



**A/D Flash MCU with LCD Driver**

**HT67F2355**

Revision: V1.20    Date: September 07, 2023

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

## Table of Contents

<b>Features .....</b>	<b>7</b>
CPU Features .....	7
Peripheral Features.....	7
<b>General Description.....</b>	<b>8</b>
<b>Block Diagram.....</b>	<b>9</b>
<b>Pin Assignment.....</b>	<b>9</b>
<b>Pin Description .....</b>	<b>10</b>
<b>Absolute Maximum Ratings.....</b>	<b>15</b>
<b>D.C. Characteristics.....</b>	<b>15</b>
Operating Voltage Characteristics.....	15
Operating Current Characteristics.....	16
Standby Current Characteristics .....	16
<b>A.C. Characteristics.....</b>	<b>17</b>
High Speed Internal Oscillator – HIRC – Frequency Accuracy .....	17
Low Speed Internal Oscillator Characteristics – LIRC .....	18
Low Speed Crystal Oscillator Characteristics – LXT.....	18
Operating Frequency Characteristic Curves .....	19
System Start Up Time Characteristics .....	19
<b>Input/Output Characteristics .....</b>	<b>20</b>
<b>A/D Converter Electrical Characteristics.....</b>	<b>21</b>
<b>Memory Electrical Characteristics .....</b>	<b>22</b>
<b>Internal Reference Voltage Characteristics.....</b>	<b>22</b>
<b>LVD/LVR Electrical Characteristics .....</b>	<b>23</b>
<b>LCD Electrical Characteristics .....</b>	<b>23</b>
<b>I<sup>2</sup>C Electrical Characteristics .....</b>	<b>24</b>
<b>Power-on Reset Characteristics.....</b>	<b>25</b>
<b>System Architecture .....</b>	<b>25</b>
Clocking and Pipelining.....	25
Program Counter.....	26
Stack .....	26
Arithmetic and Logic Unit – ALU .....	27
<b>Flash Program Memory.....</b>	<b>28</b>
Structure.....	28
Special Vectors .....	28
Look-up Table.....	28
Table Program Example.....	29
In Circuit Programming – ICP .....	30
On-Chip Debug Support – OCDs .....	30
In Application Programming – IAP .....	31

<b>Data Memory .....</b>	<b>47</b>
Structure.....	47
Data Memory Addressing .....	48
General Purpose Data Memory .....	48
Special Purpose Data Memory .....	48
<b>Special Function Register Description.....</b>	<b>50</b>
Indirect Addressing Registers – IAR0, IAR1, IAR2 .....	50
Memory Pointers – MP0, MP1L/MP1H, MP2L/MP2H.....	50
Accumulator – ACC.....	51
Program Counter Low Byte Register – PCL.....	52
Look-up Table Registers – TBLP, TBHP, TBLH.....	52
Option Memory Mapping Register – ORMC .....	52
Status Register – STATUS .....	53
<b>EEPROM Data Memory.....</b>	<b>54</b>
EEPROM Data Memory Structure .....	54
EEPROM Registers .....	54
Read Operation from the EEPROM.....	56
Page Erase Operation to the EEPROM.....	57
Write Operation to the EEPROM .....	58
Write Protection.....	59
EEPROM Interrupt .....	59
Programming Considerations.....	59
<b>Oscillators .....</b>	<b>62</b>
Oscillator Overview .....	62
System Clock Configurations.....	63
External Crystal/Ceramic Oscillator – HXT .....	63
Internal High Speed RC Oscillator – HIRC .....	64
External 32.768kHz Crystal Oscillator – LXT .....	64
Internal 32kHz Oscillator – LIRC.....	65
<b>Operating Modes and System Clocks .....</b>	<b>66</b>
System Clocks .....	66
System Operation Modes.....	67
Control Registers .....	68
Operating Mode Switching .....	71
Standby Current Considerations.....	74
Wake-up .....	74
<b>Watchdog Timer.....</b>	<b>75</b>
Watchdog Timer Clock Source.....	75
Watchdog Timer Control Register .....	75
Watchdog Timer Operation .....	76
<b>Reset and Initialisation.....</b>	<b>77</b>
Reset Functions .....	77
Reset Initial Conditions .....	82

<b>Input/Output Ports .....</b>	<b>86</b>
Pull-high Resistors .....	86
Port A Wake-up .....	87
I/O Port Control Registers .....	87
I/O Port Source Current Control .....	88
Pin-shared Functions .....	90
I/O Pin Structures .....	97
Programming Considerations .....	98
<b>Timer Modules – TM .....</b>	<b>98</b>
Introduction .....	98
TM Operation .....	99
TM Clock Source .....	99
TM Interrupts .....	99
TM External Pins .....	99
Programming Considerations .....	100
<b>Compact Type TM – CTM .....</b>	<b>101</b>
Compact Type TM Operation .....	101
Compact Type TM Register Description .....	102
Compact Type TM Operating Modes .....	106
<b>Periodic Type TM – PTM .....</b>	<b>112</b>
Periodic TM Operation .....	112
Periodic Type TM Register Description .....	112
Periodic Type TM Operating Modes .....	116
<b>Analog to Digital Converter – ADC .....</b>	<b>123</b>
A/D Converter Overview .....	123
Registers Descriptions .....	123
A/D Converter Reference Voltage .....	127
A/D Converter Input Signals .....	127
A/D Conversion Operation .....	128
Conversion Rate and Timing Diagram .....	129
Summary of A/D Conversion Steps .....	130
Programming Considerations .....	131
A/D Transfer Function .....	131
A/D Programming Examples .....	131
<b>Serial Interface Module – SIM .....</b>	<b>133</b>
SPI Interface .....	133
I <sup>2</sup> C Interface .....	141
<b>UART Interface .....</b>	<b>151</b>
UART External Pins .....	152
UART Single Wire Mode .....	152
UART Data Transfer Scheme .....	152
UART Status and Control Registers .....	153
Baud Rate Generator .....	158

UART Setup and Control.....	159
UART Transmitter.....	160
UART Receiver .....	162
Managing Receiver Errors .....	163
UART Interrupt Structure.....	164
UART Power Down and Wake-up.....	165
<b>LCD Driver .....</b>	<b>166</b>
LCD Memory .....	170
LCD Clock Source.....	170
LCD Register.....	171
LCD Voltage Source and Biasing.....	173
LCD Reset Function.....	173
LCD Driver Output.....	173
Programming Considerations.....	186
<b>Low Voltage Detector – LVD .....</b>	<b>187</b>
LVD Register .....	187
LVD Operation.....	188
<b>Interrupts .....</b>	<b>188</b>
Interrupt Registers.....	188
Interrupt Operation .....	193
External Interrupts.....	194
A/D Converter Interrupt .....	195
Multi-function Interrupts.....	195
Timer Module Interrupts .....	195
Time Base Interrupts .....	195
Serial Interface Module Interrupt.....	197
LVD Interrupt.....	197
EEPROM Interrupt .....	197
UART Transfer Interrupt.....	198
Interrupt Wake-up Function.....	198
Programming Considerations.....	198
<b>Configuration Options.....</b>	<b>199</b>
<b>Application Circuits .....</b>	<b>199</b>
<b>Instruction Set.....</b>	<b>200</b>
Introduction .....	200
Instruction Timing .....	200
Moving and Transferring Data.....	200
Arithmetic Operations.....	200
Logical and Rotate Operation .....	201
Branches and Control Transfer .....	201
Bit Operations .....	201
Table Read Operations .....	201
Other Operations.....	201

<b>Instruction Set Summary .....</b>	<b>202</b>
Table Conventions.....	202
Extended Instruction Set.....	204
<b>Instruction Definition.....</b>	<b>206</b>
Extended Instruction Definition .....	215
<b>Package Information .....</b>	<b>222</b>
44-pin LQFP (10mm×10mm) (FP2.0mm) Outline Dimensions .....	223
48-pin LQFP (7mm×7mm) Outline Dimensions .....	224

## Features

### CPU Features

- Operating Voltage
  - ♦  $f_{SYS}=4\text{MHz}$ : 1.8V~5.5V
  - ♦  $f_{SYS}=8\text{MHz}$ : 1.8V~5.5V
  - ♦  $f_{SYS}=12\text{MHz}$ : 2.7V~5.5V
- Up to 0.33 $\mu\text{s}$  instruction cycle with 12MHz system clock at  $V_{DD}=5\text{V}$
- Power down and wake-up functions to reduce power consumption
- Oscillators
  - ♦ External High Speed Crystal – HXT
  - ♦ Internal High Speed 4/8/12MHz RC – HIRC
  - ♦ External Low Speed 32.768kHz Crystal – LXT
  - ♦ Internal 32kHz RC – LIRC
- Fully integrated internal oscillators require no external components
- Multi-mode operation: FAST, SLOW, IDLE and SLEEP
- All instructions executed in 1~3 instruction cycles
- Table read instructions
- 115 powerful instructions
- 8-level subroutine nesting
- Bit manipulation instruction

### Peripheral Features

- Flash Program Memory: 8K $\times$ 16
- RAM Data Memory: 512 $\times$ 8
- True EEPROM Memory: 512 $\times$ 8
- In Application Programming – IAP
- Watchdog Timer function
- Up to 46 bidirectional I/O lines
- LCD Driver function
  - ♦ SEGs $\times$ COMs: 32 $\times$ 4, 31 $\times$ 5, 30 $\times$ 6 or 28 $\times$ 8
  - ♦ Duty type: 1/4 duty, 1/5 duty, 1/6 duty or 1/8 duty
  - ♦ Bias level: 1/2 bias, 1/3 bias or 1/4 bias
  - ♦ Bias type: R type
  - ♦ Waveform type: type A or type B
- Programmable I/O source current for LED applications
- Four external interrupt lines shared with I/O pins
- Multiple Timer Modules for time measurement, compare match output, PWM output function or single pulse output function
- Serial Interface Module – SIM includes SPI and I<sup>2</sup>C interfaces
- Single Fully-duplex or Half-duplex Universal Asynchronous Receiver and Transmitter Interface – UART

- Dual Time Base functions for generation of fixed time interrupt signals
- 10 external channel 12-bit resolution A/D converter with internal reference voltage  $V_R$
- Low Voltage Reset function – LVR
- Low Voltage Detect function – LVD
- Package types: 44/48-pin LQFP

## General Description

The HT67F2355 is a Flash Memory A/D 8-bit high performance RISC architecture microcontroller with LCD driver designed for a wide range of applications that interface directly to analog signals and require an LCD display.

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

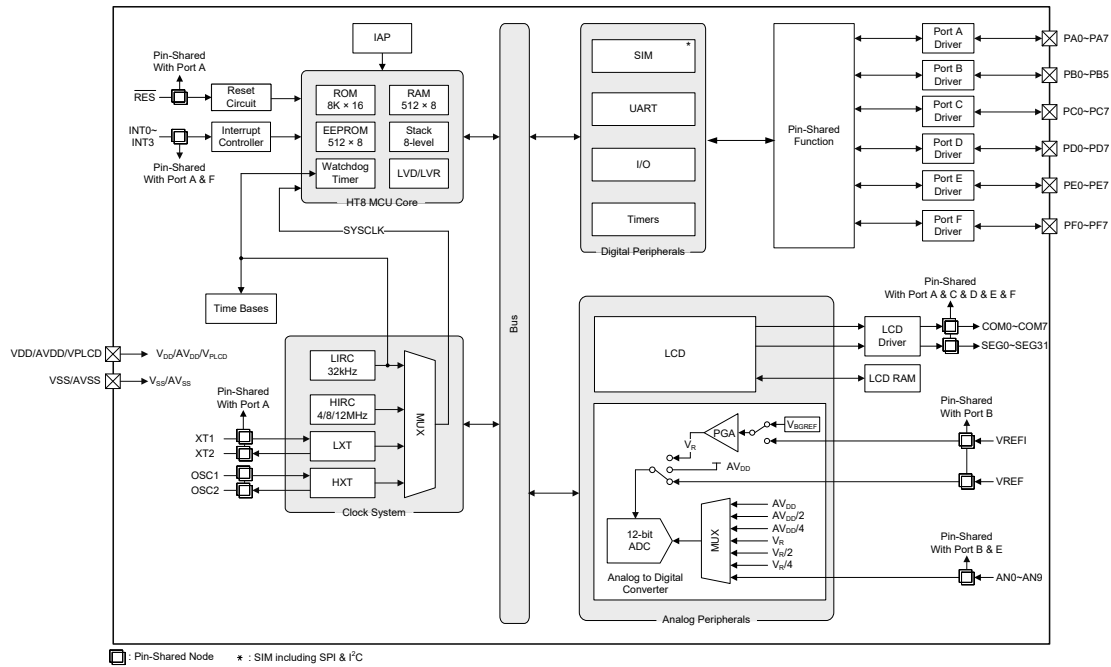
Analog features include a multi-channel 12-bit A/D converter function. Multiple and extremely flexible Timer Modules provide timing, pulse generation and PWM generation functions. Communication with the outside world is catered for by including fully integrated SPI, UART or I<sup>2</sup>C interface functions, the popular interface which provide designers with a means of easy communication with external peripheral hardware. Protective features such as an internal Watchdog Timer, Low Voltage Reset and Low Voltage Detector coupled with excellent noise immunity and ESD protection ensure that reliable operation is maintained in hostile electrical environments.

A full choice of external, internal, high and low oscillator is provided including two fully integrated system oscillators which require no external components for its implementation. The ability to operate and switch dynamically between a range of operating modes using different clock sources gives users the ability to optimise microcontroller operation and minimise power consumption. The inclusion of both LCD and LED driver functions allows for easy and cost effective solutions in applications that require interface to these display types.

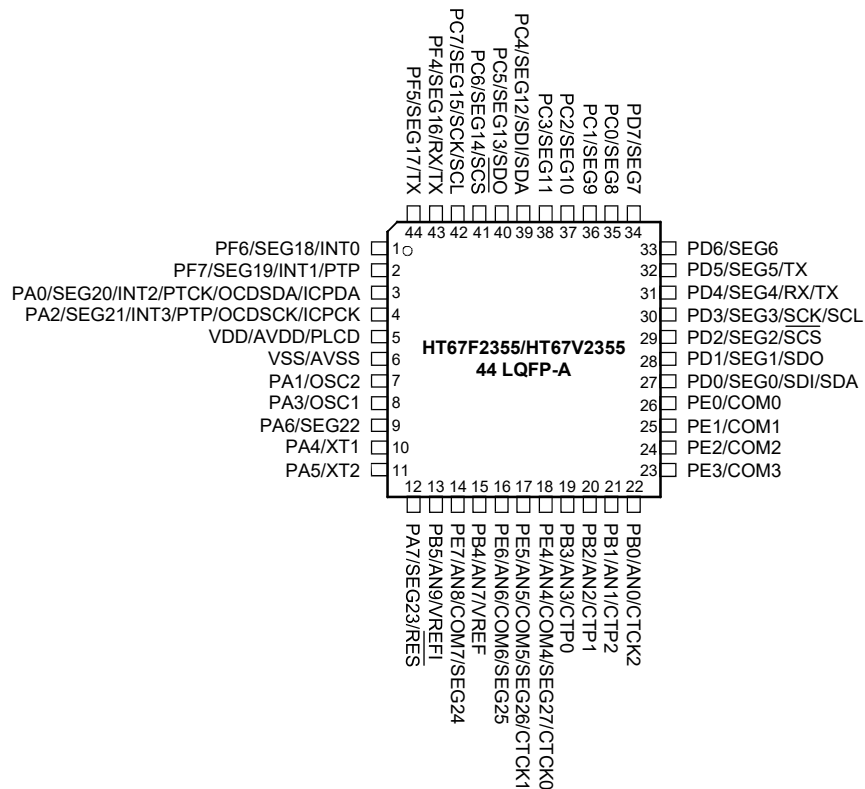
The inclusion of flexible I/O programming features, Time Base functions along with many other features ensure that the device will find excellent use in LED/LCD product applications such as temperature controllers, air conditioner remote controls, electric vehicle meters and household appliance control panels.

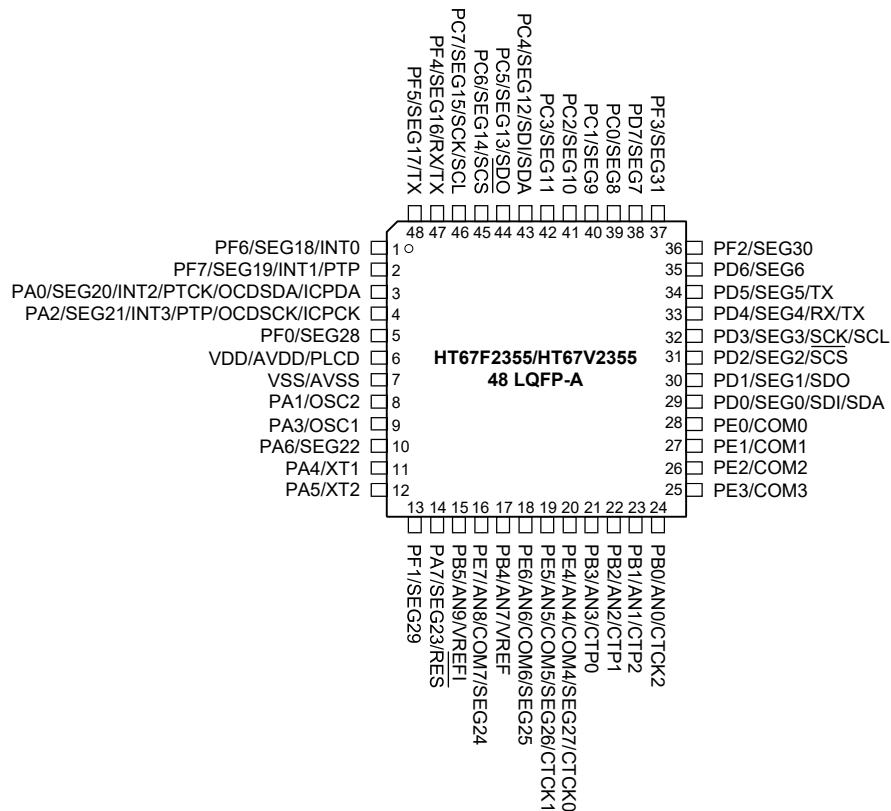


## Block Diagram



## 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 OCSDSA and OCDSCK pins are supplied as OCDS dedicated pins and as such only available for the HT67V2355 device which is the OCDS EV chip for the HT67F2355 device.
3. For less pin-count package types there will be unbonded pins which should be properly configured to avoid unwanted current consumption resulting from floating input condition. Refer to the “Standby Current Considerations” and “Input/Output Ports” sections.

## Pin Description

The function of each pin is listed in the following table, however the details behind how each pin is configured is contained in other sections of the datasheet. Note that where more than one package type exists the table will reflect the situation for the larger package type.

Pin Name	Function	OPT	I/T	O/T	Description
PA0/SEG20/INT2/ PTCK/OCSDSA/ ICPDA	PA0	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SEG20	PAS0	—	LCD	LCD segment output
	INT2	INTEG INTC1 PAS0	ST	—	External Interrupt input 2
	PTCK	PAS0	ST	—	PTM clock input
	OCSDSA	—	ST	CMOS	OCDS data/address pin, for EV chip only
	ICPDA	—	ST	CMOS	ICP data/address pin

Pin Name	Function	OPT	I/T	O/T	Description
PA1/OSC2	PA1	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OSC2	PAS0	—	HXT	HXT oscillator pin
PA2/SEG21/INT3/PTP/ OCDSCK/ICPCK	PA2	PAWU PAPU	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SEG21	PAS0	—	LCD	LCD segment output
	INT3	INTEG INTC2 PAS0	ST	—	External Interrupt input 3
	PTP	PAS0	—	CMOS	PTM output
	OCDSCK	—	ST	—	OCDS clock input, for EV chip only
	ICPCK	—	ST	CMOS	ICP clock pin
PA3/OSC1	PA3	PAWU PAPU PAS0	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	OSC1	PAS0	HXT	—	HXT oscillator pin
PA4/XT1	PA4	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	XT1	PAS1	LXT	—	LXT oscillator pin
PA5/XT2	PA5	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	XT2	PAS1	—	LXT	LXT oscillator pin
PA6/SEG22	PA6	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SEG22	PAS1	—	LCD	LCD segment output
PA7/SEG23/ $\overline{\text{RES}}$	PA7	PAWU PAPU PAS1	ST	CMOS	General purpose I/O. Register enabled pull-up and wake-up
	SEG23	PAS1	—	LCD	LCD segment output
	$\overline{\text{RES}}$	RSTC	ST	—	External reset input
PB0/AN0/CTCK2	PB0	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN0	PBS0	AN	—	A/D Converter external analog input
	CTCK2	PBS0	ST	—	CTM2 clock input
PB1/AN1/CTP2	PB1	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN1	PBS0	AN	—	A/D Converter external input channel
	CTP2	PBS0	—	CMOS	CTM2 output
PB2/AN2/CTP1	PB2	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN2	PBS0	AN	—	A/D Converter external input channel
	CTP1	PBS0	—	CMOS	CTM1 output
PB3/AN3/CTP0	PB3	PBPU PBS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN3	PBS0	AN	—	A/D Converter external input channel
	CTP0	PBS0	—	CMOS	CTM0 output

Pin Name	Function	OPT	I/T	O/T	Description
PB4/AN7/VREF	PB4	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN7	PBS1	AN	—	A/D Converter external input channel
	VREF	PBS1	AN	—	A/D Converter external reference voltage input
PB5/AN9/VREFI	PB5	PBPU PBS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN9	PBS1	AN	—	A/D Converter external input channel
	VREFI	PBS1	AN	—	A/D Converter PGA input
PC0/SEG8	PC0	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG8	PCS0	—	LCD	LCD segment output
PC1/SEG9	PC1	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG9	PCS0	—	LCD	LCD segment output
PC2/SEG10	PC2	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG10	PCS0	—	LCD	LCD segment output
PC3/SEG11	PC3	PCPU PCS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG11	PCS0	—	LCD	LCD segment output
PC4/SEG12/SDI/SDA	PC4	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG12	PCS1	—	LCD	LCD segment output
	SDI	PCS1 IFS	ST	—	SIM SPI data input
	SDA	PCS1 IFS	ST	NMOS	SIM I <sup>2</sup> C data line
PC5/SEG13/SDO	PC5	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG13	PCS1	—	LCD	LCD segment output
	SDO	PCS1	—	CMOS	SIM SPI data output
PC6/SEG14/ $\overline{\text{SCS}}$	PC6	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG14	PCS1	—	LCD	LCD segment output
	$\overline{\text{SCS}}$	PCS1 IFS	ST	CMOS	SIM SPI slave select
PC7/SEG15/SCK/SCL	PC7	PCPU PCS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG15	PCS1	—	LCD	LCD segment output
	SCK	PCS1 IFS	ST	CMOS	SIM SPI serial clock
	SCL	PCS1 IFS	ST	NMOS	SIM I <sup>2</sup> C clock line
PD0/SEG0/SDI/SDA	PD0	PDPUP PDS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG0	PDS0	—	LCD	LCD segment output
	SDI	PDS0 IFS	ST	—	SIM SPI data input
	SDA	PDS0 IFS	ST	NMOS	SIM I <sup>2</sup> C data line

Pin Name	Function	OPT	I/T	O/T	Description
PD1/SEG1/SDO	PD1	PDPUPDS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG1	PDS0	—	LCD	LCD segment output
	SDO	PDS0	—	CMOS	SIM SPI data output
PD2/SEG2/ $\overline{\text{SCS}}$	PD2	PDPUPDS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG2	PDS0	—	LCD	LCD segment output
	$\overline{\text{SCS}}$	PDS0IFS	ST	CMOS	SIM SPI slave select
PD3/SEG3/SCK/SCL	PD3	PDPUPDS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG3	PDS0	—	LCD	LCD segment output
	SCK	PDS0IFS	ST	CMOS	SIM SPI serial clock
	SCL	PDS0IFS	ST	NMOS	SIM I <sup>2</sup> C clock line
PD4/SEG4/RX/TX	PD4	PDPUPDS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG4	PDS1	—	LCD	LCD segment output
	RX/TX	PDS1IFS	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication
PD5/SEG5/TX	PD5	PDPUPDS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG5	PDS1	—	LCD	LCD segment output
	TX	PDS1	ST	—	UART TX serial data output
PD6/SEG6	PD6	PDPUPDS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG6	PDS1	—	LCD	LCD segment output
PD7/SEG7	PD7	PDPUPDS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG7	PDS1	—	LCD	LCD segment output
PE0/COM0	PE0	PEPUPES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	COM0	PES0	—	LCD	LCD segment output
PE1/COM1	PE1	PEPUPES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	COM1	PES0	—	LCD	LCD common output
PE2/COM2	PE2	PEPUPES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	COM2	PES0	—	LCD	LCD common output
PE3/COM3	PE3	PEPUPES0	ST	CMOS	General purpose I/O. Register enabled pull-up
	COM3	PES0	—	LCD	LCD common output
PE4/AN4/COM4/SEG27/CTCK0	PE4	PEPUPES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN4	PES1	AN	—	A/D Converter external input channel
	COM4	PES1	—	LCD	LCD common output
	SEG27	PES1	—	LCD	LCD segment output
	CTCK0	PES1	ST	—	CTM0 clock input

Pin Name	Function	OPT	I/T	O/T	Description
PE5/AN5/COM5/ SEG26/CTCK1	PE5	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN5	PES1	AN	—	A/D Converter external input channel
	COM5	PES1	—	LCD	LCD common output
	SEG26	PES1	—	LCD	LCD segment output
	CTCK1	PES1	ST	—	CTM1 clock input
PE6/AN6/COM6/ SEG25	PE6	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN6	PES1	AN	—	A/D Converter external input channel
	COM6	PES1	—	LCD	LCD common output
	SEG25	PES1	—	LCD	LCD segment output
PE7/AN8/COM7/ SEG24	PE7	PEPU PES1	ST	CMOS	General purpose I/O. Register enabled pull-up
	AN8	PES1	AN	—	A/D Converter external input channel
	COM7	PES1	—	LCD	LCD common output
	SEG24	PES1	—	LCD	LCD segment output
PF0/SEG28	PF0	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG28	PFS0	—	LCD	LCD segment output
PF1/SEG29	PF1	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG29	PFS0	—	LCD	LCD segment output
PF2/SEG30	PF2	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG30	PFS0	—	LCD	LCD segment output
PF3/SEG31	PF3	PFPU PFS0	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG31	PFS0	—	LCD	LCD segment output
PF4/SEG16/RX/TX	PF4	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG16	PFS1	—	LCD	LCD segment output
	RX/TX	PFS1 IFS	ST	CMOS	UART serial data input in full-duplex communication or UART serial data input/output in Single Wire Mode communication
PF5/SEG17/TX	PF5	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG17	PFS1	—	LCD	LCD segment output
	TX	PFS1	—	CMOS	UART TX serial data output
PF6/SEG18/INT0	PF6	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG18	PFS1	—	LCD	LCD segment output
	INT0	PFS1 INTEG INTC0	ST	—	External Interrupt 0
PF7/SEG19/INT1/PTP	PF7	PFPU PFS1	ST	CMOS	General purpose I/O. Register enabled pull-up
	SEG19	PFS1	—	LCD	LCD segment output
	INT1	PFS1 INTEG INTC0	ST	—	External Interrupt 1
	PTP	PFS1	—	CMOS	PTM output

Pin Name	Function	OPT	I/T	O/T	Description
VDD/AVDD/PLCD	VDD	—	PWR	—	Positive power supply
	AVDD	—	PWR	—	Analog positive power supply
	PLCD	—	PWR	AN	LCD power supply
VSS/AVSS	VSS	—	PWR	—	Negative power supply, ground
	AVSS	—	PWR	—	Analog negative power supply, ground

Legend: I/T: Input type

OPT: Optional by register option

CMOS: CMOS output

AN: Analog signal

LXT: Low frequency crystal oscillator

LCD: LCD SEG/COM output.

O/T: Output type

ST: Schmitt Trigger input

NMOS: NMOS output

PWR: Power

HXT: High frequency crystal oscillator

## Absolute Maximum Ratings

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

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

## D.C. Characteristics

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

### Operating Voltage Characteristics

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

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
$V_{DD}$	Operating Voltage – HXT	$f_{SYS}=4MHz$	1.8	—	5.5	V
		$f_{SYS}=8MHz$	1.8	—	5.5	
		$f_{SYS}=12MHz$	2.7	—	5.5	
	Operating Voltage – HIRC	$f_{SYS}=4MHz$	1.8	—	5.5	
		$f_{SYS}=8MHz$	1.8	—	5.5	
		$f_{SYS}=12MHz$	2.7	—	5.5	
	Operating Voltage – LXT	$f_{SYS}=32768Hz$	1.8	—	5.5	
	Operating Voltage – LIRC	$f_{SYS}=32kHz$	1.8	—	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	1.8V	f <sub>sys</sub> =32kHz	—	3	24	μA
		3V		—	4	30	
		5V		—	14	40	
	SLOW Mode – LXT	1.8V	f <sub>sys</sub> =32768Hz	—	3.5	24.5	μA
		3V		—	5	31	
		5V		—	16	42	
	FAST Mode – HIRC	1.8V	f <sub>sys</sub> =4MHz	—	200	500	μA
		3V		—	250	800	
		5V		—	500	1300	
		1.8V	f <sub>sys</sub> =8MHz	—	320	700	
		3V		—	500	1000	
		5V		—	1000	1600	
		2.7V	f <sub>sys</sub> =12MHz	—	0.7	1.2	mA
		3V		—	0.75	1.60	
		5V		—	1.5	2.4	
	FAST Mode – HXT	1.8V	f <sub>sys</sub> =4MHz	—	200	500	μA
		3V		—	250	800	
		5V		—	500	1300	
		1.8V	f <sub>sys</sub> =8MHz	—	320	700	
		3V		—	500	1000	
		5V		—	1000	1600	
		2.7V	f <sub>sys</sub> =12MHz	—	0.7	1.2	mA
		3V		—	0.75	1.60	
		5V		—	2.0	2.4	

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

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

### Standby Current Characteristics

Ta=25°C, unless otherwise specify

Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V <sub>DD</sub>	Conditions					
I <sub>STB</sub>	SLEEP Mode	1.8V	WDT off	—	0.5	0.8	5.0	μA
		3V		—	0.6	1.0	6.0	
		5V		—	0.7	2.0	8.0	
		1.8V	WDT on	—	1.5	3.0	7.0	
		3V		—	1.8	3.6	8.0	
		5V		—	3	5	10	
	IDLE0 Mode – LIRC	1.8V	f <sub>SUB</sub> on	—	2.4	4.0	8.0	μA
		3V		—	3	5	12	
		5V		—	5	10	15	



Symbol	Standby Mode	Test Conditions		Min.	Typ.	Max.	Max. @85°C	Unit
		V <sub>DD</sub>	Conditions					
I <sub>STB</sub>	IDLE0 Mode – LXT	1.8V	f <sub>SUB</sub> on	—	2.4	4.0	8.0	μA
		3V		—	3	5	12	
		5V		—	5	10	15	
	IDLE1 Mode – HIRC	1.8V	f <sub>SUB</sub> on, f <sub>SYS</sub> =4MHz	—	144	200	240	μA
		3V		—	180	250	300	
		5V		—	400	600	720	
		1.8V	f <sub>SUB</sub> on, f <sub>SYS</sub> =8MHz	—	288	400	480	
		3V		—	360	500	600	
		5V		—	600	800	960	
		2.7V	f <sub>SUB</sub> on, f <sub>SYS</sub> =12MHz	—	432	600	720	
		3V		—	540	750	900	
		5V		—	800	1200	1440	
	IDLE1 Mode – HXT	1.8V	f <sub>SUB</sub> on, f <sub>SYS</sub> =4MHz	—	144	200	240	μA
		3V		—	180	250	300	
		5V		—	400	660	720	
		1.8V	f <sub>SUB</sub> on, f <sub>SYS</sub> =8MHz	—	288	400	480	
		3V		—	360	500	600	
		5V		—	600	880	960	
		2.7V	f <sub>SUB</sub> on, f <sub>SYS</sub> =12MHz	—	500	600	720	
		3V		—	600	750	900	
		5V		—	1600	2000	2200	

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

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

## A.C. Characteristics

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

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

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Temp.				
f <sub>HIRC</sub>	4MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	4	+1%	MHz
			-40°C~85°C	-2%	4	+2%	
		2.2V~5.5V	25°C	-2.5%	4	+2.5%	
			-40°C~85°C	-3%	4	+3%	
		1.8V~5.5V	25°C	-4%	4	+4%	
			-40°C~85°C	-5%	4	+5%	

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Temp.				
f <sub>HIRC</sub>	8MHz Writer Trimmed HIRC Frequency	3V/5V	25°C	-1%	8	+1%	MHz
			-40°C~85°C	-2%	8	+2%	
		2.2V~5.5V	25°C	-2.5%	8	+2.5%	
			-40°C~85°C	-3%	8	+3%	
		1.8V~5.5V	25°C	-5%	8	+5%	
			-40°C~85°C	-10%	8	+10%	
	12MHz Writer Trimmed HIRC Frequency	5V	25°C	-1%	12	+1%	
			-40°C~85°C	-2%	12	+2%	
		2.7V~5.5V	25°C	-2.5%	12	+2.5%	
			-40°C~85°C	-3%	12	+3%	

Note: 1. The 3V/5V values for V<sub>DD</sub> are provided as these are the two selectable fixed voltages at which the HIRC frequency is trimmed by the writer.

2. The row below the 3V/5V trim voltage row is provided to show the values for the full V<sub>DD</sub> range operating voltage. It is recommended that the trim voltage is fixed at 3V for application voltage ranges from 1.8V to 3.6V and fixed at 5V for application voltage ranges from 3.3V to 5.5V.

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

#### Low Speed Internal Oscillator Characteristics – LIRC

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <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%	
		1.8V~5.5V	-40°C~85°C	-10%	32	+10%	
t <sub>START</sub>	LIRC Start Up Time	—	—	—	—	100	μs

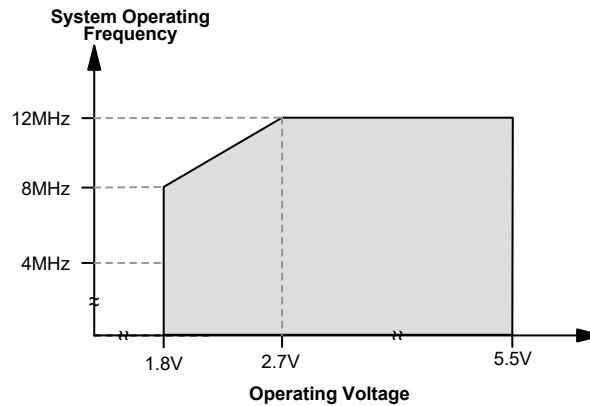
#### Low Speed Crystal Oscillator Characteristics – LXT

T<sub>a</sub>=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
f <sub>LXT</sub>	LXT Frequency	1.8V~5.5V	—	—	32768	—	Hz
t <sub>START</sub>	LXT Start Up Time	3V	—	—	—	1000	ms
		5V	—	—	—	1000	
Duty Cycle	Duty Cycle	—	—	40	—	60	%
R <sub>NEG</sub>	Negative Resistance	1.8V	—	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 condition where f <sub>SYS</sub> is off	—	f <sub>SYS</sub> =f <sub>H</sub> ~f <sub>H</sub> /64, f <sub>H</sub> =f <sub>HXT</sub>	—	128	—	t <sub>HXT</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>HIRC</sub>
		—	f <sub>SYS</sub> =f <sub>SUB</sub> =f <sub>LXT</sub>	—	1024	—	t <sub>LXT</sub>
		—	f <sub>SYS</sub> =f <sub>SUB</sub> =f <sub>LIRC</sub>	—	2	—	t <sub>LIRC</sub>
	System Start-up Time Wake-up from condition 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>	—	2	—	t <sub>H</sub>
		—	f <sub>SYS</sub> =f <sub>SUB</sub> =f <sub>LXT</sub> OR f <sub>LIRC</sub>	—	2	—	t <sub>SUB</sub>
	System Speed Switch Time FAST to SLOW Mode or SLOW to FAST Mode	—	f <sub>HXT</sub> switches from off → on	—	1024	—	t <sub>HXT</sub>
		—	f <sub>HIRC</sub> switches from off → on	—	16	—	t <sub>HIRC</sub>
		—	f <sub>LXT</sub> switches from off → on	—	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/WDTC/RSTC Software Reset	—	—	—	—	—	—
	System Reset Delay Time (Reset Source from WDT Overflow or RES Pin Reset)	—	—	14	16	18	ms

- 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>HXT</sub>, 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=25°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>IL</sub>	Input Low Voltage for I/O Ports or Input Pins	5V	—	0	—	1.5	V
		—	—	0	—	0.2V <sub>DD</sub>	
V <sub>IH</sub>	Input High Voltage for I/O Ports or Input Pins	5V	—	3.5	—	5.0	V
		—	—	0.8V <sub>DD</sub>	—	V <sub>DD</sub>	
I <sub>OL</sub>	Sink Current for I/O Pins	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	16	32	—	mA
		5V		32	65	—	
I <sub>OH</sub>	Source Current for I/O Pins	3V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDCn[m+1:m]=00, (n=0~2; m=0, 2, 4, 6)	-0.7	-1.5	—	mA
		5V		-1.5	-2.9	—	
		3V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDCn[m+1:m]=01, (n=0~2; m=0, 2, 4, 6)	-1.3	-2.5	—	
		5V		-2.5	-5.1	—	
		3V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDCn[m+1:m]=10, (n=0~2; m=0, 2, 4, 6)	-1.8	-3.6	—	
		5V		-3.6	-7.3	—	
		3V	V <sub>OH</sub> =0.9V <sub>DD</sub> , SLEDCn[m+1:m]=11, (n=0~2; m=0, 2, 4, 6)	-4	-8	—	
		5V		-8	-16	—	
R <sub>PH</sub>	Pull-high Resistance for I/O Ports (Note)	3V	LVPU=0	20	60	100	kΩ
		5V	PxPU=FFH (Px: PA~PF)	10	30	50	
		3V	LVPU=1	6.67	15.00	23.00	
		5V	PxPU=FFH (Px: PA~PF)	3.5	7.5	12.0	
I <sub>LEAK</sub>	Input Leakage Current for I/O Ports	3V	V <sub>IN</sub> =V <sub>DD</sub> or V <sub>IN</sub> =V <sub>SS</sub>	—	—	±1	μA
		5V		—	—	±1	
t <sub>TCK</sub>	TM Clock Input Pin Minimum Pulse Width	—	—	0.3	—	—	μs
t <sub>INT</sub>	External Interrupt Minimum Pulse Width	—	—	10	—	—	μs
t <sub>SRESET</sub>	Minimum Software Reset Width to Reset	—	—	45	90	120	μs

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-up 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.

## A/D Converter Electrical Characteristics

Ta=25°C, unless otherwise specify

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>ADI</sub>	Input Voltage	—	—	0	—	V <sub>REF</sub>	V
V <sub>REF</sub>	Reference Voltage	—	—	1.8	—	V <sub>DD</sub>	V
N <sub>R</sub>	Resolution	—	—	—	—	12	Bit
DNL	Differential Non-linearity	1.8V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =2.0μs	-3	—	3	LSB
		2V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs				
		3V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs				
		5V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs				
		1.8V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =10μs				
		3V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =10μs				
INL	Integral Non-linearity	1.8V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =2.0μs Ta=-40°C~85°C	-4	—	4	LSB
		2V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs Ta=-40°C~85°C				
		3V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs Ta=-40°C~85°C				
		5V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =0.5μs Ta=-40°C~85°C				
		1.8V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =10μs Ta=-40°C~85°C				
		3V	SAINS[3:0]=0000B, SAVRS[1:0]=01B, V <sub>REF</sub> =V <sub>DD</sub> , t <sub>ADCK</sub> =10μs Ta=-40°C~85°C				
I <sub>ADC</sub>	Additional Current Consumption for A/D Converter Enable	1.8V	No load, t <sub>ADCK</sub> =2.0μs	—	280	400	μA
		3V	No load, t <sub>ADCK</sub> =0.5μs	—	450	600	μA
		5V	No load, t <sub>ADCK</sub> =0.5μs	—	850	1000	μA
t <sub>ADCK</sub>	Clock Period	—	1.8V≤V <sub>DD</sub> <2.0V	2	—	10	μs
			2.0V≤V <sub>DD</sub> ≤5.5V	0.5	—	10.0	μs
t <sub>ON2ST</sub>	A/D Converter On-to-Start Time	—	—	4	—	—	μs
t <sub>ADC</sub>	Conversion Time (Including A/D Converter Sample and Hold Time)	—	—	—	16	—	t <sub>ADCK</sub>
I <sub>PGA</sub>	Additional Current for PGA Enable	2.2V	No load, PGAIS=1, PGAGS[1:0]=01	—	250	500	μA
		3V		—	300	600	
		5V		—	400	700	
V <sub>OR</sub>	PGA Maximum Output Voltage Range	2.2V	—	V <sub>SS</sub> +0.1	—	V <sub>DD</sub> -0.1	V
		3V					
		5V					
V <sub>VR</sub>	Fix Voltage Output of PGA	2.2V~5.5V	Ta=-40°C~85°C, V <sub>RI</sub> =V <sub>BGREF</sub> (PGAIS=1)	-1%	2	+1%	V
		3.2V~5.5V		-1%	3	+1%	
		4.2V~5.5V		-1%	4	+1%	
V <sub>IR</sub>	PGA Input Voltage Range	3V	Gain=1, PGAIS=0 Relative gain Gain error<±5%	V <sub>SS</sub> +0.1	—	V <sub>DD</sub> -1.4	V
		5V		V <sub>SS</sub> +0.1	—	V <sub>DD</sub> -1.4	V

## Memory Electrical Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>RW</sub>	V <sub>DD</sub> for Read/Write	—	—	V <sub>DDmin</sub>	—	V <sub>DDmax</sub>	V
<b>Flash Program Memory</b>							
t <sub>FWR</sub>	Write Time	—	—	1.364	1.500	1.667	ms
t <sub>FER</sub>	Erase Time	—	—	2.273	2.500	2.778	ms
E <sub>P</sub>	Cell Endurance	—	—	10K	—	—	E/W
t <sub>RETD</sub>	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
t <sub>ACTV</sub>	ROM Activation Time – Wake-up from Power Down Mode <sup>(Note)</sup>	—	—	32	—	64	μs
<b>Data EEPROM Memory</b>							
t <sub>EEWR</sub>	Write Time (byte mode)	—	EWERTS bit=0	—	5.4	6.6	ms
		—	EWERTS bit=1	—	6.7	8.1	ms
	Write Time (page mode)	—	EWERTS bit=0	—	2.2	2.7	ms
		—	EWERTS bit=1	—	3.0	3.6	ms
t <sub>EEER</sub>	Erase Time	—	EWERTS bit=0	—	3.2	3.9	ms
		—	EWERTS bit=1	—	3.7	4.5	ms
E <sub>P</sub>	Cell Endurance	—	—	100K	—	—	E/W
t <sub>RETD</sub>	ROM Data Retention Time	—	Ta=25°C	—	40	—	Year
<b>RAM Data Memory</b>							
V <sub>DR</sub>	RAM Data Retention Voltage	—	—	1	—	—	V

Note: 1. 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 power down mode.

2. “E/W” means Erase/Write times.

## Internal Reference Voltage Characteristics

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

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
I <sub>BGREF</sub>	Operating Current	5.5V	—	—	25	35	μA
PSRR	Power Supply Rejection Ratio	—	Ta=25°C, V <sub>RIIPPLE</sub> =1V <sub>P-P</sub> , f <sub>RIIPPLE</sub> =100Hz	75	—	—	dB
En	Output Noise	—	Ta=25°C, no load current, f=0.1Hz~10Hz	—	300	—	μV <sub>RMS</sub>
I <sub>SD</sub>	Shutdown Current	—	VBGREN=0	—	—	0.1	μA
t <sub>START</sub>	Start Up Time	1.8V~5.5V	Ta=25°C	—	—	400	μs

Note: 1. All the above parameters are measured under conditions of no load condition unless otherwise described.

2. A 0.1μF ceramic capacitor should be connected between VDD and GND.

## LVD/LVR Electrical Characteristics

Ta=-40°C~85°C

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>LVR</sub>	Low Voltage Reset Voltage	—	LVR enable, voltage select 1.7V	-5%	1.7	+5%	V
			LVR enable, voltage select 1.9V		1.9		
			LVR enable, voltage select 2.55V	-3%	2.55	+3%	
			LVR enable, voltage select 3.15V		3.15		
			LVR enable, voltage select 3.8V		3.8		
V <sub>LVD</sub>	Low Voltage Detection Voltage	—	LVD enable, voltage select 1.8V	-5%	1.8	+5%	V
			LVD enable, voltage select 2.0V		2.0		
			LVD enable, voltage select 2.4V		2.4		
			LVD enable, voltage select 2.7V		2.7		
			LVD enable, voltage select 3.0V		3.0		
			LVD enable, voltage select 3.3V		3.3		
			LVD enable, voltage select 3.6V		3.6		
			LVD enable, voltage select 4.0V		4.0		
I <sub>LVRLVD</sub>	Operating Current	3V	LVD enable, LVR enable, V <sub>LVR</sub> =1.9V, V <sub>LVD</sub> =2V	—	—	10	μA
		5V		—	10	15	
t <sub>LVDS</sub>	LVDO Stable Time	—	For LVR enable, LVD off → on	—	—	18	μs
		—	For LVR disable, LVD off → on	—	—	150	μs
t <sub>LVR</sub>	Minimum Low Voltage Width to Reset	—	—	120	240	480	μs
t <sub>LVD</sub>	Minimum Low Voltage Width to Interrupt	—	—	60	120	240	μs
I <sub>LVR</sub>	Additional Current for LVR Enable	5V	LVD disable	—	—	14	μA
I <sub>LVD</sub>	Additional Current for LVD Enable	5V	LVR disable	—	—	14	μA

## LCD Electrical Characteristics

Ta=-40°C~85°C

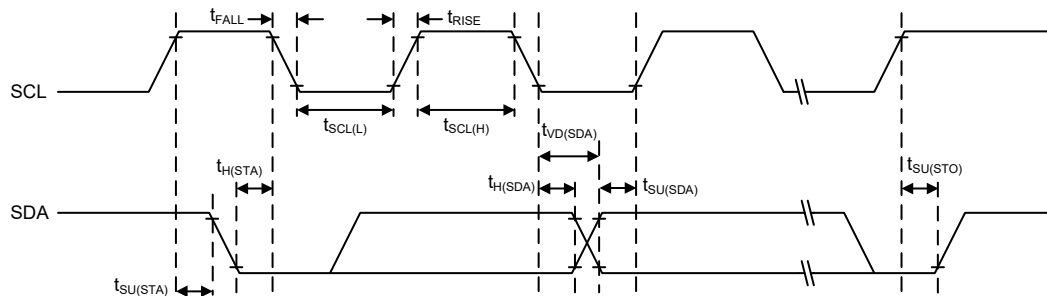
Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
V <sub>IN</sub>	LCD Operating Voltage	—	Power supply from PLCD pin, PLCD[3:0]=1xxxB	2.1	—	5.5	V
I <sub>LCD</sub>	Additional Current for LCD Enable	3V	No load, R <sub>T</sub> =1170kΩ, V <sub>A</sub> =V <sub>PLCD</sub> =V <sub>DD</sub> , 1/3 bias	—	3	6	μA
		5V		—	5	10	
		3V	No load, R <sub>T</sub> =225kΩ, V <sub>A</sub> =V <sub>PLCD</sub> =V <sub>DD</sub> , 1/3 bias	—	16	28	
		5V		—	21	40	
		3V	No load, R <sub>T</sub> =60kΩ, V <sub>A</sub> =V <sub>PLCD</sub> =V <sub>DD</sub> , 1/3 bias & 1/4 bias	—	50	75	
		5V		—	80	120	
I <sub>LCDOL</sub>	LCD Common and Segment Sink Current	3V	V <sub>OL</sub> =0.1V <sub>DD</sub>	210	420	—	μA
		5V		350	700	—	
I <sub>LCDOH</sub>	LCD Common and Segment Source Current	3V	V <sub>OH</sub> =0.9V <sub>DD</sub>	-80	-160	—	μA
		5V		-180	-360	—	

## 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	—	0.6	—	—	μs
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.



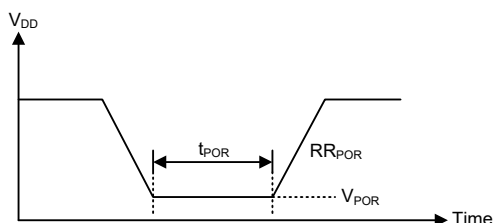
I<sup>2</sup>C Timing Diagram



## Power-on Reset Characteristics

Ta=-40°C~85°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



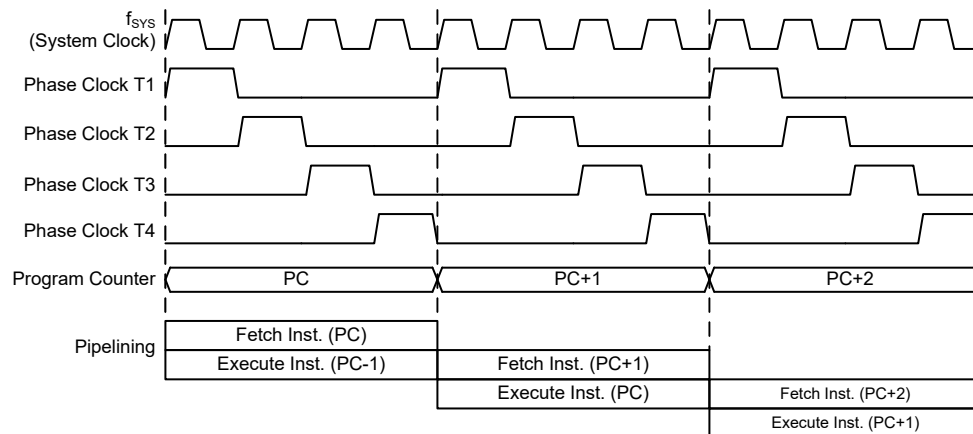
## System Architecture

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 this 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.

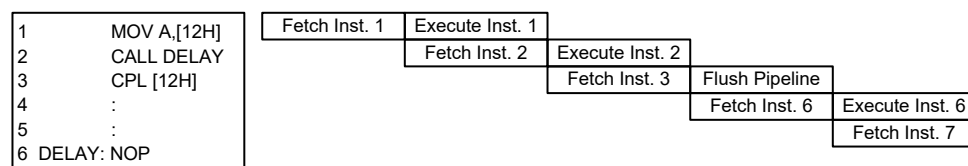
### Clocking and Pipelining

The main system clock, derived from either an HIRC, LIRC, HXT 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 demands a jump to a non-consecutive Program Memory address. Only the lower 8 bits, known as the Program Counter Low Register, are directly addressable by the application program.

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

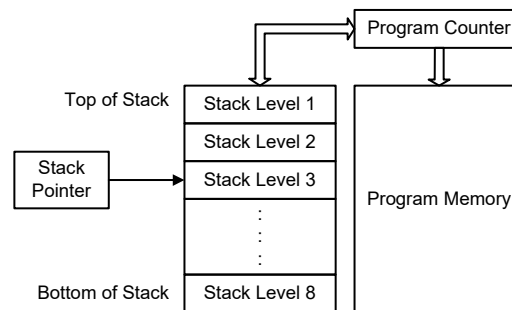
Program Counter	
High Byte	PCL Register
PC12~PC8	PCL7~PCL0

**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 prefetch.

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



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

## Arithmetic and Logic Unit – ALU

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

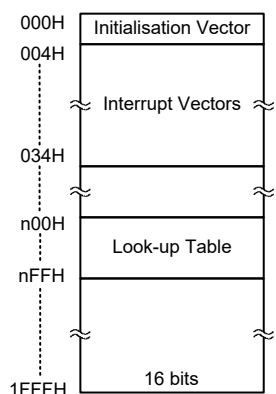
- Arithmetic operations:  
ADD, ADDM, ADC, ADCM, SUB, SUBM, SBC, SBCM, DAA,  
LADD, LADDM, LADC, LADCM, LSUB, LSUBM, LSBC, LSBCM, LDAA
- Logic operations:  
AND, OR, XOR, ANDM, ORM, XORM, CPL, CPLA,  
LAND, LANDM, LOR, LORM, LXOR, LXORM, LCPL, LCPLA
- Rotation:  
RRA, RR, RRCA, RRC, RLA, RL, RLCA, RLC,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

## Flash Program Memory

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

### Structure

The Program Memory has a capacity of 8K×16 bits. The Program Memory is addressed by the Program Counter and also contains data, table information and interrupts entries. Table data, which can be setup in any location within the Program Memory, is addressed by a separate table pointer register.



**Program Memory Structure**

### Special Vectors

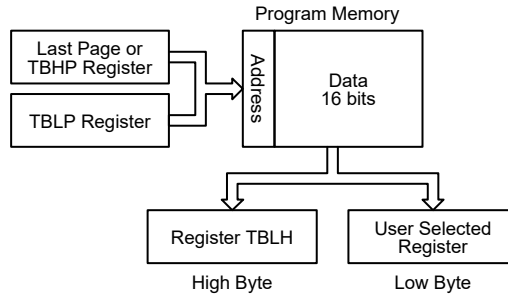
Within the Program Memory, certain locations are reserved for the reset and interrupts. The location 0000H is reserved for use by the device reset for program initialisation. After a device reset is initiated, the program will jump to this location and begin execution.

### 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 register, TBLP and TBHP. These registers define the total address of the look-up table.

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

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



### Table Program Example

The following example shows how the table pointer and table data is defined and retrieved from the microcontroller. This example uses raw table data located in the Program Memory which is stored there using the ORG statement. The value at this ORG statement is “1F00H” which refers to the start address of the last page within the 8K Program Memory of the microcontroller. The table pointer low byte register is setup here to have an initial value of “06H”. This will ensure that the first data read from the data table will be at the Program Memory address “1F06H” or 6 locations after the start of the last page. Note that the value for the table pointer is referenced to the address specified by TBLP and TBHP if the “TABRD [m]” or “LTABRD [m]” instruction is being used. The high byte of the table data which in this case is equal to zero will be transferred to the TBLH register automatically when the “TABRD [m]” or “LTABRD [m]” instruction is executed.

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

### Table Read Program Example

```

tempreg1 db ? ; temporary register #1
tempreg2 db ? ; temporary register #2
:
:
mov a,06h      ; initialise low table pointer - note that this address is referenced
mov tblp,a    ; to the last page or the page that tbhp pointed
mov a,1Fh     ; initialise high table pointer
mov tbhp,a
:
:
tabrd tempreg1 ; transfers value in table referenced by table pointer data at program
               ; memory address "1F06H" 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 "1F05H" transferred to
               ; tempreg2 and TBLH in this example the data "1AH" is
               ; transferred to tempreg1 and data "0FH" to register tempreg2
:
org 1F00h      ; sets initial address of program memory
dc 00Ah, 00Bh, 00Ch, 00Dh, 00Eh, 00Fh, 01Ah, 01Bh
:

```

## In Circuit Programming – ICP

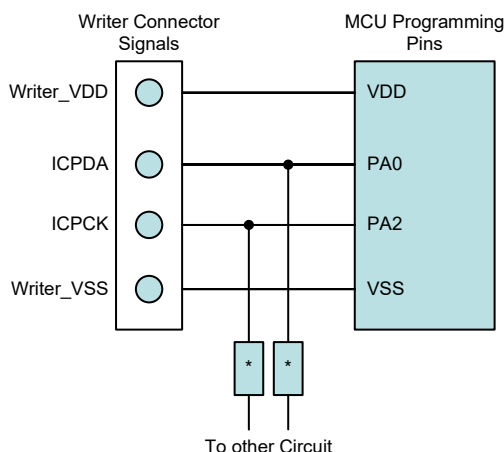
The provision of Flash type Program Memory provides the user with a means of convenient and easy upgrades and modifications to their programs on the same device. As an additional convenience, Holtek has provided a means of programming the microcontroller in-circuit using a 4-pin interface. This provides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

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

Holtek Writer Pins	MCU Programming Pins	Pin Description
ICPDA	PA0	Programming serial data/address
ICPCK	PA2	Programming clock
VDD	VDD	Power supply
VSS	VSS	Ground

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

There is an EV chip named HT67V2355 which is used to emulate the real MCU device named HT67F2355. The EV chip device also provides an “On-Chip Debug” function to debug the real MCU device during the development process. The EV chip and the real MCU device are almost functionally compatible except for “On-Chip Debug” function. Users can use the EV chip device to emulate the real chip device behavior by connecting the OCSDA and OCDSCK pins to the Holtek HT-IDE development tools. The OCSDA pin is the OCDS Data/Address input/output pin while the OCDSCK pin is the OCDS clock input pin. When users use the EV chip for debugging, other functions which are shared with the OCSDA and OCDSCK pins in the device will have no effect

in the EV chip. For more detailed OCDS information, refer to the corresponding document named “Holtek e-Link for 8-bit MCU OCDS User’s Guide”.

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

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

### Flash Memory Read/Write Size

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

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

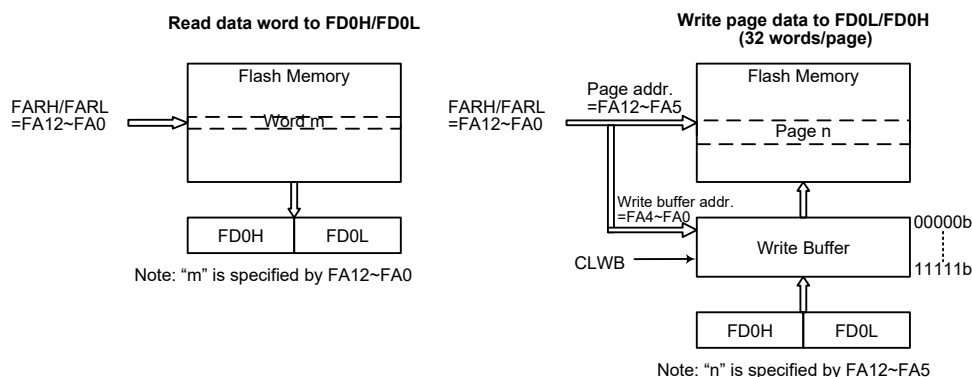
The Read operation can be carried out by executing a specific read procedure. The FRDEN bit is used to enable the read function and the FRD bit is used to initiate the read process by application programs and then indicate the read operation status. When the read process is finished, this bit will be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

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

### IAP Read/Erase/Write Format

Page	FARH	FARL [7:5]	FARL [4:0]
0	0000 0000	000	Tag Address
1	0000 0000	001	
2	0000 0000	010	
3	0000 0000	011	
4	0000 0000	100	
5	0000 0000	101	
6	0000 0000	110	
7	0000 0000	111	
8	0000 0001	000	
9	0000 0001	001	
:	:	:	
254	0001 1111	110	
255	0001 1111	111	

**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 FC2 register. The CLWB bit can be set high to enable the Clear Write Buffer procedure. When the procedure is finished this bit will be cleared to low by the hardware. It is recommended that the write buffer should be cleared by setting the CLWB bit high before the write buffer is used for the first time or when the data in the write buffer is updated.

The write buffer size is 32 words corresponding to a page. The write buffer address is mapped to a specific Flash memory page specified by the memory address bits, FA12~FA5. The data written into the FD0L and FD0H registers will be loaded into the write buffer. When data is written into the high byte data register, FD0H, it will result in the data stored in the high and low byte data registers both being written into the write buffer. It will also cause the flash memory address to be incremented by one, after which the new address will be loaded into the FARH and FARL address registers. When the flash memory address reaches the page boundary, 11111b of a page with 32 words, the address will now not be incremented but will stop at the last address of the page. At this point a new page address should be specified for any other erase/write operations.



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

### IAP Flash Program Memory Registers

There are two address registers, four 16-bit data registers and three control registers, located in Sector 1. 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 register. Several registers control the overall operation of the internal Flash Program Memory.

Register Name	Bit							
	7	6	5	4	3	2	1	0
FC0	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
FC1	D7	D6	D5	D4	D3	D2	D1	D0
FC2	—	—	—	—	—	—	FWERTS	CLWB
FARL	FA7	FA6	FA5	FA4	FA3	FA2	FA1	FA0
FARH	—	—	—	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

#### • FC0 Register

Bit	7	6	5	4	3	2	1	0
Name	CFWEN	FMOD2	FMOD1	FMOD0	FWPEN	FWT	FRDEN	FRD
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **CFWEN**: Flash Memory Erase/Write 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 the hardware, it means that the Flash memory erase/write function is enabled successfully. Otherwise, the Flash memory erase/write function is disabled if the bit is zero.

Bit 6~4 **FMOD2~FMOD0**: Flash memory mode selection

000: Write Mode

001: Page Erase Mode

010: Reserved

011: Read Mode

100: Reserved

101: Reserved

110: Flash memory Erase/Write function Enable Mode

111: Reserved

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

- Bit 3**      **FWPEN:** Flash memory Erase/Write function enable procedure trigger  
               0: Erase/Write function enable procedure is not triggered or procedure timer times out  
               1: Erase/Write function enable procedure is triggered and procedure timer starts to count
- This bit is used to activate the Flash memory Erase/Write function enable procedure and an internal timer. It is set by the application programs and then cleared 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 FWPEN bit is set high.
- Bit 2**      **FWT:** Flash memory write initiate control  
               0: Do not initiate Flash memory write or indicating that a Flash memory write process has completed  
               1: Initiate Flash memory write process
- This bit is set by software and cleared to 0 by the 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 to 0 by the 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/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/write operation is totally complete before executing other operations.

• **FC1 Register**

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

- Bit 7~0**      **D7~D0:** Chip reset pattern
- When a specific value of “55H” is written into this register, a reset signal will be generated to reset the whole chip.

• FC2 Register

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

Bit 7~2 Unimplemented, read as “0”

Bit 1 **FWERTS**: Erase time and Write time select  
 0: Erase time is 3.2ms ( $t_{FER}$ )/Write time is 2.2ms ( $t_{FWR}$ )  
 1: Erase time is 3.7ms ( $t_{FER}$ )/Write time is 3.0ms ( $t_{FWR}$ )

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 to 0 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	—	—	—	FA12	FA11	FA10	FA9	FA8
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 **FA12~FA8**: Flash memory address bit 12 ~ 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

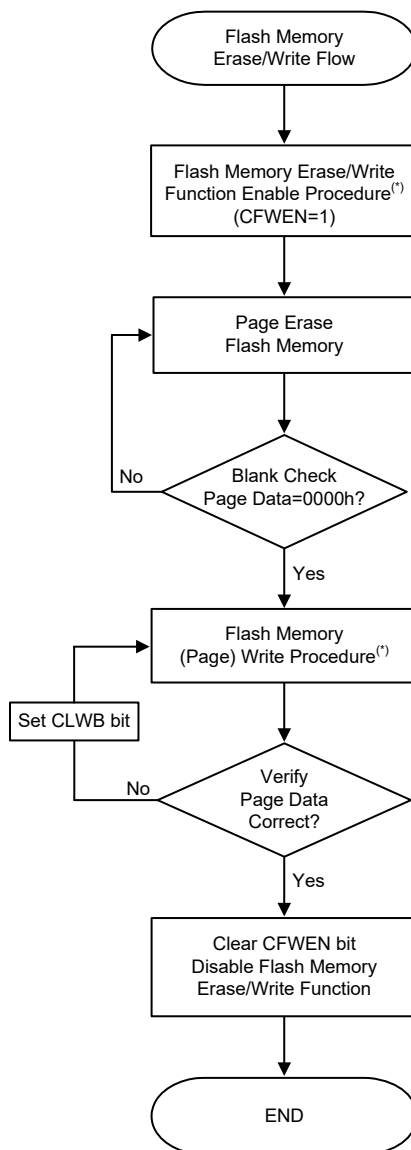
Bit 7~0      **D15~D8**: The fourth Flash memory data word bit 15 ~ bit 8

### **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 FC0 register will automatically be set high by hardware. After this, Erase or Write operations can be executed on the Flash memory. Refer to the “Flash Memory Erase/Write Function Enable Procedure” for details.
2. Configure the flash memory address to select the desired erase page, 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, 11111b, 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.

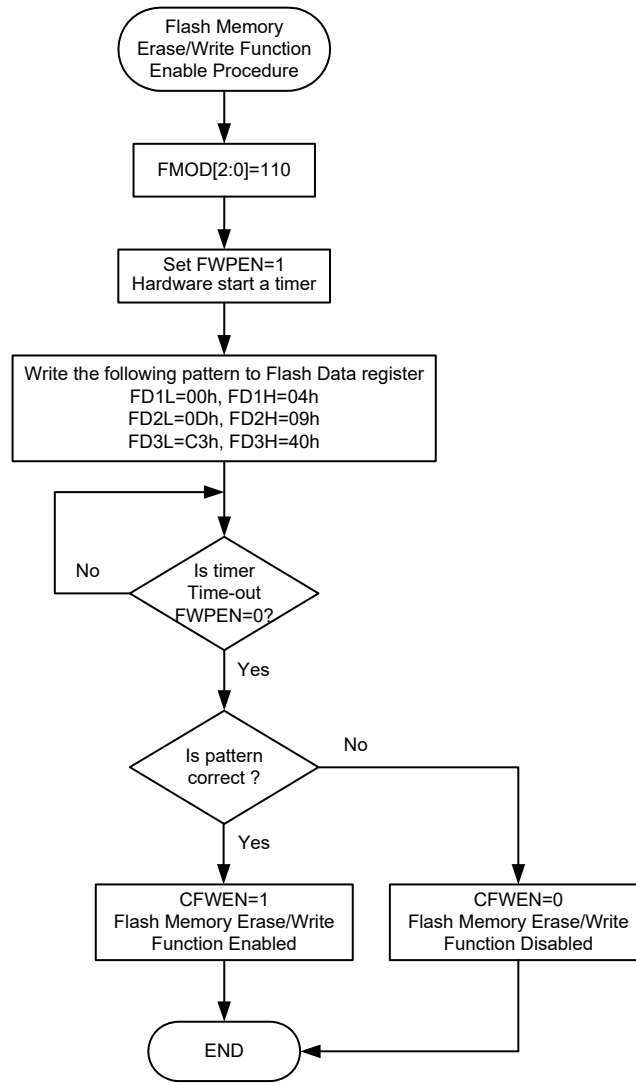
**Flash Memory Erase/Write Function Enable Procedure**

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 FC0 register to select the Flash Memory Erase/Write Function Enable Mode.
2. Set the FWPEN bit in the FC0 register to “1” to activate the Flash Memory Erase/Write Function. This will also activate an internal timer.
3. Write the correct data pattern into the Flash data registers, FD1L~FD3L and FD1H~FD3H, as soon as possible after the FWPEN bit is set high. The enable Flash memory erase/write function data pattern is 00H, 0DH, C3H, 04H, 09H and 40H corresponding to the FD1L~FD3L and FD1H~FD3H registers respectively.
4. Once the timer has timed out, the FWPEN bit will automatically be cleared to 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 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 write function, the CFWEN bit in the FC0 register can be cleared. There is no need to execute the above procedure.



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



### **Flash Memory Write Procedure**

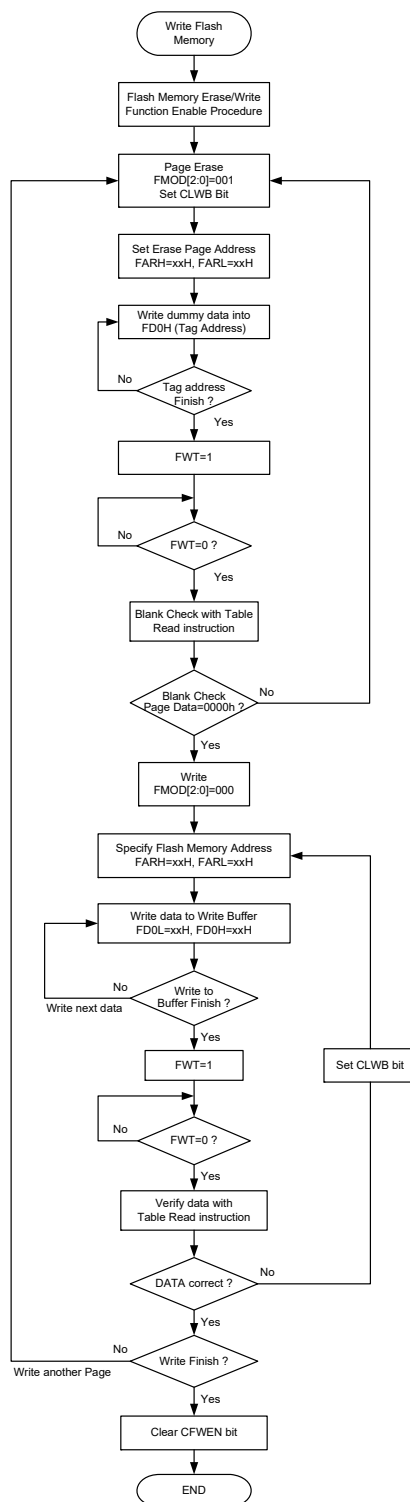
After the Flash memory erase/write function has been successfully enabled as the CFWEN bit is set high, the data to be written into the flash memory can be loaded into the write buffer. The selected flash memory page data should be erased by properly configuring the IAP control registers before the data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

The write buffer size is 32 words, known as a page, whose address is mapped to a specific flash memory page specified by the memory address bits, FA12~FA5. It is important to ensure that the page where the write buffer data is located is the same one which the memory address bits, FA12~FA5, specify.

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

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

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



**Flash Memory Consecutive Write Procedure**

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

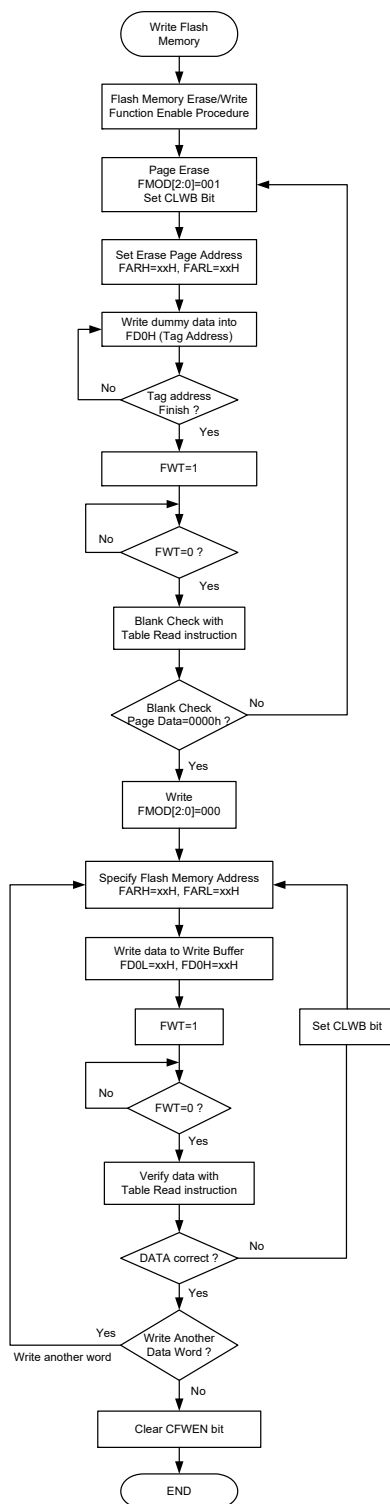
2. It will take certain time for the FWT bit state changing from high to low in the erase or write operation, which can be selected by the FWERTS bit in the FC2 register.

### **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 FMOD field 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 FMOD field to “000” to select the write operation.
5. Setup the desired address ADDR1 in the FARH and FARL registers. Write the desired data word DATA1 first into the FD0L register and then into the FD0H register.
6. Set the FWT bit high to transfer the data word from the write buffer to the flash memory. Wait until the FWT bit goes low.
7. Verify the data using the table read instruction to ensure that the write operation has successfully completed.  
If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 5.  
Go to step 8 if the write operation is successful.
8. Setup the desired address ADDR2 in the FARH and FARL registers. Write the desired data word DATA2 first into the FD0L register and then into the FD0H register.
9. Set the FWT bit high to transfer the data word from the write buffer to the flash memory. Wait until the FWT bit goes low.
10. Verify the data using the table read instruction to ensure that the write operation has successfully completed.  
If the write operation has not successfully completed, set the CLWB bit high to clear the write buffer and then go to step 8.  
Go to step 11 if the write operation is successful.
11. Clear the CFWEN bit low to disable the Flash memory erase/write function.



#### Flash Memory Non-Consecutive Write Procedure

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

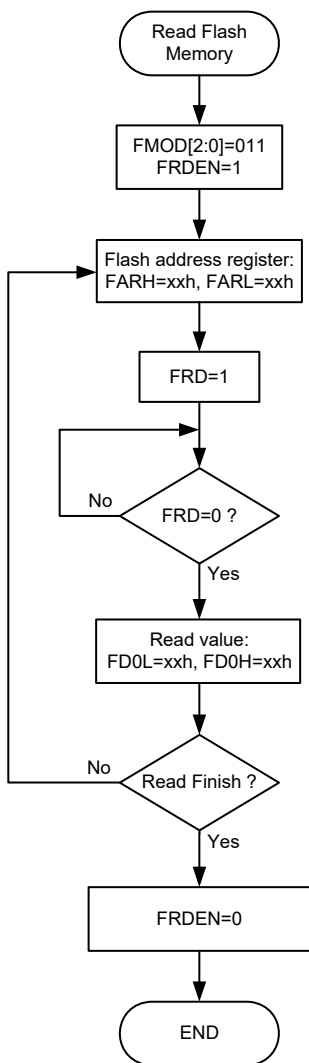
2. It will take certain time for the FWT bit state changing from high to low in the erase or write operation, which can be selected by the FWERTS bit in the FC2 register.

#### **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. 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.
5. The system frequency should be setup to the maximum application frequency when data write and data check operations are executed using the IAP function.

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

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



**Flash Memory Read Procedure**

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

## Data Memory

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

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

### Structure

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

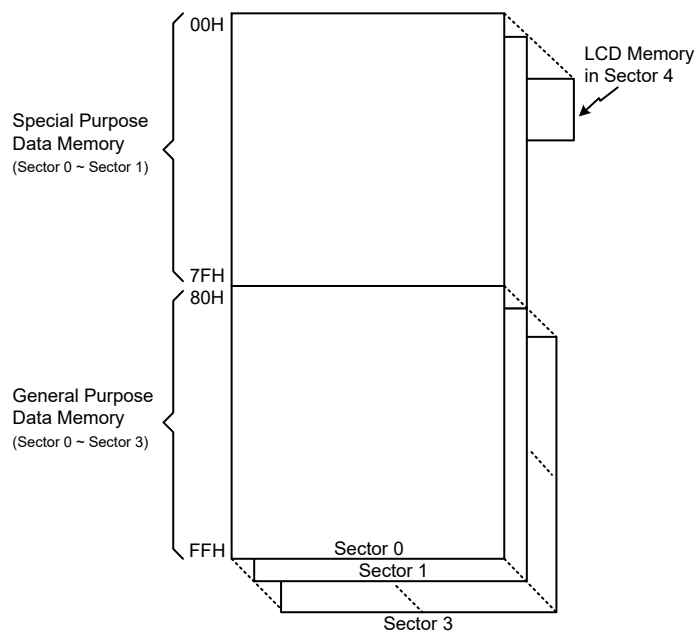
Each of the Data Memory sectors is categorized into two types, the Special Purpose Data Memory and the General Purpose Data Memory. However, the Data Memory Sector 4 is reserved for the LCD display data.

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.

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

Special Purpose Data Memory	General Purpose Data Memory		LCD Data Memory	
Located Sectors	Capacity	Sector: Address	Capacity	Sector: Address
0, 1	512×8	0: 80H~FFH 1: 80H~FFH 2: 80H~FFH 3: 80H~FFH	32×8	4: 00H~1FH

**Data Memory Summary**



**Data Memory Structure**

### Data Memory Addressing

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

Direct Addressing can be used in all sectors using the extended instructions which can address all available data memory space. For the accessed data memory which is located in any data memory sectors except Sector 0, the extended instructions can be used to access the data memory instead of using the indirect addressing access. The main difference between standard instructions and extended instructions is that the data memory address “m” in the extended instructions has 11 valid bits for this device, the high byte indicates a sector and the low byte indicates a specific address.

### General Purpose Data Memory


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

### Special Purpose Data Memory

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



	Sector 0	Sector 1		Sector 0	Sector 1
00H	IAR0	FC0	40H	TB1C	EEC
01H	MP0	FC1	41H	CTM0C0	
02H	IAR1	FC2	42H	CTM0C1	
03H	MP1L	FARL	43H	CTM0DL	
04H	MP1H	FARH	44H	CTM0DH	
05H	ACC	FD0L	45H	CTM0AL	
06H	PCL	FD0H	46H	CTM0AH	
07H	TBLP	FD1L	47H	PTMC0	
08H	TBLH	FD1H	48H	PTMC1	
09H	TBHP	FD2L	49H	PTMDL	
0AH	STATUS	FD2H	4AH	PTMDH	
0BH		FD3L	4BH	PTMAL	
0CH	IAR2	FD3H	4CH	PTMAH	
0DH	MP2L		4DH	PTMRPL	
0EH	MP2H	PAS0	4EH	PTMRPH	
0FH	RSTFC	PAS1	4FH		
10H	INTC0	PBS0	50H		
11H	INTC1	PBS1	51H		
12H	INTC2	PCS0	52H	SADOL	
13H	INTC3	PCS1	53H	SADOH	
14H	PA	PDS0	54H	SADC0	
15H	PAC	PDS1	55H	SADC1	
16H	PAPU	PES0	56H	SADC2	
17H	PAWU	PES1	57H	LVRC	
18H	PB	PFS0	58H	LVDC	
19H	PBC	PFS1	59H	VBGRC	
1AH	PBPU		5AH	LCDC0	
1BH			5BH	LCDC1	
1CH			5CH	LCDC2	
1DH			5DH	CTM1C0	
1EH			5EH	CTM1C1	
1FH	INTEG		5FH	CTM1DL	
20H	MF10		60H	CTM1DH	
21H	MF11		61H	CTM1AL	
22H	WDTC		62H	CTM1AH	
23H	EEAL		63H	CTM2C0	
24H	EEAH		64H	CTM2C1	
25H	EED		65H	CTM2DL	
26H	RSTC		66H	CTM2DH	
27H	LVPUC		67H	CTM2AL	
28H	ORMC		68H	CTM2AH	
29H	USR		69H		
2AH	UCR1		6AH		
2BH	UCR2		6BH		
2CH	UCR3		6CH		
2DH	TXR_RXR		6DH		
2EH	BRG		6EH		
2FH	SIMC0		6FH		
30H	SIMC1		70H	PC	
31H	SIMC2/SIMA		71H	PCC	
32H	SIMD		72H	PCPU	
33H	SIMTOC		73H	PD	
34H	IFS		74H	PDC	
35H	SLEDC0		75H	PDPU	
36H	SLEDC1		76H	PE	
37H	SLEDC2		77H	PEC	
38H			78H	PEPU	
39H	SCC		79H	PF	
3AH	HIRCC		7AH	PFC	
3BH	HXTC		7BH	PFPUP	
3CH	LXTC		7CH		
3DH	PSC0R		7DH		
3EH	TB0C		7EH		
3FH	PSC1R		7FH		

 : Unused, read as 00H

### Special Purpose Data Memory

## Special Function Register Description

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

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

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

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

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

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

#### Indirect Addressing Program Example 1

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

### Indirect Addressing Program Example 2

```
data .section 'data'
adres1 db ?
adres2 db ?
adres3 db ?
adres4 db ?
block db ?
code .section at 0 'code'
org 00h
start:
    mov a, 04h          ; setup size of block
    mov block, a
    mov a, 01h          ; setup the memory sector
    mov 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 Data Memory addresses.

### Direct Addressing Program Example using extended instructions

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

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

### Accumulator – ACC

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

### Program Counter Low Byte Register – PCL

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

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

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

### 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, both “TABRD [m]” and “TABRDL [m]” instructions 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 SC flag, CZ flag, zero flag (Z), carry flag (C), auxiliary carry flag (AC), overflow flag (OV), power down flag (PDF), and watchdog time-out flag (TO). These arithmetic/logical operation and system management flags are used to record the status and operation of the microcontroller.

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

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

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

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

### • STATUS Register

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

“x”: Unknown

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

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

Bit 5	<b>TO:</b> Watchdog time-out flag 0: After power up or executing the “CLR WDT” or “HALT” instruction 1: A watchdog time-out occurred
Bit 4	<b>PDF:</b> Power down flag 0: After power up or executing the «CLR WDT” instruction 1: By executing the “HALT” instruction
Bit 3	<b>OV:</b> 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	<b>Z:</b> 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	<b>AC:</b> 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	<b>C:</b> 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 512×8 bits for the device. Unlike the Program Memory and RAM Data Memory, the EEPROM Data Memory is not directly mapped into memory space and is therefore not directly addressable in the same way as the other types of memory. Read and Write operations to the EEPROM are carried out in single byte operations using an address register pair and a data register in sector 0 and a single control register in sector 1.

### 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	—	—	—	—	—	—	—	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 low byte address bit 7 ~ bit 0

• **EEAH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	—	EEAH0
R/W	—	—	—	—	—	—	—	R/W
POR	—	—	—	—	—	—	—	0

Bit 7~1 Unimplemented, read as “0”

Bit 0 **EEAH0**: Data EEPROM high byte address 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 selection

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 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

	<p>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 has not first been set high.</p>
Bit4	<p><b>MODE:</b> Data EEPROM operation mode selection</p> <p>0: Byte operation mode</p> <p>1: Page operation mode</p> <p>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.</p>
Bit 3	<p><b>WREN:</b> Data EEPROM write enable</p> <p>0: Disable</p> <p>1: Enable</p> <p>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.</p>
Bit 2	<p><b>WR:</b> Data EEPROM write control</p> <p>0: Write cycle has finished</p> <p>1: Activate a write cycle</p> <p>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.</p>
Bit 1	<p><b>RDEN:</b> Data EEPROM read enable</p> <p>0: Disable</p> <p>1: Enable</p> <p>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.</p>
Bit 0	<p><b>RD:</b> Data EEPROM read control</p> <p>0: Read cycle has finished</p> <p>1: Activate a read cycle</p> <p>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.</p>

- Note: 1. The EREN, ER, WREN, WR, RDEN and RD cannot be set to “1” at the same time in one instruction. The WR and RD cannot be set to “1” at the same time.
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 the 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 the RD bit high only will not initiate a read operation if the RDEN bit is not set high. When the read cycle terminates, the EEPROM data can be read from the EED register and the RD bit will automatically be cleared to zero. 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 the RD bit high only will not initiate a read operation if the RDEN bit is not set high. When the current byte read cycle terminates, the EEPROM data can be read from the EED register and then the current address will be incremented by one by hardware. After this the RD bit will automatically be cleared to zero. The data which is stored in the next EEPROM address can continuously be read when the RD bit is again set high 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 5 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 5-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 5 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 5-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, informing the user 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 set low 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, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended. After the write operation is finished, the WREN bit will be set low 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 5 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 5-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, informing the user that the data has been written to the EEPROM. The application program can therefore poll the WR bit to determine when the write cycle has ended. After the write operation is finished, the WREN bit will be set low 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 write 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. When an EEPROM erase or write cycle ends, the DEF request flag will be set. If the global and EEPROM interrupts are enabled and the stack is not full, a jump to the associated EEPROM interrupt vector will take place. When the interrupt is serviced, the EEPROM interrupt flag will be automatically reset. More details can be obtained in the Interrupt section.

### **Programming Considerations**

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

When writing data the WR bit must be set high immediately after the WREN bit has been set high, to ensure the write cycle executes correctly. 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, erase or write operation is totally complete. Otherwise, the EEPROM read, erase or write operation will fail.

## Programming Examples

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

```
MOV A, 040H          ; setup memory pointer low 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, 040H          ; setup memory pointer low byte MP1L
MOV MP1L, A          ; MP1L points to EEC register
MOV A, 01H           ; setup 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, 040H          ; setup memory pointer low byte MP1L
MOV MP1L, A          ; MP1L points to EEC register
MOV A, 01H           ; setup 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 erase operations
SET IAR1.5           ; start Erase Cycle - set ER bit - executed immediately
                     ; after setting EREN bit

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

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

```
MOV A, 040H          ; setup memory pointer low 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
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, 040H          ; setup memory pointer low byte MP1L
MOV MP1L, A          ; MP1L points to EEC register
MOV A, 01H           ; setup 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 defined 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 MP1H

```

## Oscillators

Various oscillator options offer the user a wide range of functions according to their various application requirements. The flexible features of the oscillator functions ensure that the best optimization can be achieved in terms of speed and power saving. Oscillator operation is 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 oscillators provide higher performance but carry with it the disadvantage of higher power requirements, while the opposite is of course true for the lower frequency oscillator. With the capability of dynamically switching between fast and slow system clock, the device have the flexibility to optimize the performance/power ratio, a feature especially important in power sensitive portable applications.

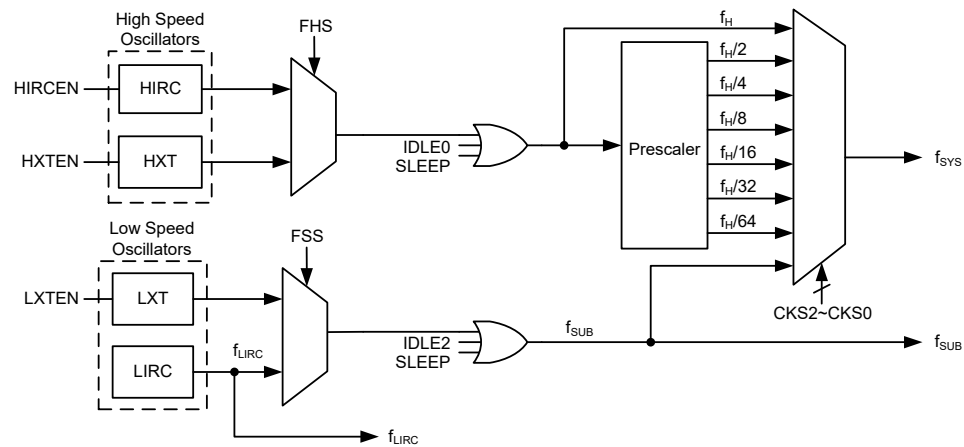
Type	Name	Frequency	Pins
External High Speed Crystal	HXT	400kHz~12MHz	OSC1/OSC2
Internal High Speed RC	HIRC	4/8/12MHz	—
External Low Speed Crystal	LXT	32.768kHz	XT1/XT2
Internal Low Speed RC	LIRC	32kHz	—

**Oscillator Types**

## System Clock Configurations

There are four methods of generating the system clock, two high speed oscillators and two low speed oscillators. The high speed oscillator is the external crystal/ceramic oscillator, HXT, and the internal 4/8/12MHz RC oscillator, HIRC. The low speed oscillators are 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. It is not possible to choose a no-oscillator selection for either the high or low speed oscillator.

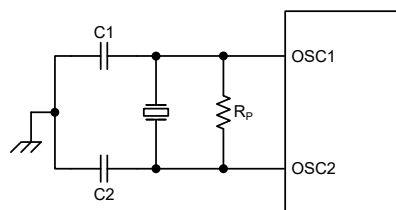


**System Clock Configurations**

## 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 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
12MHz	0pF	0pF
8MHz	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 internal RC oscillator is one of the high frequency oscillator choices, which is selected via a software control bit, FHS. It is a fully integrated system oscillator requiring no external components. The internal RC oscillator has three fixed frequencies of 4/8/12MHz, which are selected by the HIRC1~HIRC0 bits in the HIRCC register. These bits must also be setup to match the selected configuration option frequency to ensure that the HIRC frequency accuracy specified in the A.C. Characteristics is achieved. Device trimming during the manufacturing process and the inclusion of internal frequency compensation circuits are used to ensure that the influence of the power supply voltage, temperature and process variations on the oscillation frequency are minimised. Note that if this internal system clock is selected, as it requires no external pins for its operation, I/O pins are free for use as normal I/O pins or other pin-shared functional pins.

### 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.

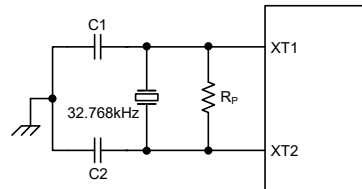
However, for some crystals, to ensure oscillation and accurate frequency generation, it is necessary to add two small value external capacitors, C1 and C2. The exact values of C1 and C2 should be selected in consultation with the crystal or resonator manufacturer's specification. The external parallel feedback resistor,  $R_p$ , 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.  $R_p$ , 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.  $R_p=5M\Omega\sim10M\Omega$  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 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 CKS bit field and FSS bit in the SCC register, the LXT oscillator operating mode can not be changed.

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

#### 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.

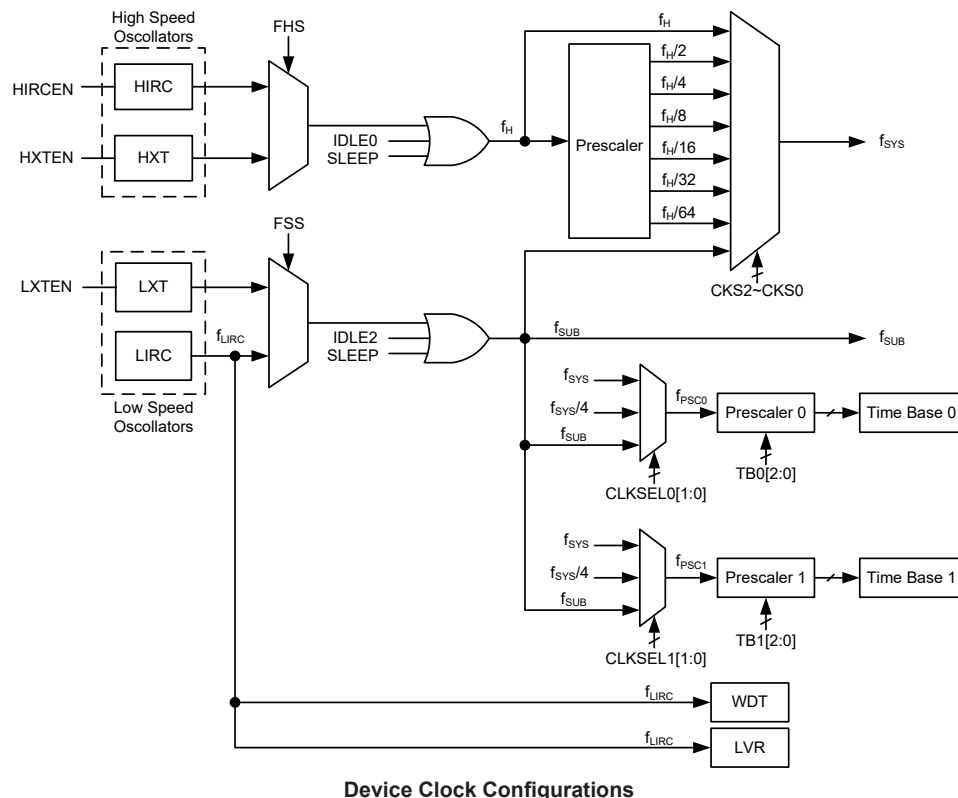
## Operating Modes and System Clocks

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 both high and low speed clock sources are provided 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 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 is sourced from an HXT or HIRC oscillator, selected via configuring the FHS bit in the SCC 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 from 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  $f_H/2 \sim f_H/64$ .



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

## System Operation Modes

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

Operation Mode	CPU	Register Setting			f <sub>sys</sub>	f <sub>H</sub>	f <sub>SUB</sub>	f <sub>LIRC</sub>
		FHIDEN	FSIDEN	CKS2~CKS0				
FAST	On	x	x	000~110	f <sub>H</sub> ~f <sub>H</sub> /64	On	On	On
SLOW	On	x	x	111	f <sub>SUB</sub>	On/Off <sup>(1)</sup>	On	On
IDLE0	Off	0	1	000~110	Off	Off	On	On
				111	On			
IDLE1	Off	1	1	xxx	On	On	On	On
IDLE2	Off	1	0	000~110	On	On	Off	On
				111	Off			
SLEEP	Off	0	0	xxx	Off	Off	Off	On/Off <sup>(2)</sup>

"x": Don't care

Note: 1. The f<sub>H</sub> clock will be switched on or off by configuring the corresponding oscillator enable bit in the SLOW mode.

2. The f<sub>LIRC</sub> clock can be switched on or off which is controlled by the WDT function being enabled or disabled in the SLEEP mode.

### FAST Mode

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

### SLOW Mode

This is also a mode where the microcontroller operates normally although now with a slower speed clock source. The clock source used will be from f<sub>SUB</sub>. The f<sub>SUB</sub> clock is derived from the LXT or LIRC oscillator determined by 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 bit are low. In the SLEEP mode the CPU will be stopped and both the high and low speed oscillators will be switched off. However the f<sub>LIRC</sub> clock can still continue to operate if the WDT function is enabled by the WDTC register.

### IDLE0 Mode

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

### IDLE1 Mode

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

### IDLE2 Mode

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

## Control Registers

The registers, SCC, HIRCC, HXTC and LXTC, 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	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
HXTC	—	—	—	—	—	HXTM	HXTF	HXTEN
LXTC	—	—	—	—	—	LXTSP	LXTF	LXTEN

**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	0	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 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	—	—	—	—	HIRC1	HIRC0	HIRCF	HIRCEN
R/W	—	—	—	—	R/W	R/W	R	R/W
POR	—	—	—	—	0	0	0	1

- Bit 7~4      Unimplemented, read as “0”
- Bit 3~2      **HIRC1~HIRC0**: HIRC frequency selection  
00: 4MHz  
01: 8MHz  
10: 12MHz  
11: 4MHz  
When the HIRC oscillator is enabled or the HIRC frequency selection is changed by the application program, the clock frequency will automatically be changed after the HIRCF flag is set to 1.  
It is recommended that the HIRC frequency selected by these two bits should be the same with the frequency determined by the configuration option to achieve the HIRC frequency accuracy specified in the A.C. Characteristics.
- Bit 1      **HIRCF**: HIRC oscillator stable flag  
0: HIRC unstable  
1: HIRC stable  
This bit is used to indicate whether the HIRC oscillator is stable or not. When the HIRCEN bit is set to 1 to enable the HIRC oscillator, the HIRCF bit will first be cleared to 0 and then set to 1 after the HIRC oscillator is stable.
- Bit 0      **HIRCEN**: HIRC oscillator enable control  
0: Disable  
1: Enable

• **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 ≤ 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

1: Enable

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

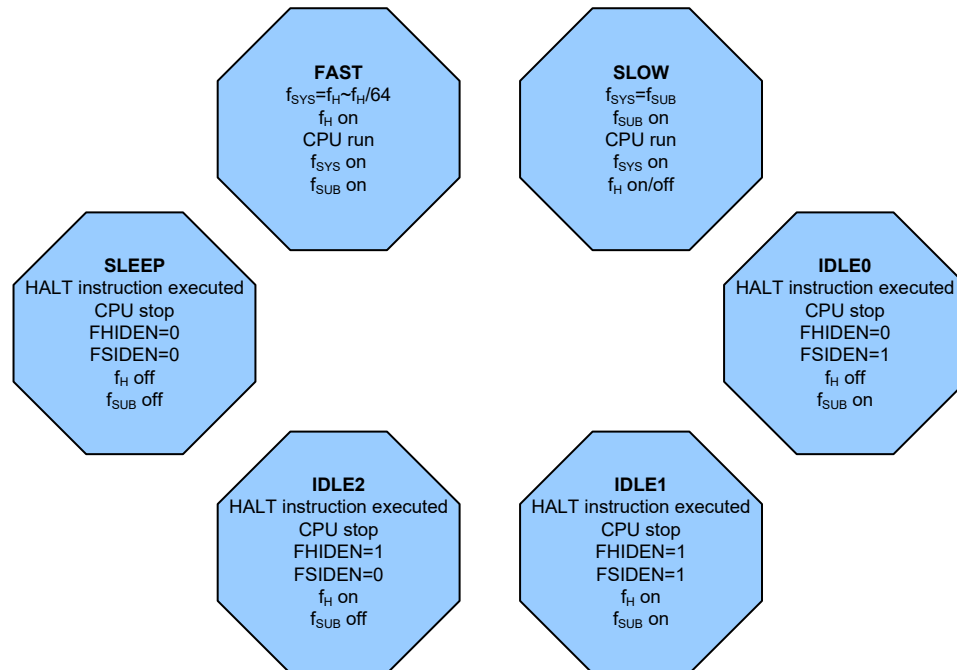
0: Disable

1: Enable

## 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.

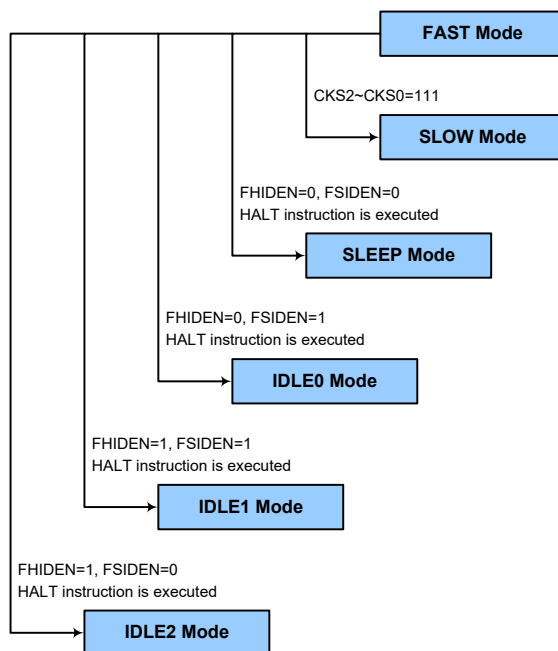
In simple terms, Mode Switching between the FAST Mode and SLOW Mode is executed using the CKS2~CKS0 bits in the SCC register while Mode Switching from the FAST/SLOW Modes to the SLEEP/IDLE Modes is executed via the HALT instruction. When a HALT instruction is executed, whether the device enters the IDLE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

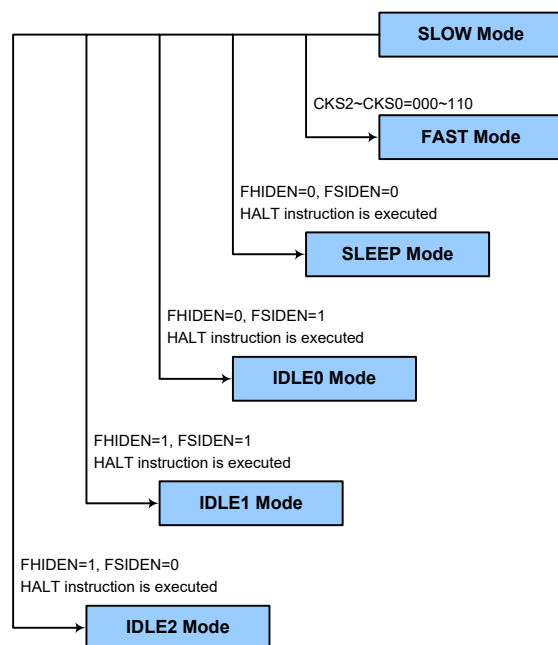
The SLOW Mode is sourced from the LXT or LIRC oscillator determined by the FSS bit in the SCC register and therefore requires these oscillators to be stable before full mode switching occurs.



### 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. The time duration required for the high speed system oscillator stabilization is specified in the System Start Up Time Characteristics.





### **Entering the SLEEP Mode**

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

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

### **Entering the IDLE0 Mode**

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

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

### **Entering the IDLE1 Mode**

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

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

### **Entering the IDLE2 Mode**

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

- The  $f_H$  clock will be on but the  $f_{SUB}$  clock will be off and the application program will stop at the “HALT” instruction.

- The Data Memory contents and registers will maintain their present condition.
- The I/O ports will maintain their present conditions.
- In the status register, the Power Down flag PDF will be set, and WDT timeout flag TO will be cleared.
- The WDT will be cleared and resume counting if the WDT function is enabled. If the WDT function is disabled, the WDT will be cleared and then stopped.

### **Standby Current Considerations**

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

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

### **Wake-up**

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

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

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

If the system is woken up by an external  $\overline{\text{RES}}$  pin 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,  $f_{LIRC}$ . The LIRC internal oscillator has an approximate frequency of 32kHz and this specified internal clock period can vary with  $V_{DD}$ , temperature and process variations. The Watchdog Timer source clock is then subdivided by a ratio of  $2^8$  to  $2^{18}$  to give longer timeouts, the actual value being chosen using the WS2~WS0 bits in the WDTC register.

### Watchdog Timer Control Register

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

#### • WDTC Register

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

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

10101: Disable  
 01010: Enable  
 Others: Reset MCU

When these bits are changed by the environmental noise or software setting to reset the microcontroller, the reset operation will be activated after a delay time,  $t_{SRESET}$ , and the WRF bit in the RSTFC register will be set high.

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

000:  $2^8/f_{LIRC}$   
 001:  $2^{10}/f_{LIRC}$   
 010:  $2^{12}/f_{LIRC}$   
 011:  $2^{14}/f_{LIRC}$   
 100:  $2^{15}/f_{LIRC}$   
 101:  $2^{16}/f_{LIRC}$   
 110:  $2^{17}/f_{LIRC}$   
 111:  $2^{18}/f_{LIRC}$

These three bits determine the division ratio of the watchdog timer source clock, which in turn determines the 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**: LVR control register software reset flag  
Refer to the “Low Voltage Reset” section

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

This bit is set high by the WDT control register software reset and cleared by the application program. Note that this bit can only be cleared to zero by the application program.

## Watchdog Timer Operation

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

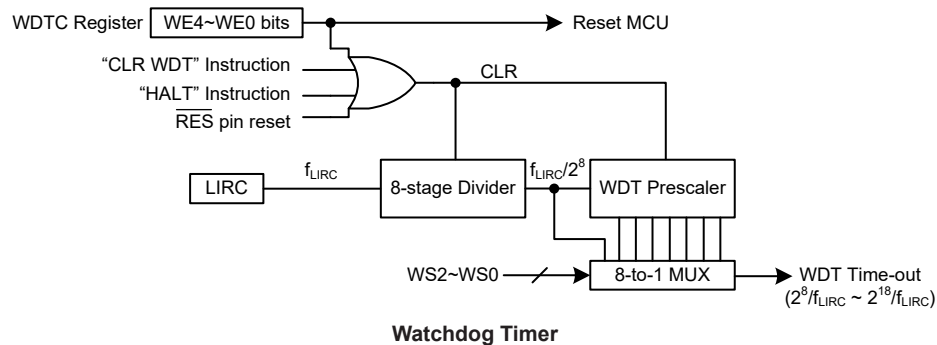
WE4~WE0 Bits	WDT Function
10101B	Disable
01010B	Enable
Any other values	Reset MCU

### 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 and PDF bits in the status register will be set and only the Program Counter and Stack Pointer will be reset. Four methods can be adopted to clear the contents of the Watchdog Timer. The first is a WDT software reset, which means a certain value except 01010B and 10101B written into the WE4~WE0 bit field, the second is using the Watchdog Timer software clear instruction and the third is via a HALT instruction. The last is an external hardware reset, which means a low level on the external reset pin if the external reset pin is selected by the RSTC register.

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

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



## Reset and Initialisation

A reset function is a fundamental part of any microcontroller ensuring that the 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, situations may arise where it is necessary to forcefully apply a reset condition when the device is running. One example of this is where after power has been applied and the device is already running, the  $\overline{\text{RES}}$  line is forcefully pulled low. In such a case, known as a normal operation reset, some of the registers remain unchanged allowing the device to proceed with normal operation after the reset line is allowed to return high.

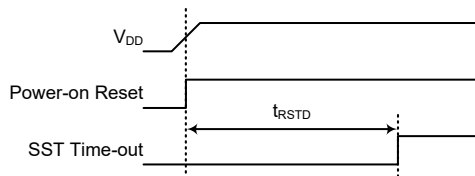
Another type of reset is when the Watchdog Timer overflows and resets. All types of reset operations result in different register conditions being setup. Another reset exists in the form of a Low Voltage Reset, LVR, where a full reset, is implemented in situations where the power supply voltage falls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

## Reset Functions

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

### 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 I/O ports will be first set to inputs.

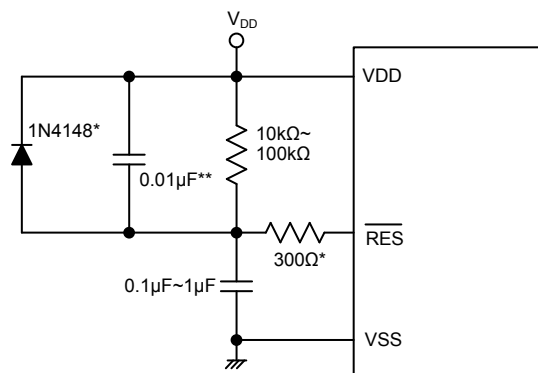


Note:  $t_{RSTD}$  is power-on delay specified in System Start Up Time Characteristics.

**Power-On Reset Timing Chart**

### RES Pin Reset

As the reset pin is shared with I/O pins, the reset function must be selected using a control register, RSTC. Although the microcontroller has an internal RC reset function, if the  $V_{DD}$  power supply rise time is not fast enough or does not stabilise quickly at power-on, the internal reset function may be incapable of providing proper reset operation. For this reason it is recommended that an external RC network is connected to the  $\overline{RES}$  pin, whose additional time delay will ensure that the  $\overline{RES}$  pin remains low for an extended period to allow the power supply to stabilise. During this time delay, normal operation of the microcontroller will be inhibited. After the  $\overline{RES}$  line reaches a certain voltage value, the reset delay time,  $t_{RSTD}$ , is invoked to provide an extra delay time after which the microcontroller will begin normal operation. The abbreviation SST in the figures stands for System Start-up Time. For most applications a resistor connected between  $V_{DD}$  and the  $\overline{RES}$  line and a capacitor connected between  $VSS$  and the  $\overline{RES}$  pin will provide a suitable external reset circuit. Any wiring connected to the  $\overline{RES}$  pin should be kept as short as possible to minimise any stray noise interference. For applications that operate within an environment where more noise is present the Enhanced Reset Circuit shown is recommended.

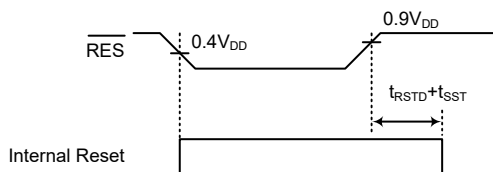


**External RES Circuit**

Note: “\*” It is recommended that this component is added for added ESD protection.

“\*\*\*” It is recommended that this component is added in environments where power line noise is significant.

Pulling the  $\overline{RES}$  pin low using external hardware will also execute a device reset. In this case, as in the case of other resets, the Program Counter will reset to zero and program execution initiated from this point.



**RES Reset Timing Chart**

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	I/O
10101010B	RES
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: I/O pin

10101010: RES pin

Other values: Reset MCU

If these bits are changed due to adverse environmental conditions, the microcontroller will be reset. The reset operation will be activated after a delay time,  $t_{SRESET}$ , and the RSTF bit in the RSTFC register will be set to 1. All resets will reset this register to POR value except the WDT time out hardware warm reset. Note that if the register is set to 10101010 to select the RES pin, this configuration has higher priority than other related pin-shared controls.

• **RSTFC Register**

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

“x”: Unknown

Bit 7~4 Unimplemented, read as “0”

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

0: Not occurred

1: Occurred

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

Bit 2 **LVRF**: LVR function reset flag

Refer to the “Low Voltage Reset” section

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

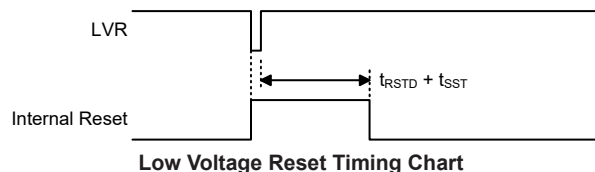
Refer to the “Low Voltage Reset” section

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

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 should the value fall below a certain predefined level. The LVR function can be enabled or disabled by the LVRC control register. If the LVRC control register is configured to enable the LVR function, the LVR function will be always enabled except in the SLEEP or IDLE mode. If the supply voltage of the device drops to within a range of  $0.9V \sim V_{LVR}$  such as might occur when changing the battery, the LVR will automatically reset the device internally and the LVRF bit in the RSTFC register will also be set high. For a valid LVR signal, a low supply voltage, i.e., a voltage in the range between  $0.9V \sim V_{LVR}$  must exist for a time greater than that specified by  $t_{LVR}$  in the LVD/LVR Electrical Characteristics. If the low supply voltage state does not exceed this value, the LVR will ignore the low supply voltage and will not perform a reset function. The actual  $V_{LVR}$  value can be selected by the LVS7~LVS0 bits in the LVRC register. If the LVS7~LVS0 bits are changed to some certain values by the environmental noise or software setting, the LVR will reset the device after a delay time,  $t_{SRESET}$ . When this happens, the LRF bit in the RSTFC register will be set high. After power on the register will have the value of 01100110B. Note that the LVR function will be automatically disabled when the device enters the IDLE/SLEEP mode.



#### • LVRC Register

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

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

01100110: 1.7V

01010101: 1.9V

00110011: 2.55V

10011001: 3.15V

10101010: 3.8V

11110000: LVR disable

Other values: MCU reset (register is reset to POR value)

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

Any register value, other than 11110000B and the five defined LVR values above, will also result in the generation of an MCU reset. The reset operation will be activated after a delay time,  $t_{SRESET}$ . However in this situation the register contents will be reset to the POR value.



• **RSTFC Register**

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

“x”: Unknown

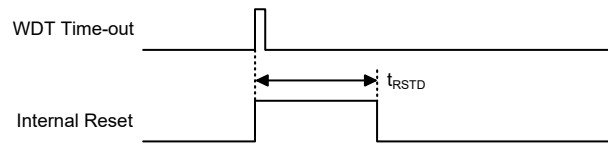
- Bit 7~4      Unimplemented, read as “0”
- Bit 3      **RSTF**: Reset control register software reset flag  
Refer to the “Internal Reset Control” section
- Bit 2      **LVRF**: LVR function reset flag  
0: Not occur  
1: Occurred  
This bit is set high when a specific Low Voltage Reset situation occurs. This bit can only be cleared to zero by the application program.
- Bit 1      **LRF**: LVR control register software reset flag  
0: Not occur  
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 the application program.
- Bit 0      **WRF**: WDT control register software reset flag  
Refer to the “Watchdog Timer Control Register” section

**In Application Programming Reset**

When a specific value of “55H” is written into the FC1 register, a reset signal will be generated to reset the whole device. Refer to the IAP section for more associated details.

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

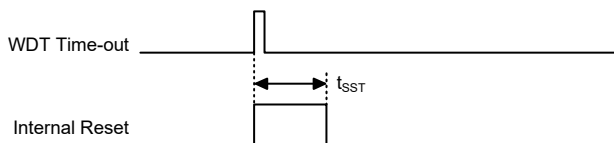
When a Watchdog time-out Reset occurs during normal operation, the Watchdog time-out flag TO will be set to “1”.



**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 zero and the TO and PDF flags will be set high. 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	RES or LVR reset during FAST or SLOW Mode operation
1	u	WDT time-out reset during FAST or SLOW Mode operation
1	1	WDT time-out reset during IDLE or SLEEP Mode operation

“u”: Unchanged

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

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

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

Register	Reset (Power On)	WDT Time-out (Normal Operation)	RES Reset (Normal Operation)	RES Reset (IDLE/SLEEP)	WDT Time-out (IDLE/SLEEP)
IAR0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
IAR1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP1H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
ACC	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
PCL	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
TBLP	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBLH	xxxx xxxx	uuuu uuuu	uuuu uuuu	uuuu uuuu	uuuu uuuu
TBHP	---x xxxx	---u uuuu	---u uuuu	---u uuuu	---u uuuu
STATUS	xx00 xxxx	uu1u uuuu	uuuu uuuu	uu01 uuuu	uu11 uuuu
IAR2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MP2H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTFC	---- 0x00	---- uuuu	---- uuuu	---- uuuu	---- uuuu
INTC0	-000 0000	-000 0000	-000 0000	-000 0000	-uuu uuuu
INTC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
INTC3	--00 --00	--00 --00	--00 --00	--00 --00	--00 --00
PA	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PAPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	Reset (Power On)	WDT Time-out (Normal Operation)	RES Reset (Normal Operation)	RES Reset (IDLE/SLEEP)	WDT Time-out (IDLE/SLEEP)
PAWU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PB	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu
PBC	--11 1111	--11 1111	--11 1111	--11 1111	--uu uuuu
PBPU	--00 0000	--00 0000	--00 0000	--00 0000	--uu uuuu
INTEG	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MFIO	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
MF11	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
WDTC	0101 0011	0101 0011	0101 0011	0101 0011	uuuu uuuu
EEAL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEAH	---- --0	---- --0	---- --0	---- --0	---- --u
EED	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
RSTC	0101 0101	0101 0101	0101 0101	0101 0101	uuuu uuuu
LVPUC	---- --0	---- --0	---- --0	---- --0	---- --u
ORMC	0000 0000	0000 0000	0000 0000	0000 0000	0000 0000
USR	0000 1011	0000 1011	0000 1011	0000 1011	uuuu uuuu
UCR1	0000 00x0	0000 00x0	0000 00x0	0000 00x0	uuuu uuuu
UCR2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
UCR3	---- --0	---- --0	---- --0	---- --0	---- --u
TXR_RXR	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
BRG	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMC0	111- 0000	111- 0000	111- 0000	111- 0000	uuu- uuuu
SIMC1	1000 0001	1000 0001	1000 0001	1000 0001	uuuu uuuu
SIMC2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMA	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SIMD	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu
SIMTOC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
IFS	---0 0000	---0 0000	---0 0000	---0 0000	---u uuuu
SLEDC0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SLEDC2	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SCC	000- 0000	000- 0000	000- 0000	000- 0000	uuu- uuuu
HIRCC	---- 0001	---- 0001	---- 0001	---- 0001	---- uuuu
HXTC	---- -000	---- -000	---- -000	---- -000	---- -uuu
LXTC	---- -000	---- -000	---- -000	---- -000	---- -uuu
PSC0R	---- --00	---- --00	---- --00	---- --00	---- --uu
TB0C	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu
PSC1R	---- --00	---- --00	---- --00	---- --00	---- --uu
TB1C	0--- -000	0--- -000	0--- -000	0--- -000	u--- -uuu
CTM0C0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0C1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0DH	---- --00	---- --00	---- --00	---- --00	---- --uu
CTM0AL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM0AH	---- --00	---- --00	---- --00	---- --00	---- --uu
PTMC0	0000 0---	0000 0---	0000 0---	0000 0---	uuuu u---
PTMC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMDL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	Reset (Power On)	WDT Time-out (Normal Operation)	RES Reset (Normal Operation)	RES Reset (IDLE/SLEEP)	WDT Time-out (IDLE/SLEEP)
PTMDH	---- --00	---- --00	---- --00	---- --00	---- --uu
PTMAL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMAH	---- --00	---- --00	---- --00	---- --00	---- --uu
PTMRPL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PTMRPH	---- --00	---- --00	---- --00	---- --00	---- --uu
SADOL	xxxx ----	xxxx ----	xxxx ----	xxxx ----	uuuu ---- (ADRF=0)
					uuuu uuuu (ADRF=1)
SADOH	xxxx xxxx	xxxx xxxx	xxxx xxxx	xxxx xxxx	uuuu uuuu (ADRF=0)
					---- uuuu (ADRF=1)
SADC0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
SADC1	0000 -000	0000 -000	0000 -000	0000 -000	uuuu -uuu
SADC2	0-00 0000	0-00 0000	0-00 0000	0-00 0000	u-uu uuuu
LVRC	0110 0110	0110 0110	0110 0110	0110 0110	uuuu uuuu
LVDC	--00 -000	--00 -000	--00 -000	--00 -000	--uu -uuu
VBGRC	---- ---0	---- ---0	---- ---0	---- ---0	---- ---u
LCDC0	0--- 0000	0--- 0000	0--- 0000	0--- 0000	u--- uuuu
LCDC1	000- 0000	000- 0000	000- 0000	000- 0000	uuu- uuuu
LCDC2	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
CTM1C0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1C1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1DH	---- --00	---- --00	---- --00	---- --00	---- --uu
CTM1AL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM1AH	---- --00	---- --00	---- --00	---- --00	---- --uu
CTM2C0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM2C1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM2DL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM2DH	---- --00	---- --00	---- --00	---- --00	---- --uu
CTM2AL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
CTM2AH	---- --00	---- --00	---- --00	---- --00	---- --uu
PC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PCPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PD	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PDP	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PE	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PEPU	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PF	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFC	1111 1111	1111 1111	1111 1111	1111 1111	uuuu uuuu
PFP	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FC1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

Register	Reset (Power On)	WDT Time-out (Normal Operation)	RES Reset (Normal Operation)	RES Reset (IDLE/SLEEP)	WDT Time-out (IDLE/SLEEP)
FC2	---- --00	---- --00	---- --00	---- --00	---- --uu
FARL	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FARH	---0 0000	---0 0000	---0 0000	---0 0000	---u uuuu
FD0L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD0H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD1H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD2H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3L	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
FD3H	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PAS1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PBS1	---- 0000	---- 0000	---- 0000	---- 0000	---- uuuu
PCS0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PCS1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PDS1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PES0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PES1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PFS0	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
PFS1	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu
EEC	0000 0000	0000 0000	0000 0000	0000 0000	uuuu uuuu

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

## Input/Output Ports

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PA	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAC	PAC7	PAC6	PAC5	PAC4	PAC3	PAC2	PAC1	PAC0
PAPU	PAPU7	PAPU6	PAPU5	PAPU4	PAPU3	PAPU2	PAPU1	PAPU0
PAWU	PAWU7	PAWU6	PAWU5	PAWU4	PAWU3	PAWU2	PAWU1	PAWU0
PB	—	—	PB5	PB4	PB3	PB2	PB1	PB0
PBC	—	—	PBC5	PBC4	PBC3	PBC2	PBC1	PBC0
PBPU	—	—	PBPU5	PBPU4	PBPU3	PBPU2	PBPU1	PBPU0
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
PFPU	PFPU7	PFPU6	PFPU5	PFPU4	PFPU3	PFPU2	PFPU1	PFPU0
LVPUC	—	—	—	—	—	—	—	LVPU

“—”: Unimplemented

### Input/Output Logic Function Register List

## Pull-high Resistors

Many product applications require pull-high resistors for their switch inputs usually requiring the use of an external resistor. To eliminate the need for these external resistors, all I/O pins, when configured as a digital input have the capability of being connected to an internal pull-high resistor. These pull-high resistors are selected using the LVPUC and PxPU registers, and are implemented using weak PMOS transistors. The PxPU register is used to determine whether the pull-high function is enabled or not while the LVPUC register is used to select the pull-high resistors value for low voltage power supply applications.

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

• PxPU Register

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

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

0: Disable

1: Enable

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

• LVPUC Register

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

Bit 7~1 Unimplemented, read as “0”

Bit 0 **LVPU:** Pull-high resistor selection for low voltage power supply

0: All pin pull-high resistors are 60kΩ @ 3V

1: All pin pull-high resistors are 15kΩ @ 3V

This bit is used to select the pull-high resistor value for low voltage power supply applications. The LVPU bit is only available when the corresponding pin pull-high function is enabled by setting the relevant pull-high control bit high. This bit will have no effect when the pull-high function is disabled.

## 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-shared functional pin is selected as general purpose input and the MCU enters the IDLE/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:** I/O port A pin wake-up control

0: Disable

1: Enable

## I/O Port Control Registers

Each I/O port has its own control register known as PAC~PFC, to control the input/output configuration. With this control register, each I/O pin with or without pull-high resistors can be reconfigured dynamically under software control. Each pin of the I/O ports is directly mapped to a bit in its associated port control register. For the I/O pin to function as an input, the corresponding

bit of the control register must be written as a “1”. This will then allow the logic state of the input pin to be directly read by instructions. When the corresponding bit of the control register is written as a “0”, the I/O pin will be setup as a CMOS output. If the pin is currently setup as an output, instructions can still be used to read the output register.

However, it should be noted that the program will in fact only read the status of the output data latch and not the actual logic status of the output pin.

• **PxC Register**

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

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

0: Output

1: Input

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

**I/O Port Source Current Control**

The device supports different source current driving capability for each I/O port. With the corresponding selection registers, SLEDC0, SLEDC1 and SLEDC2, each I/O port can support four levels of the source current driving capability. These source current selection bits are available when the corresponding pin is configured as a CMOS output. Otherwise, these select bits have no effect. Users should refer to the Input/Output Characteristics section to select the desired source current for different applications.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SLEDC0	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
SLEDC1	SLEDC17	SLEDC16	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
SLEDC2	SLEDC27	SLEDC26	SLEDC25	SLEDC24	SLEDC23	SLEDC22	SLEDC21	SLEDC20

**I/O Port Source Current Control Register List**

• **SLEDC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC07	SLEDC06	SLEDC05	SLEDC04	SLEDC03	SLEDC02	SLEDC01	SLEDC00
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **SLEDC07~SLEDC06:** PB5~PB4 source current selection

00: Source current=Level 0 (min.)

01: Source current=Level 1

10: Source current=Level 2

11: Source current=Level 3 (max.)

Bit 5~4 **SLEDC05~SLEDC04:** PB3~PB0 source current selection

00: Source current=Level 0 (min.)

01: Source current=Level 1

10: Source current=Level 2

11: Source current=Level 3 (max.)



- Bit 3~2     **SLEDC03~SLEDC02**: PA7~PA4 source current selection  
                  00: Source current=Level 0 (min.)  
                  01: Source current=Level 1  
                  10: Source current=Level 2  
                  11: Source current=Level 3 (max.)
- Bit 1~0     **SLEDC01~SLEDC00**: PA3~PA0 source current selection  
                  00: Source current=Level 0 (min.)  
                  01: Source current=Level 1  
                  10: Source current=Level 2  
                  11: Source current=Level 3 (max.)

• **SLEDC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC17	SLEDC16	SLEDC15	SLEDC14	SLEDC13	SLEDC12	SLEDC11	SLEDC10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7~6     **SLEDC17~SLEDC16**: PD7~PD4 source current selection  
                  00: Source current=Level 0 (min.)  
                  01: Source current=Level 1  
                  10: Source current=Level 2  
                  11: Source current=Level 3 (max.)
- Bit 5~4     **SLEDC15~SLEDC14**: PD3~PD0 source current selection  
                  00: Source current=Level 0 (min.)  
                  01: Source current=Level 1  
                  10: Source current=Level 2  
                  11: Source current=Level 3 (max.)
- Bit 3~2     **SLEDC13~SLEDC12**: PC7~PC4 source current selection  
                  00: Source current=Level 0 (min.)  
                  01: Source current=Level 1  
                  10: Source current=Level 2  
                  11: Source current=Level 3 (max.)
- Bit 1~0     **SLEDC11~SLEDC10**: PC3~PC0 source current selection  
                  00: Source current=Level 0 (min.)  
                  01: Source current=Level 1  
                  10: Source current=Level 2  
                  11: Source current=Level 3 (max.)

• **SLEDC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	SLEDC27	SLEDC26	SLEDC25	SLEDC24	SLEDC23	SLEDC22	SLEDC21	SLEDC20
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     **SLEDC27~SLEDC26**: PF7~PF4 source current selection  
                  00: Source current=Level 0 (min.)  
                  01: Source current=Level 1  
                  10: Source current=Level 2  
                  11: Source current=Level 3 (max.)
- Bit 5~4     **SLEDC25~SLEDC24**: PF3~PF0 source current selection  
                  00: Source current=Level 0 (min.)  
                  01: Source current=Level 1  
                  10: Source current=Level 2  
                  11: Source current=Level 3 (max.)

- Bit 3~2     **SLEDC23~SLEDC22**: PE7~PE4 source current selection  
               00: Source current=Level 0 (min.)  
               01: Source current=Level 1  
               10: Source current=Level 2  
               11: Source current=Level 3 (max.)
- Bit 1~0     **SLEDC21~SLEDC20**: PE3~PE0 source current selection  
               00: Source current=Level 0 (min.)  
               01: Source current=Level 1  
               10: Source current=Level 2  
               11: Source current=Level 3 (max.)

## Pin-shared Functions

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

### Pin-shared Function Selection Registers

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

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

Register Name	Bit							
	7	6	5	4	3	2	1	0
PAS0	PAS07	PAS06	PAS05	PAS04	PAS03	PAS02	PAS01	PAS00
PAS1	PAS17	PAS16	PAS15	PAS14	PAS13	PAS12	PAS11	PAS10
PBS0	PBS07	PBS06	PBS05	PBS04	PBS03	PBS02	PBS01	PBS00
PBS1	—	—	—	—	PBS13	PBS12	PBS11	PBS10
PCS0	PCS07	PCS06	PCS05	PCS04	PCS03	PCS02	PCS01	PCS00
PCS1	PCS17	PCS16	PCS15	PCS14	PCS13	PCS12	PCS11	PCS10
PDS0	PDS07	PDS06	PDS05	PDS04	PDS03	PDS02	PDS01	PDS00
PDS1	PDS17	PDS16	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
IFS	—	—	—	—	SDISDAPS	SCKSCLPS	SCSBPS	RXPS

**Pin-shared Function Selection Register List**

• **PAS0 Register**

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

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

00: PA3

01: PA3

10: PA3

11: OSC1

Bit 5~4 **PAS05~PAS04:** PA2 pin-shared function selection

00: PA2/INT3

01: PA2/INT3

10: PTP

11: SEG21

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

00: PA1

01: PA1

10: PA1

11: OSC2

Bit 1~0 **PAS01~PAS00:** PA0 pin-shared function selection

00: PA0/INT2/PTCK

01: PA0/INT2/PTCK

10: PA0/INT2/PTCK

11: SEG20

• **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/RES

01: PA7/RES

10: PA7/RES

11: SEG23

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

00: PA6

01: PA6

10: PA6

11: SEG22

Bit 3~2 **PAS13~PAS12:** PA5 pin-shared function selection

00: PA5

01: PA5

10: PA5

11: XT2

Bit 1~0 **PAS11~PAS10:** PA4 pin-shared function selection

00: PA4

01: PA4

10: PA4

11: XT1

• **PBS0 Register**

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

Bit 7~6      **PBS07~PBS06:** PB3 pin-shared function selection

00: PB3  
01: PB3  
10: CTP0  
11: AN3

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

00: PB2  
01: PB2  
10: CTP1  
11: AN2

Bit 3~2      **PBS03~PBS02:** PB1 pin-shared function selection

00: PB1  
01: PB1  
10: CTP2  
11: AN1

Bit 1~0      **PBS01~PBS00:** PB0 pin-shared function selection

00: PB0/CTCK2  
01: PB0/CTCK2  
10: PB0/CTCK2  
11: AN0

• **PBS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	PBS13	PBS12	PBS11	PBS10
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

Bit 7~4      Unimplemented, read as “0”

Bit 3~2      **PBS13~PBS12:** PB5 pin-shared function selection

00: PB5  
01: PB5  
10: VREFI  
11: AN9

Bit 1~0      **PBS11~PBS10:** PB4 pin-shared function selection

00: PB4  
01: PB4  
10: VREF  
11: AN7

• **PCS0 Register**

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

Bit 7~6      **PCS07~PCS06:** PC3 pin-shared function selection

00: PC3  
01: PC3  
10: PC3  
11: SEG11

Bit 5~4      **PCS05~PCS04:** PC2 pin-shared function selection

00: PC2  
01: PC2  
10: PC2  
11: SEG10

Bit 3~2      **PCS03~PCS02:** PC1 pin-shared function selection

00: PC1  
01: PC1  
10: PC1  
11: SEG9

Bit 1~0      **PCS01~PCS00:** PC0 pin-shared function selection

00: PC0  
01: PC0  
10: PC0  
11: SEG8

• **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: PC7  
10: SCK/SCL  
11: SEG15

Bit 5~4      **PCS15~PCS14:** PC6 pin-shared function selection

00: PC6  
01: PC6  
10: SCS  
11: SEG14

Bit 3~2      **PCS13~PCS12:** PC5 pin-shared function selection

00: PC5  
01: PC5  
10: SDO  
11: SEG13

Bit 1~0      **PCS11~PCS10:** PC4 pin-shared function selection

00: PC4  
01: PC4  
10: SDI/SDA  
11: SEG12

• **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: SCK/SCL  
11: SEG3

Bit 5~4 **PDS05~PDS04**: PD2 pin-shared function selection

00: PD2  
01: PD2  
10:  $\overline{SCS}$   
11: SEG2

Bit 3~2 **PDS03~PDS02**: PD1 pin-shared function selection

00: PD1  
01: PD1  
10: SDO  
11: SEG1

Bit 1~0 **PDS01~PDS00**: PD0 pin-shared function selection

00: PD0  
01: PD0  
10: SDI/SDA  
11: SEG0

• **PDS1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PDS17	PDS16	PDS15	PDS14	PDS13	PDS12	PDS11	PDS10
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **PDS17~PDS16**: PD7 pin-shared function selection

00: PD7  
01: PD7  
10: PD7  
11: SEG7

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

00: PD6  
01: PD6  
10: PD6  
11: SEG6

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

00: PD5  
01: PD5  
10: TX  
11: SEG5

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

00: PD4  
01: PD4  
10: RX/TX  
11: SEG4

• **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: COM3

Bit 5~4 **PES05~PES04:** PE2 pin-shared function selection

00: PE2  
 01: PE2  
 10: PE2  
 11: COM2

Bit 3~2 **PES03~PES02:** PE1 pin-shared function selection

00: PE1  
 01: PE1  
 10: PE1  
 11: COM1

Bit 1~0 **PES01~PES00:** PE0 pin-shared function selection

00: PE0  
 01: PE0  
 10: PE0  
 11: COM0

• **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: AN8  
 10: COM7  
 11: SEG24

Bit 5~4 **PES15~PES14:** PE6 pin-shared function selection

00: PE6  
 01: AN6  
 10: COM6  
 11: SEG25

Bit 3~2 **PES13~PES12:** PE5 pin-shared function selection

00: PE5/CTCK1  
 01: AN5  
 10: COM5  
 11: SEG26

Bit 1~0 **PES11~PES10:** PE4 pin-shared function selection

00: PE4/CTCK0  
 01: AN4  
 10: COM4  
 11: SEG27

• **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: SEG31

Bit 5~4      **PFS05~PFS04:** PF2 pin-shared function selection

00: PF2  
01: PF2  
10: PF2  
11: SEG30

Bit 3~2      **PFS03~PFS02:** PF1 pin-shared function selection

00: PF1  
01: PF1  
10: PF1  
11: SEG29

Bit 1~0      **PFS01~PFS00:** PF0 pin-shared function selection

00: PF0  
01: PF0  
10: PF0  
11: SEG28

• **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/INT1  
01: PF7/INT1  
10: PTP  
11: SEG19

Bit 5~4      **PFS15~PFS14:** PF6 pin-shared function selection

00: PF6/INT0  
01: PF6/INT0  
10: PF6/INT0  
11: SEG18

Bit 3~2      **PFS13~PFS12:** PF5 pin-shared function selection

00: PF5  
01: PF5  
10: TX  
11: SEG17

Bit 1~0      **PFS11~PFS10:** PF4 pin-shared function selection

00: PF4  
01: PF4  
10: RX/TX  
11: SEG16



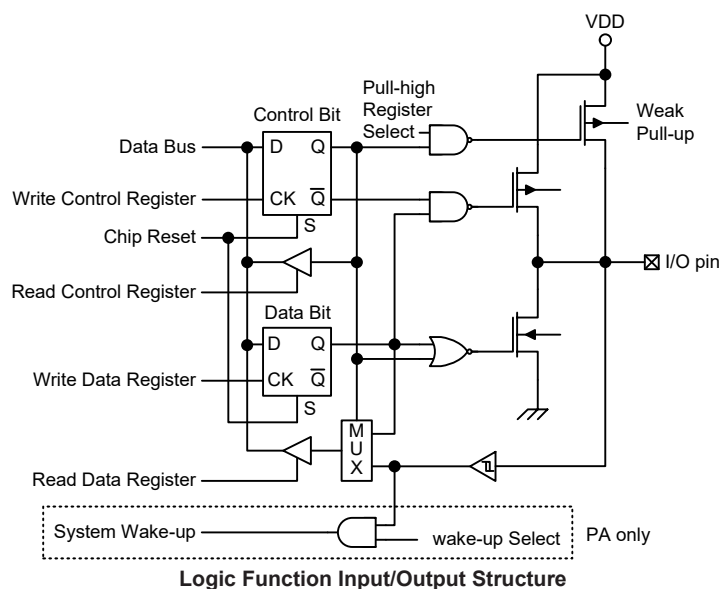
• IFS Register

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	SDISDAPS	SCKSCLPS	SCSBPS	RXPS
R/W	—	—	—	—	R/W	R/W	R/W	R/W
POR	—	—	—	—	0	0	0	0

- Bit 7~4      Unimplemented, read as “0”
- Bit 3      **SDISDAPS**: SDI/SDA input source pin selection  
0: PC4  
1: PD0
- Bit 2      **SCKSCLPS**: SCK/SCL input source pin selection  
0: PC7  
1: PD3
- Bit 1      **SCSBPS**:  $\overline{SCS}$  input source pin selection  
0: PC6  
1: PD2
- Bit 0      **RXPS**: RX/TX input source pin selection  
0: PF4  
1: PD4

**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 diagram, 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.



## 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 to high. This means that all I/O pins will be defaulted 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 devices 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, Compare Match Output and Single Pulse Output as well as being the functional unit for the generation of PWM signals. Each of the TMs has two interrupts. The addition of input and output pins for TM ensures that users are provided with timing units with a wide and flexible range of features.

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

### Introduction

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

TM Function	CTM	PTM
Timer/Counter	√	√
Compare Match Output	√	√
PWM Output	1	√
Single Pulse Output	—	√
PWM Alignment	Edge	Edge
PWM Adjustment Period & Duty	Duty or Period	Duty or Period

**TM Function Summary**

## TM Operation

The TMs 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 comparator. When the free running counter has the same value as the pre-programmed comparator, known as a compare match situation, a TM interrupt signal will be generated which can clear the counter and perhaps also change the condition of the TM output pin. The internal TM counter is driven by a user selectable clock source, which can be an internal clock or an external pin.

## TM Clock Source

The clock source which drives the main counter in the each TM can originate from various sources. The selection of the required clock source is implemented using the xTnCK2~xTnCK0 bits in the xTMn control registers, where “x” stands for C or P type TM and “n” stands for the specific TM serial number. The clock source can be a ratio of the system clock  $f_{SYS}$  or the internal high clock  $f_H$ , the  $f_{SUB}$  clock source or the external xTCKn pin. The xTCKn pin clock source is used to allow an external signal to drive the TM as an external clock source or for event counting.

## TM Interrupts

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

## TM External Pins

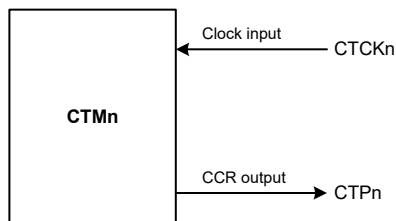
Each of the TMs, irrespective of what type, has one TM input pin, with the label xTCKn. The xTMn input pin, xTCKn, is essentially a clock source for the xTMn and is selected using the xTnCK2~xTnCK0 bits in the xTMnC0 register. This external TM input pin allows an external clock source to drive the internal TM. The xTCKn input pin can be chosen to have either a rising or falling active edge. The PTCK pin is also used as the external trigger input pin in single pulse output mode for the PTM.

The TMs each have one output pins, xTPn. When the TM is in the Compare Match Output Mode, these pins can be controlled by the TM to switch to a high or low level or to toggle when a compare match situation occurs. The output pins are also the pins where the TM generates the PWM output waveform.

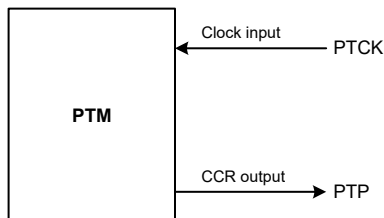
As the TM input/output pins are pin-shared with other functions, the TM input/output function must first be setup using relevant pin-shared function selection registers. The details of the pin-shared function selection are described in the pin-shared function section.

CTM		PTM	
Input	Output	Input	Output
CTCK0 CTCK1 CTCK2	CTP0 CTP1 CTP2	PTCK	PTP

**TM External Pins**



**CTMn Function Pin Block Diagram (n=0~2)**

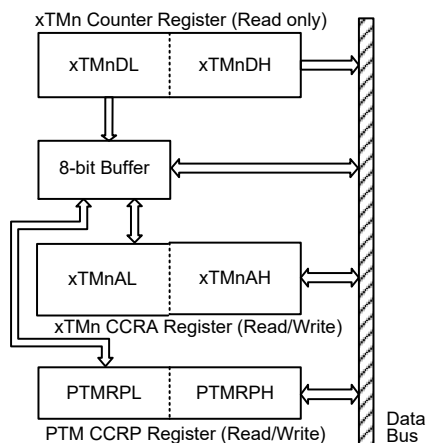


**PTM Function Pin Block Diagram**

### Programming Considerations

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

As the CCRA and CCRP registers are implemented in the way shown in the following diagram and accessing these register pairs is carried out in a specific way as described above, it is recommended to use the "MOV" instruction to access the CCRA and CCRP low byte registers, named xTMnAL and PTMRPL, using the following access procedures. Accessing the CCRA or CCRP low byte registers without following these access procedures will result in unpredictable values.

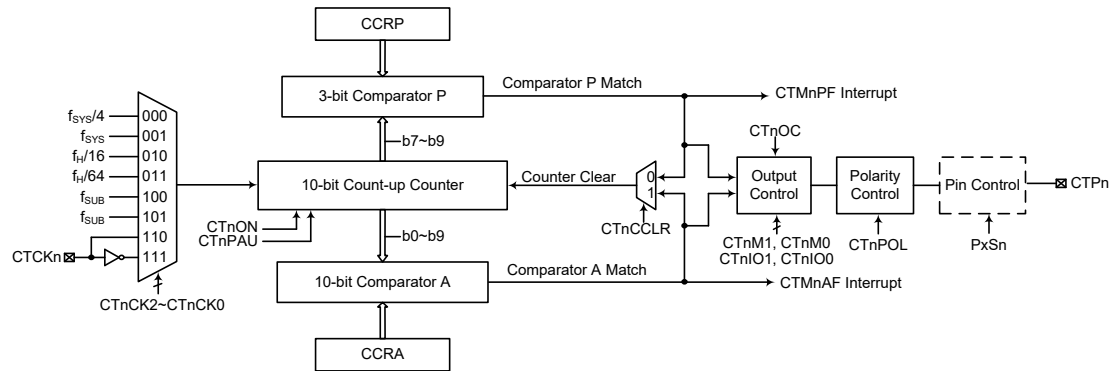


The following steps show the read and write procedures:

- Writing Data to CCRA or CCRP
  - ♦ Step 1. Write data to low byte xTMnAL or PTMRPL
    - Note that here data is only written to the 8-bit buffer.
  - ♦ Step 2. Write data to high byte xTMnAH or PTMRPH
    - Here data is written directly to the high byte registers and simultaneously data is latched from the 8-bit buffer to the low byte registers.
- Reading Data from the Counter Registers and CCRA or CCRP
  - ♦ Step 1. Read data from the high byte xTMnDH, xTMnAH or PTMRPH
    - Here data is read directly from the high byte registers and simultaneously data is latched from the Low Byte register into the 8-bit buffer.
  - ♦ Step 2. Read data from the low byte xTMnDL, xTMnAL or PTMRPL
    - This step reads data from the 8-bit buffer.

## Compact Type TM – CTM

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



Note: The CTMn external pins are pin-shared with other functions, so before using the CTMn function, ensure that the pin-shared function registers have been set properly to enable the CTMn pin function.

**10-bit Compact Type TM Block Diagram (n=0~2)**

## Compact Type TM Operation

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

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the CTnON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a CTMn interrupt signal will also usually be generated. The Compact Type TM can operate in a number of different operational modes, can be driven by different clock sources including an input pin. All operating setup conditions are selected using relevant internal registers.

## Compact Type TM Register Description

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

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

**10-bit Compact Type TM Register List (n=0~2)**

### • CTMnC0 Register

Bit	7	6	5	4	3	2	1	0
Name	CTnPAU	CTnCK2	CTnCK1	CTnCK0	CTnON	CTnRP2	CTnRP1	CTnRP0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **CTnPAU**: CTMn Counter Pause Control

0: Run  
1: Pause

The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the CTMn will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **CTCK2~CTCK0**: Select CTMn Counter clock

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

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

Bit 3 **CTnON**: CTMn Counter On/Off Control

0: Off  
1: On

This bit controls the overall on/off function of the CTMn. Setting the bit high enables the counter to run, clearing the bit to 0 disables the CTMn. Clearing this bit to zero will stop the counter from counting and turn off the CTMn which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again.

If the CTMn is in the Compare Match Output Mode or the PWM Output Mode then the CTMn output pin will be reset to its initial condition, as specified by the CTnOC bit, when the CTnON bit changes from low to high.

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

Comparator P Match Period

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

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

#### • CTMnC1 Register

Bit	7	6	5	4	3	2	1	0
Name	CTnM1	CTnM0	CTnIO1	CTnIO0	CTnOC	CTnPOL	CTnDPX	CTnCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6 **CTnM1~CTnM0**: Select CTMn Operating Mode

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

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

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

Compare Match Output Mode

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

PWM Output Mode

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

Timer/Counter Mode

Unused

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

In the Compare Match Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn output pin changes state when a compare match occurs from the Comparator A. The CTMn output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the CTMn output pin should be setup using the CTnOC bit in the CTMnC1 register. Note that the output level requested by the CTnIO1 and CTnIO0 bits must be different from the initial value setup using the CTnOC bit otherwise no change will occur on the CTMn output pin when a compare match occurs. After the CTMn output pin changes state it can be reset to its initial level by changing the level of the CTnON bit from low to high.

In the PWM Output Mode, the CTnIO1 and CTnIO0 bits determine how the CTMn output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the CTnIO1 and CTnIO0 bits only after the CTMn has been switched off. Unpredictable PWM outputs will occur if the CTnIO1 and CTnIO0 bits are changed when The CTMn is running.

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

Compare Match Output Mode

0: Initial low

1: Initial high

PWM Output Mode

0: Active low

1: Active high

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

Bit 2      **CTnPOL:** CTMn CTPn Output polarity Control

0: Non-invert

1: Invert

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

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

0: CCRP – period; CCRA – duty

1: CCRP – duty; CCRA – period

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

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

0: CTMn Comparatror P match

1: CTMn Comparatror A match

This bit is used to select the method which clears the counter. Remember that the Compact TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the CTnCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The CTnCCLR bit is not used in the PWM Output Mode.



• **CTMnDL Register**

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

Bit 7~0      **D7~D0**: CTMn Counter Low Byte Register bit 7 ~ bit 0  
 CTMn 10-bit Counter bit 7 ~ bit 0

• **CTMnDH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R	R
POR	—	—	—	—	—	—	0	0

Bit 7~2      Unimplemented, read as “0”  
 Bit 1~0      **D9~D8**: CTMn Counter High Byte Register bit 1 ~ bit 0  
 CTMn 10-bit Counter bit 9 ~ bit 8

• **CTMnAL Register**

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

Bit 7~0      **D7~D0**: CTMn CCRA Low Byte Register bit 7 ~ bit 0  
 CTMn 10-bit CCRA bit 7 ~ bit 0

• **CTMnAH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2      Unimplemented, read as “0”  
 Bit 1~0      **D9~D8**: CTMn CCRA High Byte Register bit 1 ~ bit 0  
 CTMn 10-bit CCRA bit 9 ~ bit 8

## **Compact Type TM Operating Modes**

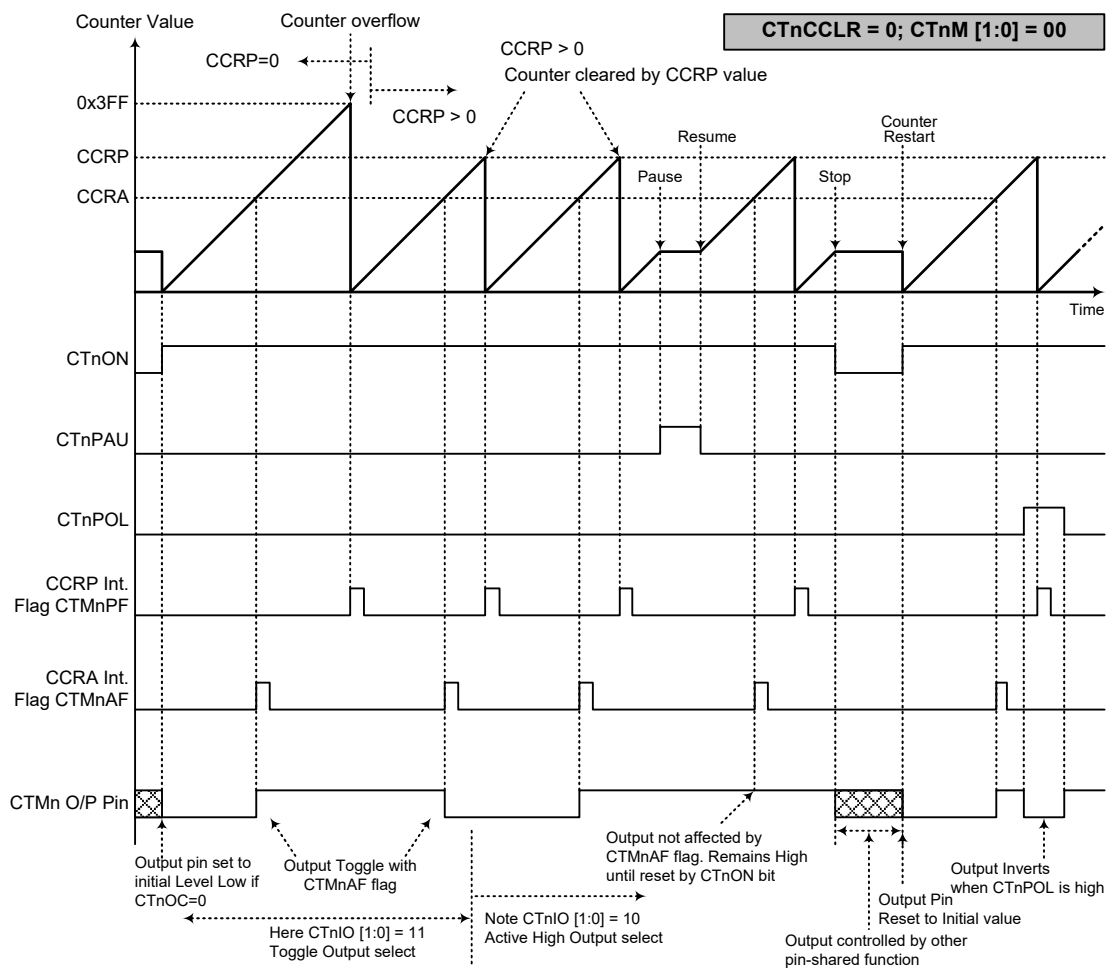
The Compact Type TM can operate in one of three operating modes, Compare Match Output Mode, PWM Output Mode or Timer/Counter Mode. The operating mode is selected using the CTnM1 and CTnM0 bits in the CTMnC1 register.

### **Compare Match Output Mode**

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the CTnCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both CTMnAF and CTMnPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

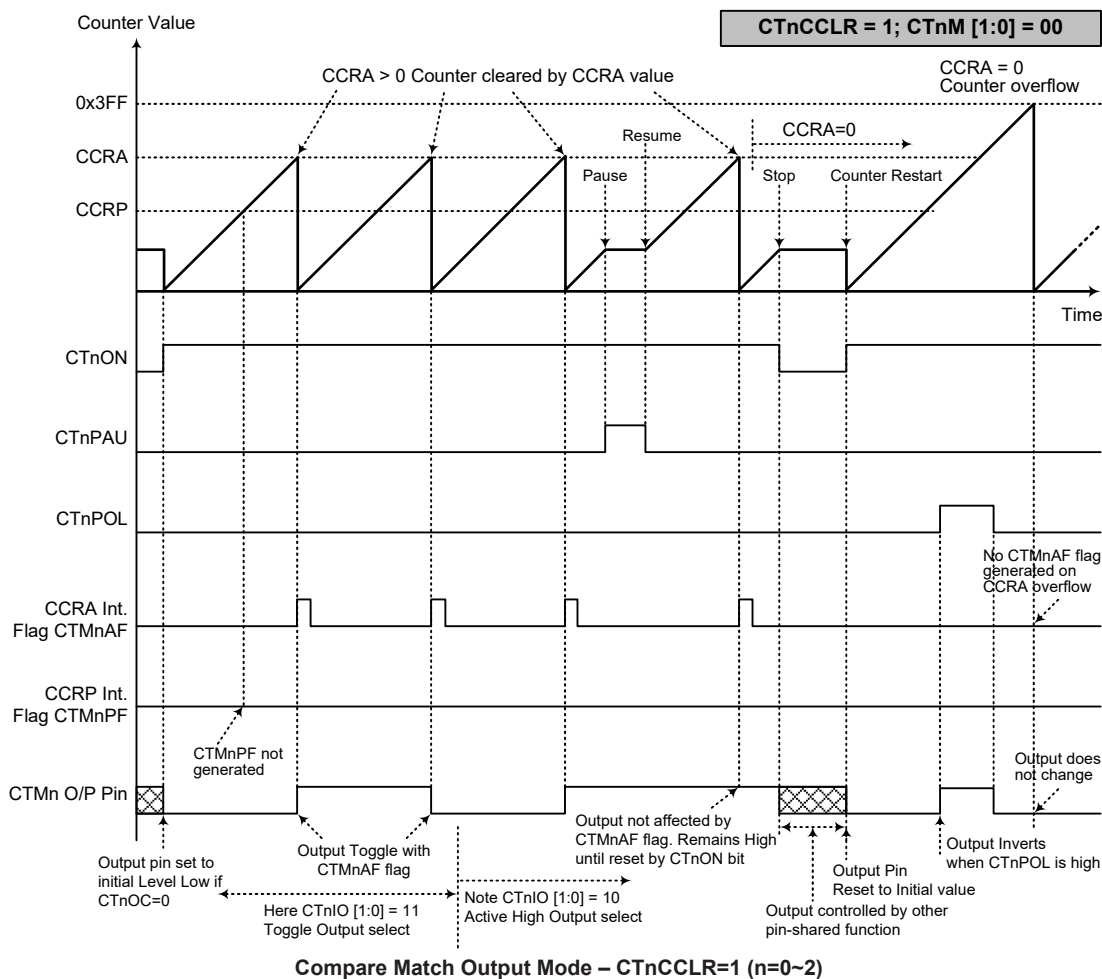
If the CTnCCLR bit in the CTMnC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the CTMnAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore, when CTnCCLR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

As the name of the mode suggests, after a comparison is made, the CTMn output pin will change state. The CTMn output pin condition however only changes state when a CTMnAF interrupt request flag is generated after a compare match occurs from Comparator A. The CTMnPF interrupt request flag, generated from a compare match occurs from Comparator P, will have no effect on the CTMn output pin. The way in which the CTMn output pin changes state are determined by the condition of the CTnIO1 and CTnIO0 bits in the CTMnC1 register. The CTMn output pin can be selected using the CTnIO1 and CTnIO0 bits to go high, to go low or to toggle from its present condition when a compare match occurs from Comparator A. The initial condition of the CTMn output pin, which is setup after the CTnON bit changes from low to high, is setup using the CTnOC bit. Note that if the CTnIO1 and CTnIO0 bits are zero then no pin change will take place.



**Compare Match Output Mode – CTnCCR=0 (n=0~2)**

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



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

### Timer/Counter Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the CTMn output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the CTMn output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

### PWM Output Mode

To select this mode, bits CTnM1 and CTnM0 in the CTMnC1 register should be set to 10 respectively. The PWM function within the CTMn is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the CTMn output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the CTnCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, one register is used to clear the internal counter and thus control the PWM waveform frequency, while the other one is used to control the duty cycle. Which register is used to control either frequency or duty cycle is determined using the CTnDPX bit in the CTMnC1 register. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRA and CCRP registers.

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

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

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

If  $f_{SYS}=12\text{MHz}$ , CTMn clock source is  $f_{SYS}/4$ , CCRP=100b, CCRA=128,

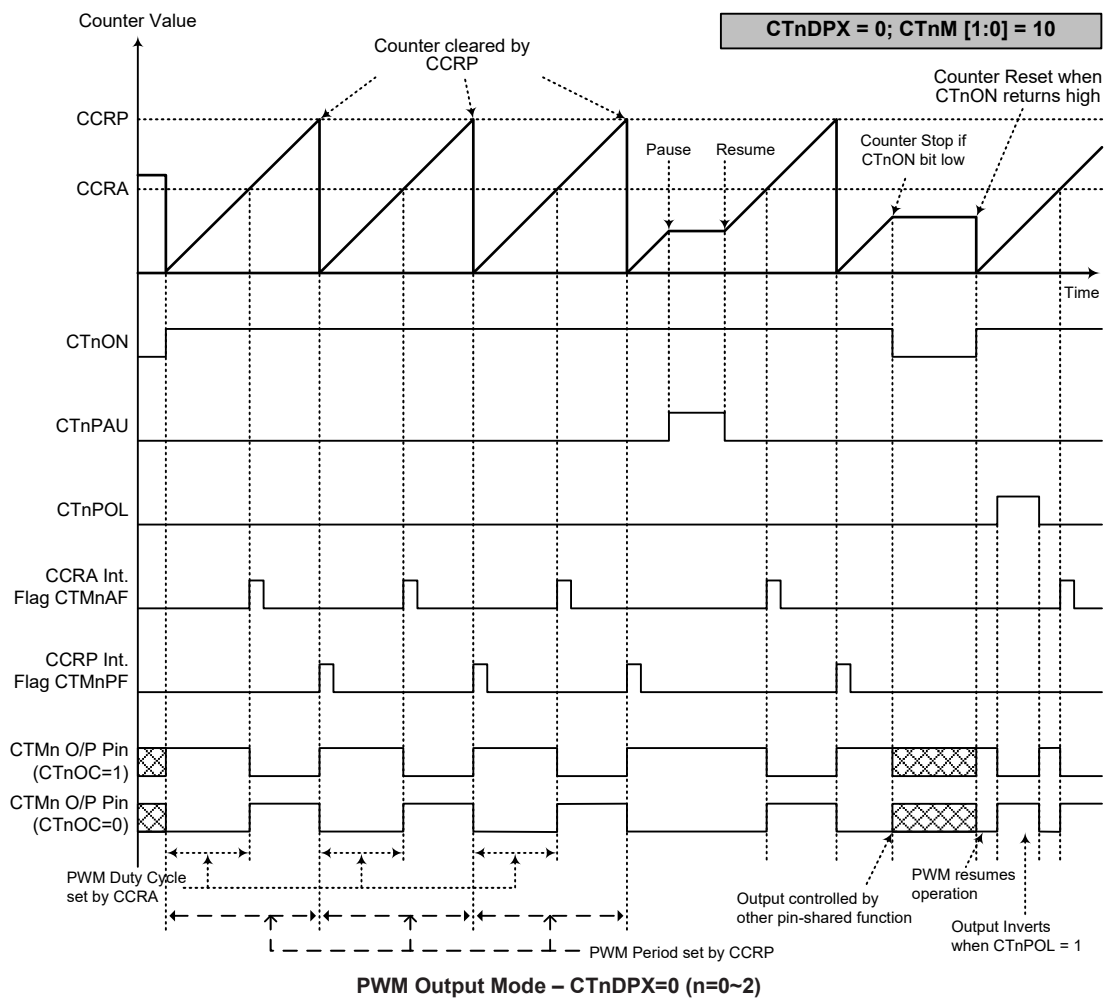
The CTMn PWM output frequency= $(f_{SYS}/4)/512=f_{SYS}/2048=5.859\text{kHz}$ , duty=128/512=25%.

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%.

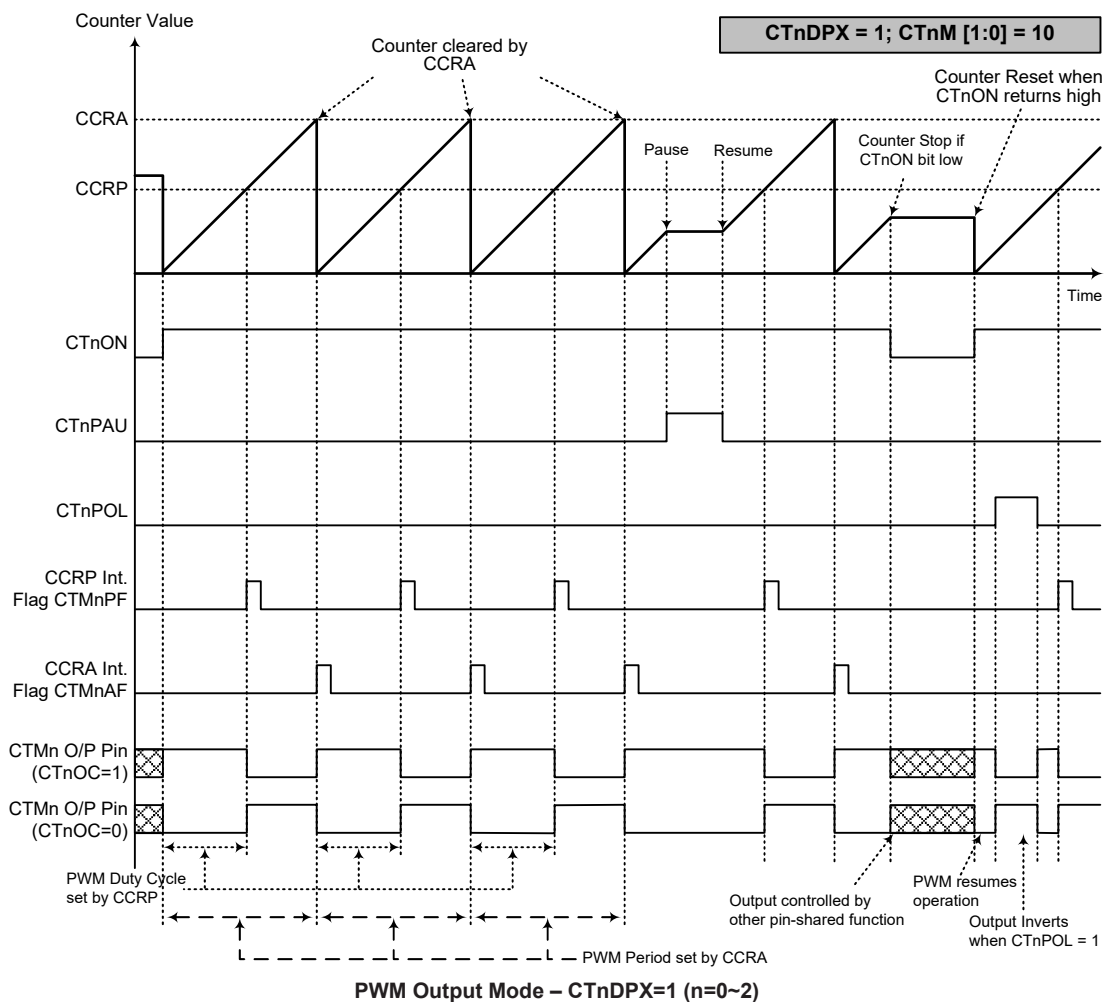
#### • 10-bit CTMn, PWM Output Mode, Edge-aligned Mode, CTnDPX=1

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

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



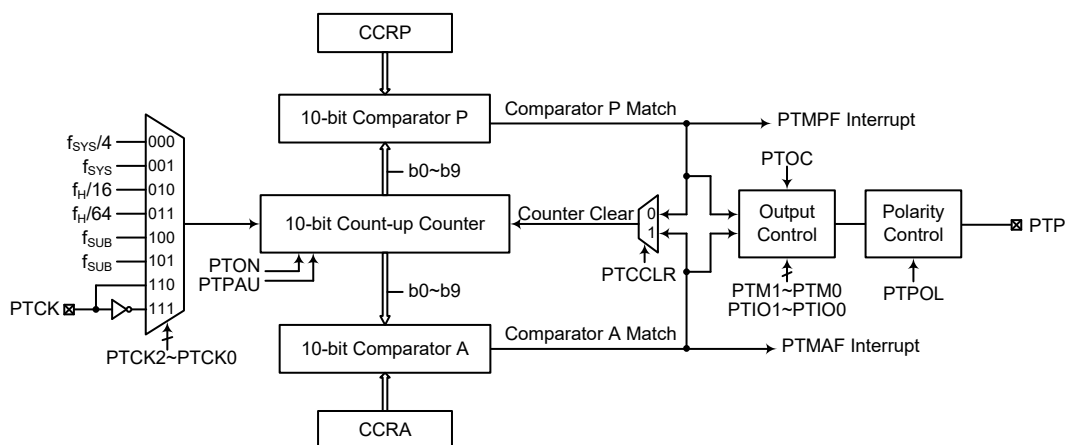
- Note: 1. Here CTnDPX=0 – Counter cleared by CCRP
2. A counter clear sets PWM Period
3. The internal PWM function continues running even when CTnIO[1:0]=00 or 01
4. The CTnCCLR bit has no influence on PWM operation



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

## Periodic Type TM – PTM

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



Note: The PTM external pins are pin-shared with other functions, so before using the PTM function, ensure that the pin-shared function registers have be set properly to enable the PTM pin function.

10-bit Periodic Type TM Block Diagram

## Periodic TM Operation

The size of the 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 the CCRA comparators are both 10-bit wide whose values are compared with all the bits in the counter.

The only way of changing the value of the 10-bit counter using the application program, is to clear the counter by changing the PTON bit from low to high. The counter will also be cleared automatically by a counter overflow or a compare match with one of its associated comparators. When these conditions occur, a PTM interrupt signal will also usually be generated. The Periodic Type TM can operate in a number of different operational modes, can be driven by different clock sources including external input pins and can also control more than one output pins. All operating setup conditions are selected using relevant internal registers.

## Periodic Type TM Register Description

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



Register Name	Bit							
	7	6	5	4	3	2	1	0
PTMC0	PTPAU	PTCK2	PTCK1	PTCK0	PTON	—	—	—
PTMC1	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	D1	PTCCLR
PTMDL	D7	D6	D5	D4	D3	D2	D1	D0
PTMDH	—	—	—	—	—	—	D9	D8
PTMAL	D7	D6	D5	D4	D3	D2	D1	D0
PTMAH	—	—	—	—	—	—	D9	D8
PTMRPL	D7	D6	D5	D4	D3	D2	D1	D0
PTMRPH	—	—	—	—	—	—	D9	D8

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

• **PTMC0 Register**

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

Bit 7 **PTPAU**: PTM Counter Pause Control

0: Run  
1: Pause

The counter can be paused by setting this bit high. Clearing the bit to zero restores normal counter operation. When in a Pause condition the PTM will remain powered up and continue to consume power. The counter will retain its residual value when this bit changes from low to high and resume counting from this value when the bit changes to a low value again.

Bit 6~4 **PTCK2~PTCK0**: Select PTM Counter clock

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

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

Bit 3 **PTON**: PTM Counter On/Off Control

0: Off  
1: On

This bit controls the overall on/off function of the PTM. Setting the bit high enables the counter to run while clearing the bit disables the PTM. Clearing this bit to zero will stop the counter from counting and turn off the PTM which will reduce its power consumption. When the bit changes state from low to high the internal counter value will be reset to zero, however when the bit changes from high to low, the internal counter will retain its residual value until the bit returns high again.

If the PTM is in the Compare Match Output Mode, the PWM Output Mode or the Single Pulse Output Mode then the PTM output pin will be reset to its initial condition, as specified by the PTOC bit, when the PTON bit changes from low to high.

Bit 2~0 Unimplemented, read as “0”

• **PTMC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTM1	PTM0	PTIO1	PTIO0	PTOC	PTPOL	D1	PTCCLR
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7~6     **PTM1~PTM0**: Select PTM Operating Mode  
               00: Compare Match Output Mode  
               01: Undefined  
               10: PWM Output Mode or Single Pulse Output Mode  
               11: Timer/Counter Mode

These bits setup the required operating mode for the PTM. To ensure reliable operation the PTM should be switched off before any changes are made to the PTM1 and PTM0 bits. In the Timer/Counter Mode, the PTM output pin state is undefined.

Bit 5~4     **PTIO1~PTIO0**: Select PTM external pin function

Compare Match Output Mode

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

PWM Output Mode/Single Pulse Output Mode

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

Timer/Counter Mode

Unused

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

In the Compare Match Output Mode, the PTIO1 and PTIO0 bits determine how the PTM output pin changes state when a compare match occurs from the Comparator A. The PTM output pin can be setup to switch high, switch low or to toggle its present state when a compare match occurs from the Comparator A. When the bits are both zero, then no change will take place on the output. The initial value of the PTM output pin should be setup using the PTOC bit in the PTMC1 register. Note that the output level requested by the PTIO1 and PTIO0 bits must be different from the initial value setup using the PTOC bit otherwise no change will occur on the PTM output pin when a compare match occurs. After the PTM output pin changes state, it can be reset to its initial level by changing the level of the PTON bit from low to high.

In the PWM Output Mode, the PTIO1 and PTIO0 bits determine how the PTM output pin changes state when a certain compare match condition occurs. The PWM output function is modified by changing these two bits. It is necessary to only change the values of the PTIO1 and PTIO0 bits only after the PTM has been switched off. Unpredictable PWM outputs will occur if the PTIO1 and PTIO0 bits are changed when the PTM is running.

Bit 3        **PTOC**: PTM PTP Output control bit

Compare Match Output Mode

- 0: Initial low
- 1: Initial high

PWM Output Mode/Single Pulse Output Mode

- 0: Active low
- 1: Active high

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

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

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

Bit 1 **D1**: Reserved, must be fixed at “0”

Bit 0 **PTCCLR**: PTM Counter Clear condition selection  
 0: Comparator P match  
 1: Comparator A match

This bit is used to select the method which clears the counter. Remember that the Periodic TM contains two comparators, Comparator A and Comparator P, either of which can be selected to clear the internal counter. With the PTCCLR bit set high, the counter will be cleared when a compare match occurs from the Comparator A. When the bit is low, the counter will be cleared when a compare match occurs from the Comparator P or with a counter overflow. A counter overflow clearing method can only be implemented if the CCRP bits are all cleared to zero. The PTCCLR bit is not used in the PWM Output Mode or Single Pulse Output Mode.

• **PTMDL Register**

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

Bit 7~0 **D7~D0**: PTM Counter Low Byte Register bit 7 ~ bit 0  
 PTM 10-bit Counter bit 7 ~ bit 0

• **PTMDH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R	R
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”  
 Bit 1~0 **D9~D8**: PTM Counter High Byte Register bit 1 ~ bit 0  
 PTM 10-bit Counter bit 9 ~ bit 8

• **PTMAL Register**

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

Bit 7~0 **D7~D0**: PTM CCRA Low Byte Register bit 7 ~ bit 0  
 PTM 10-bit CCRA bit 7 ~ bit 0

• **PTMAH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **D9~D8**: PTM CCRA High Byte Register bit 1 ~ bit 0  
PTM 10-bit CCRA bit 9 ~ bit 8

• **PTMRPL Register**

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

Bit 7~0 **D7~D0**: PTM CCRP Low Byte Register bit 7 ~ bit 0  
PTM 10-bit CCRP bit 7 ~ bit 0

• **PTMRPH Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	D9	D8
R/W	—	—	—	—	—	—	R/W	R/W
POR	—	—	—	—	—	—	0	0

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **D9~D8**: PTM CCRP High Byte Register bit 1 ~ bit 0  
PTM 10-bit CCRP bit 9 ~ bit 8

## Periodic Type TM Operating Modes

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

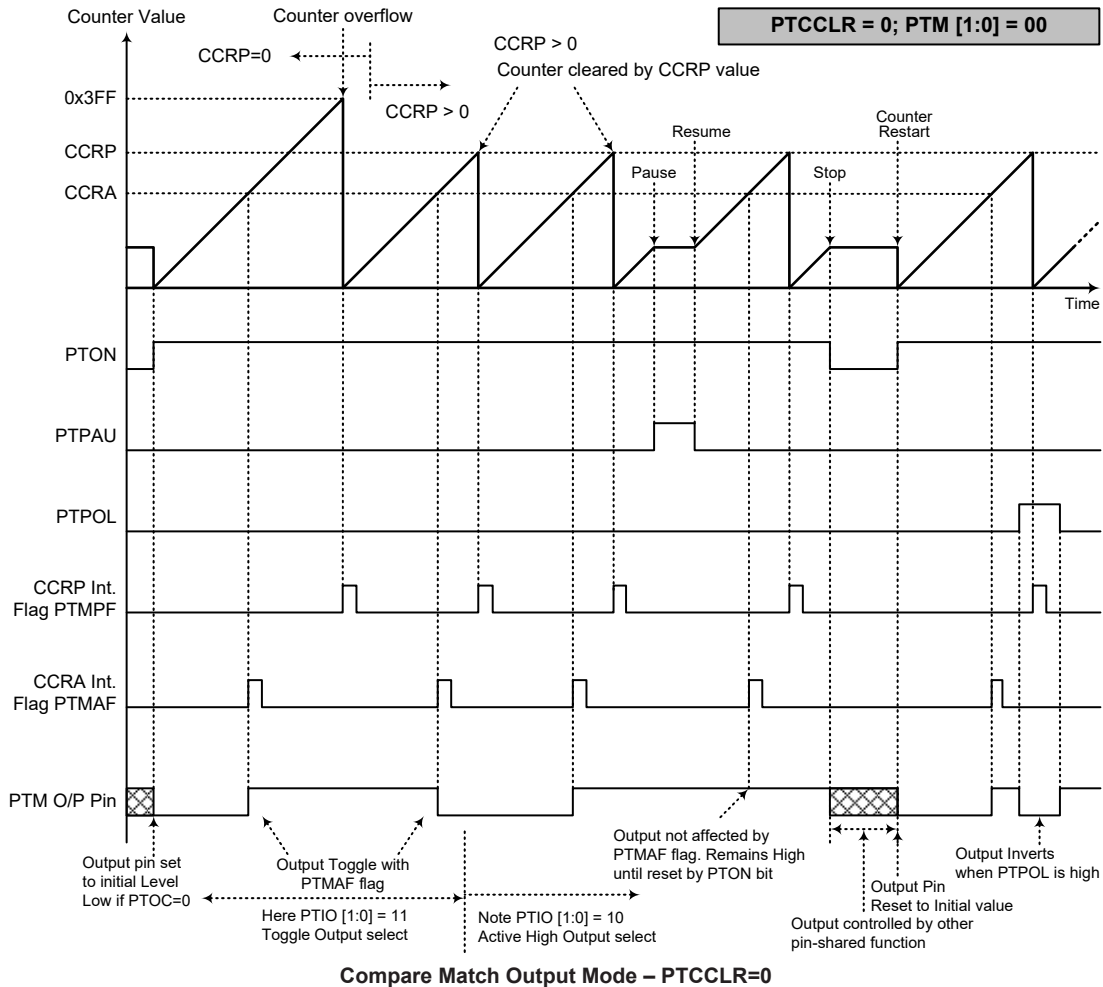
### Compare Match Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register, should be set to 00 respectively. In this mode once the counter is enabled and running it can be cleared by three methods. These are a counter overflow, a compare match from Comparator A and a compare match from Comparator P. When the PTCCLR bit is low, there are two ways in which the counter can be cleared. One is when a compare match from Comparator P, the other is when the CCRP bits are all zero which allows the counter to overflow. Here both PTMAF and PTMPF interrupt request flags for Comparator A and Comparator P respectively, will both be generated.

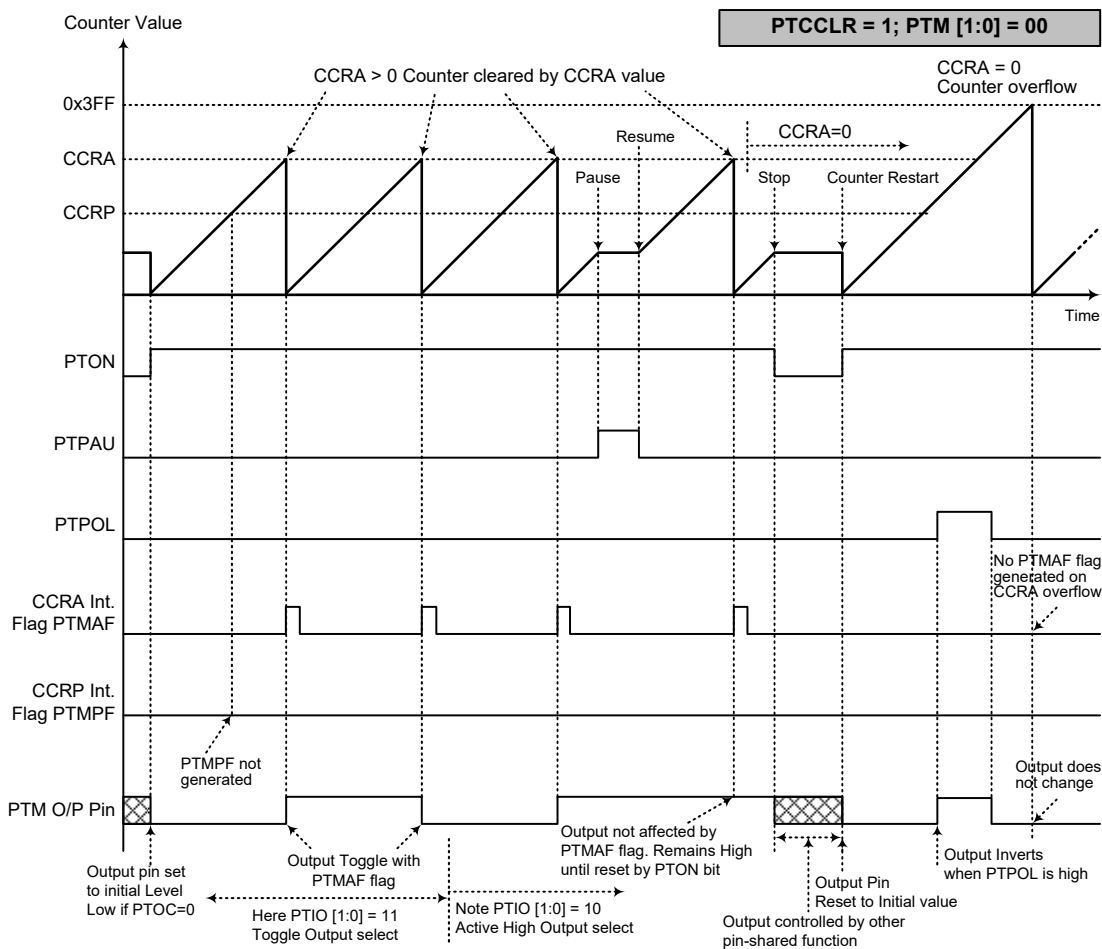
If the PTCCLR bit in the PTMC1 register is high then the counter will be cleared when a compare match occurs from Comparator A. However, here only the PTMAF interrupt request flag will be generated even if the value of the CCRP bits is less than that of the CCRA registers. Therefore when PTCCLR is high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

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

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



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



#### Compare Match Output Mode – PTCCLR=1

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

### Timer/Counter Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 11 respectively. The Timer/Counter Mode operates in an identical way to the Compare Match Output Mode generating the same interrupt flags. The exception is that in the Timer/Counter Mode the PTM output pin is not used. Therefore the above description and Timing Diagrams for the Compare Match Output Mode can be used to understand its function. As the PTM output pin is not used in this mode, the pin can be used as a normal I/O pin or other pin-shared function.

### PWM Output Mode

To select this mode, bits PTM1 and PTM0 in the PTMC1 register should be set to 10 respectively and also the PTIO1 and PTIO0 bits should be set to 10 respectively. The PWM function within the PTM is useful for applications which require functions such as motor control, heating control, illumination control etc. By providing a signal of fixed frequency but of varying duty cycle on the PTM output pin, a square wave AC waveform can be generated with varying equivalent DC RMS values.

As both the period and duty cycle of the PWM waveform can be controlled, the choice of generated waveform is extremely flexible. In the PWM Output Mode, the PTCCLR bit has no effect on the PWM operation. Both of the CCRA and CCRP registers are used to generate the PWM waveform, the CCRP is used to clear the internal counter and thus control the PWM waveform frequency, while the CCRA is used to control the duty cycle. The PWM waveform frequency and duty cycle can therefore be controlled by the values in the CCRP and CCRA registers.

An interrupt flag, one for each of the CCRA and CCRP, will be generated when a compare match occurs from either Comparator A or Comparator P. The PTOC bit in the PTMC1 register is used to select the required polarity of the PWM waveform while the two PTIO1 and PTIO0 bits are used to enable the PWM output or to force the PTM output pin to a fixed high or low level. The PTPOL bit is used to reverse the polarity of the PWM output waveform.

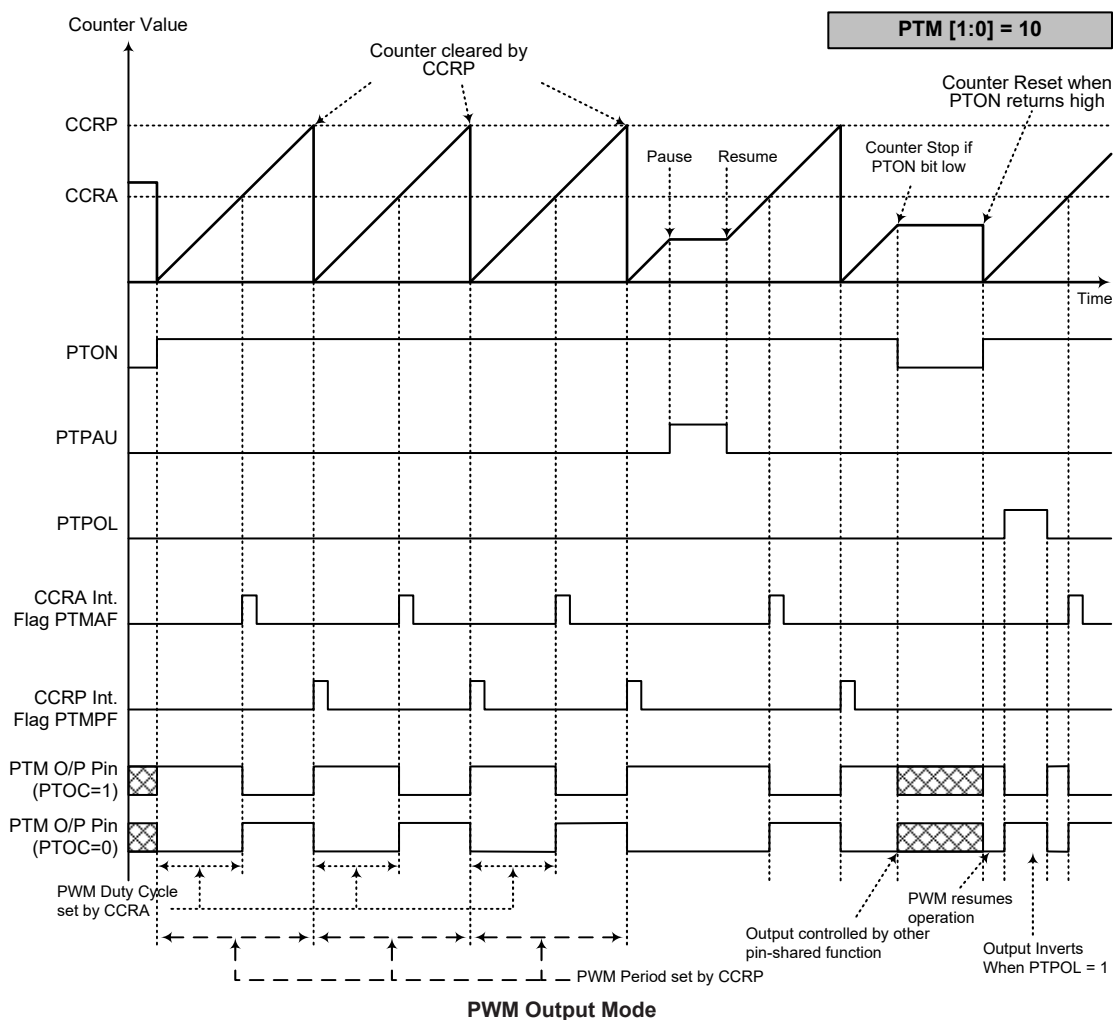
#### • 10-bit PTM, PWM Output Mode, Edge-aligned Mode

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

If  $f_{SYS}=12\text{MHz}$ , PTM clock source select  $f_{SYS}/4$ , CCRP=512 and CCRA=128,

The PTM PWM output frequency= $(f_{SYS}/4)/512=f_{SYS}/2048=5.859\text{kHz}$ , duty= $128/512=25\%$ .

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



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

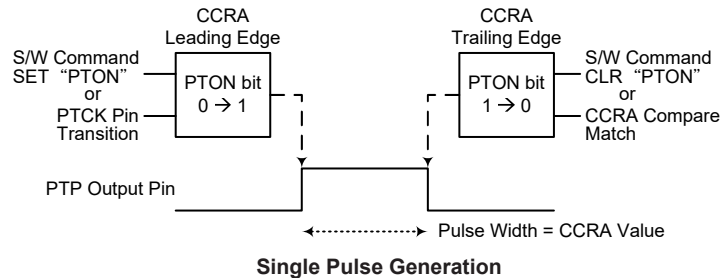


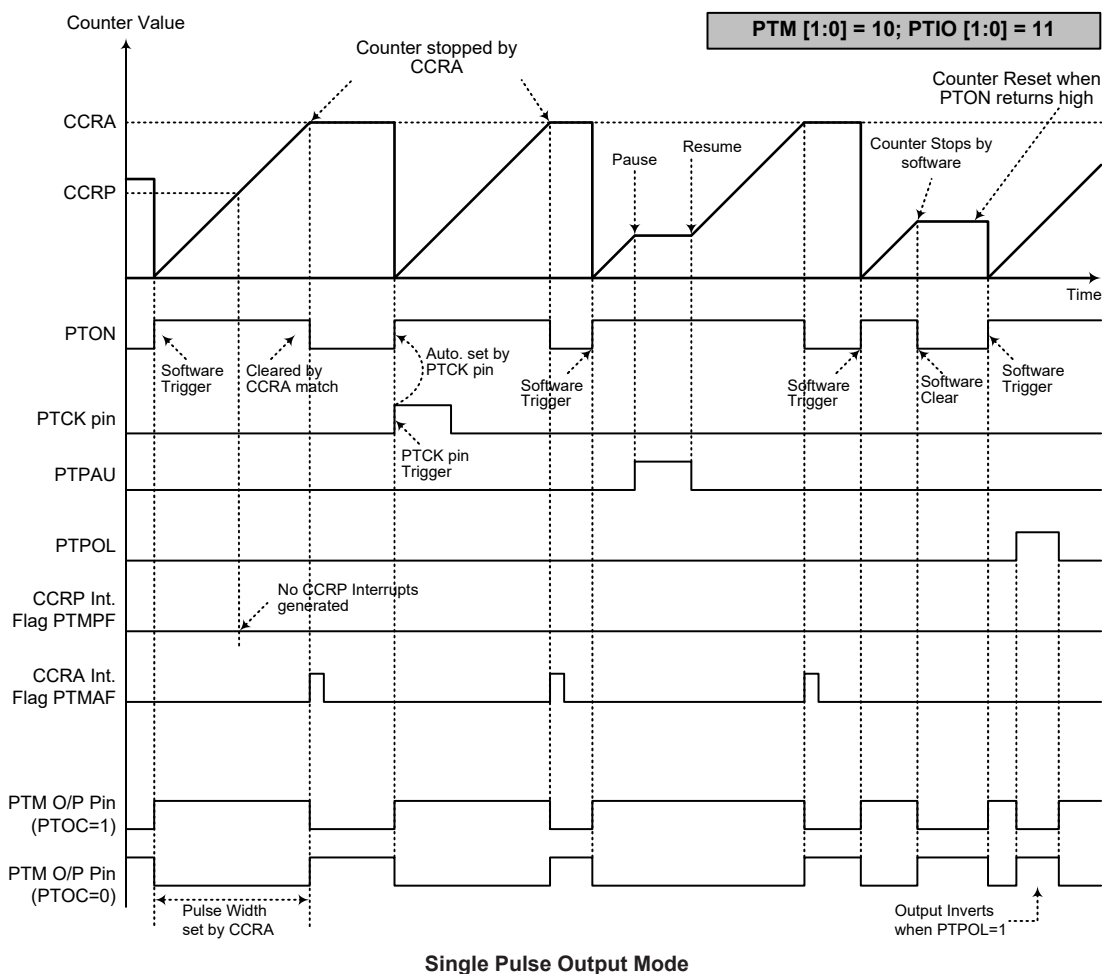
### Single Pulse Output Mode

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

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

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





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

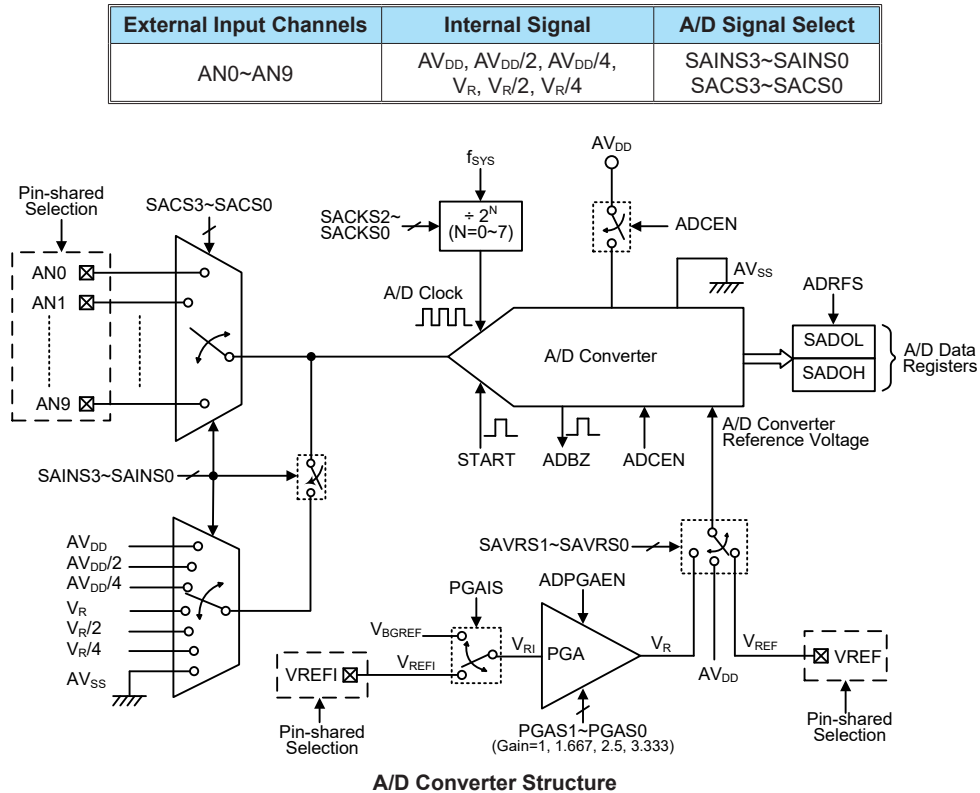
## Analog to Digital Converter – ADC

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

The device contains a multi-channel analog to digital converter which can directly interface to external analog signals, such as that from sensors or other control signals and convert these signals directly into a 12-bit digital value. It also can convert the internal signals, such as the internal reference voltage, into a 12-bit digital value. The external or internal analog signal to be converted is determined by the SAINS and SACS bit fields. Note that when the internal analog signal is selected to be converted using the SAINS field, the external channel analog input will automatically be switched off. More detailed information about the A/D input signal selection will be described in the “A/D Converter Input Signals” section.

The accompanying block diagram shows the internal structure of the A/D converter with its associated registers and control bits.



### Registers Descriptions

Overall operation of the A/D converter is controlled using six registers. A read only register pair exists to store the A/D Converter data 12-bit value. Three registers, SADC0, SADC1 and SADC2, are the control registers which setup the operating conditions and control function of the A/D converter. The VBGRC register contains the VBGREN bit to control the bandgap reference voltage.

Register Name	Bit							
	7	6	5	4	3	2	1	0
SADOL (ADRF5=0)	D3	D2	D1	D0	—	—	—	—
SADOL (ADRF5=1)	D7	D6	D5	D4	D3	D2	D1	D0
SADOH (ADRF5=0)	D11	D10	D9	D8	D7	D6	D5	D4
SADOH (ADRF5=1)	—	—	—	—	D11	D10	D9	D8
SADC0	START	ADBZ	ADCEN	ADRF5	SACS3	SACS2	SACS1	SACS0
SADC1	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
SADC2	ADPGAEN	—	—	PGAIS	SAVRS1	SAVRS0	PGAGS1	PGAGS0
VBGRC	—	—	—	—	—	—	—	VBGREN

**A/D Converter Register List**
**A/D Converter Data Registers – SADOL, SADOH**

As the device contains an internal 12-bit A/D converter, it requires two data registers to store the converted value. These are a high byte register, known as SADOH, and a low byte register, known as SADOL. After the conversion process takes place, these registers can be directly read by the microcontroller to obtain the digitised conversion value. As only 12 bits of the 16-bit register space is utilised, the format in which the data is stored is controlled by the ADRF5 bit in the SADC0 register as shown in the accompanying table. D0~D11 are the A/D conversion result data bits. Any unused bits will be read as zero. The A/D data registers contents will be unchanged if the A/D converter is disabled.

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

**A/D Converter Data Registers**
**A/D Converter Control Registers – SADC0, SADC1, SADC2**

To control the function and operation of the A/D converter, three control registers known as SADC0, SADC1 and SADC2 are provided. These 8-bit registers define functions such as the selection of which analog signal is connected to the internal A/D converter, the digitised data format, the A/D clock source as well as controlling the start function and monitoring the A/D converter busy status. As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The SAINS field in the SADC1 register and SACS field in the SADC0 register are used to determine which analog signal derived from the external or internal signals will be connected to the A/D converter. The A/D converter also contains a programmable gain amplifier, PGA, to generate the A/D converter internal reference voltage. The overall operation of the PGA is controlled using the SADC2 register.

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

• **SADC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	START	ADBZ	ADCEN	ADRF5	SACS3	SACS2	SACS1	SACS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7      **START**: Start the A/D Conversion  
0→1→0: Start  
This bit is used to initiate an A/D conversion process. The bit is normally low but if set high and then cleared low again, the A/D converter will initiate a conversion process.
- Bit 6      **ADBZ**: A/D Converter busy flag  
0: No A/D conversion is in progress  
1: A/D conversion is in progress  
This read only flag is used to indicate whether the A/D conversion is in progress or not. When the START bit is set from low to high and then to low again, the ADBZ flag will be set to 1 to indicate that the A/D conversion is initiated. The ADBZ flag will be cleared to 0 after the A/D conversion is complete.
- Bit 5      **ADCEN**: A/D Converter function enable control  
0: Disable  
1: Enable  
This bit controls the A/D internal function. This bit should be set to one to enable the A/D converter. If the bit is set low, then the A/D converter will be switched off reducing the device power consumption. When the A/D converter function is disabled, the contents of the A/D data register pair known as SADOL and SADOH will be unchanged.
- Bit 4      **ADRF5**: A/D conversion data format select  
0: A/D converter data format → SADOH=D [11:4]; SADOL=D [3:0]  
1: A/D converter data format → SADOH=D [11:8]; SADOL=D [7:0]  
This bit controls the format of the 12-bit converted A/D value in the two A/D data registers. Details are provided in the A/D converter data register section.
- Bit 3~0    **SACS3~SACS0**: A/D converter external analog input channel select  
0000: External AN0 input  
0001: External AN1 input  
0010: External AN2 input  
0011: External AN3 input  
0100: External AN4 input  
0101: External AN5 input  
0110: External AN6 input  
0111: External AN7 input  
1000: External AN8 input  
1001: External AN9 input  
1010~1111: Non-existed channel, the input will be floating

• **SADC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	SAINS3	SAINS2	SAINS1	SAINS0	—	SACKS2	SACKS1	SACKS0
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 7~4

**SAINS3~SAINS0:** A/D converter input signal select

- 0000: External source – External analog channel input, ANn
- 0001: Internal source – Internal signal derived from AV<sub>DD</sub>
- 0010: Internal source – Internal signal derived from AV<sub>DD</sub>/2
- 0011: Internal source – Internal signal derived from AV<sub>DD</sub>/4
- 0100: External source – External analog channel input, ANn
- 0101: Internal source – Internal signal derived from PGA output V<sub>R</sub>
- 0110: Internal source – Internal signal derived from PGA output V<sub>R</sub>/2
- 0111: Internal source – Internal signal derived from PGA output V<sub>R</sub>/4
- 10xx: Internal source – Ground
- 11xx: External source – External analog channel input, ANn

When the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off regardless of the SACKS field value. It will prevent the external channel input from being connected together with the internal analog signal.

Bit 3

Unimplemented, read as “0”

Bit 2~0

**SACKS2~SACKS0:** A/D conversion clock source select

- 000: f<sub>sys</sub>
- 001: f<sub>sys</sub>/2
- 010: f<sub>sys</sub>/4
- 011: f<sub>sys</sub>/8
- 100: f<sub>sys</sub>/16
- 101: f<sub>sys</sub>/32
- 110: f<sub>sys</sub>/64
- 111: f<sub>sys</sub>/128

• **SADC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	ADPGAEN	—	—	PGAIS	SAVRS1	SAVRS0	PGAGS1	PGAGS0
R/W	R/W	—	—	R/W	R/W	R/W	R/W	R/W
POR	0	—	—	0	0	0	0	0

Bit 7

**ADPGAEN:** PGA enable control

- 0: Disable
- 1: Enable

Bit 6~5

Unimplemented, read as “0”

Bit 4

**PGAIS:** PGA input voltage selection

- 0: From VREFI pin
- 1: From internal reference voltage VBGREF

When the internal independent reference voltage VBGREF is selected as the PGA input, the external reference voltage on the VREFI pin will be automatically switched off. In addition, the internal bandgap reference VBGREF should be enabled by setting the VBGREN bit in the VBGRC register to “1”.

Bit 3~2

**SAVRS1~SAVRS0:** A/D converter reference voltage select

- 00: Internal A/D converter power, AV<sub>DD</sub>.
- 01: External VREF pin
- 1x: Internal PGA output voltage, V<sub>R</sub>.

These bits are used to select the A/D converter reference voltage source. When the internal reference voltage source is selected, the reference voltage derived from the external VREF pin will automatically be switched off.

- Bit 1~0 **PGAGS1~PGAGS0**: PGA gain select  
 00: Gain=1  
 01: Gain=1.667 –  $V_R=2V$  as  $V_{RI}=1.2V$   
 10: Gain=2.5 –  $V_R=3V$  as  $V_{RI}=1.2V$   
 11: Gain=3.333 –  $V_R=4V$  as  $V_{RI}=1.2V$

These bits are used to select the PGA gain. Note that here the gain is guaranteed only when the PGA input voltage is equal to 1.2V.

• **VBGRC Register**

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

Bit 7~1 Unimplemented, read as “0”

- Bit 0 **VBGREN**: Bandgap reference voltage control  
 0: Disable  
 1: Enable

This bit is used to enable the internal Bandgap reference circuit. The internal Bandgap reference circuit should first be enabled before the  $V_{BGREF}$  voltage is selected to be used. A specific start-up time is necessary for the Bandgap circuit to become stable and accurate.

## A/D Converter Reference Voltage

The actual reference voltage supply to the A/D Converter can be supplied from the positive power supply,  $AV_{DD}$ , an external reference source supplied on pin VREF or an internal reference voltage  $V_R$  determined by the SAVRS1~SAVRS0 bits in the SADC2 register. The internal reference voltage is amplified through a programmable gain amplifier, PGA, which is controlled by the ADPGAEN bit in the SADC2 register. The PGA gain can be equal to 1, 1.667, 2.5 or 3.333 and selected using the PGAGS1~PGAGS0 bits in the SADC2 register. The PGA input can come from the external reference input pin, VREFI, or an internal Bandgap reference voltage,  $V_{BGREF}$ , selected by the PGAIS bit in the SADC2 register. As the VREFI and VREF pins both are pin-shared with other functions, when the VREFI or VREF pin is selected as the reference voltage pin, the VREFI or VREF pin-shared function selection bits should first be properly configured to disable other pin-shared functions. However, if the internal reference signal is selected as the reference source, the external reference input from the VREFI or VREF pin will automatically be switched off by hardware.

Note that the internal Bandgap reference circuit should first be enabled before the  $V_{BGREF}$  is selected to be used. A specific start-up time is necessary for the Bandgap circuit to become stable and accurate.

## A/D Converter Input Signals

All of the external A/D analog input pins are pin-shared with the I/O pins as well as other functions. The corresponding pin-shared function selection bits in the PxS1 and PxS0 registers, determine whether the external input pins are setup as A/D converter analog channel inputs or whether they have other functions. If the corresponding pin is setup to be an A/D converter analog channel input, the original pin function will be disabled. In this way, pins can be changed under program control to change their function between A/D inputs and other functions. All pull-high resistors, which are setup through register programming, will be automatically disconnected if the pins are setup as A/D

inputs. Note that it is not necessary to first setup the A/D pin as an input in the port control register to enable the A/D input as when the relevant A/D input function selection bits enable an A/D input, the status of the port control register will be overridden.

As the device contains only one actual analog to digital converter hardware circuit, each of the external and internal analog signals must be routed to the converter. The SAINS3~SAINS0 bits in the SADC1 register are used to determine that the analog signal to be converted comes from the external channel input or internal analog signal. The SACS3~SACS0 bits in the SADC0 register are used to determine which external channel input is selected to be converted. If the SAINS3~SAINS0 bits are set to “0000”, “0100” or “11xx”, the external channel input will be selected to be converted and the SACS3~SACS0 bits can determine which external channel is selected.

When the SAINS field is set to the value of “0x01”, “0x10” or “0x11”, the internal analog signal will be selected. If the internal analog signal is selected to be converted, the external channel signal input will automatically be switched off regardless of the SACS field value. It will prevent the external channel input from being connected together with the internal analog signal.

SAINS [3:0]	SACS [3:0]	Input Signals	Description
0000, 0100, 11xx	0000~1001 1010~1111	AN0~AN9 —	External channel analog input ANn Non-existed channel, input is floating.
0001	xxxx	AV <sub>DD</sub>	Internal signal derived from AV <sub>DD</sub>
0010	xxxx	AV <sub>DD</sub> /2	Internal signal derived from AV <sub>DD</sub> /2
0011	xxxx	AV <sub>DD</sub> /4	Internal signal derived from AV <sub>DD</sub> /4
0101	xxxx	V <sub>R</sub>	Internal signal derived from PGA output V <sub>R</sub>
0110	xxxx	V <sub>R</sub> /2	Internal signal derived from PGA output V <sub>R</sub> /2
0111	xxxx	V <sub>R</sub> /4	Internal signal derived from PGA output V <sub>R</sub> /4
10xx	xxxx	AV <sub>SS</sub>	Connected to the ground

**A/D Converter Input Signal Selection**

### A/D Conversion Operation

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

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

The clock source for the A/D converter, which originates from the system clock  $f_{SYS}$ , can be chosen to be either  $f_{SYS}$  or a subdivided version of  $f_{SYS}$ . The division ratio value is determined by the SACKS2~SACKS0 bits in the SADC1 register. Although the A/D clock source is determined by the system clock  $f_{SYS}$  and by bits SACKS2~SACKS0, there are some limitations on the maximum A/D clock source speed that can be selected. As the recommended range of permissible A/D clock period,  $t_{ADCK}$ , is from 0.5 $\mu$ s to 10 $\mu$ s @ 2.0V $\leq$ V<sub>DD</sub> $\leq$ 5.5V, care must be taken for system clock frequencies. For example, if the system clock operates at a frequency of 4MHz, the SACKS2~SACKS0 bits should not be set to 000, 001 or 111. Doing so will give A/D clock periods that are less than the



minimum A/D clock period or greater than the maximum A/D clock period which may result in inaccurate A/D conversion values. Refer to the following table for examples, where values marked with an asterisk \* show where, special care must be taken.

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

**A/D Clock Period Examples @ 2.0V $\leq$ V<sub>DD</sub> $\leq$ 5.5V**

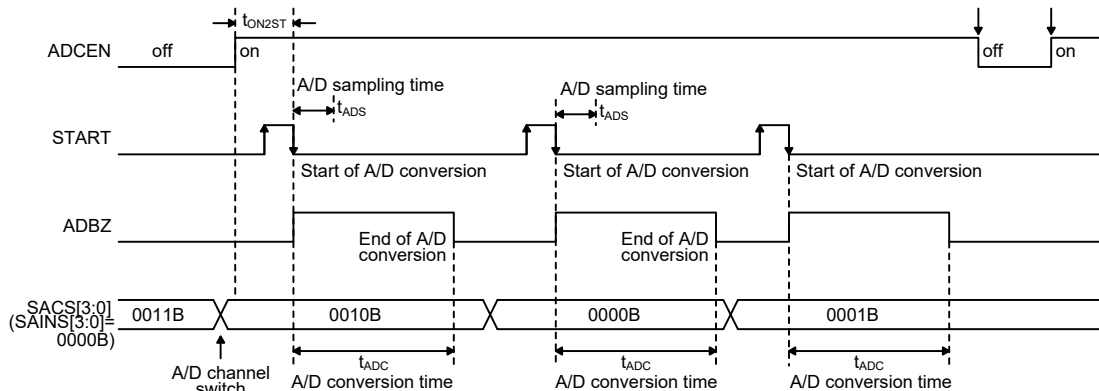
Controlling the power on/off function of the A/D converter circuitry is implemented using the ADCEN bit in the SADC0 register. This bit must be set high to power on the A/D converter. When the ADCEN bit is set high to power on the A/D converter internal circuitry, a certain delay as indicated in the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

## Conversion Rate and Timing Diagram

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

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

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



**A/D Conversion Timing – External Channel Input**

## Summary of A/D Conversion Steps

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

- Step 1  
Select the required A/D conversion clock by properly programming the SACKS2~SACKS0 bits in the SADC1 register.
- Step 2  
Enable the A/D converter by setting the ADCEN bit in the SADC0 register to one.
- Step 3  
Select which signal is to be connected to the internal A/D converter by correctly configuring the SACS and SAINS bit fields  
Selecting the external channel input to be converted, go to Step 4.  
Selecting the internal analog signal to be converted, go to Step 5.
- Step 4  
If the SAINS field is 0000, 0100 or 11xx, the external channel input can be selected. The desired external channel input is selected by configuring the SACS field. When the A/D input signal comes from the external channel input, the corresponding pin should be configured as an A/D input function by selecting the relevant pin-shared function control bits. Then go to Step 6.
- Step 5  
If the SAINS field is set to 0x01, 0x10 or 0x11, the relevant internal analog signal will be selected. When the internal analog signal is selected to be converted, the external channel analog input will automatically be disconnected. Then go to Step 6.
- Step 6  
Select the A/D converter output data format by configuring the ADRFS bit.
- Step 7  
Select the A/D converter reference voltage source by configuring the SAVRS bit field.  
Select the PGA input signal and the desired PGA gain if the PGA output voltage,  $V_R$ , is selected as the A/D converter reference voltage.
- Step 8  
If A/D conversion interrupt is used, the interrupt control registers must be correctly configured to ensure the A/D interrupt function is active. The master interrupt control bit, EMI, and the A/D conversion interrupt control bit, ADE, must both be set high in advance.
- Step 9  
The A/D conversion procedure can now be initialized by setting the START bit from low to high and then low again.
- Step 10  
If A/D conversion is in progress, the ADBZ flag will be set high. After the A/D conversion process is complete, the ADBZ flag will go low and then the output data can be read from SADOH and SADOL registers.

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

## Programming Considerations

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

## A/D Transfer Function

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

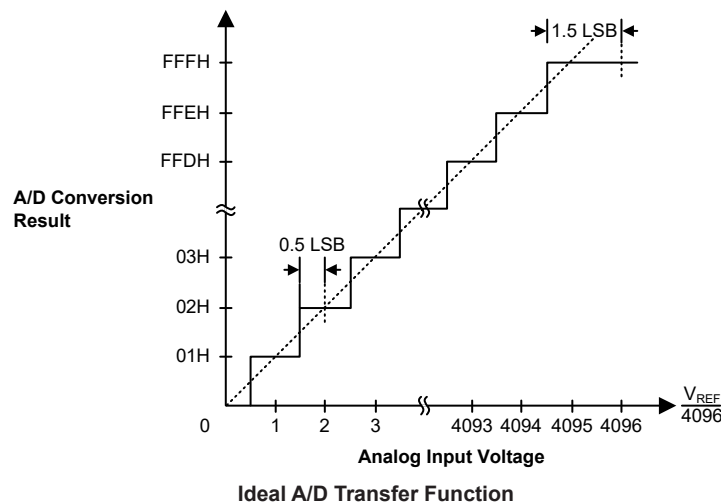
$$1 \text{ LSB} = V_{REF}/4096$$

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

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

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

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



## A/D Programming Examples

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

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

```
clr ADE           ; disable ADC interrupt
mov a,03H         ; select fsys/8 as A/D clock and A/D input
mov SADC1,a       ; signal comes from external channel
mov a,00H         ; select AVDD as the A/D reference voltage source
```

```

mov SADC2,a
mov a,03H          ; setup PBS0 to configure pin AN0
mov PBS0,a
mov a,20H          ; enable A/D converter and select AN0 as the A/D external channel
                      ; input

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

```

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

```

clr ADE            ; disable ADC interrupt
mov a,03H          ; select fsys/8 as A/D clock and A/D input
mov SADC1,a        ; signal comes from external channel
mov a,00H          ; select AVDD as the A/D reference voltage source
mov SADC2,a
mov a,03h          ; setup PBS0 to configure pin AN0
mov PBS0,a
mov a,20h
mov SADC0,a        ; enable A/D converter and select AN0 as the A/D external channel
                      ; input
:
Start_conversion:
clr START          ; high pulse on START bit to initiate conversion
set START          ; reset A/D
clr START          ; start A/D
clr ADF            ; clear ADC interrupt request flag
set ADE            ; enable ADC interrupt
set EMI            ; enable global interrupt
:
:
ADC_ISR:           ; ADC interrupt service routine
mov acc_stack,a    ; save ACC to user defined memory
mov a,STATUS
mov status_stack,a ; save STATUS to user defined memory
:
mov a,SADOL        ; read low byte conversion result value
mov SADOL_buffer,a ; save result to user defined register
mov a,SADOH        ; read high byte conversion result value
mov SADOH_buffer,a ; save result to user defined register
:
EXIT_INT_ISR:
mov a,status_stack
mov STATUS,a       ; restore STATUS from user defined memory
mov a,acc_stack    ; restore ACC from user defined memory
reti

```

## 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.

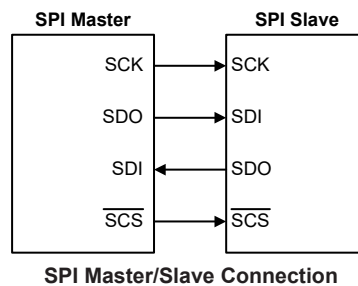
### SPI Interface

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  $\overline{\text{SCS}}$  pin. If the master needs to control multiple slave devices from a single master, the master can use I/O pin to select the slave devices.

### SPI Interface Operation

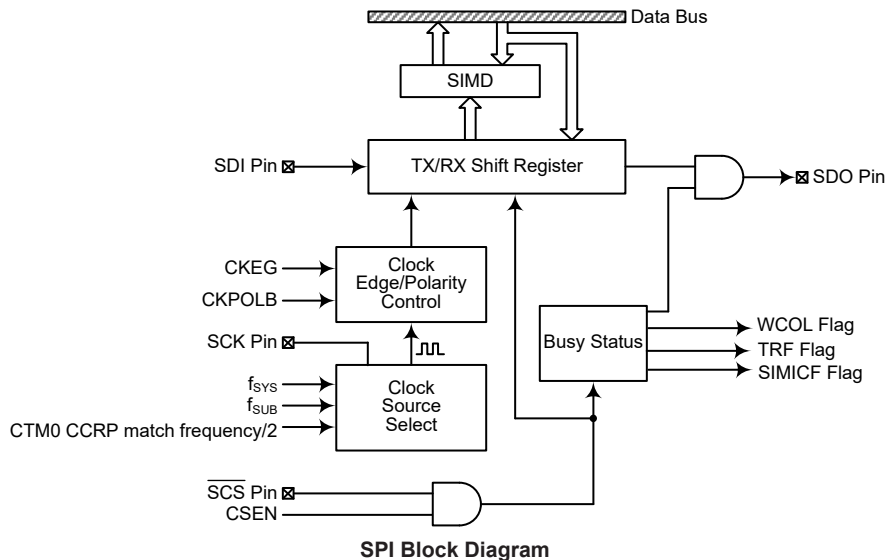
The SPI interface is a full duplex synchronous serial data link. It is a four-line interface with pin names SDI, SDO, SCK and  $\overline{\text{SCS}}$ . Pins SDI and SDO are the Serial Data Input and Serial Data Output lines, SCK is the Serial Clock line and  $\overline{\text{SCS}}$  is the Slave Select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



The SPI function 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.

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 CTM0 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 CTM0 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 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 slave mode Incomplete Transfer Flag  
           0: SIM SPI slave mode incomplete condition not occurred  
           1: SIM SPI slave mode incomplete condition occurred

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

• **SIMC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	D7	D6	CKPOLB	CKEG	MLS	CSEN	WCOL	TRF
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7~6     **D7~D6**: Undefined bits  
 These bits can be read or written by the application program.

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

- Bit 4     **CKEG**: SPI SCK clock active edge type selection  
 CKPOLB=0  
           0: SCK is high base level and data capture at SCK rising edge  
           1: SCK is high base level and data capture at SCK falling edge  
 CKPOLB=1  
           0: SCK is low base level and data capture at SCK falling edge  
           1: SCK is low base level and data capture at SCK rising edge

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

- Bit 3     **MLS**: SPI data shift order  
           0: LSB first  
           1: MSB first

This is the data shift select bit and is used to select how the data is transferred, either MSB or LSB first. Setting the bit high will select MSB first and low for LSB first.

- Bit 2     **CSEN**: SPI  $\overline{SCS}$  pin control  
           0: Disable  
           1: Enable

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

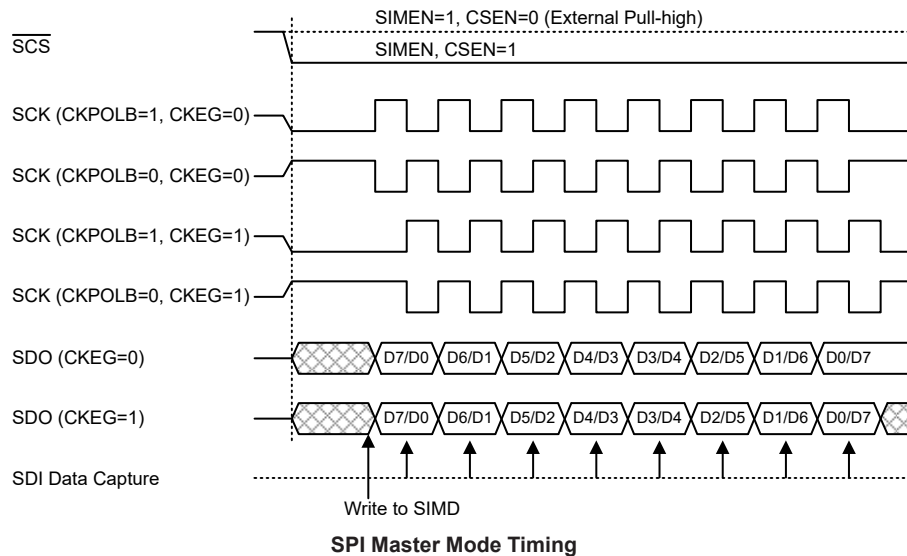


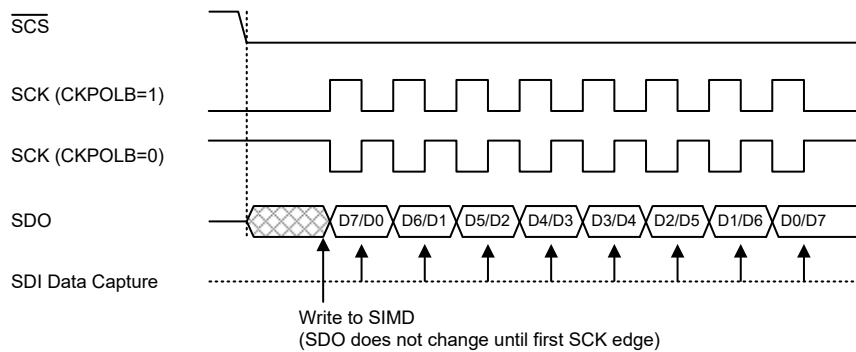
Bit 1	<b>WCOL:</b> SPI write collision flag 0: No collision 1: Collision The WCOL flag is used to detect whether a data collision has occurred or not. If this bit is high, it means that data has been attempted to be written to the SIMD register during a data transfer operation. This writing operation will be ignored if data is being transferred. This bit can be cleared by the application program.
Bit 0	<b>TRF:</b> 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

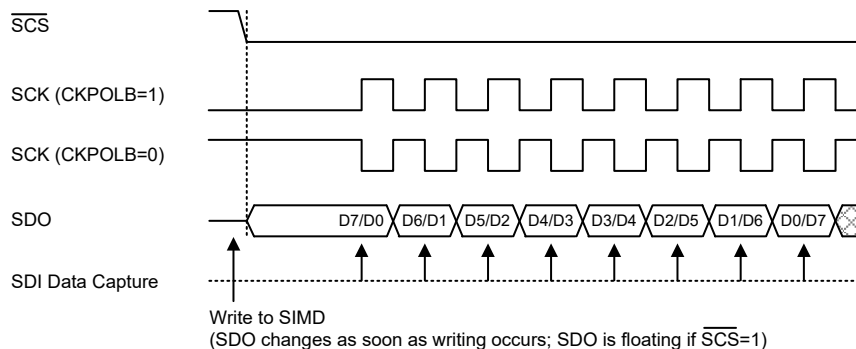
After the SPI interface is enabled by setting the SIMEN bit high, then in the Master Mode, when data is written to the SIMD register, transmission/reception will begin simultaneously. When the data transfer is complete, the TRF flag will be set automatically, but must be cleared using the application 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 will continue to function in certain IDLE Modes if the clock source used by the SPI interface is still active.



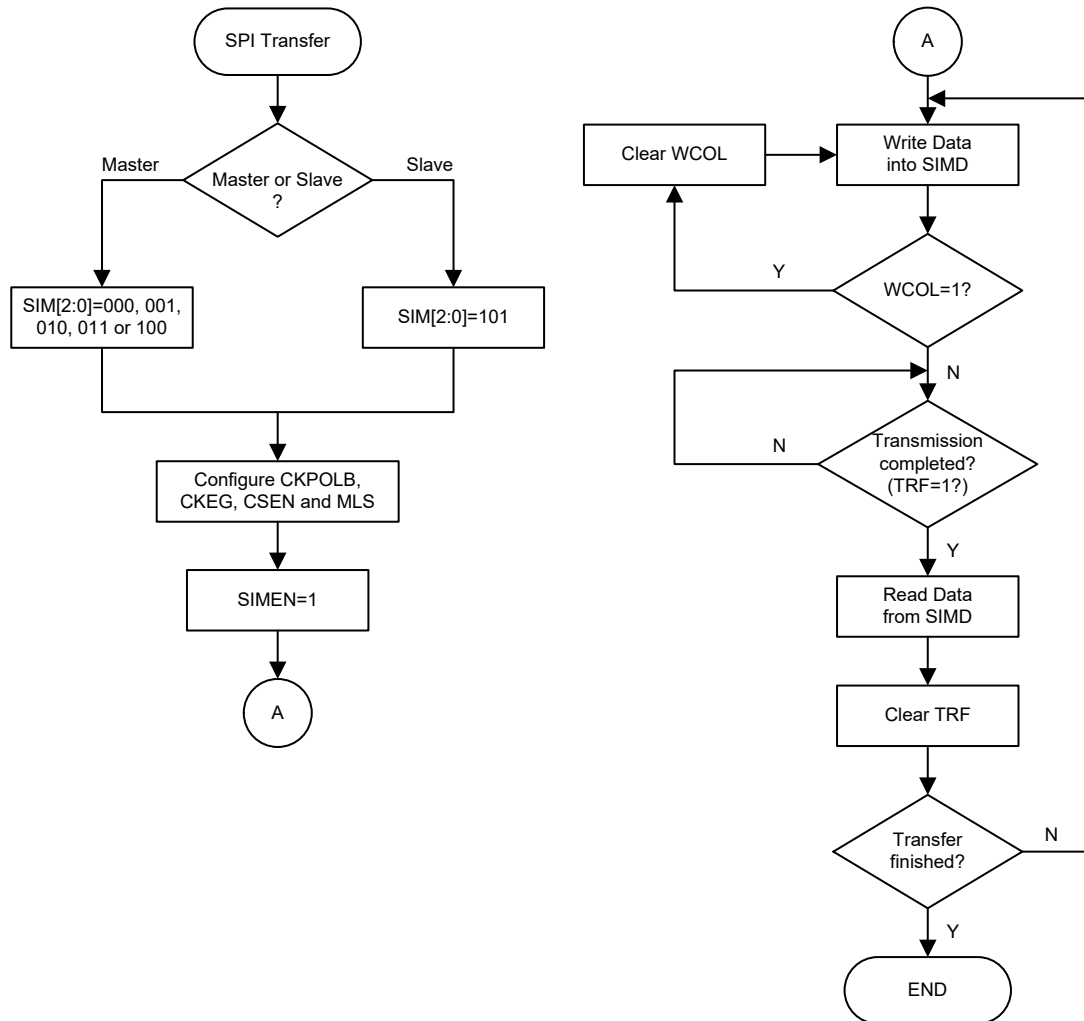


**SPI Slave Mode Timing – CKEG=0**



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

**SPI Slave Mode Timing – CKEG=1**



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

### SPI Operation Steps

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

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

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

**Master Mode**

- Step 1  
Select the SPI Master mode and clock source using the SIM2~SIM0 bits in the SIMC0 control register.
- Step 2  
Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Slave devices.
- Step 3  
Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.
- Step 4  
For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then use the SCK and 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 a SPI serial bus interrupt.
- Step 7  
Read data from the SIMD register.
- Step 8  
Clear TRF.
- Step 9  
Go to step 4.

**Slave Mode**

- Step 1  
Select the SPI Slave mode using the SIM2~SIM0 bits in the SIMC0 control register
- Step 2  
Setup the CSEN bit and setup the MLS bit to choose if the data is MSB or LSB first, this setting must be the same with the Master devices.
- Step 3  
Setup the SIMEN bit in the SIMC0 control register to enable the SPI interface.
- Step 4  
For write operations: write the data to the SIMD register, which will actually place the data into the TXRX buffer. Then wait for the master clock SCK and  $\overline{SCS}$  signal. After this, go to step 5.

For read operations: the data transferred in on the SDI line will be stored in the TXRX buffer until all the data has been received at which point it will be latched into the SIMD register.

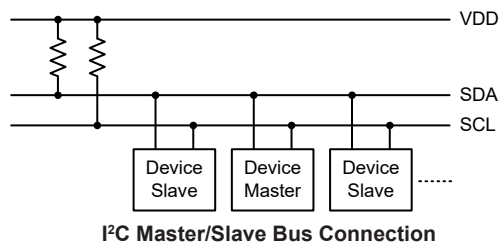
- Step 5  
Check the WCOL bit if set high then a collision error has occurred so return to step 4. If equal to zero then go to the following step.
- Step 6  
Check the TRF bit or wait for a SPI serial bus interrupt.
- Step 7  
Read data from the SIMD register.
- Step 8  
Clear TRF.
- Step 9  
Go to step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

### Error Detection

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

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

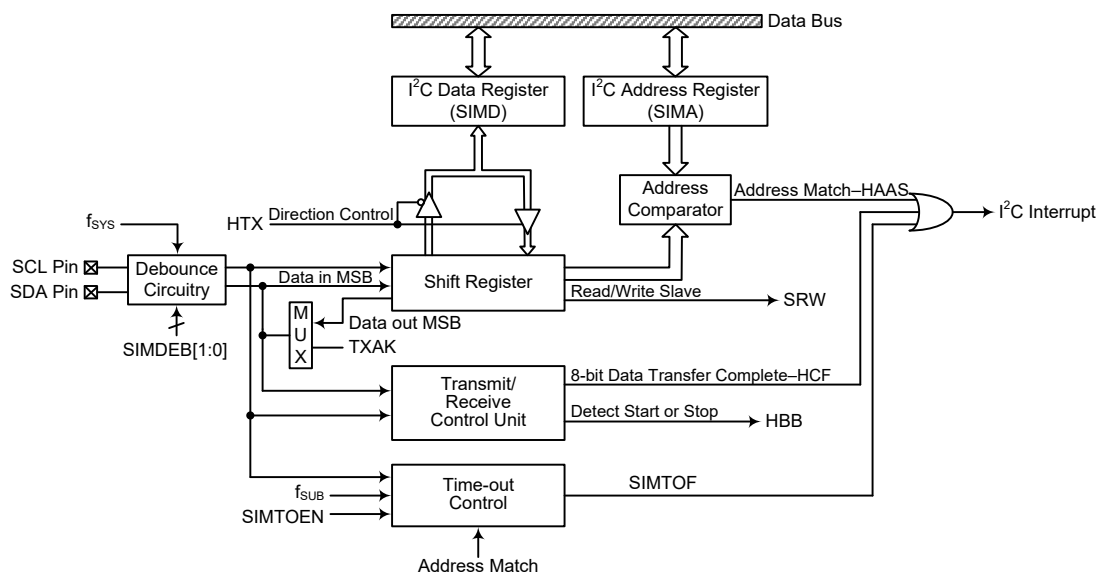
The I<sup>2</sup>C interface is used to communicate with external peripheral devices such as sensors, EEPROM memory etc. Originally developed by Philips, it is a two-line low speed serial interface for synchronous serial data transfer. The advantage of only two lines for communication, relatively simple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



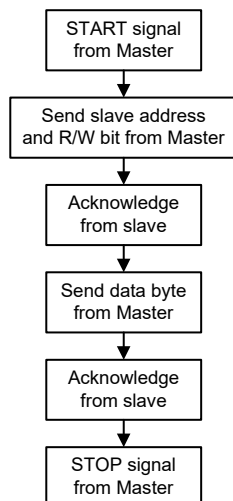
### I<sup>2</sup>C Interface Operation

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.



I²C Block Diagram



I²C Interface Operation

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

I²C Debounce Time Selection	I²C Standard Mode (100kHz)	I²C Fast Mode (400kHz)
No Debounce	$f_{sys} > 2\text{MHz}$	$f_{sys} > 4\text{MHz}$
2 system clock debounce	$f_{sys} > 4\text{MHz}$	$f_{sys} > 8\text{MHz}$
4 system clock debounce	$f_{sys} > 4\text{MHz}$	$f_{sys} > 8\text{MHz}$

I²C 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. Bit 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 set the data transmission clock frequency. The SIMC1 register contains the relevant flags which are used to indicate the I<sup>2</sup>C communication status. Another register, SIMTOC, is used to control the I<sup>2</sup>C time-out function and is described in the corresponding section.

### • SIMC0 Register

Bit	7	6	5	4	3	2	1	0
Name	SIM2	SIM1	SIM0	—	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 CTM0 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 CTM0 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

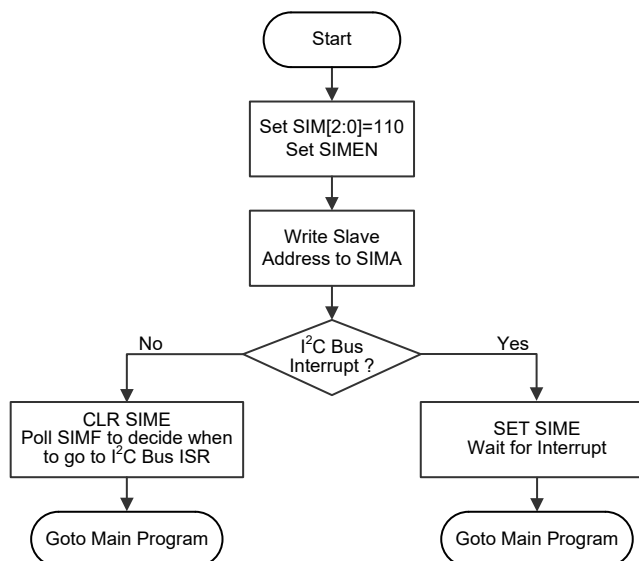
- Bit 7 HCF:** I<sup>2</sup>C Bus data transfer completion flag  
0: Data is being transferred  
1: Completion of an 8-bit data transfer  
The HCF flag is the data transfer flag. This flag will be zero when data is being transferred. Upon completion of an 8-bit data transfer the flag will go high and an interrupt will be generated.
- Bit 6 HAAS:** I<sup>2</sup>C Bus 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 set to “0” when the bus is free which will occur when a STOP signal is detected.
- Bit 4 HTX:** I<sup>2</sup>C slave device 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 send acknowledge flag  
1: Slave does not send acknowledge flag  
The TXAK bit is the transmit acknowledge flag. After the slave device receipt of 8-bits of data, this bit will be transmitted to the bus on the 9<sup>th</sup> clock from the slave device. The slave device must always set TXAK bit to “0” before further data is received.
- Bit 2 SRW:** I<sup>2</sup>C slave read/write flag  
0: Slave device should be in receive mode  
1: Slave device should be in transmit mode  
The SRW flag is the I<sup>2</sup>C Slave Read/Write flag. This flag determines whether the master device wishes to transmit or receive data from the I<sup>2</sup>C bus. When the transmitted address and slave address is match, that is when the HAAS flag is set high, the slave device will check the SRW flag to determine whether it should be in transmit mode or receive mode. If the SRW flag is high, the master is requesting to read data from the bus, so the slave device should be in transmit mode. When the SRW flag is zero, the master will write data to the bus, therefore the slave device should be in receive mode to read this data.
- Bit 1 IAMWU:** I<sup>2</sup>C Address Match Wake-Up control  
0: Disable  
1: Enable – 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 RXAK:** 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 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 I<sup>2</sup>C interrupt will be generated. After entering the interrupt service routine, the slave device must first check the condition of the HAAS and SIMTOF bits to determine whether the interrupt source originates from an address match, 8-bit data transfer completion or 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 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 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 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 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.

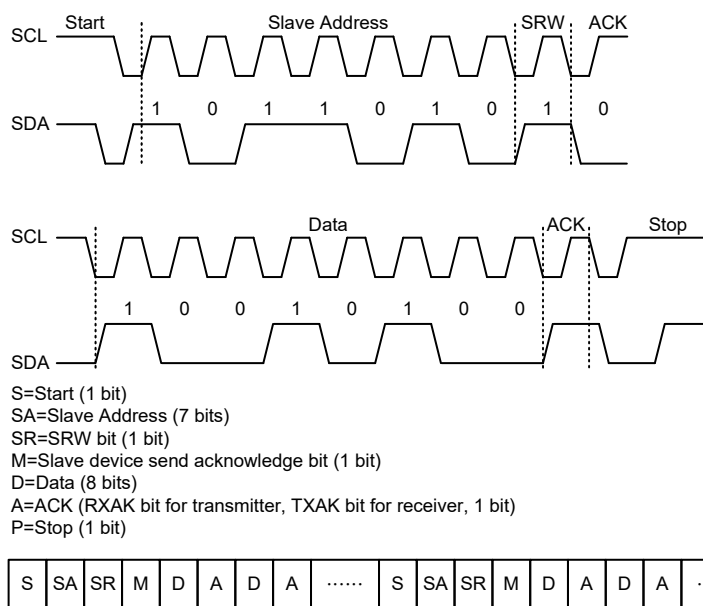
### **I<sup>2</sup>C Bus Slave Address Acknowledge Signal**

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 set to "0".

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

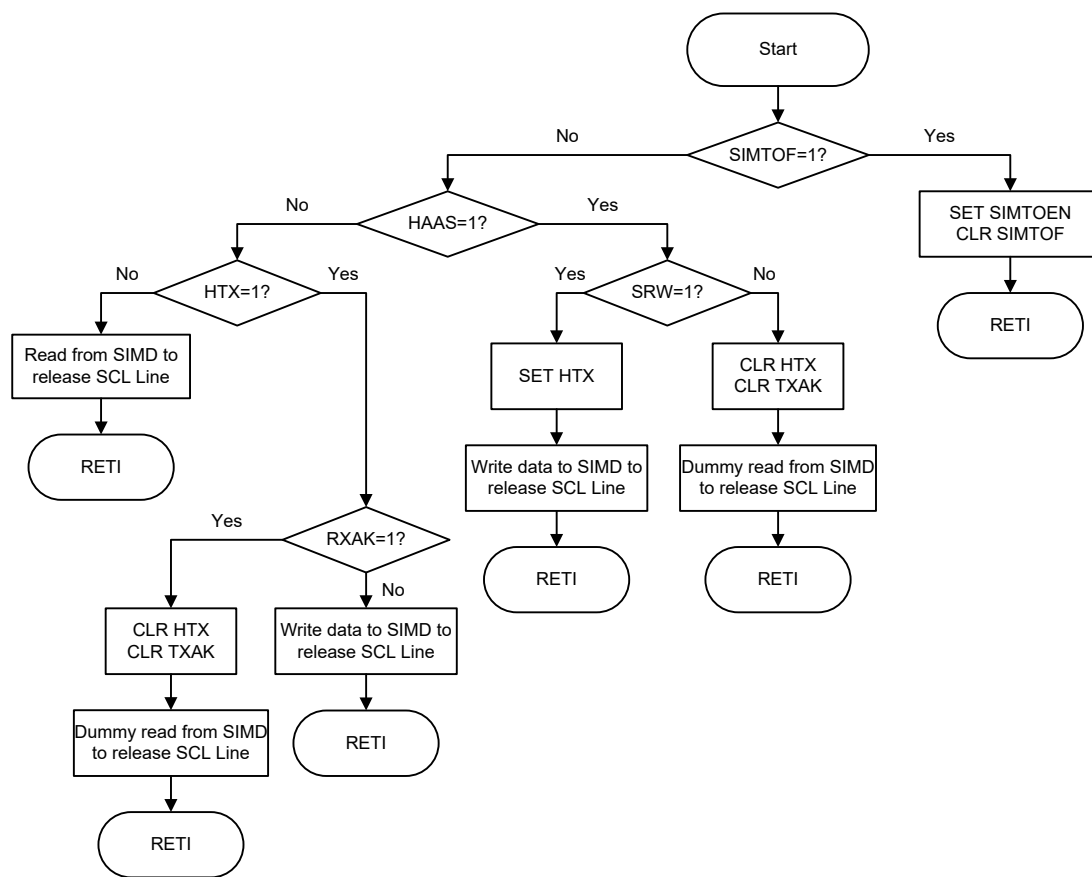
The transmitted data is 8-bits wide and is transmitted after the slave device has acknowledged receipt of its slave address. The order of serial bit transmission is the MSB first and the LSB last. After receipt of 8-bits of data, the receiver must transmit an acknowledge signal, level “0”, before it 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.

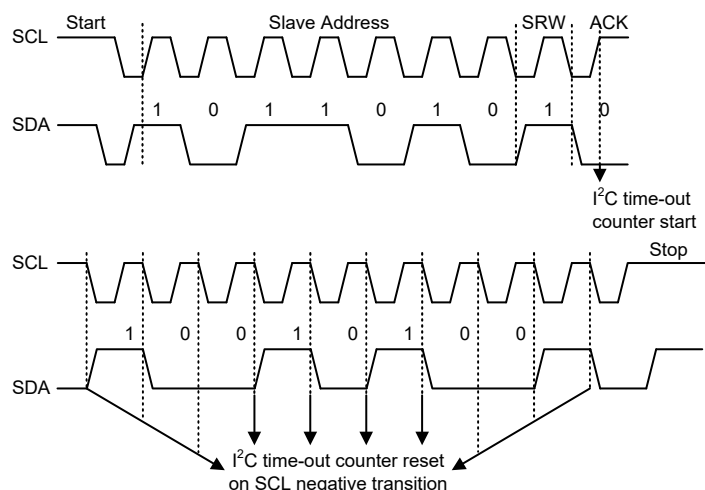
### I<sup>2</sup>C Communication Timing Diagram



I²C Bus ISR Flow Chart

### I²C Time-out Control

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



### I<sup>2</sup>C Time-out

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 I<sup>2</sup>C 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 SIMTOS bits in the SIMTOC register. The time-out duration is calculated by the formula:  $((1 \sim 64) \times (32/f_{SUB}))$ . This gives a time-out period which ranges from about 1ms to 64ms.

### • SIMTOC Register

Bit	7	6	5	4	3	2	1	0
Name	SIMTOEN	SIMTOF	SIMTOS5	SIMTOS4	SIMTOS3	SIMTOS2	SIMTOS1	SIMTOS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

Bit 7 **SIMTOEN**: 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

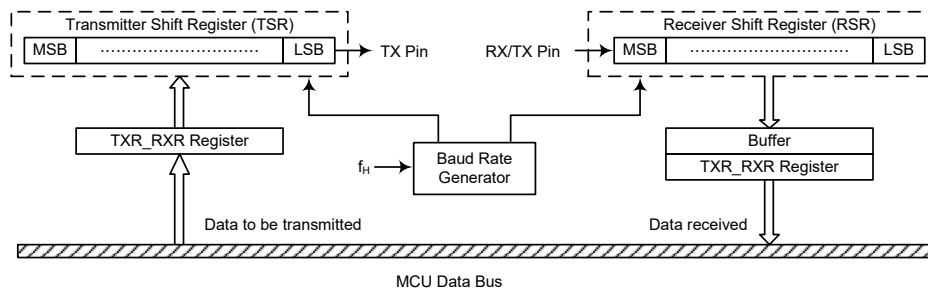
I<sup>2</sup>C Time-out clock source is  $f_{SUB}/32$   
I<sup>2</sup>C Time-out period is equal to  $(SIMTOS[5:0]+1) \times (32/f_{SUB})$

## UART Interface

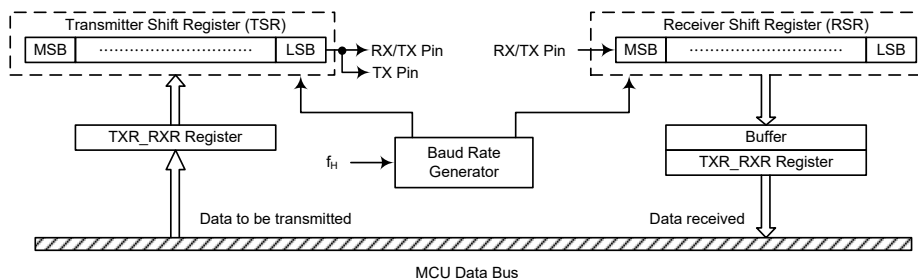
The device contains one integrated full-duplex or half-duplex asynchronous serial communications UART interface that enables communication with external devices that contain a serial interface. The UART function has many features and can transmit and receive data serially by transferring a frame of data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

The integrated UART function contains the following features:

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



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



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

## UART External Pins

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

## UART Single Wire Mode

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

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

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

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

## UART Data Transfer Scheme

The following block diagram shows the overall data transfer structure arrangement for the UART. The actual data to be transmitted from the MCU is first transferred to the TXR\_RXR register by the application program. The data will then be transferred to the Transmit Shift Register from where it will be shifted out, LSB first, onto the TX pin at a rate controlled by the Baud Rate Generator. Only the TXR\_RXR register is mapped onto the MCU Data Memory, the Transmit Shift Register is not mapped and is therefore inaccessible to the application program.

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

It should be noted that the actual register for data transmission and reception only exists as a single shared register in the Data Memory. This shared register known as the TXR\_RXR register is used for both data transmission and data reception.



## UART Status and Control Registers

There are six control registers associated with the UART function. The SWM bit in the UCR3 register is used to enable/disable the UART Single Wire Mode. The USR, UCR1 and UCR2 registers control the overall function of the UART, while the BRG register controls the Baud rate. The actual data to be transmitted and received on the serial interface is managed through the TXR\_RXR data register.

Register Name	Bit							
	7	6	5	4	3	2	1	0
USR	PERR	NF	FERR	OERR	RIDLE	RXIF	TIDLE	TXIF
UCR1	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
UCR2	TXEN	RXEN	BRGH	ADDEN	WAKE	RIE	TIIE	TEIE
UCR3	—	—	—	—	—	—	—	SWM
TXR_RXR	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
BRG	D7	D6	D5	D4	D3	D2	D1	D0

UART Register List

### • USR Register

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

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

Bit 7 **PERR:** Parity error flag  
 0: No parity error is detected  
 1: Parity error is detected

The PERR flag is the parity error flag. When this read only flag is “0”, it indicates a parity error has not been detected. When the flag is “1”, it indicates that the parity of the received word is incorrect. This error flag is applicable only if Parity mode (odd or even) is selected. The flag can also be cleared to zero by a software sequence which involves a read to the status register USR followed by an access to the TXR\_RXR data register.

Bit 6 **NF:** Noise flag  
 0: No noise is detected  
 1: Noise is detected

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

Bit 5 **FERR:** Framing error flag  
 0: No framing error is detected  
 1: Framing error is detected

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

Bit 4	<p><b>OERR:</b> Overrun error flag</p> <p>0: No overrun error is detected</p> <p>1: Overrun error is detected</p> <p>The OERR flag is the overrun error flag which indicates when the receiver buffer has overflowed. When this read only flag is “0”, it indicates that there is no overrun error. When the flag is “1”, it indicates that an overrun error occurs which will inhibit further transfers to the TXR_RXR receive data register. The flag is cleared to zero by a software sequence, which is a read to the status register USR followed by an access to the TXR_RXR data register.</p>
Bit 3	<p><b>RIDLE:</b> Receiver status</p> <p>0: Data reception is in progress (Data being received)</p> <p>1: No data reception is in progress (Receiver is idle)</p> <p>The RIDLE flag is the receiver status flag. When this read only flag is “0”, it indicates that the receiver is between the initial detection of the start bit and the completion of the stop bit. When the flag is “1”, it indicates that the receiver is idle. Between the completion of the stop bit and the detection of the next start bit, the RIDLE bit is “1” indicating that the UART receiver is idle and the RX/TX pin stays in logic high condition.</p>
Bit 2	<p><b>RXIF:</b> Receive TXR_RXR data register status</p> <p>0: TXR_RXR data register is empty</p> <p>1: TXR_RXR data register has available data</p> <p>The RXIF flag is the receive data register status flag. When this read only flag is “0”, it indicates that the TXR_RXR read data register is empty. When the flag is “1”, it indicates that the TXR_RXR read data register contains new data. When the contents of the shift register are transferred to the TXR_RXR register, an interrupt is generated if RIE=1 in the UCR2 register. If one or more errors are detected in the received word, the appropriate receive-related flags NF, FERR, and/or PERR are set within the same clock cycle. The RXIF flag will eventually be cleared to zero when the USR register is read with RXIF set, followed by a read from the TXR_RXR register, and if the TXR_RXR register has no more new data available.</p>
Bit 1	<p><b>TIDLE:</b> Transmission idle</p> <p>0: Data transmission is in progress (Data being transmitted)</p> <p>1: No data transmission is in progress (Transmitter is idle)</p> <p>The TIDLE flag is known as the transmission complete flag. When this read only flag is “0”, it indicates that a transmission is in progress. This flag will be set high when the TXIF flag is “1” and when there is no transmit data or break character being transmitted. When TIDLE is equal to “1”, the TX pin becomes idle with the pin state in logic high condition. The TIDLE flag is cleared to zero by reading the USR register with TIDLE set and then writing to the TXR_RXR register. The flag is not generated when a data character or a break is queued and ready to be sent.</p>
Bit 0	<p><b>TXIF:</b> Transmit TXR_RXR data register status</p> <p>0: Character is not transferred to the transmit shift register</p> <p>1: Character has transferred to the transmit shift register (TXR_RXR data register is empty)</p> <p>The TXIF flag is the transmit data register empty flag. When this read only flag is “0”, it indicates that the character is not transferred to the transmitter shift register. When the flag is “1”, it indicates that the transmitter shift register has received a character from the TXR_RXR data register. The TXIF flag is cleared to zero by reading the UART status register (USR) with TXIF set and then writing to the TXR_RXR data register. Note that when the TXEN bit is set, the TXIF flag bit will also be set since the transmit data register is not yet full.</p>

• **UCR 1 Register**

The UCR1 register together with the UCR2 and UCR3 registers are the three UART control registers that are used to set the various options for the UART function, such as overall on/off control, parity control, data transfer bit length, single wire mode communication etc. Further explanation on each of the bits is given below:

Bit	7	6	5	4	3	2	1	0
Name	UARTEN	BNO	PREN	PRT	STOPS	TXBRK	RX8	TX8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	W
POR	0	0	0	0	0	0	x	0

“x”: unknown

Bit 7 **UARTEN**: UART function enable control

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

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

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

Bit 6 **BNO**: Number of data transfer bits selection

- 0: 8-bit data transfer
- 1: 9-bit data transfer

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

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

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

- 0: Parity function is disabled
- 1: Parity function is enabled

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

Bit 4 **PRT**: Parity type selection bit

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

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

Bit 3 **STOPS**: Number of Stop bits selection

- 0: One stop bit format is used
- 1: Two stop bits format is used

This bit determines if one or two stop bits are to be used. When this bit is equal to “1”, two stop bits are used. If this bit is equal to “0”, then only one stop bit is used.

- Bit 2      **TXBRK**: Transmit break character  
             0: No break character is transmitted  
             1: Break characters transmit  
             The TXBRK bit is the Transmit Break Character bit. When this bit is “0”, there are no break characters and the TX pin operates normally. When the bit is “1”, there are transmit break characters and the transmitter will send logic zeros. When this bit is equal to “1”, after the buffered data has been transmitted, the transmitter output is held low for a minimum of a 13-bit length and until the TXBRK bit is reset.
- Bit 1      **RX8**: Receive data bit 8 for 9-bit data transfer format (read only)  
             This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the received data known as RX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.
- Bit 0      **TX8**: Transmit data bit 8 for 9-bit data transfer format (write only)  
             This bit is only used if 9-bit data transfers are used, in which case this bit location will store the 9th bit of the transmitted data known as TX8. The BNO bit is used to determine whether data transfers are in 8-bit or 9-bit format.

• **UCR2 Register**

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

Bit	7	6	5	4	3	2	1	0
Name	TXEN	RXEN	BRGH	ADDEN	WAKE	RIE	TIIE	TEIE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7      **TXEN**: UART Transmitter enabled control  
             0: UART transmitter is disabled  
             1: UART transmitter is enabled  
             The bit named TXEN is the Transmitter Enable Bit. When this bit is equal to “0”, the transmitter will be disabled with any pending data transmissions being aborted. In addition the buffers will be reset. In this situation the TX pin will be in a floating state. If the TXEN bit is equal to “1” and the UARTEN bit is also equal to “1”, the transmitter will be enabled and the TX pin will be controlled by the UART. Clearing the TXEN bit during a transmission will cause the data transmission to be aborted and will reset the transmitter. If this situation occurs, the TX pin will be in a floating state.
- Bit 6      **RXEN**: UART Receiver enabled control  
             0: UART receiver is disabled  
             1: UART receiver is enabled  
             The bit named RXEN is the Receiver Enable Bit. When this bit is equal to “0”, the receiver will be disabled with any pending data receptions being aborted. In addition the receive buffers will be reset. In this situation the RX/TX pin will be in a floating state. If the RXEN bit is equal to “1” and the UARTEN bit is also equal to “1”, the receiver will be enabled and the RX/TX pin will be controlled by the UART. Clearing the RXEN bit during a reception will cause the data reception to be aborted and will reset the receiver. If this situation occurs, the RX/TX pin will be in a floating state.
- Bit 5      **BRGH**: Baud Rate speed selection  
             0: Low speed baud rate  
             1: High speed baud rate  
             The bit named BRGH selects the high or low speed mode of the Baud Rate Generator. This bit, together with the value placed in the baud rate register BRG, controls the Baud Rate of the UART. If this bit is equal to “1”, the high speed mode is selected. If the bit is equal to “0”, the low speed mode is selected.

- Bit 4      **ADDEN**: Address detect function enable control  
             0: Address detect function is disabled  
             1: Address detect function is enabled  
 The bit named ADDEN is the address detect function enable control bit. When this bit is equal to “1”, the address detect function is enabled. When it occurs, if the 8th bit, which corresponds to TXR\_RXR.7 if BNO=0 or the 9th bit, which corresponds to RX8 if BNO=1, has a value of “1”, then the received word will be identified as an address, rather than data. If the corresponding interrupt is enabled, an interrupt request will be generated each time the received word has the address bit set, which is the 8th or 9th bit depending on the value of BNO. If the address bit known as the 8th or 9th bit of the received word is “0” with the address detect function being enabled, an interrupt will not be generated and the received data will be discarded.
- Bit 3      **WAKE**: RX/TX pin wake-up UART function enable control  
             0: RX/TX pin wake-up UART function is disabled  
             1: RX/TX pin wake-up UART function is enabled  
 This bit is used to control the wake-up UART function when a falling edge on the RX/TX pin occurs. Note that this bit is only available when the UART clock ( $f_H$ ) is switched off. There will be no RX/TX pin wake-up UART function if the UART clock ( $f_H$ ) exists. If the WAKE bit is set to 1 as the UART clock ( $f_H$ ) is switched off, a UART wake-up request will be initiated when a falling edge on the RX/TX pin occurs. When this request happens and the corresponding interrupt is enabled, an RX/TX pin wake-up UART interrupt will be generated to inform the MCU to wake up the UART function by switching on the UART clock ( $f_H$ ) via the application program. Otherwise, the UART function can not resume even if there is a falling edge on the RX/TX pin when the WAKE bit is cleared to 0.
- Bit 2      **RIE**: Receiver interrupt enable control  
             0: Receiver related interrupt is disabled  
             1: Receiver related interrupt is enabled  
 This bit enables or disables the receiver interrupt. If this bit is equal to “1” and when the receiver overrun flag OERR or receive data available flag RXIF is set, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the OERR or RXIF flags.
- Bit 1      **TIE**: Transmitter Idle interrupt enable control  
             0: Transmitter idle interrupt is disabled  
             1: Transmitter idle interrupt is enabled  
 This bit enables or disables the transmitter idle interrupt. If this bit is equal to “1” and when the transmitter idle flag TIDLE is set, due to a transmitter idle condition, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the TIDLE flag.
- Bit 0      **TEIE**: Transmitter Empty interrupt enable control  
             0: Transmitter empty interrupt is disabled  
             1: Transmitter empty interrupt is enabled  
 This bit enables or disables the transmitter empty interrupt. If this bit is equal to “1” and when the transmitter empty flag TXIF is set, due to a transmitter empty condition, the UART interrupt request flag will be set. If this bit is equal to “0”, the UART interrupt request flag will not be influenced by the condition of the TXIF flag.

### • UCR3 Register

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

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

Bit 7~1 Unimplemented, read as “0”

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

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

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

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

### • TXR\_RXR Register

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

Bit	7	6	5	4	3	2	1	0
Name	TXRX7	TXRX6	TXRX5	TXRX4	TXRX3	TXRX2	TXRX1	TXRX0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	x	x	x	x	x	x	x	x

“x”: unknown

Bit 7~0 **TXRX7~TXRX0**: UART Transmit/Receive Data bit 7~bit 0

### • BRG 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**: Baud Rate values

By programming the BRGH bit in UCR2 Register which allows selection of the related formula described above and programming the required value in the BRG register, the required baud rate can be setup.

Note: Baud rate= $f_H/[64 \times (N+1)]$  if BRGH=0;

Baud rate= $f_H/[16 \times (N+1)]$  if BRGH=1.

### Baud Rate Generator

To setup the speed of the serial data communication, the UART function contains its own dedicated baud rate generator. The baud rate is controlled by its own internal free running 8-bit timer, the period of which is determined by two factors. The first of these is the value placed in the baud rate register BRG and the second is the value of the BRGH bit with the control register UCR2. The BRGH bit decides if the baud rate generator is to be used in a high speed mode or low speed mode, which in turn determines the formula that is used to calculate the baud rate. The value N in the BRG register which is used in the following baud rate calculation formula determines the division factor. Note that N is the decimal value placed in the BRG register and has a range of between 0 and 255.

UCR2 BRGH Bit	0	1
Baud Rate (BR)	$f_H/[64 \times (N+1)]$	$f_H/[16 \times (N+1)]$

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

#### Calculating the Baud Rate and Error Values

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

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

Re-arranging this equation gives  $N = [f_H/(BR \times 64)] - 1$

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

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

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

#### UART Setup and Control

For data transfer, the UART function utilizes a non-return-to-zero, more commonly known as NRZ, format. This is composed of one start bit, eight or nine data bits, and one or two stop bits. Parity is supported by the UART hardware, and can be setup to be even, odd or no parity. For the most common data format, 8 data bits along with no parity and one stop bit, denoted as 8, N, 1, is used as the default setting, which is the setting at power-on. The number of data bits and stop bits, along with the parity, are setup by programming the corresponding BNO, PRT, PREN, and STOPS bits in the UCR1 register. The baud rate used to transmit and receive data is setup using the internal 8-bit baud rate generator, while the data is transmitted and received LSB first. Although the UART transmitter and receiver are functionally independent, they both use the same data format and baud rate. In all cases stop bits will be used for data transmission.

#### Enabling/Disabling the UART Interface

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

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



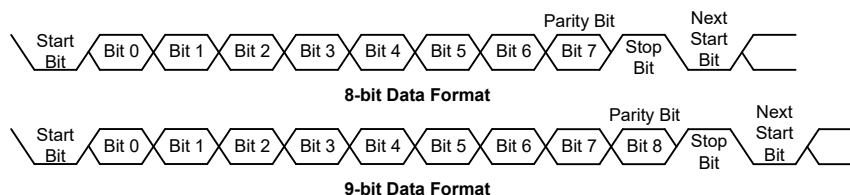
### Data, Parity and Stop Bit Selection

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

Start Bit	Data Bits	Address Bit	Parity Bit	Stop Bit
<b>Example of 8-bit Data Formats</b>				
1	8	0	0	1
1	7	0	1	1
1	7	1	0	1
<b>Example of 9-bit Data Formats</b>				
1	9	0	0	1
1	8	0	1	1
1	8	1	0	1

**Transmitter Receiver Data Format**

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



### UART Transmitter

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



### Transmitting Data

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

- Make the correct selection of the BNO, PRT, PREN and STOPS bits to define the required word length, parity type and number of stop bits.
- Setup the BRG register to select the desired baud rate.
- Set the TXEN bit to ensure that the TX pin is used as a UART transmitter pin.
- Access the USR register and write the data that is to be transmitted into the TXR\_RXR register.  
Note that this step will clear the TXIF bit.

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

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

1. A USR register access
2. A TXR\_RXR register write execution

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

During a data transmission, a write instruction to the TXR\_RXR register will place the data into the TXR\_RXR register, which will be copied to the shift register at the end of the present transmission. When there is no data transmission in progress, a write instruction to the TXR\_RXR register will place the data directly into the shift register, resulting in the commencement of data transmission, and the TXIF bit being immediately set. When a frame transmission is complete, which happens after stop bits are sent or after the break frame, the TIDLE bit will be set. To clear the TIDLE bit the following software sequence is used:

1. A USR register access
2. A TXR\_RXR register write execution

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

### Transmit Break

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

## UART Receiver

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

### Receiving Data

When the UART receiver is receiving data, the data is serially shifted in on the external RX/TX pin input, LSB first. In the read mode, the TXR\_RXR register forms a buffer between the internal bus and the receiver shift register. The TXR\_RXR register is a two-byte deep FIFO data buffer, where two bytes can be held in the FIFO while a third byte can continue to be received. Note that the application program must ensure that the data is read from TXR\_RXR before the third byte has been completely shifted in, otherwise this third byte will be discarded and an overrun error OERR will be subsequently indicated. The steps to initiate a data transfer can be summarized as follows:

- Make the correct selection of BNO, PRT and PREN bits to define the word length, parity type.
- Setup the BRG register to select the desired baud rate.
- Set the RXEN bit to ensure that the RX/TX pin is used as a UART receiver pin.

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

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

- The RXIF bit in the USR register will be set when the TXR\_RXR register has data available. There will be at most one more character available before an overrun error occurs.
- When the contents of the shift register have been transferred to the TXR\_RXR register, then if the RIE bit is set, an interrupt will be generated.
- If during reception, a frame error, noise error, parity error, or an overrun error has been detected, then the error flags can be set.

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

1. A USR register access
2. A TXR\_RXR register read execution

### Receiving Break

Any break character received by the UART will be managed as a framing error. The receiver will count and expect a certain number of bit times as specified by the values programmed into the BNO bit plus one stop bit. If the break is much longer than 13 bit times, the reception will be considered as complete after the number of bit times specified by BNO plus one stop bit. The RXIF bit is set, FERR is set, zeros are loaded into the receive data register, interrupts are generated if appropriate and the RIDLE bit is set. A break is regarded as a character that contains only zeros with the FERR flag set. If a long break signal has been detected, the receiver will regard it as a data frame including a start bit, data bits and the invalid stop bit and the FERR flag will be set. The receiver must wait for a valid stop bit before looking for the next start bit. The receiver will not make the assumption that

the break condition on the line is the next start bit. The break character will be loaded into the buffer and no further data will be received until stop bits are received. It should be noted that the RIDLE read only flag will go high when the stop bits have not yet been received. The reception of a break character on the UART registers will result in the following:

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

#### **Idle Status**

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

#### **Receiver Interrupt**

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

### **Managing Receiver Errors**

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

#### **Overrun Error – OERR**

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

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

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

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

#### **Noise Error – NF**

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

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

Note that the NF flag is reset by a USR register read operation followed by a TXR\_RXR register read operation.

**Framing Error – FERR**

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

**Parity Error – PERR**

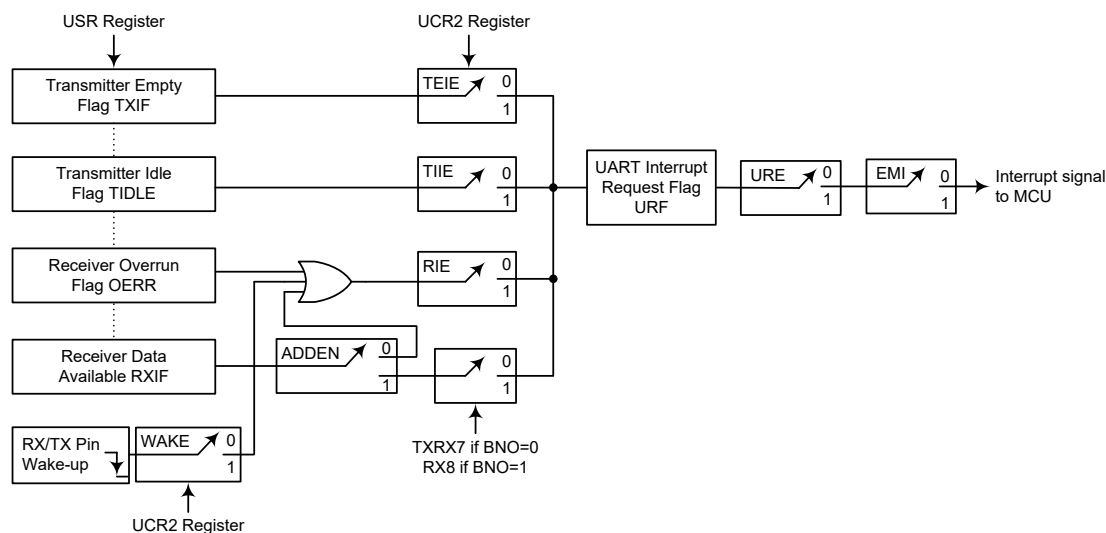
The read only parity error flag, PERR, in the USR register, is set if the parity of the received word is incorrect. This error flag is only applicable if the parity is enabled, PREN=1, and if the parity type, odd or even is selected. The read only PERR flag and the received data will be recorded in the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

**UART Interrupt Structure**

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

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

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



**UART Interrupt Structure**

### Address Detect Mode

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

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

**ADDEN Bit Function**

### UART Power Down and Wake-up

When the UART clock,  $f_H$ , is switched off, the UART will cease to function. If the MCU switches off the UART clock,  $f_H$ , and enters the power down mode while a transmission is still in progress, then the transmission will be paused until the UART clock source derived from the microcontroller is activated. In a similar way, if the MCU switches off the UART clock  $f_H$  and enters the IDLE or SLEEP mode by executing the “HALT” instruction while receiving data, then the reception of data will likewise be paused. When the MCU enters the IDLE or SLEEP mode, note that the USR, UCR1, UCR2, transmit and receive registers, as well as the BRG register will not be affected. It is recommended to make sure first that the UART data transmission or reception has been finished before the microcontroller enters the IDLE or SLEEP mode.

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

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

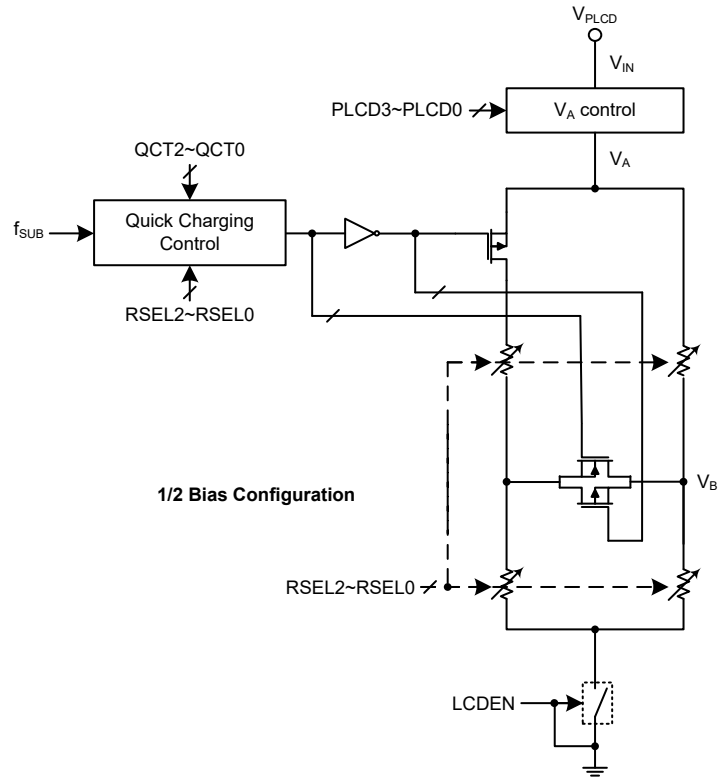
## LCD Driver

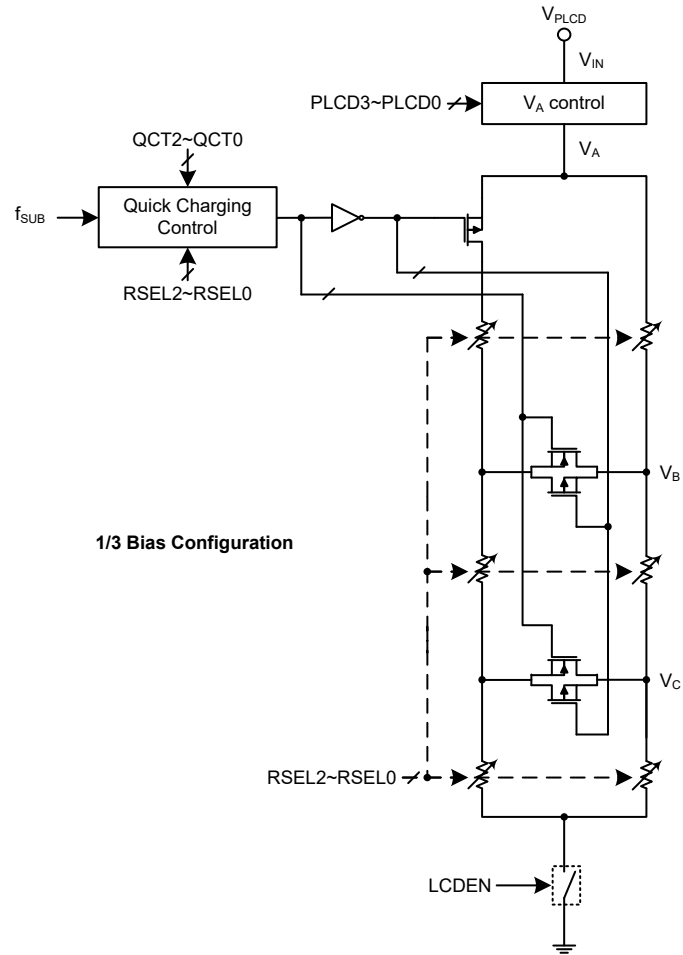
For large volume applications, which incorporate an LCD in their design, the use of a custom display rather than a more expensive character based display reduces costs significantly. However, the corresponding COM and SEG signals required, which vary in both amplitude and time, to drive such a custom 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.

The 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 for the device range.

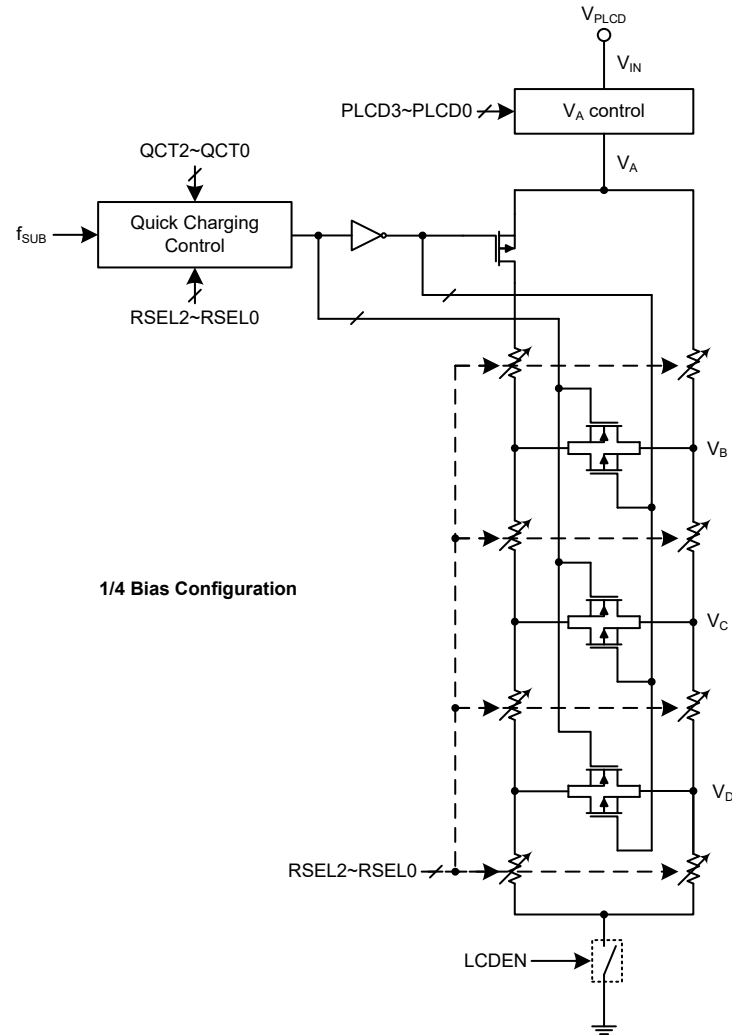
Driver No.		Duty	Bias Level	Bias Type	Waveform Type
48LQFP	44LQFP				
32×4	28×4	1/4	1/2 or 1/3	R	A or B
31×5	27×5	1/5	1/2	R	A or B
30×6	26×6	1/6	1/3	R	A or B
28×8	24×8	1/8	1/3 or 1/4	R	A or B

**LCD Driver Output Selection**



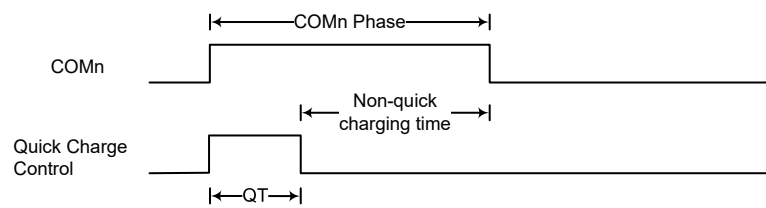






Note: When the R type LCD is disabled, the DC path will be switched off.

**R Type Bias Configurations – 1/2 bias, 1/3 bias & 1/4 bias**



QT: Quick charging time, determined by QCT[2:0].

**Quick Charging Mode**



## LCD Register

There are three control registers, named as LCDC0, LCDC1 and LCDC2, in the Data Memory which is used to control the various setup features of the LCD Driver.

Various bits in these registers control functions such as LCD waveform type, bias type, supply power selection, total bias resistor selection together with the overall LCD enable and disable control. The LCDEN bit in the LCDC0 register, which provides the overall LCD enable/disable function, will only be effective when the device is in the FAST, SLOW or IDLE Mode. If the device is in the SLEEP Mode then the display will always be disabled. Bits, RSEL2~RSEL0, in the LCDC0 register select the internal total bias resistors to supply the LCD panel with the proper bias current. A choice to best match the LCD panel used in the application can be selected also to minimise bias current. The TYPE bit in the LCDC0 register is used to select whether Type A or Type B LCD waveform signals are used.

The PLCD3~PLCD0 bits in the LCDC1 register are used to select the  $V_A$  voltage for R type bias circuitry. The QCT2~QCT0 bits in the same register are used to determine the quick charge time period.

Register Name	Bit							
	7	6	5	4	3	2	1	0
LCDC0	TYPE	—	—	—	RSEL2	RSEL1	RSEL0	LCDEN
LCDC1	QCT2	QCT1	QCT0	—	PLCD3	PLCD2	PLCD1	PLCD0
LCDC2	—	—	—	—	DTYC1	DTYC0	BIAS1	BIAS0

**LCD Register List**

### • LCDC0 Register

Bit	7	6	5	4	3	2	1	0
Name	TYPE	—	—	—	RSEL2	RSEL1	RSEL0	LCDEN
R/W	R/W	—	—	—	R/W	R/W	R/W	R/W
POR	0	—	—	—	0	0	0	0

Bit 7 **TYPE**: LCD waveform type selection

0: Type A

1: Type B

Bit 6~4 Unimplemented, read as “0”

Bit 3~1 **RSEL2~RSEL0**: R type total bias resistors selection

000: 1170k $\Omega$

001: 225k $\Omega$

010: 60k $\Omega$

011: Quick charging mode – switching between 60 k $\Omega$  and 1170 k $\Omega$

1xx: Quick charging mode – switching between 60 k $\Omega$  and 225 k $\Omega$

The device provides the low power quick charging mode for LCD display. In quick charging mode the LCD will provide LCD bias current by  $R_T=60k\Omega$  at the beginning of LCD display refreshes (i.e the moment on LCD COM changes). After quick charging time, the bias resistor will change to 225 k $\Omega$ /1170 k $\Omega$ .

Note that the bias resistor for 1/3 bias is  $R_T/3$ .

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. However, in the SLEEP mode, the LCD function is always switched off.

• **LCDC1 Register**

Bit	7	6	5	4	3	2	1	0
Name	QCT2	QCT1	QCT0	—	PLCD3	PLCD2	PLCD1	PLCD0
R/W	R/W	R/W	R/W	—	R/W	R/W	R/W	R/W
POR	0	0	0	—	0	0	0	0

Bit 7~5      **QCT2~QCT0**: R type quick charging time selection

000: 1  $t_{SUB}$

001: 2  $t_{SUB}$

010: 3  $t_{SUB}$

011: 4  $t_{SUB}$

100: 5  $t_{SUB}$

101: 6  $t_{SUB}$

110: 7  $t_{SUB}$

111: 8  $t_{SUB}$

The  $t_{SUB}$  is the period of the LCD clock source  $f_{SUB}$ , i.e.,  $1/f_{SUB}$ .

Bit 4      Unimplemented, read as “0”

Bit 3~0      **PLCD3~PLCD0**: R type bias supply voltage selection for  $V_A$  mode

0000:  $8/16 \times V_{PLCD}$

0001:  $9/16 \times V_{PLCD}$

0010:  $10/16 \times V_{PLCD}$

0011:  $11/16 \times V_{PLCD}$

0100:  $12/16 \times V_{PLCD}$

0101:  $13/16 \times V_{PLCD}$

0110:  $14/16 \times V_{PLCD}$

0111:  $15/16 \times V_{PLCD}$

1xxx:  $V_{PLCD}$

Note that the  $V_A$  voltage level has to be equal to or greater than 2.1V.

• **LCDC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	DTYC1	DTYC0	BIAS1	BIAS0
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      **DTYC1~DTYC0**: LCD duty selection

00: 1/4 Duty (COM0~COM3)

01: 1/6 Duty (COM0~COM5)

10: 1/8 Duty (COM0~COM7)

11: 1/5 Duty (COM0~COM4)

The unused COM pin can be configured as other pin-shared functions using the corresponding pin-shared selection register.

Bit 1~0      **BIAS1~BIAS0**: LCD duty selection

00: 1/3 bias

01: 1/4 bias

10: 1/2 bias

11: Unimplemented

## 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.

### R Type Biasing

For R type biasing an external LCD voltage source must be supplied on pin PLCD to generate the internal biasing voltages. This could be the microcontroller power supply VDD or some other voltage source equal to or less than VDD. For the R type 1/2 bias scheme, three voltage levels  $V_{SS}$ ,  $V_A$ , and  $V_B$  are utilised. For the R type 1/3 bias scheme, four voltage levels  $V_{SS}$ ,  $V_A$ ,  $V_B$  and  $V_C$  are utilised. For the R type 1/4 bias scheme, four voltage levels  $V_{SS}$ ,  $V_A$ ,  $V_B$ ,  $V_C$  and  $V_D$  are utilised. The voltage  $V_A$  is selected by the PLCD3~PLCD0 bits to be equal to a specific ratio of  $V_{PLCD}$  varying from  $8/16 V_{PLCD}$  to  $V_{PLCD}$ .

Different values of internal bias resistors can be selected using the RSEL2~RSEL0 bits in the LCD0 register. This along with the voltage on pin PLCD will determine the bias current. Note that no external capacitors or resistors are required to be connected if R type biasing is used.

Bias Selection	Bias Voltage
1/2 Bias	$V_A=8/16V_{PLCD} \sim V_{PLCD}$ ; $V_B=1/2V_A$
1/3 Bias	$V_A=8/16V_{PLCD} \sim V_{PLCD}$ ; $V_B=2/3V_A$ ; $V_C=1/3V_A$
1/4 Bias	$V_A=8/16V_{PLCD} \sim V_{PLCD}$ ; $V_B=3/4V_A$ ; $V_C=2/4V_A$ ; $V_D=1/4V_A$

R Type Bias Voltage

## LCD Reset Function

The LCD has an internal reset function that is an OR function of the inverted LCDEN bit in the LCD0 register and the SLEEP function. When the LCDEN bit is set to 1 to enable the LCD driver function before the device enters the SLEEP mode, the LCD function will be reset after the device enters the SLEEP mode. Clearing the LCDEN bit to zero will also reset the LCD function.

MCU Reset	SLEEP Mode	LCDEN	LCD Reset	COM & SEG Voltage Level
No	Off	1	No	Normal Operation
No	Off	0	Yes	Low
No	On	x	Yes	Low
Yes	x	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 Function

## LCD Driver Output

The number of COM and SEG outputs supplied by the LCD driver, as well as its biasing and waveform type selections, are dependent upon how the LCD control bits are programmed. The Bias Type, whether C or R type is also selected by a software control 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. The duty, which is to have a value of 1/4 and which 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 Type, 4 COM, 1/2 Bias

#### LCD Display Off Mode

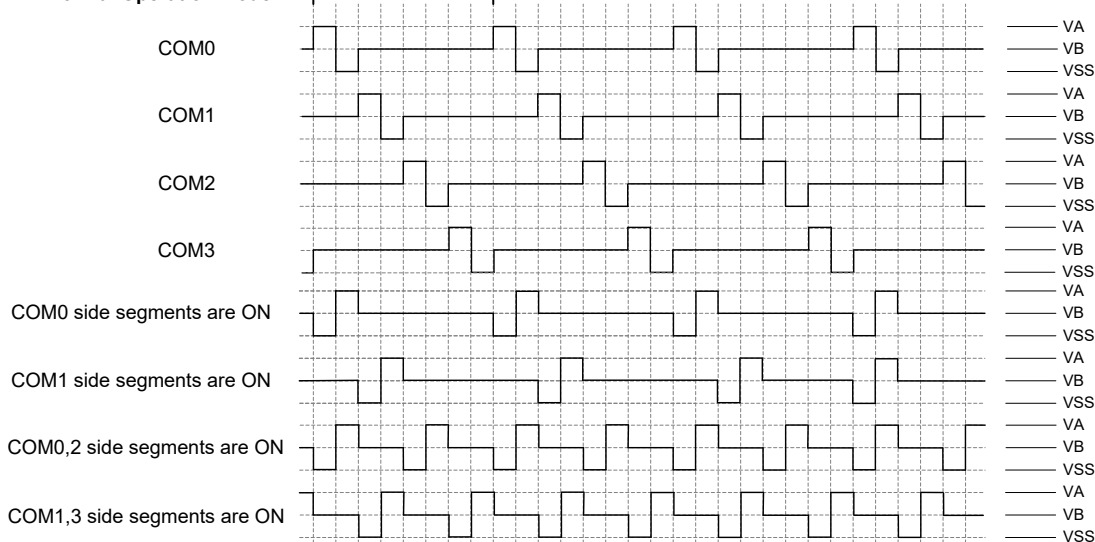
COM0 ~ COM3

All segment outputs



#### Normal Operation Mode

1 Frame



LCD Driver Output – Type A, 1/4 Duty, 1/2 Bias

COM1,3 side segments are ON

### LCD Driver Output – Type B, 1/4 Duty, 1/2 Bias

— VA  
— VB  
— VSS  
— VA  
— VB  
— VSS

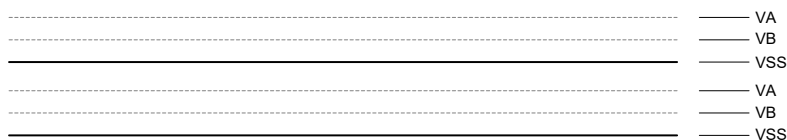
— VA  
— VB  
— VSS  
— VA  
— VB  
— VSS  
— VA  
— VB  
— VSS  
— VA  
— VB  
— VSS  
— VA  
— VB  
— VSS  
— VA  
— VB  
— VSS

**R Type, 5 COM, 1/2 Bias**

**LCD Display Off Mode**

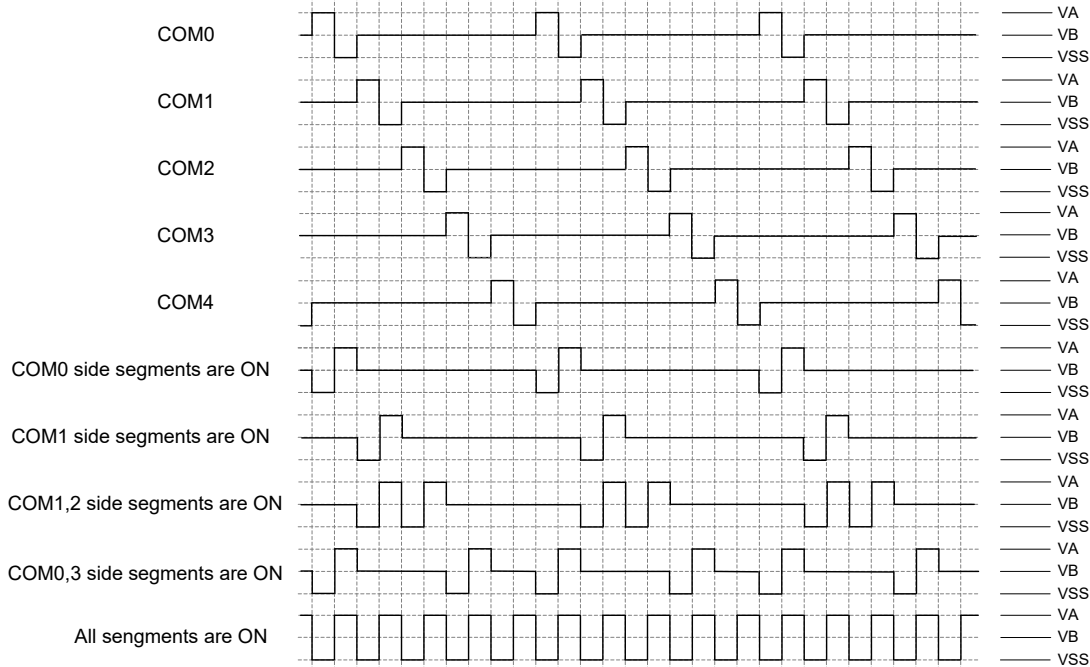
COM0 ~ COM4

All sengment outputs



**Normal Operation Mode**

1 Frame



**LCD Driver Output – Type A, 1/5 Duty, 1/2 Bias**



All segment outputs

\_\_\_\_\_ VA  
 \_\_\_\_\_ VB  
 \_\_\_\_\_ VSS  
 \_\_\_\_\_ VA  
 \_\_\_\_\_ VB  
 \_\_\_\_\_ VSS

A horizontal line with vertical end caps. Above the right end cap is the text "1 Frame".

All segments are ON

[illegible]

### LCD Driver Output – Type B, 1/5 Duty, 1/2 Bias

**R 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

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

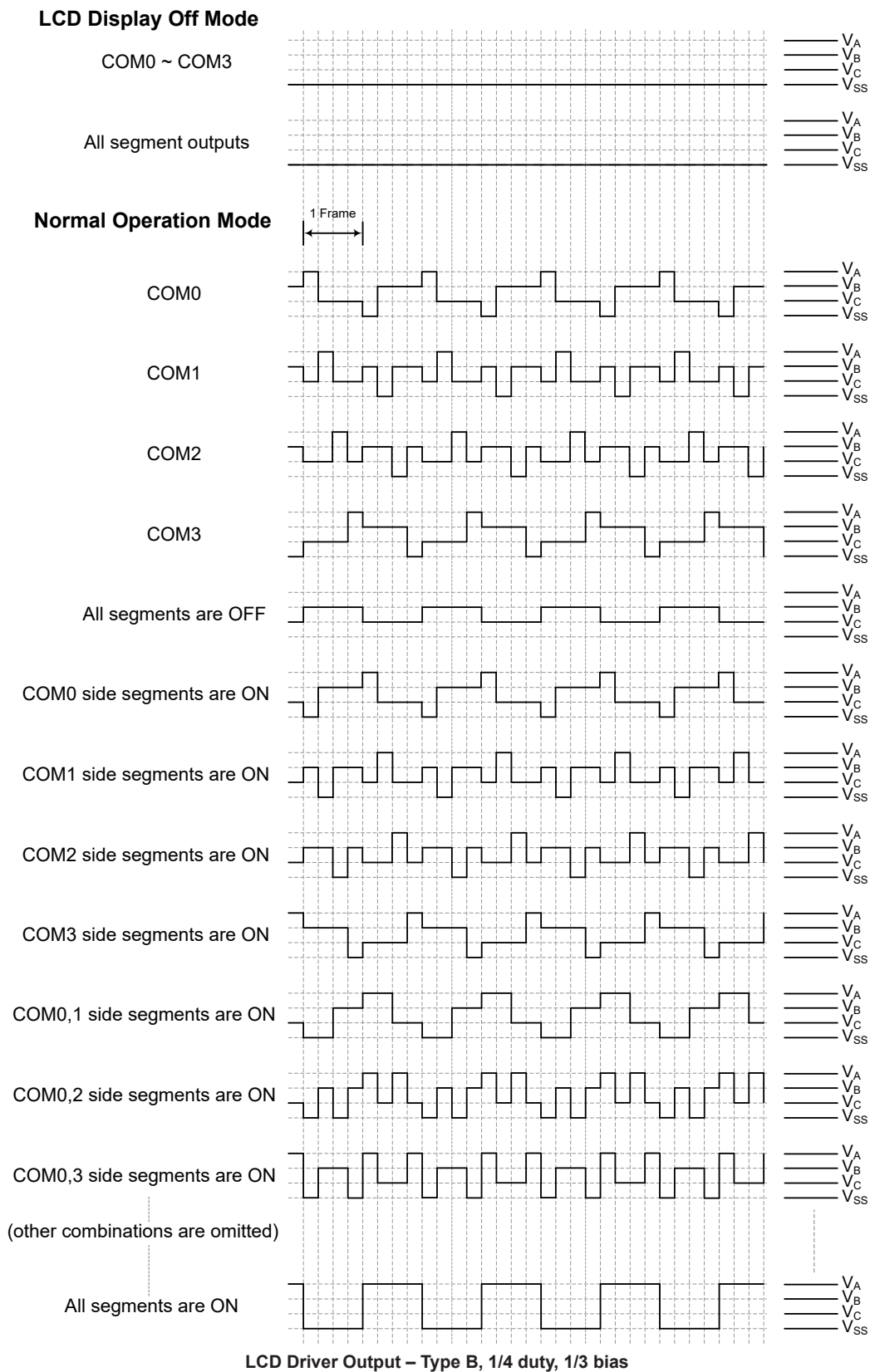
V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

V<sub>A</sub>  
V<sub>B</sub>  
V<sub>C</sub>  
V<sub>SS</sub>

**LCD Driver Output – Type A, 1/4 duty, 1/3 bias**



**R Type, 6 COM, 1/3 Bias**

**LCD Display Off Mode**

COM0 ~ COM5

All segment outputs

**Normal Operation Mode**

1 Frame

COM0

COM1

COM2

COM3

COM4

COM5

All segments are OFF

COM0 side segments are ON

COM1 side segments are ON

COM2 side segments are ON

COM3 side segments are ON

COM4 side segments are ON

COM5 side segments are ON

COM0,1 side segments are ON

COM0,2 side segments are ON

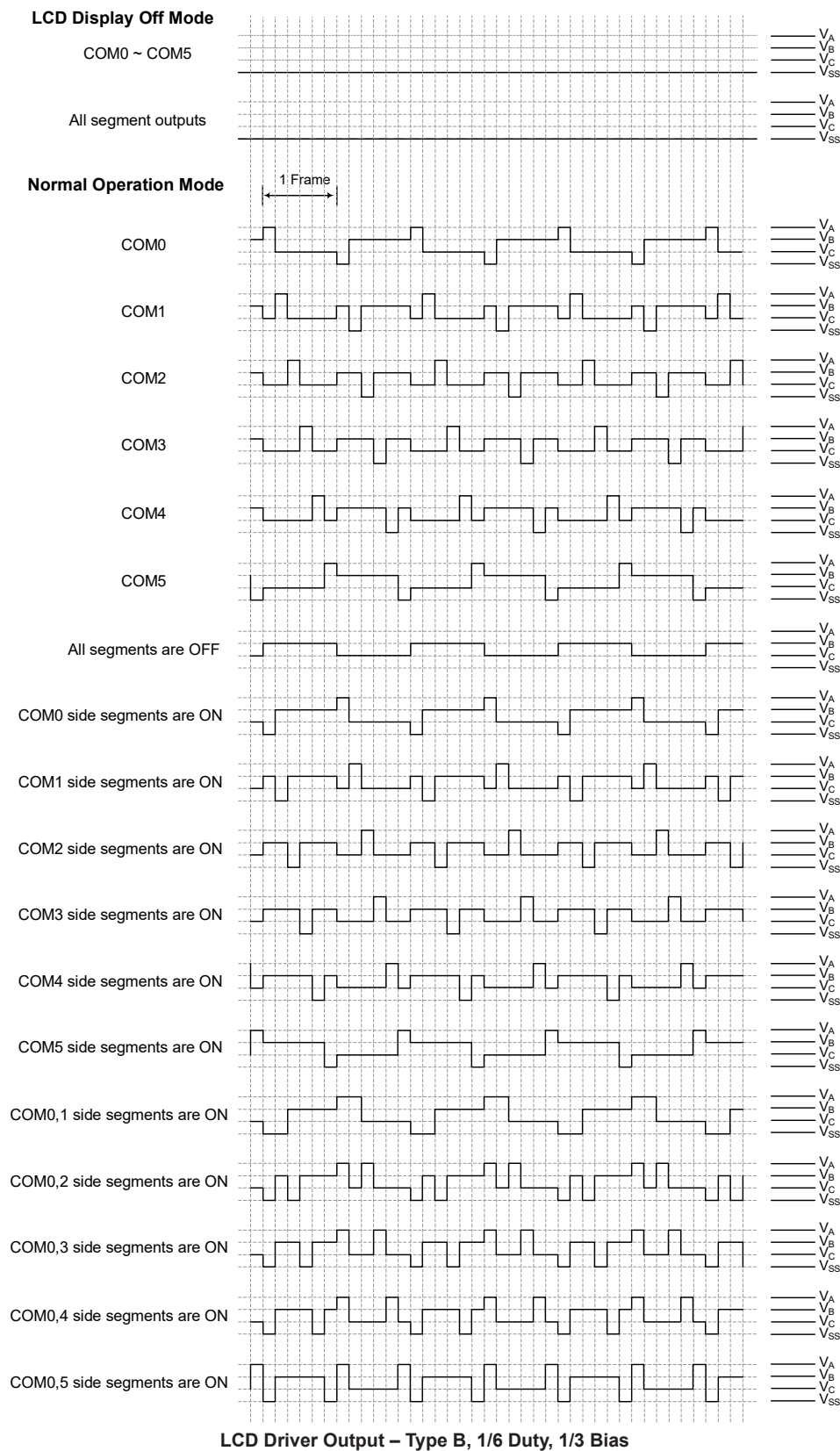
COM0,3 side segments are ON

COM0,4 side segments are ON

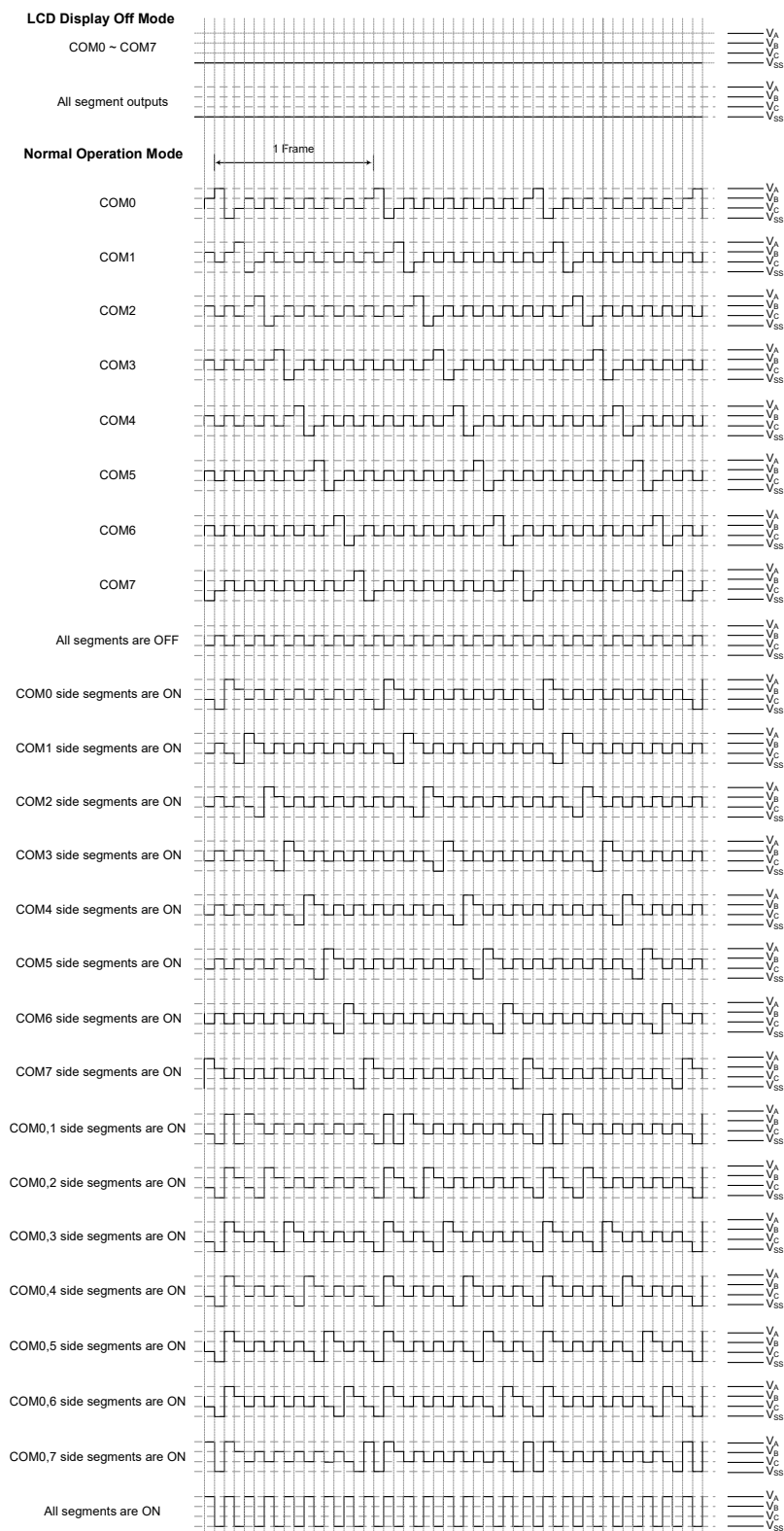
COM0,5 side segments are ON

All segments are ON

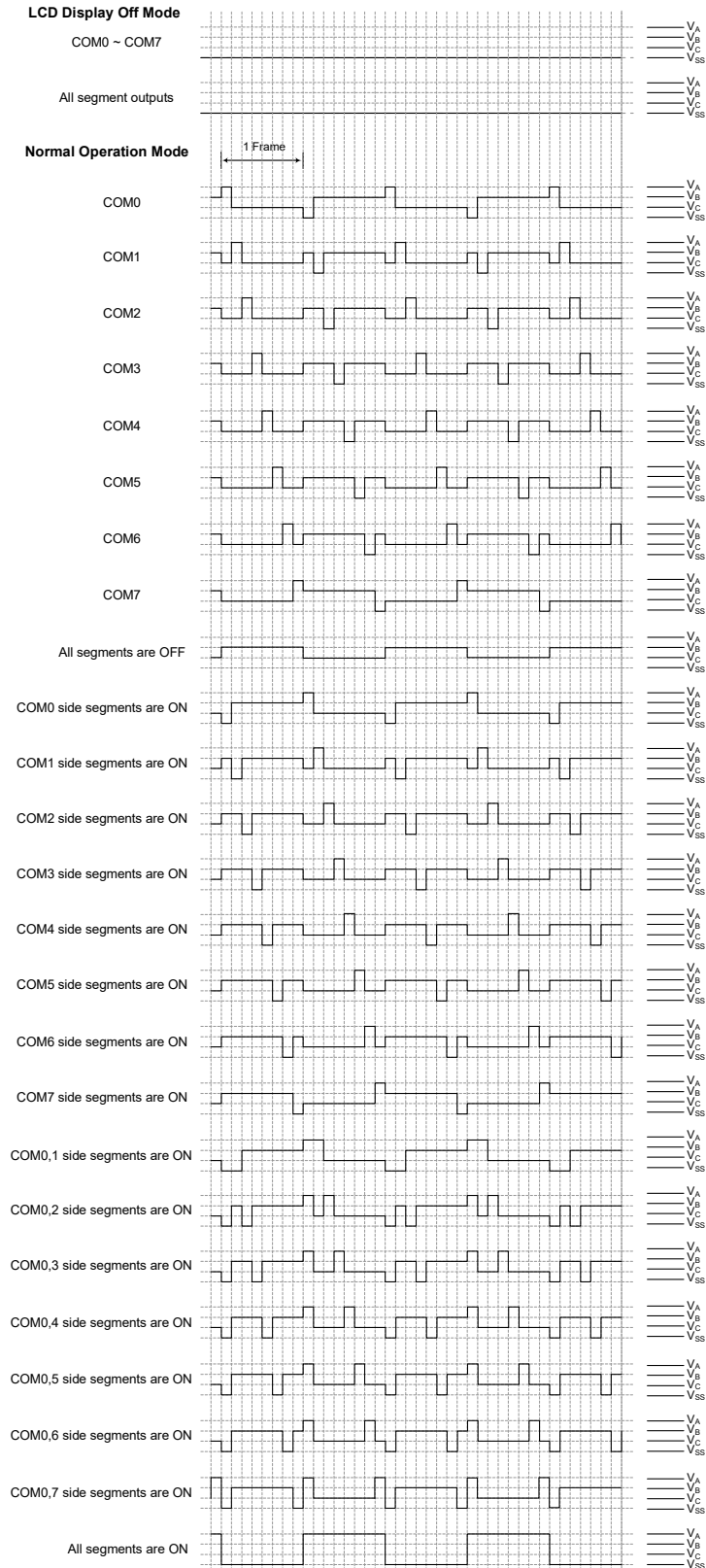
**LCD Driver Output – Type A, 1/6 Duty, 1/3 Bias**



**R Type, 8 COM, 1/3 Bias**

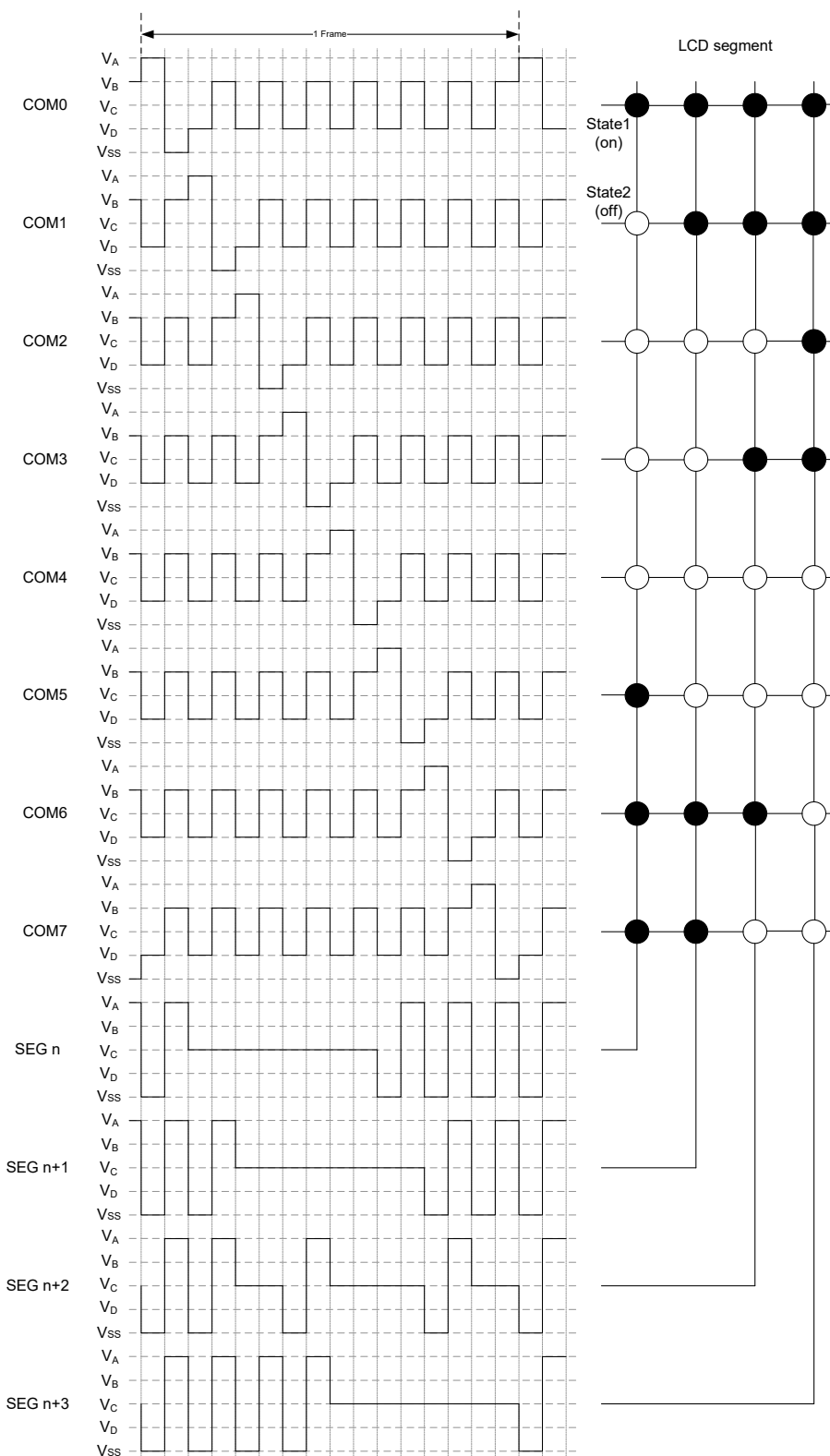


**LCD Driver Output – Type A, 1/8 Duty, 1/3 Bias**



**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**





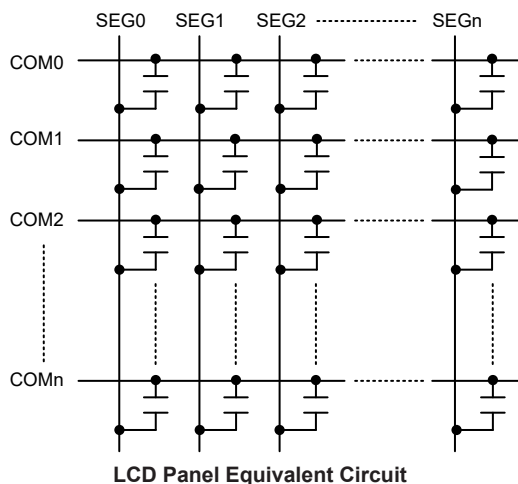
## Programming Considerations

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

### LVD Register

The Low Voltage Detector function is controlled using a single register with the name LVDC. Three bits in this register, VLVD2~VLVD0, are used to select one of eight fixed voltages below which a low voltage condition will be determined. A low voltage condition is indicated when the LVDO bit is set. If the LVDO bit is low, this indicates that the  $V_{DD}$  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	—	—	LVDO	LVDEN	—	VLVD2	VLVD1	VLVD0
R/W	—	—	R	R/W	—	R/W	R/W	R/W
POR	—	—	0	0	—	0	0	0

Bit 7~6 Unimplemented, read as “0”

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

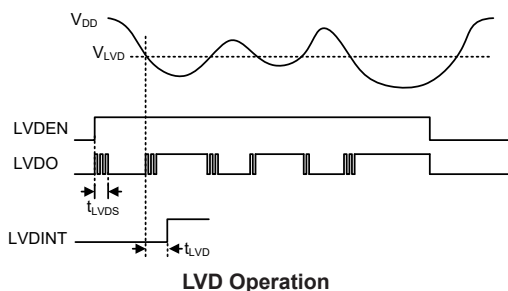
Bit 4 **LVDEN**: Low Voltage Detector function control  
 0: Disable  
 1: Enable

Bit3 Unimplemented, read as “0”

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

## LVD Operation

The Low Voltage Detector function operates by comparing the power supply voltage,  $V_{DD}$ , with a pre-specified voltage level stored in the LVDC register. This has a range of between 1.8V and 4.0V. When the power supply voltage,  $V_{DD}$ , fall below this pre-determined value, the LVDO bit will be set high indicating a low power supply voltage condition. When the device is in the SLEEP mode, the low voltage detector will be disabled even if the LVDEN bit is high. After enabling the Low Voltage Detector, a time delay  $t_{LVDS}$  should be allowed for the circuitry to stabilise before reading the LVDO bit. Note also that as the  $V_{DD}$  voltage may rise and fall rather slowly, at the voltage nears that of  $V_{LVD}$ , there may be multiple bit LVDO transitions.



The Low Voltage Detector also has its own interrupt, providing an alternative means of low voltage detection, in addition to polling the LVDO bit. The interrupt will only be generated after a delay of  $t_{LVD}$  after the LVDO bit has been set high by a low voltage condition. In this case, the LVF interrupt request flag will be set, causing an interrupt to be generated if  $V_{DD}$  falls below the preset LVD voltage. This will cause the device to wake up from the IDLE Mode, however if the Low Voltage Detector wake-up function is not required then the LVF flag should be first set high before the device enters the IDLE Mode.

## Interrupts

Interrupts are an important part of any microcontroller system. When an external event or an internal function such as a Timer Module or an A/D converter requires microcontroller attention, their corresponding interrupt will enforce a temporary suspension of the main program allowing the microcontroller to direct attention to their respective needs. The device contains several external interrupt and internal interrupt functions. The external interrupts are generated by the action of the external INT0~INT3 pins, while the internal interrupts are generated by various internal functions such as the Timer Modules, Time Bases, Low Voltage Detector (LVD), EEPROM and the A/D converter.

### 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. The registers fall into three categories. The first is the INTC0~INTC3 registers which setup the primary interrupts, the second is the MFI0~MFI1 registers which setup the Multi-function interrupts. Finally there is an INTEG register to setup the external interrupts trigger edge type.

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

Function	Enable Bit	Request Flag	Notes
Global	EMI	—	—
INTn Pin	INTnE	INTnF	n=0~3
A/D Converter	ADE	ADF	—
Multi-function	MFnE	MFnF	n=0~1
Time Base	TBnE	TBnF	n=0~1
LVD	LVE	LVF	—
EEPROM	DEE	DEF	—
UART	URE	URF	—
SIM	SIME	SIMF	—
PTM	PTMPE	PTMPF	—
	PTMAE	PTMAF	
CTM	CTMnPE	CTMnPF	n=0~2
	CTMnAE	CTMnAF	

**Interrupt Register Bit Naming Conventions**

Register Name	Bit							
	7	6	5	4	3	2	1	0
INTEG	INT3S1	INT3S0	INT2S1	INT2S0	INT1S1	INT1S0	INT0S1	INT0S0
INTC0	—	MF0F	INT1F	INT0F	MF0E	INT1E	INT0E	EMI
INTC1	INT2F	URF	SIMF	MF1F	INT2E	URE	SIME	MF1E
INTC2	ADF	TB1F	TB0F	INT3F	ADE	TB1E	TB0E	INT3E
INTC3	—	—	DEF	LVF	—	—	DEE	LVE
MF10	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE
MF11	PTMAF	PTMPF	CTM2AF	CTM2PF	PTMAE	PTMPE	CTM2AE	CTM2PE

**Interrupt Register List**

• **INTEG 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

• **INTC0 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	MF0F	INT1F	INT0F	MF0E	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      **MF0F**: Multi-function 0 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5      **INT1F**: External interrupt 1 request flag  
0: No request  
1: Interrupt request
- Bit 4      **INT0F**: External interrupt 0 request flag  
0: No request  
1: Interrupt request
- Bit 3      **MF0E**: Multi-function 0 interrupt control  
0: Disable  
1: Enable
- Bit 2      **INT1E**: External interrupt 1 control  
0: Disable  
1: Enable
- Bit 1      **INT0E**: External interrupt 0 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	INT2F	URF	SIMF	MF1F	INT2E	URE	SIME	MF1E
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      **INT2F**: INT2 Interrupt Request Flag  
0: No request  
1: Interrupt request
- Bit 6      **URF**: UART interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5      **SIMF**: SIM Interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **MF1F**: Multi-function 1 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3      **INT2E**: INT2 interrupt control  
0: Disable  
1: Enable
- Bit 2      **URE**: UART interrupt control  
0: Disable  
1: Enable

- Bit 1      **SIME**: SIM Interrupt control  
0: Disable  
1: Enable
- Bit 0      **MF1E**: Multi-function 1 interrupt control  
0: Disable  
1: Enable

• **INTC2 Register**

Bit	7	6	5	4	3	2	1	0
Name	ADF	TB1F	TB0F	INT3F	ADE	TB1E	TB0E	INT3E
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7      **ADF**: A/D converter interrupt request flag  
0: No request  
1: Interrupt request
- Bit 6      **TB1F**: Time Base 1 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5      **TB0F**: Time Base 0 interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **INT3F**: INT3 Interrupt Request Flag  
0: No request  
1: Interrupt request
- Bit 3      **ADE**: A/D converter interrupt control  
0: Disable  
1: Enable
- Bit 2      **TB1E**: Time Base 1 interrupt control  
0: Disable  
1: Enable
- Bit 1      **TB0E**: Time Base 0 interrupt control  
0: Disable  
1: Enable
- Bit 0      **INT3E**: INT3 interrupt control  
0: Disable  
1: Enable

• **INTC3 Register**

Bit	7	6	5	4	3	2	1	0
Name	—	—	DEF	LVF	—	—	DEE	LVE
R/W	—	—	R/W	R/W	—	—	R/W	R/W
POR	—	—	0	0	—	—	0	0

- Bit 7~6      Unimplemented, read as “0”
- Bit 5      **DEF**: Data EEPROM interrupt request flag  
0: No request  
1: Interrupt request
- Bit 4      **LVF**: LVD interrupt request flag  
0: No request  
1: Interrupt request
- Bit 3~2      Unimplemented, read as “0”

- Bit 1      **DEE**: Data EEPROM interrupt control  
0: Disable  
1: Enable
- Bit 0      **LVE**: LVD interrupt control  
0: Disable  
1: Enable

• **MFIO Register**

Bit	7	6	5	4	3	2	1	0
Name	CTM1AF	CTM1PF	CTM0AF	CTM0PF	CTM1AE	CTM1PE	CTM0AE	CTM0PE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

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

• **MF11 Register**

Bit	7	6	5	4	3	2	1	0
Name	PTMAF	PTMPF	CTM2AF	CTM2PF	PTMAE	PTMPE	CTM2AE	CTM2PE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

- Bit 7      **PTMAF**: PTM CCRA comparator interrupt request flag  
0: No request  
1: Interrupt request
- Bit 6      **PTMPF**: PTM CCRP comparator interrupt request flag  
0: No request  
1: Interrupt request
- Bit 5      **CTM2AF**: CTM2 CCRA comparator interrupt request flag  
0: No request  
1: Interrupt request



Bit 4	<b>CTM2PF</b> : CTM2 CCRP comparator interrupt request flag 0: No request 1: Interrupt request
Bit 3	<b>PTMAE</b> : PTM CCRA comparator interrupt control 0: Disable 1: Enable
Bit 2	<b>PTMPE</b> : PTM CCRP comparator interrupt control 0: Disable 1: Enable
Bit 1	<b>CTM2AE</b> : CTM2 CCRA comparator interrupt control 0: Disable 1: Enable
Bit 0	<b>CTM2PE</b> : CTM2 CCRP comparator interrupt control 0: Disable 1: Enable

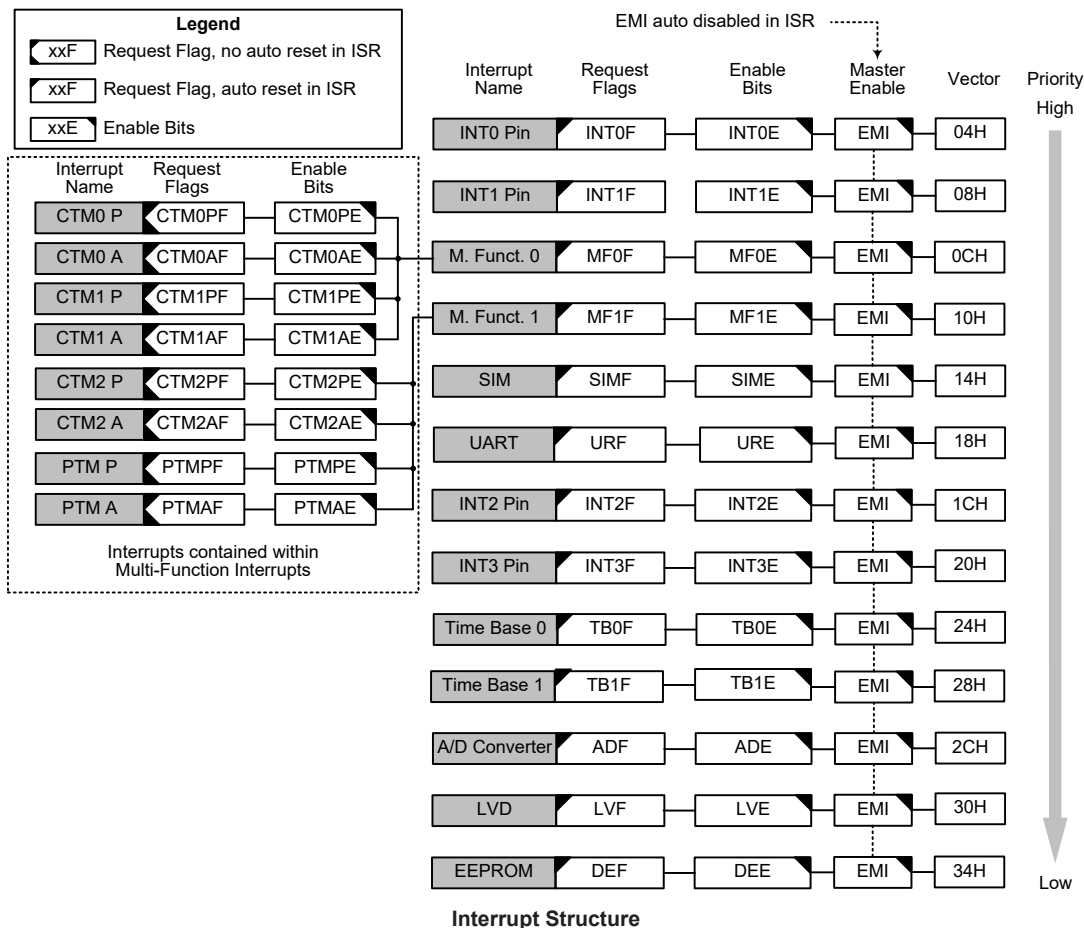
### Interrupt Operation

When the conditions for an interrupt event occur, such as a TM Comparator P, 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 an “RETI”, which retrieves the original Program Counter address from the stack and allows the microcontroller to continue with normal execution at the point where the interrupt occurred.

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

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



## External Interrupts

The external interrupts are controlled by signal transitions on the pins INT0~INT3. An external interrupt request will take place when the external interrupt request flags, INT0F~INT3F, are set, which will occur when a transition, whose type is chosen by the edge select bits, appears on the external interrupt pins. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and respective external interrupt enable bit, INT0E~INT3E, must first be set. Additionally the correct interrupt edge type must be selected using the INTEG register to enable the external interrupt function and to choose the trigger edge type. As the external interrupt pins are pin-shared with I/O pins, they can only be configured as external interrupt pins if their external interrupt enable bit in the corresponding interrupt register has been set and the external interrupt pin is selected by the corresponding pin-shared function selection bits. The pin must also be setup as an input by setting the corresponding bit in the port control register.

When the interrupt is enabled, the stack is not full and the correct transition type appears on the external interrupt pin, a subroutine call to the external interrupt vector, will take place. When the interrupt is serviced, the external interrupt request flags, INT0F~INT3F, will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts. Note that any pull-high resistor selections on the external interrupt pins will remain valid even if the pin is used as an external interrupt input. The INTEG register is used to select the type of active edge that will trigger the external interrupt. A choice of either rising or falling or both edge types can be chosen to trigger an external interrupt. Note that the INTEG register can also be used to disable the external interrupt function.

## **A/D Converter Interrupt**

The A/D converter interrupt is controlled by the termination of an A/D conversion process. An A/D converter interrupt request will take place when the A/D Converter Interrupt request flag, ADF, is set, which occurs when the A/D conversion process finishes. To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, and A/D converter interrupt enable bit, ADE, must first be set. When the interrupt is enabled, the stack is not full and the A/D conversion process has ended, a subroutine call to the A/D converter interrupt vector, will take place. When the 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 this device there are several 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.

A multi-function interrupt request will take place when any of the multi-function interrupt request flags, MF<sub>n</sub>F are set. The multi-function interrupt flags will be set when any of their included functions generate an interrupt request flag. When the multi-function interrupt is enabled and the stack is not full, and either one of the interrupts contained within each of multi-function interrupt occurs, a subroutine call to one of the multi-function interrupt vectors will take place. When the interrupt is serviced, the related multi-function request flag will be automatically reset and the EMI bit will be automatically cleared to disable other interrupts.

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

## **Timer Module Interrupts**

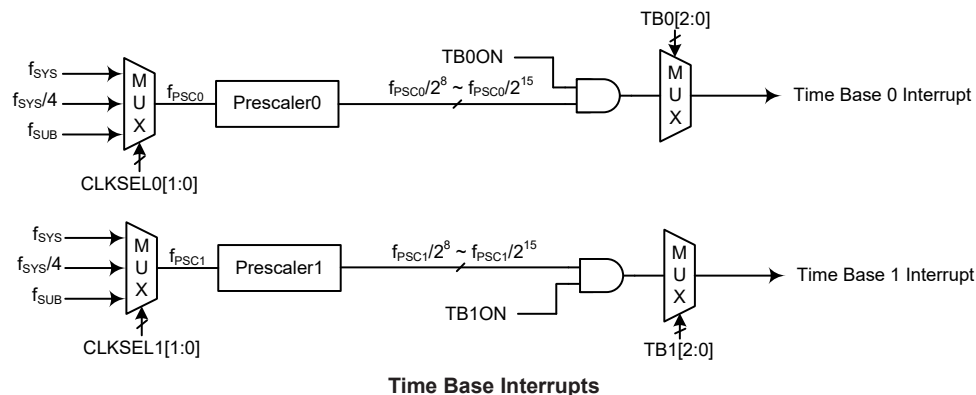
Each of the Compact and Periodic Type TM has two interrupts. All of the TM interrupts are contained within the Multi-function Interrupts. For all of the TM types there are two interrupt request flags of CTM<sub>n</sub>PF, CTM<sub>n</sub>AF and PTMPF, PTMAF, and two enable bits of CTM<sub>n</sub>PE, CTM<sub>n</sub>AE and PTMPE, PTMAE. A TM interrupt request will take place when any of the TM request flags are set, a situation which occurs when a TM comparator P or A match situation happens.

To allow the program to branch to its respective interrupt vector address, the global interrupt enable bit, EMI, respective TM interrupt enable bit, and relevant multi-function interrupt enable bit, MF<sub>n</sub>E, must first be set. When the interrupt is enabled, the stack is not full and a TM comparator match situation occurs, a subroutine call to the relevant multi-function interrupt vector locations, will take place. When the TM interrupt is serviced, the EMI bit will be automatically cleared to disable other interrupts, however only the related MF<sub>n</sub>F 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.

## **Time Base Interrupts**

The function of the Time Base interrupts is to provide regular time signal in the form of an internal interrupt. They are controlled by the overflow signals from their respective timer functions. When these happen 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, TB<sub>n</sub>F, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

The purpose of the Time Base interrupt is to provide an interrupt signal at fixed time periods. Its clock source,  $f_{PSC0}$  or  $f_{PSC1}$ , originates from the internal clock source  $f_{SYS}$ ,  $f_{SYS}/4$  or  $f_{SUB}$  and then passes through a divider, the division ratio of which is selected by programming the appropriate bits in the TB0C and TB1C registers to obtain longer interrupt periods whose value ranges. The clock source which in turn controls the Time Base interrupt period is selected using the CLKSEL0[1:0] and CLKSEL1[1:0] bits in the PSC0R and PSC1R register.



#### • PSCnR Register (n=0~1)

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

Bit 7~2 Unimplemented, read as “0”

Bit 1~0 **CLKSELn1~CLKSELn0**: Prescaler clock source  $f_{PSCn}$  selection  
 00:  $f_{SYS}$   
 01:  $f_{SYS}/4$   
 1x:  $f_{SUB}$

#### • TB0C Register

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

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

Bit 6~3 Unimplemented, read as “0”

Bit 2~0 **TB02~TB00**: Select Time Base 0 time-out period  
 000:  $2^8/f_{PSC0}$   
 001:  $2^9/f_{PSC0}$   
 010:  $2^{10}/f_{PSC0}$   
 011:  $2^{11}/f_{PSC0}$   
 100:  $2^{12}/f_{PSC0}$   
 101:  $2^{13}/f_{PSC0}$   
 110:  $2^{14}/f_{PSC0}$   
 111:  $2^{15}/f_{PSC0}$

• **TB1C Register**

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

- Bit 7      **TB1ON**: Time Base 1 control  
0: Disable  
1: Enable
- Bit 6~3      Unimplemented, read as “0”
- Bit 2~0      **TB12~TB10**: Select Time Base 1 time-out period  
000:  $2^8/f_{PSC1}$   
001:  $2^9/f_{PSC1}$   
010:  $2^{10}/f_{PSC1}$   
011:  $2^{11}/f_{PSC1}$   
100:  $2^{12}/f_{PSC1}$   
101:  $2^{13}/f_{PSC1}$   
110:  $2^{14}/f_{PSC1}$   
111:  $2^{15}/f_{PSC1}$

### Serial Interface Module Interrupt

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

### LVD Interrupt

An LVD interrupt request will take place when the LVD interrupt request flag, LVF, is set, which occurs when the Low Voltage Detector function detects a low power supply voltage. To allow the program to branch to its interrupt vector address, the global interrupt enable bit, EMI, and Low Voltage interrupt enable bit, LVE, must first be set. When the interrupt is enabled, the stack is not full and a low voltage condition occurs, a subroutine call to the LCD interrupt vector, will take place. When the Low Voltage interrupt is serviced, the respective interrupt request flag, LVF, will be automatically reset and the EMI bit will be cleared to disable other interrupts.

### EEPROM Interrupt

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

## UART Transfer Interrupt

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

## Interrupt Wake-up Function

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

## Programming Considerations

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

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

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

Every interrupt has the capability of waking up the microcontroller when it is in SLEEP or IDLE Mode, the wake-up being generated when the interrupt request flag changes from low to high. If it is required to prevent a certain interrupt from waking up the microcontroller then its respective request flag should be first set high before enter SLEEP or IDLE Mode.

As only the Program Counter is pushed onto the stack, then when the interrupt is serviced, if the contents of the accumulator, status register or other registers are altered by the interrupt service program, their contents should be saved to the memory at the beginning of the interrupt service routine. To return from an interrupt subroutine, either an RET or RETI instruction may be executed. The RETI instruction in addition to executing a return to the main program also automatically sets the EMI bit high to allow further interrupts. The RET instruction however only executes a return to the main program leaving the EMI bit in its present zero state and therefore disabling the execution of further interrupts.

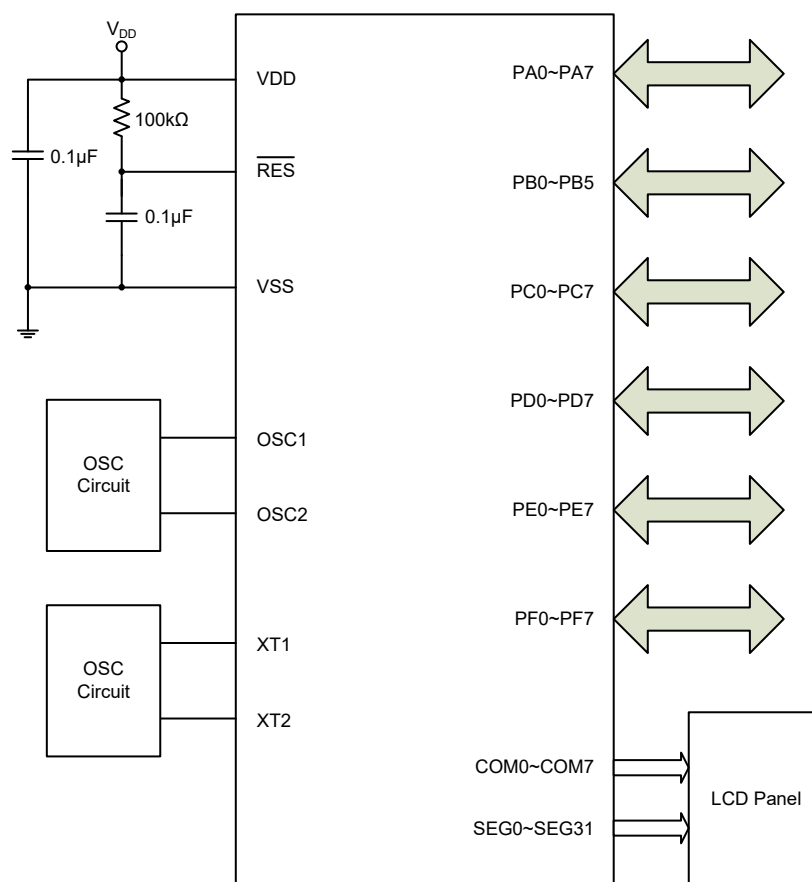
## Configuration Options

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

No.	Options
<b>Oscillator Option</b>	
1	HIRC Frequency Selection – $f_{HIRC}$ : 4MHz, 8MHz or 12MHz

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

## Application Circuits



## Instruction Set

### Introduction

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

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

### Instruction Timing

Most instructions are implemented within one instruction cycle. The exceptions to this are branch, call, or table read instructions where two instruction cycles are required. One instruction cycle is equal to 4 system clock cycles, therefore in the case of an 8MHz system oscillator, most instructions would be implemented within 0.5 $\mu$ s and branch or call instructions would be implemented within 1 $\mu$ s. Although instructions which require one more cycle to implement are generally limited to the JMP, CALL, RET, RETI and table read instructions, it is important to realize that any other instructions which involve manipulation of the Program Counter Low register or PCL will also take one more cycle to implement. As instructions which change the contents of the PCL will imply a direct jump to that new address, one more cycle will be required. Examples of such instructions would be "CLR PCL" or "MOV PCL, A". For the case of skip instructions, it must be noted that if the result of the comparison involves a skip operation then this will also take one more cycle, if no skip is involved then only one cycle is required.

### Moving and Transferring Data

The transfer of data within the microcontroller program is one of the most frequently used operations. Making use of several kinds of MOV instructions, data can be transferred from registers to the Accumulator and vice-versa as well as being able to move specific immediate data directly into the Accumulator. One of the most important data transfer applications is to receive data from the input ports and transfer data to the output ports.

### Arithmetic Operations

The ability to perform certain arithmetic operations and data manipulation is a necessary feature of most microcontroller applications. Within the Holtek microcontroller instruction set are a range of add and subtract instruction mnemonics to enable the necessary arithmetic to be carried out. Care must be taken to ensure correct handling of carry and borrow data when results exceed 255 for addition and less than 0 for subtraction. The increment and decrement instructions such as INC, INCA, DEC and DECA provide a simple means of increasing or decreasing by a value of one of the values in the destination specified.



## Logical and Rotate Operation

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

## Branches and Control Transfer

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

## Bit Operations

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

## Table Read Operations

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

## Other Operations

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

## Instruction Set Summary

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

### Table Conventions

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

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

Mnemonic	Description	Cycles	Flag Affected
<b>Data Move</b>			
MOV A,[m]	Move Data Memory to ACC	1	None
MOV [m],A	Move ACC to Data Memory	1 <sup>Note</sup>	None
MOV A,x	Move immediate data to ACC	1	None
<b>Bit Operation</b>			
CLR [m].i	Clear bit of Data Memory	1 <sup>Note</sup>	None
SET [m].i	Set bit of Data Memory	1 <sup>Note</sup>	None
<b>Branch Operation</b>			
JMP addr	Jump unconditionally	2	None
SZ [m]	Skip if Data Memory is zero	1 <sup>Note</sup>	None
SZA [m]	Skip if Data Memory is zero with data movement to ACC	1 <sup>Note</sup>	None
SZ [m].i	Skip if bit i of Data Memory is zero	1 <sup>Note</sup>	None
SNZ [m]	Skip if Data Memory is not zero	1 <sup>Note</sup>	None
SNZ [m].i	Skip if bit i of Data Memory is not zero	1 <sup>Note</sup>	None
SIZ [m]	Skip if increment Data Memory is zero	1 <sup>Note</sup>	None
SDZ [m]	Skip if decrement Data Memory is zero	1 <sup>Note</sup>	None
SIZA [m]	Skip if increment Data Memory is zero with result in ACC	1 <sup>Note</sup>	None
SDZA [m]	Skip if decrement Data Memory is zero with result in ACC	1 <sup>Note</sup>	None
CALL addr	Subroutine call	2	None
RET	Return from subroutine	2	None
RET A,x	Return from subroutine and load immediate data to ACC	2	None
RETI	Return from interrupt	2	None
<b>Table Read Operation</b>			
TABRD [m]	Read table (specific page) to TBLH and Data Memory	2 <sup>Note</sup>	None
TABRDL [m]	Read table (last page) to TBLH and Data Memory	2 <sup>Note</sup>	None
ITABRD [m]	Increment table pointer TBLP first and Read table (specific page) to TBLH and Data Memory	2 <sup>Note</sup>	None
ITABRDL [m]	Increment table pointer TBLP first and Read table (last page) to TBLH and Data Memory	2 <sup>Note</sup>	None
<b>Miscellaneous</b>			
NOP	No operation	1	None
CLR [m]	Clear Data Memory	1 <sup>Note</sup>	None
SET [m]	Set Data Memory	1 <sup>Note</sup>	None
CLR WDT	Clear Watchdog Timer	1	TO, PDF
SWAP [m]	Swap nibbles of Data Memory	1 <sup>Note</sup>	None
SWAPA [m]	Swap nibbles of Data Memory with result in ACC	1	None
HALT	Enter power down mode	1	TO, PDF

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

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

## Extended Instruction Set

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

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

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

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

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

## Instruction Definition

<b>ADC A,[m]</b>	Add Data Memory to ACC with Carry
Description	The contents of the specified Data Memory, Accumulator and the carry flag are added. 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 $\leftarrow$ Program Counter + 1 Program Counter $\leftarrow$ addr
Affected flag(s)	None
<b>CLR [m]</b>	Clear Data Memory
Description	Each bit of the specified Data Memory is cleared to 0.
Operation	[m] $\leftarrow$ 00H
Affected flag(s)	None
<b>CLR [m].i</b>	Clear bit of Data Memory
Description	Bit i of the specified Data Memory is cleared to 0.
Operation	[m].i $\leftarrow$ 0
Affected flag(s)	None
<b>CLR WDT</b>	Clear Watchdog Timer
Description	The TO, PDF flags and the WDT are all cleared.
Operation	WDT cleared TO $\leftarrow$ 0 PDF $\leftarrow$ 0
Affected flag(s)	TO, PDF
<b>CPL [m]</b>	Complement Data Memory
Description	Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa.
Operation	[m] $\leftarrow$ $\overline{[m]}$
Affected flag(s)	Z
<b>CPLA [m]</b>	Complement Data Memory with result in ACC
Description	Each bit of the specified Data Memory is logically complemented (1's complement). Bits which previously contained a 1 are changed to 0 and vice versa. The complemented result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	ACC $\leftarrow$ $\overline{[m]}$
Affected flag(s)	Z
<b>DAA [m]</b>	Decimal-Adjust ACC for addition with result in Data Memory
Description	Convert the contents of the Accumulator value to a BCD (Binary Coded Decimal) value resulting from the previous addition of two BCD variables. If the low nibble is greater than 9 or if AC flag is set, then a value of 6 will be added to the low nibble. Otherwise the low nibble remains unchanged. If the high nibble is greater than 9 or if the C flag is set, then a value of 6 will be added to the high nibble. Essentially, the decimal conversion is performed by adding 00H, 06H, 60H or 66H depending on the Accumulator and flag conditions. Only the C flag may be affected by this instruction which indicates that if the original BCD sum is greater than 100, it allows multiple precision decimal addition.
Operation	[m] $\leftarrow$ ACC + 00H or [m] $\leftarrow$ ACC + 06H or [m] $\leftarrow$ ACC + 60H or [m] $\leftarrow$ ACC + 66H
Affected flag(s)	C

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



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

<b>RLA [m]</b>	Rotate Data Memory left with result in ACC
Description	The contents of the specified Data Memory are rotated left by 1 bit with bit 7 rotated into bit 0. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC.(i+1) \leftarrow [m].i; (i=0\sim6)$ $ACC.0 \leftarrow [m].7$
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	$ACC.i \leftarrow [m].(i+1); (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
Affected flag(s)	C
<b>SBC A,[m]</b>	Subtract Data Memory from ACC with Carry
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
<b>SBC A, x</b>	Subtract immediate data from ACC with Carry
Description	The immediate data and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
<b>SBCM A,[m]</b>	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[m] \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
<b>SDZ [m]</b>	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] - 1$ Skip if $[m]=0$
Affected flag(s)	None
<b>SDZA [m]</b>	Skip if decrement Data Memory is zero with result in ACC
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0, the following instruction is skipped. The result is stored in the Accumulator but the specified Data Memory contents remain unchanged. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a two cycle instruction. If the result is not 0, the program proceeds with the following instruction.
Operation	$ACC \leftarrow [m] - 1$ 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>TABRDL [m]</b>	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
<b>ITABRD [m]</b>	Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
<b>ITABRDL [m]</b>	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) 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	$[m].i \leftarrow [m].(i+1); (i=0\sim6)$ $[m].7 \leftarrow [m].0$
Affected flag(s)	None
<b>LRRA [m]</b>	Rotate Data Memory right with result in ACC
Description	Data in the specified Data Memory is rotated right by 1 bit with bit 0 rotated into bit 7. The rotated result is stored in the Accumulator and the contents of the Data Memory remain unchanged.
Operation	$ACC.i \leftarrow [m].(i+1); (i=0\sim6)$ $ACC.7 \leftarrow [m].0$
Affected flag(s)	None
<b>LRRC [m]</b>	Rotate Data Memory right through Carry
Description	The contents of the specified Data Memory and the carry flag are rotated right by 1 bit. Bit 0 replaces the Carry bit and the original carry flag is rotated into bit 7.
Operation	$[m].i \leftarrow [m].(i+1); (i=0\sim6)$ $[m].7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC.i \leftarrow [m].(i+1); (i=0\sim6)$ $ACC.7 \leftarrow C$ $C \leftarrow [m].0$
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	$ACC \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ
<b>LSBCM A,[m]</b>	Subtract Data Memory from ACC with Carry and result in Data Memory
Description	The contents of the specified Data Memory and the complement of the carry flag are subtracted from the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[m] \leftarrow ACC - [m] - \overline{C}$
Affected flag(s)	OV, Z, AC, C, SC, CZ

<b>LSDZ [m]</b>	Skip if decrement Data Memory is 0
Description	The contents of the specified Data Memory are first decremented by 1. If the result is 0 the following instruction is skipped. As this requires the insertion of a dummy instruction while the next instruction is fetched, it is a three cycle instruction. If the result is not 0 the program proceeds with the following instruction.
Operation	$[m] \leftarrow [m] - 1$ 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] $\neq$ 0
Affected flag(s)	None
<b>LSUB A,[m]</b>	Subtract Data Memory from ACC
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Accumulator. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$ACC \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C, SC, CZ
<b>LSUBM A,[m]</b>	Subtract Data Memory from ACC with result in Data Memory
Description	The specified Data Memory is subtracted from the contents of the Accumulator. The result is stored in the Data Memory. Note that if the result of subtraction is negative, the C flag will be cleared to 0, otherwise if the result is positive or zero, the C flag will be set to 1.
Operation	$[m] \leftarrow ACC - [m]$
Affected flag(s)	OV, Z, AC, C, SC, CZ
<b>LSWAP [m]</b>	Swap nibbles of Data Memory
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged.
Operation	$[m].3 \sim [m].0 \leftrightarrow [m].7 \sim [m].4$
Affected flag(s)	None
<b>LSWAPA [m]</b>	Swap nibbles of Data Memory with result in ACC
Description	The low-order and high-order nibbles of the specified Data Memory are interchanged. The result is stored in the Accumulator. The contents of the Data Memory remain unchanged.
Operation	$ACC.3 \sim ACC.0 \leftarrow [m].7 \sim [m].4$ $ACC.7 \sim ACC.4 \leftarrow [m].3 \sim [m].0$
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>LTABRDL [m]</b>	Read table (last page) to TBLH and Data Memory
Description	The low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
<b>LITABRD [m]</b>	Increment table pointer low byte first and read table (specific page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the program code (specific page) addressed by the table pointer (TBHP and TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) TBLH ← program code (high byte)
Affected flag(s)	None
<b>LITABRDL [m]</b>	Increment table pointer low byte first and read table (last page) to TBLH and Data Memory
Description	Increment table pointer low byte, TBLP, first and then the low byte of the program code (last page) addressed by the table pointer (TBLP) is moved to the specified Data Memory and the high byte moved to TBLH.
Operation	[m] ← program code (low byte) 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

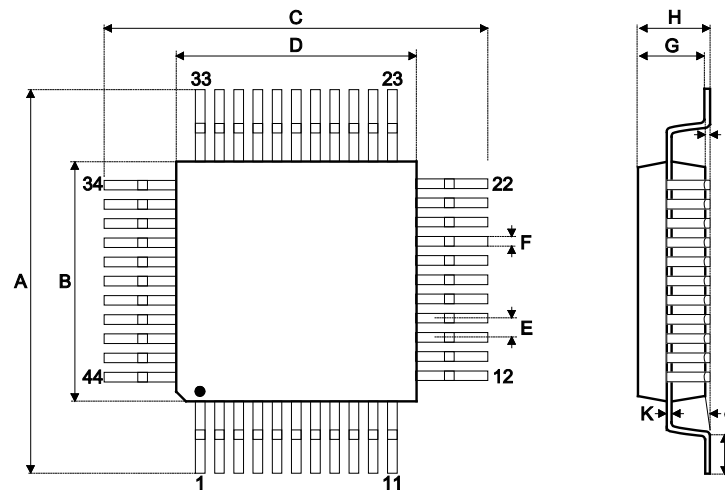
## Package Information

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 Information](#).

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

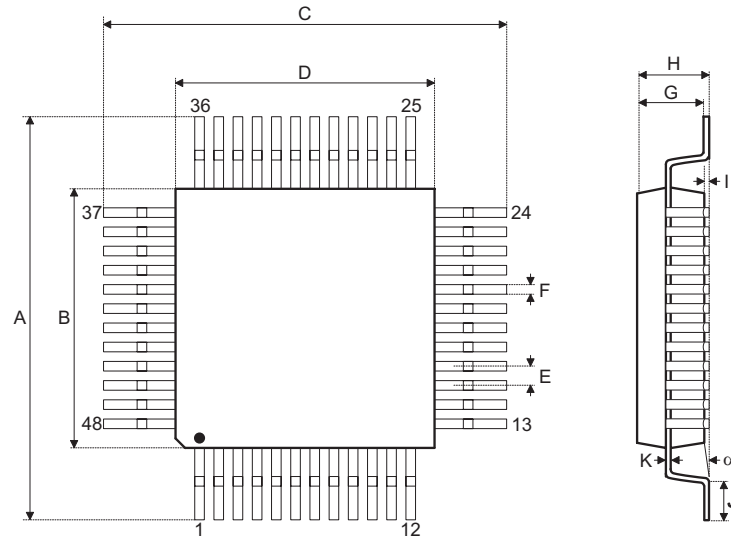
- [Package Information \(include Outline Dimensions, Product Tape and Reel Specifications\)](#)
- [The Operation Instruction of Packing Materials](#)
- [Carton information](#)

**44-pin LQFP (10mm×10mm) (FP2.0mm) 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	0.032 BSC		
F	0.012	0.015	0.018
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	0.80 BSC		
F	0.30	0.37	0.45
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°

**48-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	0.020 BSC		
F	0.007	0.009	0.011
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	0.50 BSC		
F	0.17	0.22	0.27
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© 2023 by HOLTEK SEMICONDUCTOR INC. All Rights Reserved.

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