF0E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7      **MF3F**: Multi-function 3 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 6      **MF2F**: Multi-function 2 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5      **MF1F**: Multi-function 1 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **MF0F**: Multi-function 0 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3      **MF3E**: Multi-function 3 interrupt control  
0: Disable  
1: Enable
- Bit 2      **MF2E**: Multi-function 2 interrupt control  
0: Disable  
1: Enable
- Bit 1      **MF1E**: Multi-function 1 interrupt control  
0: Disable  
1: Enable
- Bit 0      **MF0E**: Multi-function 0 interrupt control  
0: Disable  
1: Enable



• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	SOFF	USBF	TB1F	TB0F	SOFE	USBE	TB1E	TB0E
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      **SOFF**: USB start-of-frame interrupt request flag  
             0: No request  
             1: Interrupt request
- Bit 6      **USBF**: USB 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      **SOFE**: USB start-of-frame interrupt control  
             0: Disable  
             1: Enable
- Bit 2      **USBE**: USB 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

• **INTC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	SIMF	UR1F	UR0F	—	SIME	UR1E	UR0E
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      **SIMF**: SIM interrupt request flag  
             0: No request  
             1: Interrupt request
- Bit 5      **UR1F**: UART1 interrupt request flag  
             0: No request  
             1: Interrupt request
- Bit 4      **UR0F**: UART0 interrupt request flag  
             0: No request  
             1: Interrupt request
- Bit 3      Unimplemented, read as “0”
- Bit 2      **SIME**: SIM interrupt control  
             0: Disable  
             1: Enable

- Bit 1     **UR1E**: UART1 interrupt control  
           0: Disable  
           1: Enable
- Bit 0     **UR0E**: UART0 interrupt control  
           0: Disable  
           1: Enable

• **MF10 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTM1AF	PTM1PF	PTM0AF	PTM0PF	PTM1AE	PTM1PE	PTM0AE	PTM0PE
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     **PTM1AF**: PTM1 Comparator A match interrupt request flag  
           0: No request  
           1: Interrupt request  
       Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 6     **PTM1PF**: PTM1 Comparator P match interrupt request flag  
           0: No request  
           1: Interrupt request  
       Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 5     **PTM0AF**: PTM0 Comparator A match interrupt request flag  
           0: No request  
           1: Interrupt request  
       Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 4     **PTM0PF**: PTM0 Comparator P match interrupt request flag  
           0: No request  
           1: Interrupt request  
       Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 3     **PTM1AE**: PTM1 Comparator A match interrupt control  
           0: Disable  
           1: Enable
- Bit 2     **PTM1PE**: PTM1 Comparator P match interrupt control  
           0: Disable  
           1: Enable
- Bit 1     **PTM0AE**: PTM0 Comparator A match interrupt control  
           0: Disable  
           1: Enable
- Bit 0     **PTM0PE**: PTM0 Comparator P match interrupt control  
           0: Disable  
           1: Enable

• **MF1 Register**

Bit	7	6	5	4	3	2	1	0
Name	LVF	DEF	STMAF	STMPF	LVE	DEE	STMAE	STMPE
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  
 Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 6      **DEF**: Data EEPROM interrupt request flag  
 0: No request  
 1: Interrupt request  
 Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 5      **STMAF**: STM Comparator A match interrupt request flag  
 0: No request  
 1: Interrupt request  
 Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 4      **STMPF**: STM Comparator P match interrupt request flag  
 0: No request  
 1: Interrupt request  
 Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 3      **LVE**: LVD interrupt control  
 0: Disable  
 1: Enable
- Bit 2      **DEE**: Data EEPROM interrupt control  
 0: Disable  
 1: Enable
- Bit 1      **STMAE**: STM Comparator A match interrupt control  
 0: Disable  
 1: Enable
- Bit 0      **STMPE**: STM Comparator P match interrupt control  
 0: Disable  
 1: Enable

• **MF12 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	ATMBF	ATMAF	ATMPF	—	ATMBE	ATMAE	ATMPE
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      **ATMBF**: ATM Comparator B match interrupt request flag  
 0: No request  
 1: Interrupt request  
 Note that this bit must be cleared to zero by the application program when the interrupt is serviced.

- Bit 5      **ATMAF**: ATM Comparator A match interrupt request flag  
             0: No request  
             1: Interrupt request  
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 4      **ATMPF**: ATM Comparator P match interrupt request flag  
             0: No request  
             1: Interrupt request  
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 3      Unimplemented, read as “0”
- Bit 2      **ATMBE**: ATM Comparator B match interrupt control  
             0: Disable  
             1: Enable
- Bit 1      **ATMAE**: ATM Comparator A match interrupt control  
             0: Disable  
             1: Enable
- Bit 0      **ATMPE**: ATM Comparator P match interrupt control  
             0: Disable  
             1: Enable

• **MFI3 Register**

Bit	7	6	5	4	3	2	1	0
Name	INT5F	INT4F	INT3F	INT2F	INT5E	INT4E	INT3E	INT2E
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      **INT5F**: INT5 interrupt request flag  
             0: No request  
             1: Interrupt request  
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 6      **INT4F**: INT4 interrupt request flag  
             0: No request  
             1: Interrupt request  
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 5      **INT3F**: INT3 interrupt request flag  
             0: No request  
             1: Interrupt request  
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 4      **INT2F**: INT2 interrupt request flag  
             0: No request  
             1: Interrupt request  
             Note that this bit must be cleared to zero by the application program when the interrupt is serviced.
- Bit 3      **INT5E**: INT5 interrupt control  
             0: Disable  
             1: Enable
- Bit 2      **INT4E**: INT4 interrupt control  
             0: Disable  
             1: Enable

Bit 1	<b>INT3E:</b> INT3 interrupt control 0: Disable 1: Enable
Bit 0	<b>INT2E:</b> INT2 interrupt control 0: Disable 1: Enable

### Interrupt Operation

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

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

Once an interrupt subroutine is serviced, all the other interrupts will be blocked, as the global interrupt enable bit, EMI bit will be cleared automatically. This will prevent any further interrupt nesting from occurring. However, if other interrupt requests occur during this interval, although the interrupt will not be immediately serviced, the request flag will still be recorded.

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

### External Interrupts

The external interrupts are controlled by signal transitions on the pins INT0~INT5. An external interrupt request will take place when the external interrupt request flags, INT0F~INT5F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI, respective external interrupt enable bit, INT0E~INT5E, and relevant Multi-function Interrupt enable bit, MFnE, must first be set. Additionally the correct interrupt edge type must be selected using the INTEGn register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the

corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register.

When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector or the relevant Multi-function Interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT1F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. For the INT2~INT5 external pins, when the interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts and the related MFnF flag will also be automatically cleared. As the INT2~INT5 interrupt request flags, INT2F~INT5F, will not be automatically cleared, they have to be cleared by the application program. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input.

The INTEGn 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 INTEGn register can also be used to disable the external interrupt function.

### **A/D Converter Interrupt**

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

### **Multi-function Interrupts**

Within the device there are four Multi-function interrupts. Unlike the other independent interrupts, these interrupts have no independent source, but rather are formed from other existing interrupt sources, namely the TM interrupts, LVD interrupt, EEPROM interrupt and INT2~INT5 interrupt.

A Multi-function interrupt request will take place when any of the Multi-function interrupt request flags MFnF 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 their interrupt vector addresses, when the Multi-function interrupt is enabled and the stack is not full, and one of the interrupts contained within each of Multi-function interrupt occurs, a subroutine call to the relevant Multi-function interrupt vector will take place. When the interrupt is serviced, the related Multi-Function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

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

### **Timer Module Interrupts**

The Standard and Periodic type TMs each has two interrupts, while the Audio Type TM has three interrupts. All of the TM interrupts are contained within the Multi-function Interrupts. For each of the Standard and Periodic Type TMs there are two interrupt request flags and two interrupt enable bits. For the Audio Type TM there are three interrupt request flags and three enable 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, A or B match situation happens.

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

### **EEPROM Interrupt**

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

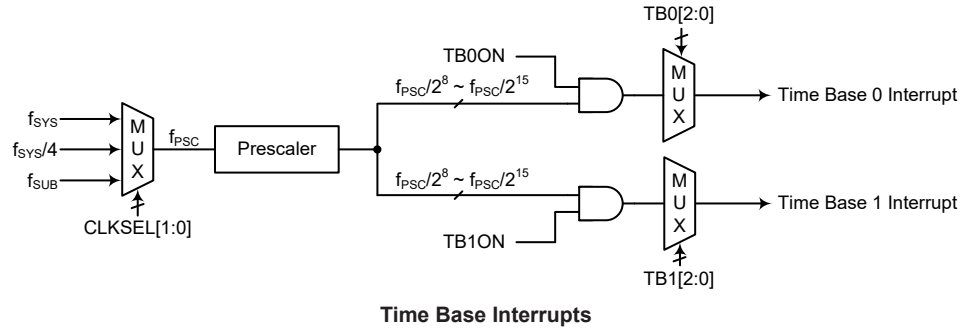
### **LVD Interrupt**

The Low Voltage Detector Interrupt is contained within the Multi-function Interrupt. An LVD Interrupt request will take place when the LVD Interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage or a low LVDIN pin input voltage. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, Low Voltage Interrupt enable bit, LVE, and associated Multi-function interrupt enable bit, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the Multi-function Interrupt vector, will take place. When the Low Voltage Interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts. However, only the Multi-function interrupt request flag will be also automatically cleared. As the LVF flag will not be automatically cleared, it has to be cleared by the application program.

### **Time Base Interrupts**

The function of the Time Base Interrupts is to provide regular time signals in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happens their respective interrupt request flags, TB0F or TB1F will be set. To allow the program to branch to their respective interrupt vector addresses, the global interrupt enable bit, EMI and Time Base enable bits, TB0E or TB1E, must first be set. When the interrupt is enabled, the stack is not full and the Time Base overflows, a subroutine call to their respective vector locations will take place. When the interrupt is serviced, the respective interrupt request flag, TB0F or TB1F, will be automatically cleared, the EMI bit will also be automatically cleared to disable other interrupts.

The purpose of the Time Base Interrupt is to provide an interrupt signal at fixed time periods. Its clock source,  $f_{PSC}$ , originates from the internal clock source  $f_{SYS}$ ,  $f_{SYS}/4$  or  $f_{SUB}$  and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source that generate  $f_{PSC}$ , which in turn controls the Time Base interrupt period, is selected using the CLKSEL[1:0] bits in the PSCR register.



• **PSCR Register**

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **CLKSEL1~CLKSELO**: Prescaler clock source  $f_{psc}$  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 Enable Control  
 0: Disable  
 1: Enable

Bit 6~3 Unimplemented, read as “0”

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

• **TB1C Register**

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

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

Bit 6~3 Unimplemented, read as “0”



Bit 2~0	<b>TB12~TB10:</b> Time Base 1 time-out period selection
	000: $2^8/f_{PSC}$
	001: $2^9/f_{PSC}$
	010: $2^{10}/f_{PSC}$
	011: $2^{11}/f_{PSC}$
	100: $2^{12}/f_{PSC}$
	101: $2^{13}/f_{PSC}$
	110: $2^{14}/f_{PSC}$
	111: $2^{15}/f_{PSC}$

## USB Interrupt

Several USB conditions can generate a USB interrupt. When one of these conditions occurs, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are the USB suspended, USB resumed, USB reset and USB endpoint FIFO access events. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and USB interrupt enable bit, USBE, must first be set. When the interrupt is enabled, the stack is not full and any of these conditions are created, a subroutine call to the USB interrupt vector, will take place. When the interrupt is serviced, the USB interrupt request flag, USBF, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

## USB Start of Frame Interrupt

The USB Start of Frame interrupt, also known as the USB SOF interrupt, will take place when the corresponding interrupt request flag, SOFF, is set, which occurs when a USB start of frame signal is detected. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and USB start of frame interrupt enable bit, SOFE, must first be set. When the interrupt is enabled, the stack is not full and a USB start of frame signal is detected, a subroutine call to the respective interrupt vector will take place. When the USB SOF interrupt is serviced, the SOFF flag will be automatically cleared, the EMI bit will also be automatically cleared to disable other interrupts.

## Serial Interface Module Interrupt

The Serial Interface Module Interrupt is also known as the SIM interrupt. A SIM Interrupt request will take place when the SIM Interrupt request flag, SIMF, is set, which occurs when a byte of data has been received or transmitted by the SIM interface, an I<sup>2</sup>C slave address match or I<sup>2</sup>C bus time-out occurs. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and SIM Interrupt enable bit, SIME, must first be set. When the interrupt is enabled, the stack is not full and any of the above described situations occurs, a subroutine call to the corresponding SIM Interrupt vector, will take place. When the interrupt is serviced, the SIMF flag will be automatically cleared and the EMI bit will be automatically cleared to disable other interrupts.

## UART Interrupts

The UARTn Interrupt is controlled by several UARTn transfer conditions. When one of these conditions occurs, an interrupt pulse will be generated to get the attention of the microcontroller. These conditions are a transmitter data register empty, transmitter idle, receiver reaching FIFO trigger level, receiver overrun, address detect and an RXn/TXn pin wake-up. To allow the program to branch to their interrupt vector addresses, the global interrupt enable bit, EMI, and UARTn Interrupt enable bit, URnE, must first be set. When the interrupt is enabled, the stack is not full and any of the conditions described above occurs, a subroutine call to the corresponding UARTn Interrupt vector, will take place. When the interrupt is serviced, the UARTn Interrupt flag, URnF,

will be automatically cleared. The EMI bit will also be automatically cleared to disable other interrupts. However, the UnSR register flags will only be cleared when certain actions are taken by the UARTn, the details of which are given in the UART section.

### **Interrupt Wake-up Function**

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

### **Programming Considerations**

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

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

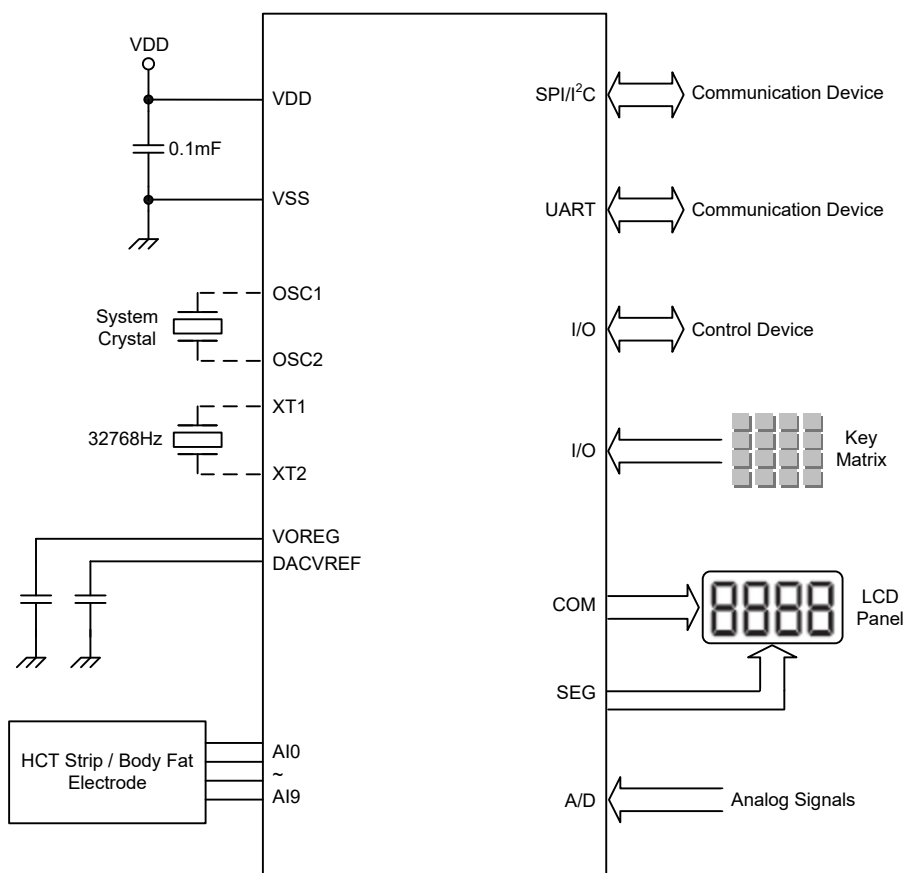
It is recommended that programs do not use the “CALL” instruction within the interrupt service subroutine. Interrupts often occur in an unpredictable manner or need to be serviced immediately. If only one stack is left and the interrupt is not well controlled, the original control sequence will be damaged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

Every interrupt has the capability of waking up the microcontroller when it is in the SLEEP or IDLE Mode, the wake up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine.

To return from an interrupt subroutine, either a RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

## Application Circuits



## Instruction Set

### Introduction

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

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

### Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5 $\mu$ 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	↑Note	Z, C, AC, OV, SC
ADD A,x	Add immediate data to ACC	1	Z, C, AC, OV, SC
ADC A,[m]	Add Data Memory to ACC with Carry	1	Z, C, AC, OV, SC
ADCM A,[m]	Add ACC to Data memory with Carry	↑Note	Z, C, AC, OV, SC
SUB A,x	Subtract immediate data from the ACC	1	Z, C, AC, OV, SC, CZ
SUB A,[m]	Subtract Data Memory from ACC	1	Z, C, AC, OV, SC, CZ
SUBM A,[m]	Subtract Data Memory from ACC with result in Data Memory	↑Note	Z, C, AC, OV, SC, CZ
SBC A,x	Subtract immediate data from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBC A,[m]	Subtract Data Memory from ACC with Carry	1	Z, C, AC, OV, SC, CZ
SBCM A,[m]	Subtract Data Memory from ACC with Carry, result in Data Memory	↑Note	Z, C, AC, OV, SC, CZ
DAA [m]	Decimal adjust ACC for Addition with result in Data Memory	↑Note	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	↑Note	Z
ORM A,[m]	Logical OR ACC to Data Memory	↑Note	Z
XORM A,[m]	Logical XOR ACC to Data Memory	↑Note	Z
AND A,x	Logical AND immediate Data to ACC	1	Z
OR A,x	Logical OR immediate Data to ACC	1	Z
XOR A,x	Logical XOR immediate Data to ACC	1	Z
CPL [m]	Complement Data Memory	↑Note	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	↑Note	Z
DECA [m]	Decrement Data Memory with result in ACC	1	Z
DEC [m]	Decrement Data Memory	↑Note	Z
<b>Rotate</b>			
RRA [m]	Rotate Data Memory right with result in ACC	1	None
RR [m]	Rotate Data Memory right	↑Note	None
RRCA [m]	Rotate Data Memory right through Carry with result in ACC	1	C
RRC [m]	Rotate Data Memory right through Carry	↑Note	C
RLA [m]	Rotate Data Memory left with result in ACC	1	None
RL [m]	Rotate Data Memory left	↑Note	None
RLCA [m]	Rotate Data Memory left through Carry with result in ACC	1	C
RLC [m]	Rotate Data Memory left through Carry	↑Note	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	↑ <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	↑ <sup>Note</sup>	None
SET [m].i	Set bit of Data Memory	↑ <sup>Note</sup>	None
<b>Branch Operation</b>			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	↑ <sup>Note</sup>	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	↑ <sup>Note</sup>	None
SZ [m].i	Skip if bit i of Data Memory is zero	↑ <sup>Note</sup>	None
SNZ [m]	Skip if Data Memory is not zero	↑ <sup>Note</sup>	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	↑ <sup>Note</sup>	None
SIZ [m]	Skip if increment Data Memory is zero	↑ <sup>Note</sup>	None
SDZ [m]	Skip if decrement Data Memory is zero	↑ <sup>Note</sup>	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	↑ <sup>Note</sup>	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	↑ <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
<b>Miscellaneous</b>			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	↑ <sup>Note</sup>	None
SET [m]	Set Data Memory	↑ <sup>Note</sup>	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	↑ <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>			
LRRRA [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
LSIZ [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
LSIZA [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
<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. 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. 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. 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. 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. 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 ← Program Counter + 1 Program Counter ← 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] ← 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 ← 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 TO ← 0 PDF ← 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] ← $\overline{[m]}$
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 ← $\overline{[m]}$
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] ← ACC + 00H or [m] ← ACC + 06H or [m] ← ACC + 60H or [m] ← 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	$TO \leftarrow 0$ $PDF \leftarrow 1$
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 ← ACC "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 ← ACC "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] ← ACC "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 ← 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 ← Stack ACC ← 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 ← Stack EMI ← 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) ← [m].i; (i=0~6) [m].0 ← [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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
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	$[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
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	$[m].i \leftarrow [m].(i+1); (i=0\sim6)$ $[m].7 \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1); (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
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	$[m].i \leftarrow [m].(i+1); (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
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 ← [m].(i+1); (i=0~6) ACC.7 ← C C ← [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 ← ACC – [m] – $\bar{C}$
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 ← ACC – [m] – $\bar{C}$
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] ← ACC – [m] – $\bar{C}$
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] ← [m] – 1 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 ← [m] – 1 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	$[m] \leftarrow [m] + 1$ 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	$ACC \leftarrow [m] + 1$ 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	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
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	$ACC \leftarrow [m]$ 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) 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. 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. 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. 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. 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 $[m] \leftarrow ACC + 06H$ or $[m] \leftarrow ACC + 60H$ or $[m] \leftarrow ACC + 66H$
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	$[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
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	$[m].(i+1) \leftarrow [m].i; (i=0\sim6)$ $[m].0 \leftarrow C$ $C \leftarrow [m].7$
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	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow C$ $C \leftarrow [m].7$
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	$[m].i \leftarrow [m].(i+1); (i=0\sim 6)$ $[m].7 \leftarrow [m].0$
Affected flag(s)	None
<b>LRRRA [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	$ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$ $ACC.7 \leftarrow [m].0$
Affected flag(s)	None
<b>LRRRC [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	$[m].i \leftarrow [m].(i+1); (i=0\sim 6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1); (i=0\sim 6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC \leftarrow ACC - [m] - \bar{C}$
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	$[m] \leftarrow ACC - [m] - \bar{C}$
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	$[m] \leftarrow [m] - 1$ 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	$ACC \leftarrow [m] - 1$ 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	$[m] \leftarrow [m] + 1$ 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	$ACC \leftarrow [m] + 1$ 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] ≠ 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	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
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	$ACC \leftarrow [m]$ 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) 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

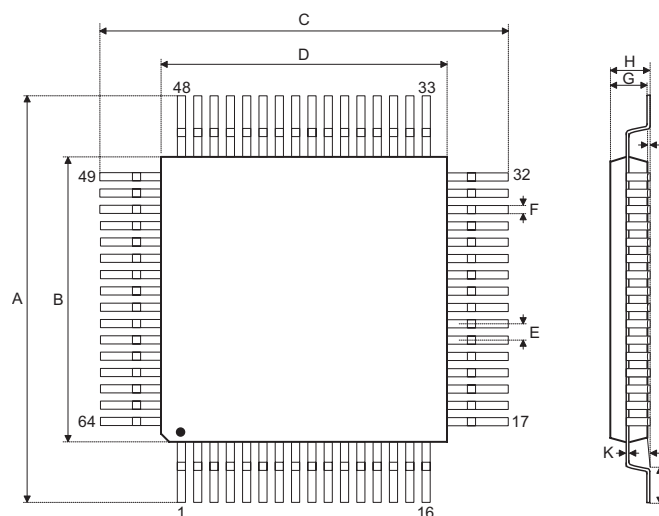
## Package Information

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

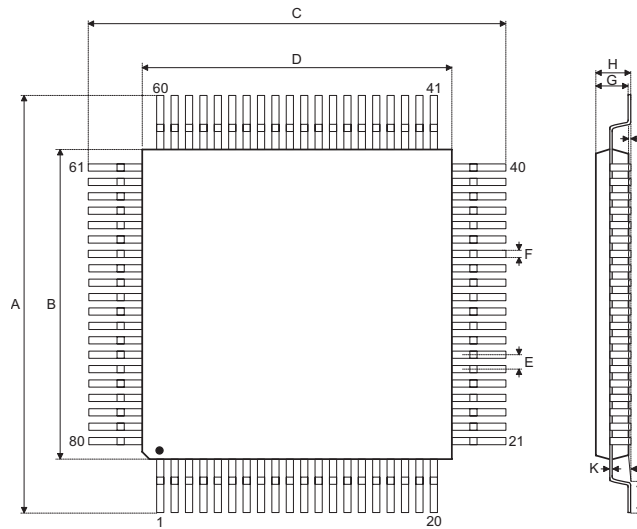
**64-pin LQFP (7mm×7mm) Outline Dimensions**



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.354 BSC		
B	0.276 BSC		
C	0.354 BSC		
D	0.276 BSC		
E	0.016 BSC		
F	0.005	0.007	0.009
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	9.00 BSC		
B	7.00 BSC		
C	9.00 BSC		
D	7.00 BSC		
E	0.40 BSC		
F	0.13	0.18	0.23
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°

**80-pin LQFP (10mm×10mm) Outline Dimensions**



Symbol	Dimensions in inch		
	Min.	Nom.	Max.
A	0.472 BSC		
B	0.394 BSC		
C	0.472 BSC		
D	0.394 BSC		
E	0.016 BSC		
F	0.005	0.007	0.009
G	0.053	0.055	0.057
H	—	—	0.063
I	0.002	—	0.006
J	0.018	0.024	0.030
K	0.004	—	0.008
α	0°	—	7°

Symbol	Dimensions in mm		
	Min.	Nom.	Max.
A	12.00 BSC		
B	10.00 BSC		
C	12.00 BSC		
D	10.00 BSC		
E	0.40 BSC		
F	0.13	0.18	0.23
G	1.35	1.40	1.45
H	—	—	1.60
I	0.05	—	0.15
J	0.45	0.60	0.75
K	0.09	—	0.20
α	0°	—	7°

Copyright© 2024 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 authorize 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.